

Hortonworks Data Platform

Security

(October 30, 2017)

Hortonworks Data Platform: Security

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

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

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

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



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

Table of Contents

1. HDP Security Overview	1
1.1. What's New in This Release	1
1.2. Understanding Data Lake Security	5
1.3. HDP Security Features	6
1.3.1. Administration	7
1.3.2. Authentication and Perimeter Security	8
1.3.3. Authorization	9
1.3.4. Audit	11
1.3.5. Data Protection	11
2. Authentication	12
2.1. Enabling Kerberos Authentication Using Ambari	12
2.1.1. Kerberos Overview	12
2.1.2. Kerberos Principals	13
2.1.3. Installing and Configuring the KDC	14
2.1.4. Enabling Kerberos Security	19
2.1.5. Kerberos Client Packages	24
2.1.6. Disabling Kerberos Security	24
2.1.7. Customizing the Attribute Template	25
2.1.8. Managing Admin Credentials	25
2.2. Configuring HDP Components for Kerberos Using Ambari	26
2.2.1. Configuring Kafka for Kerberos Using Ambari	26
2.2.2. Configuring Storm for Kerberos Using Ambari	41
2.3. Configuring Ambari Authentication with LDAP or AD	46
2.3.1. Configuring Ambari for LDAP or Active Directory Authentication	46
2.3.2. Configuring Ranger Authentication with UNIX, LDAP, or AD	52
2.3.3. Encrypting Database and LDAP Passwords in Ambari	60
2.4. Configuring LDAP Authentication in Hue	62
2.4.1. Enabling the LDAP Backend	62
2.4.2. Enabling User Authentication with Search Bind	62
2.4.3. Setting the Search Base to Find Users and Groups	63
2.4.4. Specifying the URL of the LDAP Server	64
2.4.5. Specifying LDAPS and StartTLS Support	64
2.4.6. Specifying Bind Credentials for LDAP Searches	64
2.4.7. Synchronizing Users and Groups	64
2.4.8. Setting Search Bind Authentication and Importing Users and Groups	65
2.4.9. Setting LDAP Users' Filter	65
2.4.10. Setting an LDAP Groups Filter	66
2.4.11. Setting Multiple LDAP Servers	66
2.5. Advanced Security Options for Ambari	67
2.5.1. Configuring Ambari for Non-Root	67
2.5.2. Optional: Ambari Web Inactivity Timeout	71
2.5.3. Optional: Set Up Kerberos for Ambari Server	72
2.5.4. Optional: Set Up Two-Way SSL Between Ambari Server and Ambari Agents	73
2.5.5. Optional: Configure Ciphers and Protocols for Ambari Server	73
2.5.6. Optional: HTTP Cookie Persistence	73
2.6. Enabling SPNEGO Authentication for Hadoop	74

2.6.1. Configure Ambari Server for Authenticated HTTP	74
2.6.2. Configuring HTTP Authentication for HDFS, YARN, MapReduce2, HBase, Oozie, Falcon and Storm	74
2.6.3. Enabling Browser Access to a SPNEGO-enabled Web UI	75
2.7. Setting Up Kerberos Authentication for Non-Ambari Clusters	76
2.7.1. Preparing Kerberos	76
2.7.2. Configuring HDP for Kerberos	82
2.7.3. Setting up One-Way Trust with Active Directory	110
2.7.4. Configuring Proxy Users	112
2.8. Perimeter Security with Apache Knox	112
2.8.1. Apache Knox Gateway Overview	112
2.8.2. Configuring the Knox Gateway	115
2.8.3. Defining Cluster Topologies	119
2.8.4. Configuring a Hadoop Server for Knox	120
2.8.5. Mapping the Internal Nodes to External URLs	126
2.8.6. Configuring Authentication	129
2.8.7. Configuring Identity Assertion	149
2.8.8. Configuring Service Level Authorization	158
2.8.9. Audit Gateway Activity	161
2.8.10. Gateway Security	164
2.8.11. Setting Up Knox Services for HA	167
2.8.12. Knox CLI Testing Tools	171
2.9. Knox SSO	172
2.9.1. Identity Providers (IdP)	172
2.9.2. Setting up Knox SSO for Ambari	177
2.9.3. Setting up Knox SSO for Ranger Web UI	178
2.9.4. Setting up the Knox Token Service for Ranger APIs	179
2.9.5. Setting up Knox SSO for Apache Atlas	181
3. Configuring Authorization in Hadoop	184
3.1. Installing Ranger Using Ambari	184
3.1.1. Overview	184
3.1.2. Installation Prerequisites	184
3.1.3. Ranger Installation	193
3.1.4. Enabling Ranger Plugins	234
3.1.5. Ranger Plugins - Kerberos Overview	269
3.2. Using Ranger to Provide Authorization in Hadoop	273
3.2.1. About Ranger Policies	274
3.2.2. Using the Ranger Console	279
3.2.3. Configuring Resource-Based Services	284
3.2.4. Resource-Based Policy Management	301
3.2.5. Row-level Filtering and Column Masking in Hive	334
3.2.6. Adding Tag-based Service	348
3.2.7. Tag-Based Policy Management	350
3.2.8. Users/Groups and Permissions Administration	369
3.2.9. Reports Administration	381
3.2.10. Special Requirements for High Availability Environments	385
3.2.11. Adding a New Component to Apache Ranger	386
3.2.12. Developing a Custom Authorization Module	389
3.2.13. Apache Ranger Public REST API	389
4. Data Protection: Wire Encryption	424
4.1. Enabling RPC Encryption	424

4.2. Enabling Data Transfer Protocol	425
4.3. Enabling SSL: Understanding the Hadoop SSL Keystore Factory	425
4.4. Creating and Managing SSL Certificates	427
4.4.1. Obtain a Certificate from a Trusted Third-Party Certification Authority (CA)	427
4.4.2. Create and Set Up an Internal CA (OpenSSL)	429
4.4.3. Installing Certificates in the Hadoop SSL Keystore Factory (HDFS, MapReduce, and YARN)	432
4.4.4. Using a CA-Signed Certificate	433
4.5. Enabling SSL for HDP Components	434
4.6. Enable SSL for WebHDFS, MapReduce Shuffle, Tez, and YARN	435
4.7. Enable SSL for HttpFS	438
4.8. Enable SSL on Oozie	439
4.8.1. Configure the Oozie Client to Connect Using SSL	439
4.8.2. Connect to the Oozie Web UI Using SSL	440
4.8.3. Configure Oozie HCatalogJob Properties	440
4.9. Enable SSL on the HBase REST Server	440
4.10. Enable SSL on the HBase Web UI	442
4.11. Enable SSL on HiveServer2	443
4.11.1. Setting up SSL with self-signed certificates	444
4.11.2. Selectively disabling SSL protocol versions	445
4.12. Enable SSL for Kafka Clients	445
4.12.1. Configuring the Kafka Broker	445
4.12.2. Configuring Kafka Producer and Kafka Consumer	447
4.13. Enable SSL for Accumulo	448
4.13.1. Generate a Certificate Authority	448
4.13.2. Generate a Certificate/Keystore Per Host	449
4.13.3. Configure Accumulo Servers	450
4.13.4. Configure Accumulo Clients	451
4.14. Enable SSL for Apache Atlas	451
4.14.1. Configuring Apache Atlas SSL	451
4.14.2. Credential Provider Utility Script	453
4.15. SPNEGO setup for WebHCat	454
4.16. Configure SSL for Hue	454
4.16.1. Enabling SSL on Hue by Using a Private Key	455
4.16.2. Enabling SSL on Hue Without Using a Private Key	455
4.17. Configure SSL for Knox	455
4.17.1. Self-Signed Certificate with Specific Hostname for Evaluations	455
4.17.2. CA-Signed Certificates for Production	456
4.17.3. Setting Up Trust for the Knox Gateway Clients	456
4.18. Securing Phoenix	457
4.19. Set Up SSL for Ambari	457
4.19.1. Set Up Truststore for Ambari Server	458
4.20. Configure Ambari Ranger SSL	459
4.20.1. Configuring Ambari Ranger SSL Using Public CA Certificates	459
4.20.2. Configuring Ambari Ranger SSL Using a Self-Signed Certificate	474
4.20.3. Configure Ranger Admin Database for SSL-Enabled MySQL	489
4.21. Configure Non-Ambari Ranger SSL	490
4.21.1. Configuring Non-Ambari Ranger SSL Using Public CA Certificates	490
4.21.2. Configuring Non-Ambari Ranger SSL Using a Self Signed Certificate	493

4.22. Connecting to SSL-Enabled Components	497
4.22.1. Connect to SSL Enabled HiveServer2 using JDBC	498
4.22.2. Connect to SSL Enabled Oozie Server	498
5. Auditing in Hadoop	500
5.1. Using Apache Solr for Ranger Audits	500
5.1.1. Prerequisites	501
5.1.2. Installing Externally Managed SolrCloud	501
5.1.3. Configuring Externally Managed SolrCloud	502
5.1.4. Configuring Externally Managed Solr Standalone	505
5.1.5. Configuring SolrCloud for Kerberos	506
5.2. Migrating Audit Logs from DB to Solr in Ambari Clusters	510
5.3. Manually Enabling Audit Settings in Ambari Clusters	512
5.3.1. Manually Updating Ambari Solr Audit Settings	512
5.3.2. Manually Updating HDFS Audit Settings (for Ambari installs)	513
5.4. Enabling Audit Logging in Non-Ambari Clusters	514
5.5. Managing Auditing in Ranger	515
5.5.1. View Operation Details	516
5.5.2. Differentiate Events from Multiple Clusters	517
5.5.3. Access	517
5.5.4. Admin	518
5.5.5. Login Sessions	519
5.5.6. Plugins	520
5.5.7. Plugin Status	521
6. ACLs on HDFS	522
6.1. Configuring ACLs on HDFS	522
6.2. Using CLI Commands to Create and List ACLs	522
6.3. ACL Examples	523
6.4. ACLS on HDFS Features	527
6.5. Use Cases for ACLs on HDFS	528
7. Data Protection: HDFS Encryption	532
7.1. Ranger KMS Administration	532
7.1.1. Installing the Ranger Key Management Service	532
7.1.2. Store Master Key in a Hardware Security Module (HSM)	541
7.1.3. Enable Ranger KMS Audit	549
7.1.4. Enabling SSL for Ranger KMS	552
7.1.5. Install Multiple Ranger KMS	557
7.1.6. Using the Ranger Key Management Service	558
7.1.7. Ranger KMS Properties	564
7.1.8. Troubleshooting Ranger KMS	568
7.2. HDFS "Data at Rest" Encryption	568
7.2.1. HDFS Encryption Overview	569
7.2.2. Configuring and Starting the Ranger Key Management Service (Ranger KMS)	571
7.2.3. Configuring and Using HDFS Data at Rest Encryption	571
7.2.4. Configuring HDP Services for HDFS Encryption	580
7.2.5. Appendix: Creating an HDFS Admin User	590
8. Running DataNodes as Non-Root	592
8.1. Introduction	592
8.2. Configuring DataNode SASL	592
9. Addendum	595
9.1. ZooKeeper ACLs Best Practices	595

9.1.1. Accumulo	596
9.1.2. Ambari Solr	597
9.1.3. Atlas	598
9.1.4. HBase	598
9.1.5. HDFS/WebHDFS	599
9.1.6. Hive/HCatalog	601
9.1.7. Kafka	602
9.1.8. Oozie	603
9.1.9. Ranger	604
9.1.10. Ranger KMS/Hadoop KMS	605
9.1.11. Slider	606
9.1.12. Storm	606
9.1.13. WebHCat	607
9.1.14. YARN	607
9.1.15. YARN Registry	607
9.1.16. ZooKeeper	609
9.2. Ranger AD Integration	610
9.2.1. Ranger Architecture	610
9.2.2. Ranger AD Integration	614
9.2.3. Issue Ranger Group Mapping	626

List of Figures

3.1. Installing Ranger - Main Dashboard View	194
3.2. Installing Ranger - Add Service	195
3.3. Installing Ranger - Choose Service	196
3.4. Installing Ranger - Ranger Requirements	197
3.5. Installing Ranger Assign Masters	198
3.6. Service Manager	272
7.1. HDFS Encryption Components	570

List of Tables

2.1. Browser Settings for Storm UI	43
2.2. UNIX Authentication Settings	52
2.3. Active Directory Authentication Settings	54
2.4. Active Directory Custom ranger-admin-site Settings	56
2.5. LDAP Authentication Settings	57
2.6. LDAP Custom ranger-admin-site Settings	59
2.7. Active Directory Authentication Settings	60
2.8. Service Principals	80
2.9. Service Keytab File Names	81
2.10. General core-site.xml, Knox, and Hue	85
2.11. core-site.xml Master Node Settings – Knox Gateway	86
2.12. core-site.xml Master Node Settings – Hue	86
2.13. hdfs-site.xml File Property Settings	87
2.14. yarn-site.xml Property Settings	92
2.15. mapred-site.xml Property Settings	94
2.16. hbase-site.xml Property Settings for HBase Server and Phoenix Query Server	95
2.17. hive-site.xml Property Settings	98
2.18. oozie-site.xml Property Settings	99
2.19. webhcat-site.xml Property Settings	99
2.20. Supported Component APIs: Proxy	114
2.21. Supported Component UIs: Proxy	114
2.22. Apache Service Gateway Directories	115
2.23. Cluster Topology Provider and Service Roles	119
2.24. gateway-site.xml Configuration Elements	148
2.25. Identity Assertion Providers	149
2.26. LDAP Authentication and Authorization Arguments	171
2.27. Supported Component UIs: SSO	172
3.1. Ranger DB Host	199
3.2. Driver Class Name	200
3.3. Ranger DB Username Settings	200
3.4. JDBC Connect String	201
3.5. DBA Credential Settings	201
3.6. UNIX User Sync Properties	210
3.7. LDAP/AD Common Configs	212
3.8. LDAP/AD User Configs	213
3.9. LDAP/AD Group Configs	215
3.10. Atlas Tag Source Properties	219
3.11. AtlasREST Source Properties	220
3.12. File Tag Source Properties	220
3.13. UNIX Authentication Settings	221
3.14. LDAP Authentication Settings	222
3.15. AD Settings	226
3.16. LDAP Advanced ranger-ugsync-site Settings	231
3.17. AD Advanced ranger-ugsync-site Settings	231
3.18. Advanced ranger-ugsync-site Settings for LDAP and AD	231
3.19. HDFS Plugin Properties	270
3.20. Hive Plugin Properties	271

3.21. HBase Plugin Properties	271
3.22. Knox Plugin Properties	272
3.23. Knox Configuration Properties	273
3.24. Service Details	286
3.25. Config Properties	287
3.26. Service Details	288
3.27. Config Properties	289
3.28. Service Details	290
3.29. Config Properties	290
3.30. Service Details	293
3.31. Config Properties	293
3.32. Service Details	295
3.33. Config Properties	295
3.34. Service Details	296
3.35. Config Properties	296
3.36. Service Details	298
3.37. Config Properties	298
3.38. Service Details	299
3.39. Config Properties	299
3.40. Service Details	301
3.41. Config Properties	301
3.42. Policy Details	303
3.43. Allow Conditions	304
3.44. Policy Details	306
3.45. Allow Conditions	306
3.46. Policy Details	308
3.47. Allow Conditions	309
3.48. Policy Details	312
3.49. Allow Conditions	312
3.50. Policy Details	314
3.51. Allow Conditions	314
3.52. Policy Details	316
3.53. Allow Conditions	316
3.54. Policy Details	318
3.55. Allow Conditions	318
3.56. Storm User and Group Permissions	319
3.57. Policy Details	321
3.58. Allow Conditions	321
3.59. Policy Details	323
3.60. Allow Conditions	323
3.61. Export Policy Options	327
3.62. Policy Details	337
3.63. Row Filter Conditions	337
3.64. Policy Details	341
3.65. Mask Conditions	341
3.66. Policy Details	346
3.67. Mask Conditions	346
3.68. Policy Details	353
3.69. Allow, Exclude from Allow, Deny, and Exclude from Deny Conditions	354
3.70. Policy Details	358
3.71. Allow Conditions	359

3.72. Deny Conditions	359
3.73. Exclude from Allow Conditions	359
3.74. Export Policy Options	363
4.1. Components that Support SSL	425
4.2. Configure SSL Data Protection for HDP Components	434
4.3. Configuration Properties in ssl-server.xml	436
4.4. Atlas Advanced application-properties	452
4.5. Atlas Custom application-properties	452
5.1. Solr install.properties Values for setup.sh script	502
5.2. Solr install.properties Values	502
5.3. Solr install.properties Values	505
5.4. JDBC Audit String	511
5.5. Search Criteria	517
5.6. Search Criteria	519
5.7. Search Criteria	519
5.8. Agents Search Criteria	520
5.9. Plugin Status Search Criteria	521
6.1. ACL Options	522
6.2. getfacl Options	523
7.1. Properties in Advanced dbks-site Menu (dbks-site.xml)	564
7.2. Properties in Advanced kms-env	564
7.3. Properties in Advanced kms-properties (install.properties)	565
7.4. Properties in Advanced kms-site (kms-site.xml)	565
7.5. Properties in Advanced ranger-kms-audit (ranger-kms-audit.xml)	567
7.6. Properties in Advanced ranger-kms-policymgr-ssl	568
7.7. Properties in Advanced ranger-kms-security	568
7.8. Troubleshooting Suggestions	568

List of Examples

2.1. Example Search Filter to Find the Client Bind DN	132
2.2. GroupMappingServiceProvider Example	155
2.3. knoxsso.xml with Shiro provider	173
2.4. knoxsso.xml with Okta	175
2.5. Example Knox SSO for Ambari	177

1. HDP Security Overview

Security is essential for organizations that store and process sensitive data in the Hadoop ecosystem. Many organizations must adhere to strict corporate security policies.

Hadoop is a distributed framework used for data storage and large-scale processing on clusters using commodity servers. Adding security to Hadoop is challenging because not all of the interactions follow the classic client-server pattern.

- In Hadoop, the file system is partitioned and distributed, requiring authorization checks at multiple points.
- A submitted job is executed at a later time on nodes different than the node on which the client authenticated and submitted the job.
- Secondary services such as a workflow system access Hadoop on behalf of users.
- A Hadoop cluster scales to thousands of servers and tens of thousands of concurrent tasks.

A Hadoop-powered "Data Lake" can provide a robust foundation for a new generation of Big Data analytics and insight, but can also increase the number of access points to an organization's data. As diverse types of enterprise data are pulled together into a central repository, the inherent security risks can increase.

Hortonworks understands the importance of security and governance for every business. To ensure effective protection for its customers, Hortonworks uses a holistic approach based on five core security features:

- Administration
- Authentication and perimeter security
- Authorization
- Audit
- Data protection

This chapter provides an overview of the security features implemented in the Hortonworks Data Platform (HDP). Subsequent chapters in this guide provide more details on each of these security features.

1.1. What's New in This Release

New features and changes for Apache Ranger and Apache Knox have been introduced in Hortonworks Data Platform, version 2.6.x, along with documentation updates. New features are described in the following sections.

- **Hortonworks Data Platform 2.6.3**
 - **Authentication**
 - Apache Atlas Admin UI Support through Knox: [Setting up Hadoop Service URLs](#), [Example Service Definitions](#), and [Supported Hadoop Services](#).

You can now configure the Atlas Admin UI to go through the Knox Gateway (proxy) by creating a service definition with:

```
<service>
  <role>ATLAS</role>
  <url>http://atlas-host:8443</url>
</service>
```

- Apache Zeppelin UI Support through Knox: [Setting up Hadoop Service URLs, Example Service Definitions, and Supported Hadoop Services](#).

You can now configure the Zeppelin UI to go through the Knox Gateway (proxy) by creating a service definition with:

```
<service>
  <role>ZEPPELINUI</role>
  <url>http://zeppelin-host:9995</url>
</service>

<service>
  <role>ZEPPELINWS</role>
  <url>http://zeppelin-host:9995/ws</url>
</service>
```

- [Setting up SSOCookieProvider Federation Provider](#)

The SSOCookieProvider enables the federation of the authentication event that occurred through KnoxSSO. KnoxSSO is a typical service provider-initiated webSSO mechanism that sets a cookie to be presented by browsers to participating applications and cryptographically verified.

Knox Gateway needs a pluggable mechanism for consuming these cookies and federating the KnoxSSO authentication event as an asserted identity in its interaction with the Hadoop cluster for REST API invocations. This provider is useful when an application that is integrated with KnoxSSO for authentication also consumes REST APIs through the Knox Gateway.

- **Authorization**

- [Automatically Assign ADMIN/KEYADMIN Role for External Users](#)

You can use usersync to mark specific external users, or users in a specific external group, with ADMIN or KEYADMIN role within Ranger. This is useful in cases where internal users are not allowed to login to Ranger.

- [Setting up the Knox Token Service for Ranger APIs](#)

Once logged into Knox SSO, the UI service uses a cookie named `hadoop-jwt`. The Knox Token Service enables clients to acquire this same JWT token to use for accessing REST APIs. By acquiring the token and setting it as a bearer token on a request, a client is able to access REST APIs that are protected with the [JWT Federation Provider](#).

- [Using Tag Attributes and Values in Ranger Tag-based Policy Conditions](#)

Tag-based policies enable you to control access to resources across multiple Hadoop components without creating separate services and policies in each component. You can also use Ranger TagSync to synchronize the Ranger tag store with an external metadata service such as Apache Atlas.

- Support for nested LDAP/AD Group Sync: [Configuring Ranger User Sync for LDAP/AD](#)

Enables nested group memberships in Ranger so that the policies configured for parent groups are applied for all the members in the subgroups.

- Under Hive Policy>Allow Conditions>Permissions, new **Service Admin** option to provide authorization for Hive query kill API: [Create a Hive Policy](#)

The Hive Service Name is used only in conjunction with Permissions=Service Admin. It enables a user who has Service Admin permission in Ranger to run the kill query API: `kill query <queryID>`.

- **Miscellaneous**

- Bug fixes

- **Hortonworks Data Platform 2.6.2**

- **Authorization**

- [Dynamic tag-based column masking of Hive Columns using Ranger policies](#)

Where Ranger **resource-based** masking policy for Hive anonymizes data from a Hive column identified by the database, table, and column, **tag-based** masking policy anonymizes Hive column data based on tags and tag attribute values associated with Hive column (usually specified as metadata classification in Atlas).

- **Miscellaneous**

- Bug fixes

- **Hortonworks Data Platform 2.6.1**

- **Miscellaneous**

- Bug fixes

- **Hortonworks Data Platform 2.6.0**

- **Authentication**

- [New Identity Assertion Provider: HadoopGroupProvider](#)

The Hadoop Group Lookup identity assertion provider looks up the user's 'group membership' for authenticated users using Hadoop's group mapping service (GroupMappingServiceProvider).

This allows existing investments in the Hadoop to be leveraged within Knox and used within the access control policy enforcement at the perimeter.

- [Support for PAM Authentication](#)

PAM authentication is configured by adding a "ShiroProvider" authentication provider to the cluster's topology file with PAM parameters.

There are a large number of pluggable authentication modules available for authenticating access to Hadoop through the Knox Gateway. ShiroProvider, in addition to LDAP support, also includes support for PAM-based authentication for unix-based systems.

- [Added support for WebSockets](#)

WebSocket is a communication protocol that allows full duplex communication over single TCP connection. Knox provides out-of-the-box support for WebSocket protocol, but currently, only text-based messages are supported.

- **Authorization**

- Export and import [tag-based](#) and [resource-based](#) policies

Export and import policies from Ranger Admin UI from one cluster to another when launching new clusters or moving policies from test to production clusters. Export/import a specific subset of policies (such as those that pertain to specific resources or user/groups) or clone the entire repository or multiple repositories via Ranger Admin UI.

- [Incremental Usersync](#)

When enabled, Ranger Usersync saves the latest timestamp of all the objects that are synced previously and uses that timestamp to perform the next sync. Usersync uses a polling mechanism to perform incremental sync by using LDAP attributes uSNChanged (for AD) or modifytimestamp (for LDAP). Recommended for large deployments.

- Support for [{USER} variable](#) in Ranger policies

The variable `{USER}` can be used to autofill the accessing user.

- **Auditing**

- New [Plugin Status](#) page under Audits

This tab shows policies in effect for each plugin. Includes the relevant host info and when the plugin downloaded and started enforcing the policies.

- **Miscellaneous**

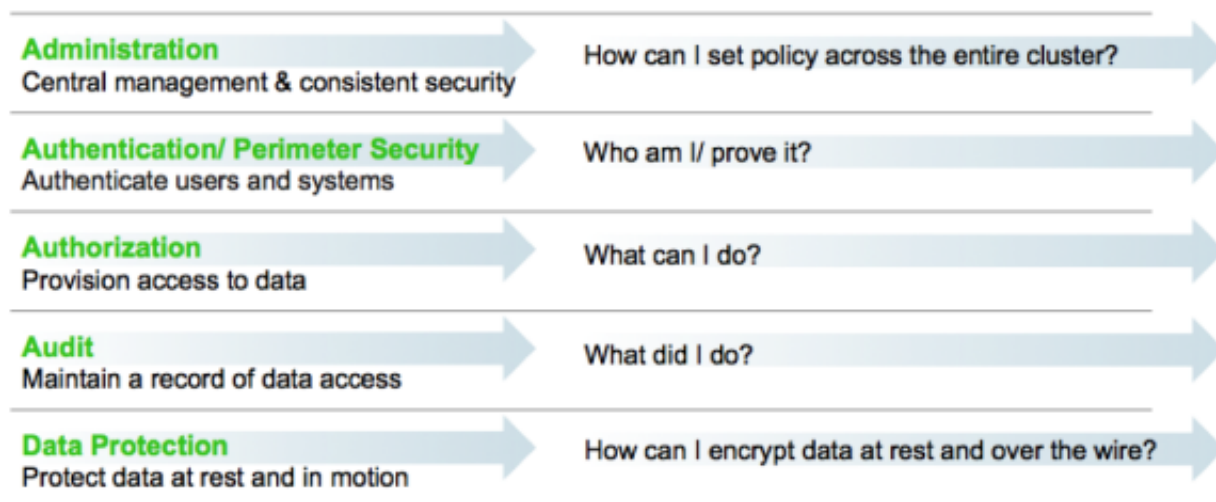
- Bug fixes

1.2. Understanding Data Lake Security

The general consensus in nearly every industry is that data is an essential new driver of competitive advantage. Hadoop plays a critical role in the modern data architecture by providing low-cost, large-scale data storage and processing. The successful Hadoop journey typically starts with data architecture optimization or new advanced analytic applications, which leads to the formation of what is known as a Data Lake. As new and existing types of data from machine sensors, server logs, clickstream data, and other sources flow into the Data Lake, it serves as a central repository based on shared Hadoop services that power deep organizational insights across a broad and diverse set of data.

The need to protect the Data Lake with comprehensive security is clear. As large and growing volumes of diverse data are channeled into the Data Lake, it will store vital and often highly sensitive business data. However, the external ecosystem of data and operational systems feeding the Data Lake is highly dynamic and can introduce new security threats on a regular basis. Users across multiple business units can access the Data Lake freely and refine, explore, and enrich its data, using methods of their own choosing, further increasing the risk of a breach. Any breach of this enterprise-wide data can result in catastrophic consequences: privacy violations, regulatory infractions, or the compromise of vital corporate intelligence. To prevent damage to the company's business, customers, finances, and reputation, a Data Lake should meet the same high standards of security as any legacy data environment.

Piecemeal protections are no more effective for a Data Lake than they would be in a traditional repository. Effective Hadoop security depends on a holistic approach that revolves around five pillars of security: administration, authentication and perimeter security, authorization, audit, and data protection.

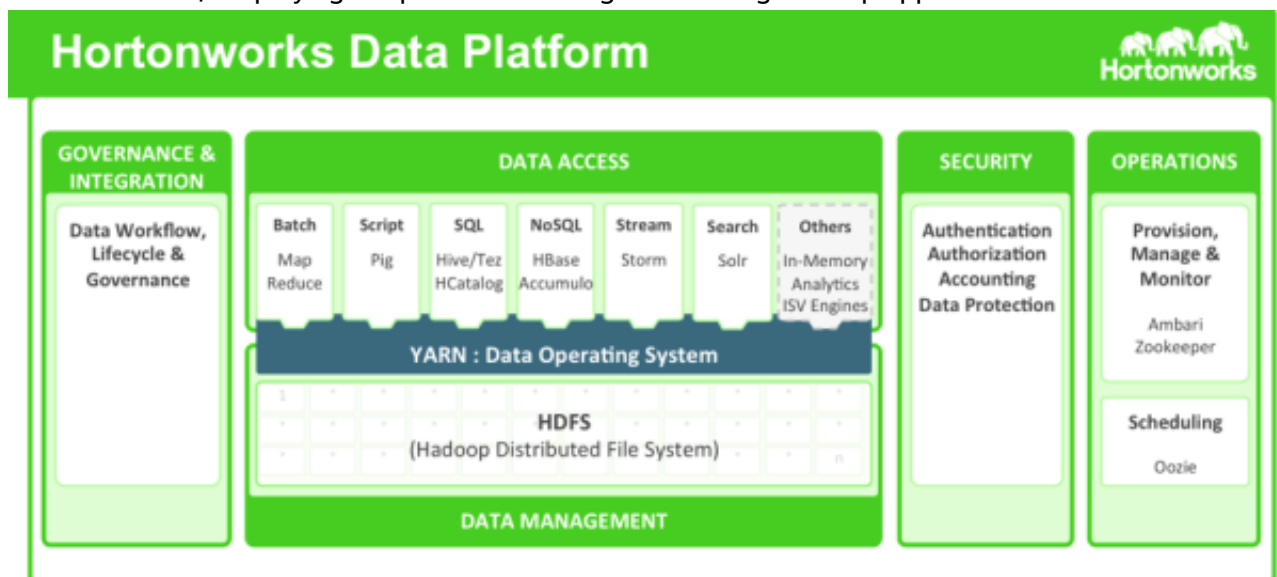


Requirements for Enterprise-Grade Security

Security administrators must address questions and provide enterprise-grade coverage across each of these areas as they design the infrastructure to secure data in Hadoop. If any of these pillars is vulnerable, it becomes a risk factor in the company's Big Data environment. A Hadoop security strategy must address all five pillars, with a consistent implementation approach to ensure effectiveness.

You cannot achieve comprehensive protection across the Hadoop stack by using an assortment of point solutions. Security must be an integral part of the platform on which your Data Lake is built. This bottom-up approach makes it possible to enforce and manage security across the stack through a central point of administration, thereby preventing gaps and inconsistencies. This approach is especially important for Hadoop implementations in which new applications or data engines are always emerging in the form of new Open Source projects — a dynamic scenario that can quickly exacerbate any vulnerability.

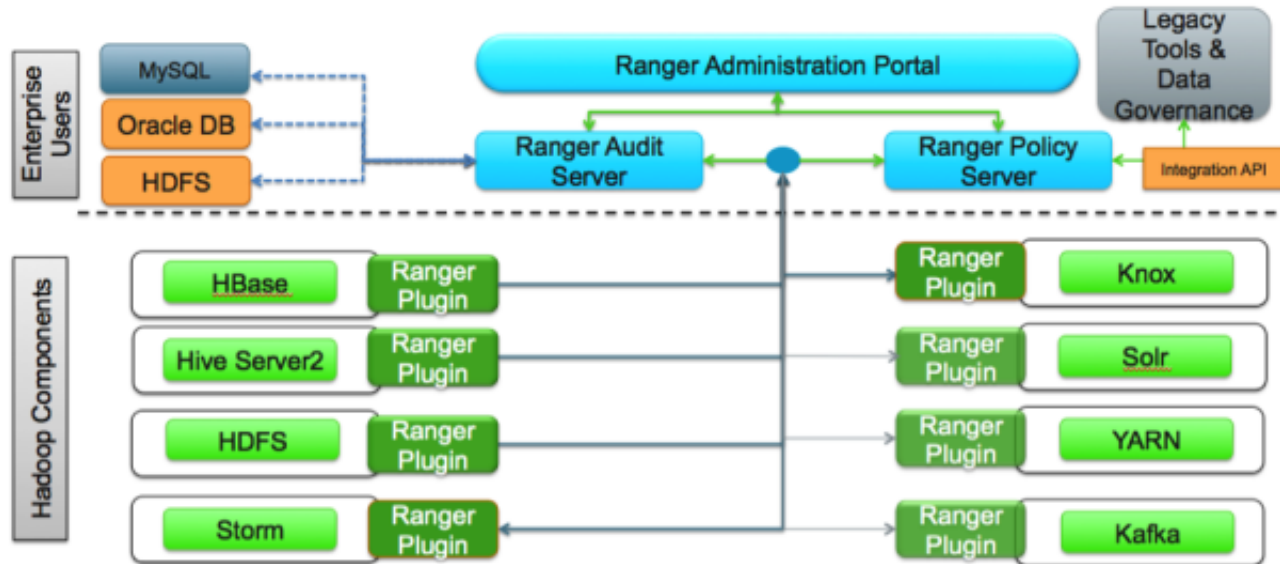
Hortonworks helps customers maintain high levels of protection for enterprise data by building centralized security administration and management into the infrastructure of the Hortonworks Data Platform. HDP provides an enterprise-ready data platform with rich capabilities spanning security, governance, and operations. HDP includes powerful data security functionality that works across component technologies and integrates with preexisting EDW, RDBMS, and MPP systems. By implementing security at the platform level, Hortonworks ensures that security is consistently administered to all of the applications across the stack, simplifying the process of adding or removing Hadoop applications.



The Hortonworks Data Platform

1.3. HDP Security Features

HDP uses Apache Ranger to provide centralized security administration and management. The Ranger Administration Portal is the central interface for security administration. You can use Ranger to create and update policies, which are then stored in a policy database. Ranger plug-ins (lightweight Java programs) are embedded within the processes of each cluster component. For example, the Ranger plug-in for Apache Hive is embedded within HiveServer2:

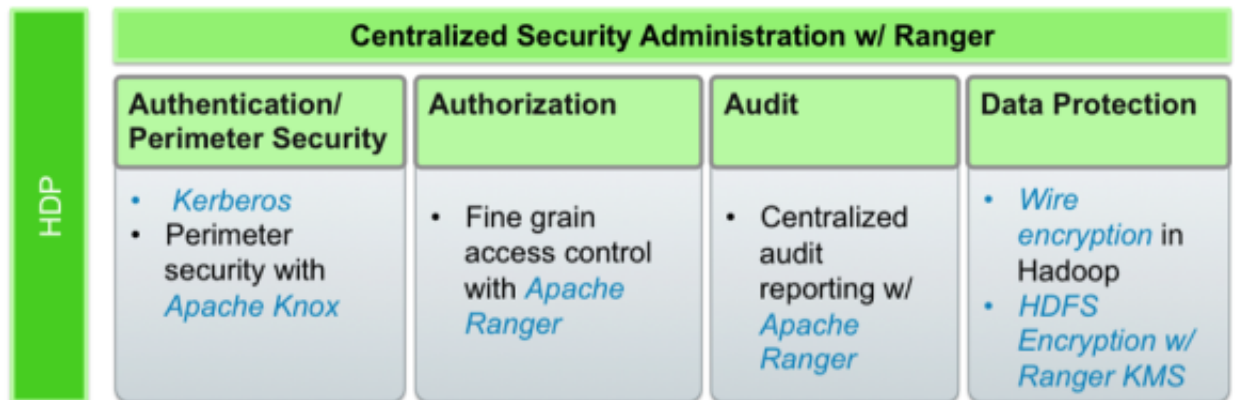


Apache Ranger Architecture

These plug-ins pull policies from a central server and store them locally in a file. When a user request comes through the component, these plug-ins intercept the request and evaluate it against the security policy. Plug-ins also collect data from the user request and follow a separate thread to send this data back to the audit server.

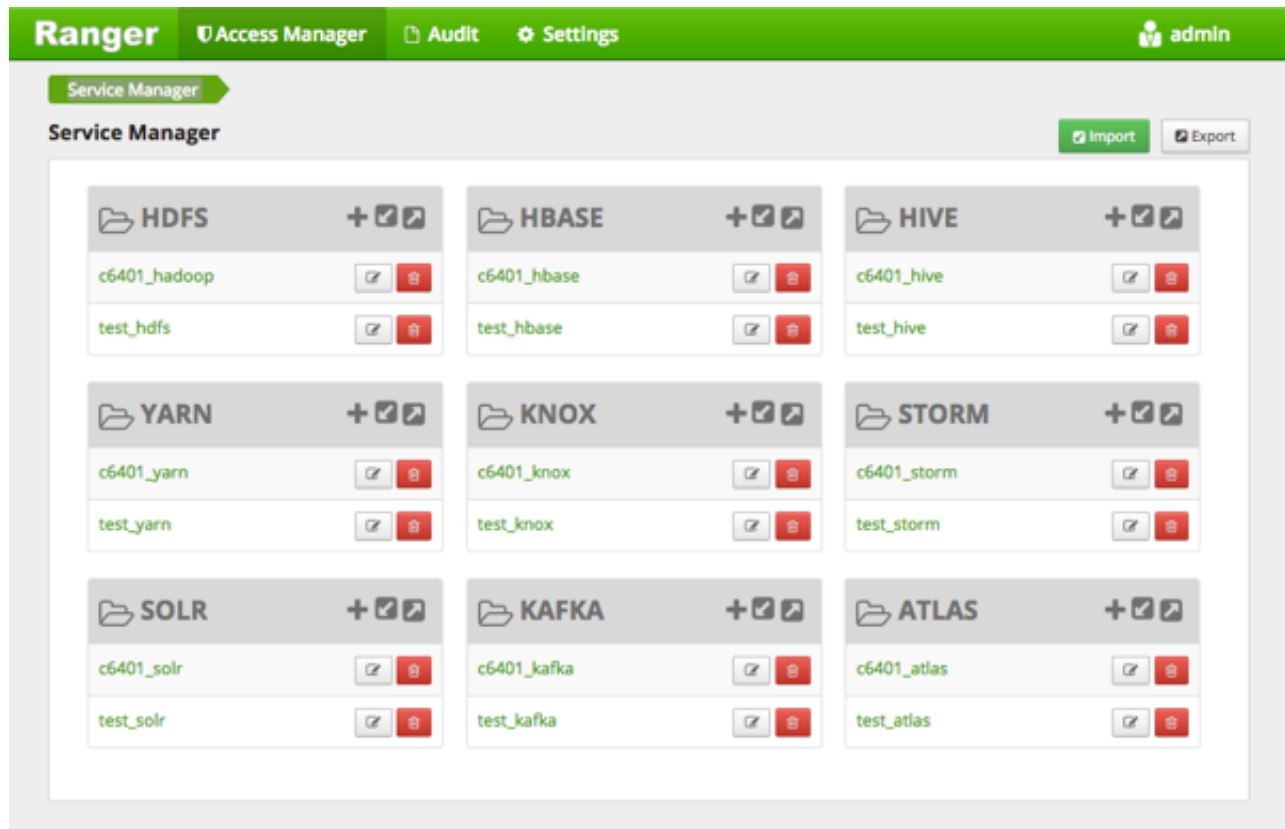
1.3.1. Administration

To deliver consistent security administration and management, Hadoop administrators require a centralized user interface they can use to define, administer and manage security policies consistently across all of the Hadoop stack components:



Ranger Centralized Security Administration

The Apache Ranger administration console provides a central point of administration for the other four pillars of Hadoop security.

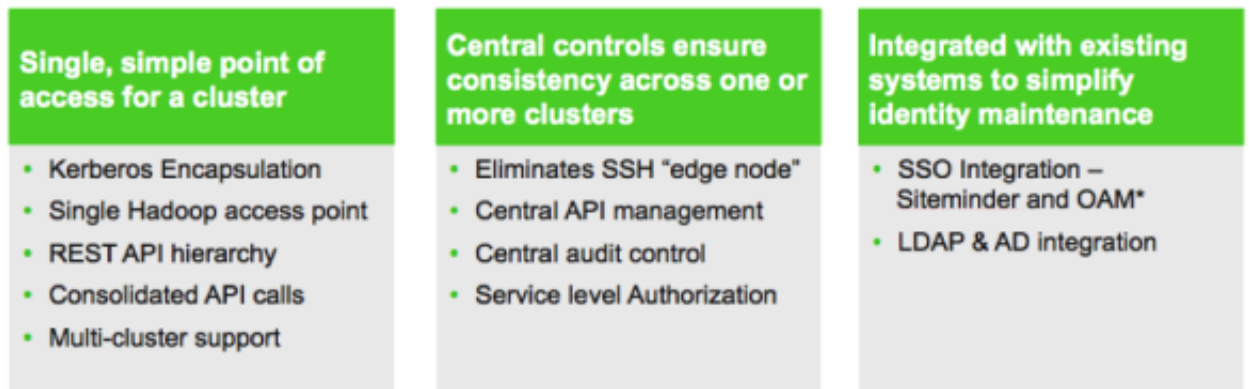


Ranger Administration Console

1.3.2. Authentication and Perimeter Security

Establishing user identity with strong authentication is the basis for secure access in Hadoop. Users need to reliably identify themselves and then have that identity propagated throughout the Hadoop cluster to access cluster resources. Hortonworks uses Kerberos for authentication. Kerberos is an industry standard used to authenticate users and resources within a Hadoop cluster. HDP also includes Ambari, which simplifies Kerberos setup, configuration, and maintenance.

Apache Knox Gateway is used to help ensure perimeter security for Hortonworks customers. With Knox, enterprises can confidently extend the Hadoop REST API to new users without Kerberos complexities, while also maintaining compliance with enterprise security policies. Knox provides a central gateway for Hadoop REST APIs that have varying degrees of authorization, authentication, SSL, and SSO capabilities to enable a single access point for Hadoop.



Apache Knox Features

1.3.3. Authorization

Ranger manages access control through a user interface that ensures consistent policy administration across Hadoop data access components. Security administrators can define security policies at the database, table, column, and file levels, and can administer permissions for specific LDAP-based groups or individual users. Rules based on dynamic conditions such as time or geolocation can also be added to an existing policy rule. The Ranger authorization model is pluggable and can be easily extended to any data source using a service-based definition.

Administrators can use Ranger to define a centralized security policy for the following Hadoop components:

- HDFS
- YARN
- Hive
- HBase
- Storm
- Knox
- Solr
- Kafka

Ranger works with standard authorization APIs in each Hadoop component and can enforce centrally administered policies for any method used to access the Data Lake.

Policy Details :

Policy Name * enabled

Hive Database * include

table include

Hive Column * include

Description

Audit Logging YES

User and Group Permissions :

Permissions

Select Group +

Select User +

Delegate Admin

[Add Permissions](#) +

add/edit permissions

- select
- update
- Create
- Drop
- Alter
- Index
- Lock
- All
- Select/Deselect All

Ranger Security Policy Definitions

Ranger provides administrators with the deep visibility into the security administration process that is required for auditing. The combination of a rich user interface and deep audit visibility makes Ranger highly intuitive to use, enhancing productivity for security administrators.

Ranger Access Manager Audit Settings admin

Service Manager > sandbox_hive Policies

List of Policies : sandbox_hive

Search for your policy... Add New Policy

Policy ID	Policy Name	Status	Audit Logging	Groups	Users	Action
3	sandbox_hive-1-20150529142947	Enabled	Enabled	--	xapolicymgr	
4	Hive Global Tables Allow	Disabled	Enabled	public	--	
5	Hive Global UDF Allow	Disabled	Enabled	public	--	
18	Call_Details_Table	Enabled	Enabled	developer	--	
19	Customer_Details_Table	Disabled	Enabled	Marketing	--	
20	Hive Demo Table Loader	Enabled	Enabled	--	hive	
21	Hive Demo UDF Loader	Enabled	Enabled	--	hive	
29	admin policy	Enabled	Enabled	--	admin	

Ranger Security Policy Overview

1.3.4. Audit

As customers deploy Hadoop into corporate data and processing environments, metadata and data governance must be vital parts of any enterprise-ready data lake. For this reason, Hortonworks [established the Data Governance Initiative \(DGI\) with Aetna, Merck, Target, and SAS](#) to introduce a common approach to Hadoop data governance into the open source community. This initiative has since evolved into a new open source project named Apache Atlas. Apache Atlas is a set of core governance services that enables enterprises to meet their compliance requirements within Hadoop, while also enabling integration with the complete enterprise data ecosystem. These services include:

- Dataset search and lineage operations
- Metadata-driven data access control
- Indexed and searchable centralized auditing
- Data lifecycle management from ingestion to disposition
- Metadata interchange with other tools

Ranger also provides a centralized framework for collecting access audit history and reporting this data, including filtering on various parameters. HDP enhances audit information that is captured within different components within Hadoop and provides insights through this centralized reporting capability.

1.3.5. Data Protection

The data protection feature makes data unreadable both in transit over the network and at rest on a disk. HDP satisfies security and compliance requirements by using both transparent data encryption (TDE) to encrypt data for HDFS files, along with a Ranger-embedded open source Hadoop key management store (KMS). Ranger enables security administrators to manage keys and authorization policies for KMS. Hortonworks is also working extensively with its encryption partners to integrate HDFS encryption with enterprise-grade key management frameworks.

Encryption in HDFS, combined with KMS access policies maintained by Ranger, prevents rogue Linux or Hadoop administrators from accessing data, and supports segregation of duties for both data access and encryption.

2. Authentication

2.1. Enabling Kerberos Authentication Using Ambari

This chapter describes how to configure Kerberos for strong authentication for Hadoop users and hosts in an Ambari-managed cluster.

- [Kerberos Overview \[12\]](#)
- [Kerberos Principals \[13\]](#)
- [Installing and Configuring the KDC \[14\]](#)
- [Enabling Kerberos Security \[19\]](#)

2.1.1. Kerberos Overview

Strongly authenticating and establishing a user's identity is the basis for secure access in Hadoop. Users need to be able to reliably "identify" themselves and then have that identity propagated throughout the Hadoop cluster. Once this is done, those users can access resources (such as files or directories) or interact with the cluster (like running MapReduce jobs). Besides users, Hadoop cluster resources themselves (such as Hosts and Services) need to authenticate with each other to avoid potential malicious systems or daemon's "posing as" trusted components of the cluster to gain access to data.

Hadoop uses Kerberos as the basis for strong authentication and identity propagation for both user and services. Kerberos is a third party authentication mechanism, in which users and services rely on a third party - the Kerberos server - to authenticate each to the other. The Kerberos server itself is known as the **Key Distribution Center**, or **KDC**. At a high level, it has three parts:

- A database of the users and services (known as **principals**) that it knows about and their respective Kerberos passwords
- An **Authentication Server (AS)** which performs the initial authentication and issues a **Ticket Granting Ticket (TGT)**
- A **Ticket Granting Server (TGS)** that issues subsequent service tickets based on the initial **TGT**

A **user principal** requests authentication from the AS. The AS returns a TGT that is encrypted using the user principal's Kerberos password, which is known only to the user principal and the AS. The user principal decrypts the TGT locally using its Kerberos password, and from that point forward, until the ticket expires, the user principal can use the TGT to get service tickets from the TGS. Service tickets are what allow a principal to access various services.

Because cluster resources (hosts or services) cannot provide a password each time to decrypt the TGT, they use a special file, called a **keytab**, which contains the resource

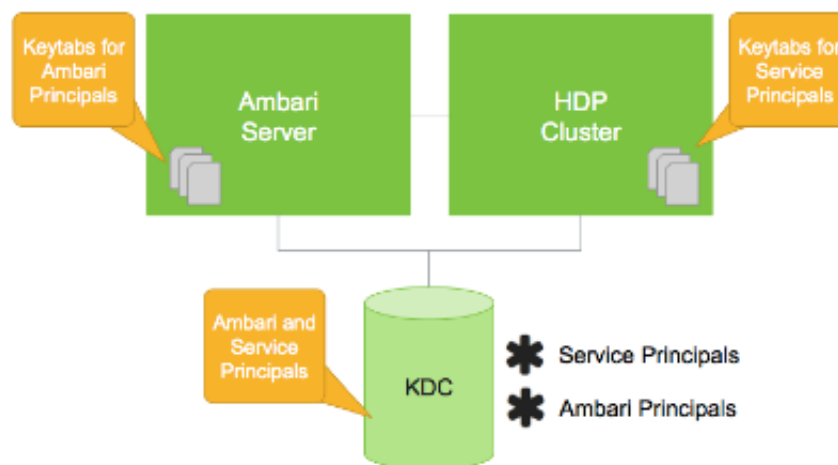
principal's authentication credentials. The set of hosts, users, and services over which the Kerberos server has control is called a **realm**.

Terminology

Term	Description
Key Distribution Center, or KDC	The trusted source for authentication in a Kerberos-enabled environment.
Kerberos KDC Server	The machine, or server, that serves as the Key Distribution Center (KDC).
Kerberos Client	Any machine in the cluster that authenticates against the KDC.
Principal	The unique name of a user or service that authenticates against the KDC.
Keytab	A file that includes one or more principals and their keys.
Realm	The Kerberos network that includes a KDC and a number of Clients.
KDC Admin Account	An administrative account used by Ambari to create principals and generate keytabs in the KDC.

2.1.2. Kerberos Principals

Each service and sub-service in Hadoop must have its own principal. A **principal** name in a given realm consists of a primary name and an instance name, in this case the instance name is the FQDN of the host that runs that service. As services do not log in with a password to acquire their tickets, their principal's authentication credentials are stored in a **keytab** file, which is extracted from the Kerberos database and stored locally in a secured directory with the service principal on the service component host.



Principal and Keytab Naming Conventions

Asset	Convention	Example
Principals	<code>\$service_component_name/\$FQDN@EXAMPLE.COM</code>	<code>nn/c6401.ambari.apache.org@EXAMPLE.COM</code>
Keytabs	<code>\$service_component_abbreviation.service.keytab</code>	<code>keytabsecurity/keytabs/nn.service.keytab</code>

Notice in the preceding example the primary name for each service principal. These primary names, such as `nn` or `hive` for example, represent the NameNode or Hive service, respectively. Each primary name has appended to it the instance name, the FQDN of the

host on which it runs. This convention provides a unique principal name for services that run on multiple hosts, like DataNodes and NodeManagers. Adding the host name serves to distinguish, for example, a request from DataNode A from a request from DataNode B. This is important for the following reasons:

- Compromised Kerberos credentials for one DataNode do not automatically lead to compromised Kerberos credentials for all DataNodes.
- If multiple DataNodes have exactly the same principal and are simultaneously connecting to the NameNode, and if the Kerberos authenticator being sent happens to have same timestamps, then the authentication is rejected as a replay request.

In addition to the Hadoop **Service Principals**, Ambari itself also requires a set of **Ambari Principals** to perform service “smoke” checks, perform alert health checks and to retrieve metrics from cluster components. Keytab files for the **Ambari Principals** reside on each cluster host, just as keytab files for the service principals.

Ambari Principals	Description
Smoke and “Headless” Service users	Used by Ambari to perform service “smoke” checks and run alert health checks.
Ambari Server user	When a cluster is enabled for Kerberos, the component REST endpoints (such as the YARN ATS component) require SPNEGO authentication. Ambari Server needs access to these APIs and requires a Kerberos principal in order to authenticate via SPNEGO against these APIs.

2.1.3. Installing and Configuring the KDC

Ambari is able to configure Kerberos in the cluster to work with an existing MIT KDC, or existing Active Directory installation. This section describes the steps necessary to prepare for this integration.



Note

If you do not have an existing KDC (MIT or Active Directory), [install a new MIT KDC](#). Please be aware that installing a KDC on a cluster host **after** installing the Kerberos client may overwrite the `krb5.conf` file generated by Ambari.

You can choose to have Ambari connect to the KDC and automatically create the necessary Service and Ambari principals, generate and distribute the keytabs (“Automated Kerberos Setup”). Ambari also provides an advanced option to manually configure Kerberos. If you choose this option, you must create the principals, generate and distribute the keytabs. Ambari will not do this automatically (“Manual Kerberos Setup”).

Supported Key Distribution Center (KDC) Versions

- Microsoft Active Directory 2008 and above
- MIT Kerberos v5
- [Use an Existing MIT KDC \[15\]](#)
- [Use an Existing Active Directory \[15\]](#)
- [Use Manual Kerberos Setup \[15\]](#)

2.1.3.1. Use an Existing MIT KDC

To use an existing MIT Kerberos v5 KDC for the cluster, you must prepare the following:

- Ambari Server and cluster hosts have network access to both the KDC and KDC admin hosts.
- KDC administrative credentials are on-hand.

Proceed with [Enabling Kerberos Security \[19\]](#).

2.1.3.2. Use an Existing Active Directory

To use an existing domain from Microsoft Active Directory 2008 and later for the cluster with Automated Kerberos Setup, you must prepare the following:

- Ambari Server and cluster hosts have network access to, and be able to resolve the DNS names of, the Domain Controllers.
- Active Directory secure LDAP (LDAPS) connectivity has been configured.
- Active Directory User container for service principals has been created and is on-hand. For example, "OU=Hadoop,OU=People,dc=apache,dc=org"
- Active Directory administrative credentials with delegated control of "Create, delete, and manage user accounts" on the previously mentioned User container are on-hand.

Proceed with [Enabling Kerberos Security \[19\]](#).



Note

You will be prompted to enter the KDC Admin Account credentials during the Kerberos setup so that Ambari can contact the KDC and perform the necessary principal and keytab generation. By default, Ambari will not retain the KDC credentials unless you have configured Ambari for encrypted passwords. Refer to [Managing Admin Credentials](#) for more information.



Note

If Centrify is installed and being used on any of the servers in the cluster, it is critical that you refer to Centrify's integration guide before attempting to enable Kerberos Security on your cluster. The documentation can be found in the Centrify Server Suite documentation library. A direct link to the Hortonworks-specific configuration guide can be found [here](#).

2.1.3.3. Use Manual Kerberos Setup

To perform Manual Kerberos Setup, you must prepare the following:

- Cluster hosts have network access to the KDC.
- Kerberos client utilities (such as kinit) have been installed on every cluster host.
- The Java Cryptography Extensions (JCE) have been setup on the Ambari Server host and all hosts in the cluster.

- The Service and Ambari Principals will be manually created in the KDC before completing this wizard.
- The keytabs for the Service and Ambari Principals will be manually created and distributed to cluster hosts before completing this wizard.

Proceed with [Enabling Kerberos Security \[19\]](#).

2.1.3.4. (Optional) Install a new MIT KDC

The following gives a very high level description of the KDC installation process. To get more information see specific Operating Systems documentation, such as [RHEL documentation](#), [CentOS documentation](#), or [SLES documentation](#).



Note

Because Kerberos is a time-sensitive protocol, all hosts in the realm must be time-synchronized, for example, by using the Network Time Protocol (NTP). If the local system time of a client differs from that of the KDC by as little as 5 minutes (the default), the client will not be able to authenticate.

Install the KDC Server

1. Install a new version of the KDC server:

RHEL/CentOS/Oracle Linux

```
yum install krb5-server krb5-libs krb5-workstation
```

SLES

```
zypper install krb5 krb5-server krb5-client
```

Ubuntu/Debian

```
apt-get install krb5-kdc krb5-admin-server
```

2. Using a text editor, open the KDC server configuration file, located by default here:

```
vi /etc/krb5.conf
```

3. Change the [realms] section of this file by replacing the default "kerberos.example.com" setting for the kdc and admin_server properties with the Fully Qualified Domain Name of the KDC server host. In the following example, "kerberos.example.com" has been replaced with "my.kdc.server".

```
[realms]
EXAMPLE.COM = {
    kdc = my.kdc.server
    admin_server = my.kdc.server
}
```



Note

For Ubuntu/Debian, the setup of the default realm for the KDC and KDC Admin hostnames is performed during the KDC server install. You can re-run

setup using `dpkg-reconfigure krb5-kdc`. Therefore, Steps 2 and 3 above are not needed for Ubuntu/Debian.

Create the Kerberos Database

- Use the utility `kdb5_util` to create the Kerberos database.

RHEL/CentOS/Oracle Linux

```
kdb5_util create -s
```

SLES

```
kdb5_util create -s
```

Ubuntu/Debian

```
krb5_newrealm
```

Start the KDC

- Start the KDC server and the KDC admin server.

RHEL/CentOS/Oracle Linux 6

```
/etc/rc.d/init.d/krb5kdc start
```

```
/etc/rc.d/init.d/kadmin start
```

RHEL/CentOS/Oracle Linux 7

```
systemctl start krb5kdc
```

```
systemctl start kadmin
```

SLES

```
rckrb5kdc start
```

```
rckadmind start
```

Ubuntu/Debian

```
service krb5-kdc restart
```

```
service krb5-admin-server restart
```



Important

When installing and managing your own MIT KDC, it is **very important** to **set up the KDC server to auto-start on boot**. For example:

RHEL/CentOS/Oracle Linux 6

```
chkconfig krb5kdc on
```

```
chkconfig kadmin on
```

RHEL/CentOS/Oracle Linux 7

```
systemctl enable krb5kdc
```

```
systemctl enable kadmin
```

SLES

```
chkconfig rckrb5kdc on
```

```
chkconfig rckadmind on
```

Ubuntu/Debian

```
update-rc.d krb5-kdc defaults
```

```
update-rc.d krb5-admin-server defaults
```

Create a Kerberos Admin

Kerberos principals can be created either on the KDC machine itself or through the network, using an "admin" principal. The following instructions assume you are using the KDC machine and using the `kadmin.local` command line administration utility. Using `kadmin.local` on the KDC machine allows you to create principals without needing to create a separate "admin" principal before you start.



Note

You will need to provide these admin account credentials to Ambari when enabling Kerberos. This allows Ambari to connect to the KDC, create the cluster principals and generate the keytabs.

1. Create a KDC admin by creating an admin principal.

```
kadmin.local -q "addprinc admin/admin"
```

2. Confirm that this admin principal has permissions in the KDC ACL. Using a text editor, open the KDC ACL file:

RHEL/CentOS/Oracle Linux

```
vi /var/kerberos/krb5kdc/kadm5.acl
```

SLES

```
vi /var/lib/kerberos/krb5kdc/kadm5.acl
```

Ubuntu/Debian

```
vi /etc/krb5kdc/kadm5.acl
```

3. Ensure that the KDC ACL file includes an entry so to allow the admin principal to administer the KDC for your specific realm. When using a realm that is different than

EXAMPLE.COM, **be sure there is an entry for the realm you are using**. If not present, principal creation will fail. For example, for an admin/admin@HADOOP.COM principal, you should have an entry:

```
*/admin@HADOOP.COM *
```

4. After editing and saving the kadm5.acl file, you must restart the kadmin process.

RHEL/CentOS/Oracle Linux 6

```
/etc/rc.d/init.d/kadmin restart
```

RHEL/CentOS/Oracle Linux 7

```
systemctl restart kadmin
```

SLES

```
rckadmind restart
```

Ubuntu/Debian

```
service krb5-admin-server restart
```

2.1.4. Enabling Kerberos Security

Whether you choose automated or manual Kerberos setup, Ambari provides a wizard to help with enabling Kerberos in the cluster. This section provides information on preparing Ambari before running the wizard, and the steps to run the wizard.

- [Installing the JCE \[20\]](#)
- [Creating Mappings Between Principals and UNIX Usernames \[83\]](#)
- [Running the Kerberos Security Wizard \[21\]](#)



Important

Prerequisites for enabling Kerberos are having the JCE installed on all hosts on the cluster (including the Ambari Server) and having the Ambari Server host as part of the cluster. This means the Ambari Server host should be running an Ambari Agent.

You should also [create mappings between principals and UNIX user names](#). Creating mappings can help resolve access issues related to case mismatches between principal and local user names.



Note

Ambari Metrics will not be secured with Kerberos unless it is configured for distributed metrics storage. By default, it uses embedded metrics storage and will not be secured as part of the Kerberos Wizard. If you wish to have Ambari Metrics secured with Kerberos, please see [this topic](#) to enable distributed metrics storage prior to running the Kerberos Wizard.



Note

If Centrify is installed and being used on any of the servers in the cluster, it is critical that you refer to Centrify's integration guide before attempting to enable Kerberos Security on your cluster. The documentation can be found in the Centrify Server Suite documentation library. A direct link to the Hortonworks-specific configuration guide can be found [here](#).

2.1.4.1. Installing the JCE

Before enabling Kerberos in the cluster, you must deploy the Java Cryptography Extension (JCE) security policy files on the Ambari Server and on all hosts in the cluster.



Important

If you are using Oracle JDK, **you must distribute and install the JCE on all hosts** in the cluster, including the Ambari Server. **Be sure to restart Ambari Server after installing the JCE.** If you are using OpenJDK, some distributions of the OpenJDK (such as RHEL/CentOS and Ubuntu) come with unlimited strength JCE automatically and therefore, installation of JCE is not required.

2.1.4.1.1. Install the JCE

1. On the Ambari Server, obtain the JCE policy file appropriate for the JDK version in your cluster.

- For Oracle JDK 1.8:

<http://www.oracle.com/technetwork/java/javase/downloads/jce8-download-2133166.html>

- For Oracle JDK 1.7:

<http://www.oracle.com/technetwork/java/javase/downloads/jce-7-download-432124.html>

For example:

```
wget --no-check-certificate --no-cookies --header "Cookie: oraclelicense=accept-securebackup-cookie" "http://download.oracle.com/otn-pub/java/jce/8/jce_policy-8.zip"
```

2. Save the policy file archive in a temporary location.
3. On Ambari Server and on each host in the cluster, add the unlimited security policy JCE jars to `$JAVA_HOME/jre/lib/security/`.

For example, run the following to extract the policy jars into the JDK installed on your host:

```
unzip -o -j -q jce_policy-8.zip -d /usr/jdk64/jdk1.8.0_40/jre/lib/security/
```

4. Restart Ambari Server: `sudo ambari-server restart`

5. Proceed to [Running the Kerberos Security Wizard \[21\]](#).

2.1.4.2. Running the Kerberos Security Wizard

Ambari provides three options for enabling Kerberos:

- Existing MIT KDC
- Existing Active Directory
- Manage Kerberos principals and keytabs manually

When choosing **Existing MIT KDC** or **Existing Active Directory**, the Kerberos Wizard prompts for information related to the KDC, the KDC Admin Account and the Service and Ambari principals. Once provided, Ambari will automatically create principals, generate keytabs and distribute keytabs to the hosts in the cluster. The services will be configured for Kerberos and the service components are restarted to authenticate against the KDC. This is the **Automated Setup** option. See [Launching the Kerberos Wizard \(Automated Setup\)](#) for more details.

When choosing **Manage Kerberos principals and keytabs manually**, you must create the principals, generate and distribute the keytabs; including you performing the [Ambari Server Kerberos setup](#). Ambari will not do this automatically. This is the **Manual Setup** option. See [Launching the Kerberos Wizard \(Manual Setup\)](#) for more details.

2.1.4.2.1. Launching the Kerberos Wizard (Automated Setup)

1. Be sure you have [Installed and Configured your KDC](#) and have [prepared the JCE](#) on each host in the cluster.
2. Log in to Ambari Web and Browse to `Admin > Kerberos`.
3. Click "Enable Kerberos" to launch the wizard.
4. Select the type of KDC you are using and confirm you have met the prerequisites.
5. Provide information about the KDC and admin account.
 - a. In the **KDC** section, enter the following information:
 - In the **KDC Host** field, the IP address or FQDN for the KDC host. Optionally a port number may be included.
 - In the **Realm name** field, the default realm to use when creating service principals.
 - (Optional) In the **Domains** field, provide a list of patterns to use to map hosts in the cluster to the appropriate realm. For example, if your hosts have a common domain in their FQDN such as `host1.hortonworks.local` and `host2.hortonworks.local`, you would set this to:

```
.hortonworks.local, hortonworks.local
```

- b. In the **Kadmin** section, enter the following information:

- In the **Kadmin Host** field, the IP address or FQDN for the KDC administrative host. Optionally a port number may be included.
- The **Admin principal** and **password** that will be used to create principals and keytabs.
- (Optional) If you have configured Ambari for encrypted passwords, the **Save Admin Credentials** option will be enabled. With this option, you can have Ambari store the KDC Admin credentials to use when making cluster changes. Refer to [Managing Admin Credentials](#) for more information on this option.

6. Modify any advanced Kerberos settings based on your environment.

- a. (Optional) To manage your Kerberos client `krb5.conf` manually (and not have Ambari manage the `krb5.conf`), expand the **Advanced krb5-conf** section and uncheck the "Manage" option. **You must have the `krb5.conf` configured on each host.**



Note

When manually managing the `krb5.conf` it is recommended to ensure that DNS is not used for looking up KDC, and REALM entries. Relying on DNS can cause negative performance, and functional impact. To ensure that DNS is not used, ensure the following entries are set in the `libdefaults` section of your configuration.

```
[libdefaults]
dns_lookup_kdc = false
dns_lookup_realm = false
```

- b. (Optional) to configure any additional KDC's to be used for this environment, add an entry for each additional KDC to the `realms` section of the **Advanced krb5-conf's** `krb5-conf` template.

```
kdc = {{kdc_host}}
kdc = otherkdc.example.com
```

- c. (Optional) To not have Ambari install the Kerberos client libraries on all hosts, expand the **Advanced kerberos-env** section and uncheck the "Install OS-specific Kerberos client package(s)" option. **You must have the Kerberos client utilities installed on each host.**
- d. (Optional) If your Kerberos client libraries are in non-standard path locations, expand the **Advanced kerberos-env** section and adjust the "Executable Search Paths" option.
- e. (Optional) If your KDC has a password policy, expand the **Advanced kerberos-env** section and adjust the Password options.
- f. (Optional) Ambari will test your Kerberos settings by generating a test principal and authenticating with that principal. To customize the test principal name that Ambari will use, expand the **Advanced kerberos-env** section and adjust the **Test Kerberos Principal** value. By default, the test principal name is a combination of cluster name and date (`${cluster_name}-${short_date}`). This test principal **will be deleted** after the test is complete.

- g. (Optional) If you need to customize the attributes for the principals Ambari will create, when using Active Directory, see [Customizing the Attribute Template](#) for more information. When using MIT KDC, you can pass **Principal Attributes** options in the **Advanced kerberos-env** section. For example, you can set options related to pre-auth or max. renew life by passing:

```
-requires_preauth -maxrenewlife "7 days"
```

7. Proceed with the install.
8. Ambari will install Kerberos clients on the hosts and test access to the KDC by testing that Ambari can create a principal, generate a keytab and distribute that keytab.
9. Customize the Kerberos identities used by Hadoop and proceed to kerberize the cluster.



Important

On the **Configure Identities** step, be sure to review the principal names, particularly the **Ambari Principals** on the **General** tab. These principal names, by default, append the name of the cluster to each of the Ambari principals. You can leave this as default or adjust these by removing the "-**{cluster-name}**" from principal name string. For example, if your cluster is named HDP and your realm is EXAMPLE.COM, the hdfs principal will be created as hdfs-HDP@EXAMPLE.COM.

10. Confirm your configuration. You can optionally download a CSV file of the principals and keytabs that Ambari will automatically create.
11. Click **Next** to start the process.
12. After principals have been created and keytabs have been generated and distributed, Ambari updates the cluster configurations, then starts and tests the Services in the cluster.
13. Exit the wizard when complete.
14. Ambari Server communicates with components in the cluster, and now with Kerberos setup, you need to make sure Ambari Server is setup for Kerberos. As part of the automated Kerberos setup process, Ambari Server has been given a keytab and setup is performed. All you need to do is restart Ambari Server for that to take affect. Therefore, restart Ambari Server at this time.

2.1.4.2.2. Launching the Kerberos Wizard (Manual Setup)

1. Be sure you have [Installed and Configured your KDC](#) and have [prepared the JCE](#) on each host in the cluster.
2. Log in to Ambari Web and Browse to Admin > Kerberos.
3. Click "Enable Kerberos" to launch the wizard.
4. Select the **Manage Kerberos principals and keytabs manually** option and confirm you have met the prerequisites.

5. Provide information about the KDC and admin account.
 - a. If your Kerberos client libraries are in non-standard path locations, expand the **Advanced kerberos-env** section and adjust the "Executable Search Paths" option.
6. Customize the Kerberos identities used by Hadoop and proceed to kerberize the cluster.



Important

On the **Configure Identities** step, be sure to review the principal names, particularly the **Ambari Principals** on the **General** tab. These principal names, by default, append the name of the cluster to each of the Ambari principals. You can leave this as default or adjust these by removing the "`-${cluster-name}`" from principal name string. For example, if your cluster is named HDP and your realm is EXAMPLE.COM, the hdfs principal will be created as `hdfs-HDP@EXAMPLE.COM`.

7. Confirm your configuration. Since you have chosen the Manual Kerberos Setup option, obtain the CSV file for the list of principals and keytabs required for the cluster to work with Kerberos. **Do not proceed until you have manually created and distributed the principals and keytabs to the cluster hosts.**
8. Click **Next** to continue.
9. Ambari updates the cluster configurations, then starts and tests the Services in the cluster.
10. Exit the wizard when complete.
11. Finish by setting up [Ambari Server for Kerberos](#).

2.1.5. Kerberos Client Packages

If you chose to enable Kerberos using the Automated Kerberos Setup option, as part of the enabling Kerberos process, Ambari installs the Kerberos clients on the cluster hosts. Depending on your operating system, the following packages are installed:

Packages installed by Ambari for the Kerberos Client

Operating System	Packages
RHEL/CentOS/Oracle Linux 7	krb5-workstation
RHEL/CentOS/Oracle Linux 6	krb5-workstation
SLES 11	krb5-client
Ubuntu/Debian	krb5-user, krb5-config

2.1.6. Disabling Kerberos Security

After [Enabling Kerberos Security \[19\]](#), you can disable Kerberos.

1. Log in to Ambari Web and Browse to Admin > Kerberos.
2. Click **Disable Kerberos** to launch the wizard.

3. Complete the wizard.



Note

If you have enabled Kerberos with an Automated Setup option, Ambari will attempt to contact the KDC and remove the principals created by Ambari. If the KDC is unavailable, the wizard will fail on the Unkerberize step. You can choose to ignore the failure and continue, but the principals will not be removed from the KDC.

2.1.7. Customizing the Attribute Template

If you are using the Kerberos Automated setup with Active Directory, depending on your KDC policies, you can customize the attributes that Ambari sets when creating principals. On the Configure Kerberos step of the wizard, in the **Advanced kerberos-env** section, you have access to the Ambari Attribute Template. This template (which is based on the [Apache Velocity](#) templating syntax) can be modified to adjust which attributes are set on the principals and how those attribute values are derived.

The following table lists the set of computed attribute variables available if you choose to modify the template:

Attribute Variables	Example
\$normalized_principal	nn/c6401.ambari.apache.org@EXAMPLE.COM
\$principal_name	nn/c6401.ambari.apache.org
\$principal_primary	nn
\$principal_digest	SHA1 hash of the \$normalized_principal
\$principal_digest_256	SHA256 hash of the \$normalized_principal
\$principal_digest_512	SHA512 hash of the \$normalized_principal
\$principal_instance	c6401.ambari.apache.org
\$realm	EXAMPLE.COM
\$password	password

2.1.8. Managing Admin Credentials

When you enable Kerberos, if you choose to use an **Existing MIT KDC** or **Existing Active Directory**, the Kerberos Wizard prompts for information related to the KDC, the KDC Admin Account credentials, and the Service and Ambari principals. Once provided, Ambari will automatically create principals, generate keytabs and distribute keytabs to the hosts in the cluster. The services will be configured for Kerberos and the service components are restarted to authenticate against the KDC. This is the **Kerberos Automated Setup** option.

By default, Ambari will not retain the KDC Admin Account credentials you provide unless you have [encrypted the passwords stored in Ambari](#). If you have not configured Ambari for password encryption, you will be prompted to provide KDC Admin Account credentials whenever cluster changes are made that require KDC principal and/or keytab changes (such as adding services, components and hosts).

If you have configured Ambari for password encryption, you will have an option to Save Admin Credentials. Ambari will use the retained KDC Admin Account credentials to make the KDC changes automatically.

Save Admin Credentials



Important

If you **do not** have password encryption enabled for Ambari, the Save Admin Credentials option **will not be** enabled.

Updating KDC Credentials

If you have chosen to Save Admin Credentials when enabling Kerberos, you can update or remove the credentials from Ambari using the following:

1. In Ambari Web, browse to **Admin > Kerberos** and click the **Manage KDC Credentials** button. The **Manage KDC Credentials** dialog is displayed.
2. If credentials have been previously saved, click **Remove** to remove the credentials currently stored in Ambari. Once removed, if cluster changes that require KDC principal and/or keytab changes (such as adding services, components and hosts), you will be prompted to enter the KDC Admin Account credentials.
3. Alternatively, to update the KDC Admin Account credentials, enter the Admin principal and password values and click **Save**.

2.2. Configuring HDP Components for Kerberos Using Ambari

This section describes how to configure Kerberos for strong authentication for HDP components in an Ambari-managed cluster.

2.2.1. Configuring Kafka for Kerberos Using Ambari

This section describes how to configure Kafka for Kerberos security on an Ambari-managed cluster.

Kerberos security for Kafka is an optional feature. When security is enabled, features include:

- Authentication of client connections (consumer, producer) to brokers
- ACL-based authorization

2.2.1.1. Preparing the Cluster

Before you enable Kerberos, your cluster must meet the following prerequisites:

Prerequisite	References*
Ambari-managed cluster with Kafka installed. <ul style="list-style-type: none"> • Ambari Version 2.1.0.0 or later • Stack version HDP 2.3.2 or later 	Installing, Configuring, and Deploying a HDP Cluster in Automated Install with Ambari
Key Distribution Center (KDC) server installed and running	Installing and Configuring the KDC

JCE installed on all hosts on the cluster (including the Ambari server)	Enabling Kerberos Authentication Using Ambari
---	---

Links are for Ambari 2.1.2.0.

When all prerequisites are fulfilled, enable Kerberos security. (For more information see [Launching the Kerberos Wizard \(Automated Setup\)](#).)

2.2.1.2. Configuring the Kafka Broker for Kerberos

During the installation process, Ambari configures a series of Kafka settings and creates a JAAS configuration file for the Kafka server.

It is not necessary to modify these settings, but for more information see Appendix A, [Appendix: Kafka Configuration Options \[37\]](#).

2.2.1.3. Creating Kafka Topics

When you use a script, command, or API to create a topic, an entry is created under ZooKeeper. The only user with access to ZooKeeper is the service account running Kafka (by default, `kafka`). Therefore, the first step toward creating a Kafka topic on a secure cluster is to run `kinit`, specifying the Kafka service keytab. The second step is to create the topic.

1. Run `kinit`, specifying the Kafka service keytab. For example:

```
kinit -k -t /etc/security/keytabs/kafka.service.keytab kafka/c6401.ambari.apache.org@EXAMPLE.COM
```

2. Next, create the topic. Run the `kafka-topics.sh` command-line tool with the following options:

```
/bin/kafka-topics.sh --zookeeper <hostname>:<port> --create --topic <topic-name> --partitions <number-of-partitions> --replication-factor <number-of-replicating-servers>
```

For example:

```
/bin/kafka-topics.sh --zookeeper c6401.ambari.apache.org:2181 --create --topic test_topic --partitions 2 --replication-factor 2  
Created topic "test_topic".
```

For more information about `kafka-topics.sh` parameters, see [Basic Kafka Operations](#) on the Apache Kafka website.

Permissions

By default, permissions are set so that only the Kafka service user has access; no other user can read or write to the new topic. In other words, if your Kafka server is running with principal `$KAFKA-USER`, only that principal will be able to write to ZooKeeper.

For information about adding permissions, see [Authorizing Access when Kerberos is Enabled](#).

2.2.1.4. Producing Events/Messages to Kafka on a Secured Cluster

Prerequisite: Make sure that you have enabled access to the topic (via Ranger or native ACLs) for the user associated with the producer process. We recommend that you use Ranger to manage permissions. For more information, see the [Apache Ranger User Guide for Kafka](#).

During the installation process, Ambari configures a series of Kafka client and producer settings, and creates a JAAS configuration file for the Kafka client. It is not necessary to modify these settings, but for more information about them see Appendix A, [Kafka Configuration Options](#).

Note: Only the Kafka Java API is supported for Kerberos. Third-party clients are not supported.

To produce events/messages:

1. Specify the path to the JAAS configuration file as one of your JVM parameters:

```
-Djava.security.auth.login.config=/usr/hdp/current/kafka-broker/config/kafka_client_jaas.conf
```

For more information about the `kafka_client_jaas` file, see "JAAS Configuration File for the Kafka Client" in [Kafka Configuration Options](#).

2. `kinit` with the principal's keytab.
3. Launch `kafka-console-producer.sh` with the following configuration options. (Note: these settings are the same as in previous versions, except for the addition of `--security-protocol SASL_PLAINTEXT`.)

```
./bin/kafka-console-producer.sh --broker-list <hostname:port  
[,hostname:port, ...]> --topic <topic-name> --security-protocol  
SASL_PLAINTEXT
```

For example:

```
./bin/kafka-console-producer.sh --broker-list  
c6401.ambari.apache.org:6667,c6402.ambari.apache.org:6667 --  
topic test_topic --security-protocol SASL_PLAINTEXT
```

Producer Code Example for a Kerberos-Enabled Cluster

The following example shows sample code for a producer in a Kerberos-enabled Kafka cluster. Note that the `SECURITY_PROTOCOL_CONFIG` property is set to `SASL_PLAINTEXT`.

```
package com.hortonworks.example.kafka.producer;  
  
import org.apache.kafka.clients.CommonClientConfigs;  
import org.apache.kafka.clients.producer.Callback;  
import org.apache.kafka.clients.producer.KafkaProducer;  
import org.apache.kafka.clients.producer.Producer;
```



```
import org.apache.kafka.clients.producer.ProducerConfig;
import org.apache.kafka.clients.producer.ProducerRecord;
import org.apache.kafka.clients.producer.RecordMetadata;

import java.util.Properties;
import java.util.Random;

public class BasicProducerExample {

    public static void main(String[] args){

        Properties props = new Properties();
        props.put(ProducerConfig.BOOTSTRAP_SERVERS_CONFIG, "kafka.example.
com:6667");

        // specify the protocol for SSL Encryption
        props.put(CommonClientConfigs.SECURITY_PROTOCOL_CONFIG,
"SASL_PLAINTEXT");

        props.put(ProducerConfig.ACKS_CONFIG, "all");
        props.put(ProducerConfig.RETRIES_CONFIG, 0);
        props.put(ProducerConfig.VALUE_SERIALIZER_CLASS_CONFIG, "org.apache.
kafka.common.serialization.StringSerializer");
        props.put(ProducerConfig.KEY_SERIALIZER_CLASS_CONFIG, "org.apache.
kafka.common.serialization.StringSerializer");

        Producer<String, String> producer = new KafkaProducer<String,
String>(props);
        TestCallback callback = new TestCallback();
        Random rnd = new Random();
        for (long i = 0; i < 100; i++) {
            ProducerRecord<String, String> data = new ProducerRecord<String,
String>(
                "test-topic", "key-" + i, "message-"+i );
            producer.send(data, callback);
        }

        producer.close();
    }

    private static class TestCallback implements Callback {
        @Override
        public void onComplete(RecordMetadata recordMetadata, Exception e) {
            if (e != null) {
                System.out.println("Error while producing message to topic :" +
recordMetadata);
                e.printStackTrace();
            } else {
                String message = String.format("sent message to topic:%s
partition:%s offset:%s", recordMetadata.topic(), recordMetadata.partition(),
recordMetadata.offset());
                System.out.println(message);
            }
        }
    }
}
```

To run the example, issue the following command:

```
$ java -Djava.security.auth.login.config=/usr/hdp/current/kafka-broker/
config/kafka_client_jaas.conf com.hortonworks.example.kafka.producer.
BasicProducerExample
```

Troubleshooting

Issue: If you launch the producer from the command-line interface without specifying the `security-protocol` option, you will see the following error:

```
2015-07-21 04:14:06,611] ERROR fetching topic metadata for topics
[Set(test_topic)] from broker
[ArrayBuffer(BrokerEndPoint(0,c6401.ambari.apache.org,6667),
BrokerEndPoint(1,c6402.ambari.apache.org,6667))] failed
(kafka.utils.CoreUtils$)
kafka.common.KafkaException: fetching topic metadata for topics
[Set(test_topic)] from broker
[ArrayBuffer(BrokerEndPoint(0,c6401.ambari.apache.org,6667),
BrokerEndPoint(1,c6402.ambari.apache.org,6667))] failed
    at kafka.client.ClientUtils$.fetchTopicMetadata(ClientUtils.scala:73)
Caused by: java.io.EOFException: Received -1 when reading from channel, socket
has likely been closed.
    at kafka.utils.CoreUtils$.read(CoreUtils.scala:193)
    at kafka.network.BoundedByteBufferReceive.
readFrom(BoundedByteBufferReceive.scala:54)
```

Solution: Add `--security-protocol SASL_PLAINTEXT` to the `kafka-console-producer.sh` runtime options.

2.2.1.5. Consuming Events/Messages from Kafka on a Secured Cluster

Prerequisite: Make sure that you have enabled access to the topic (via Ranger or native ACLs) for the user associated with the consumer process. We recommend that you use Ranger to manage permissions. For more information, see the [Apache Ranger User Guide for Kafka](#).

During the installation process, Ambari configures a series of Kafka client and producer settings, and creates a JAAS configuration file for the Kafka client. It is not necessary to modify these values, but for more information see see Appendix A, [Kafka Configuration Options](#).

Note: Only the Kafka Java API is supported for Kerberos. Third-party clients are not supported.

To consume events/messages:

1. Specify the path to the JAAS configuration file as one of your JVM parameters. For example:

```
-Djava.security.auth.login.config=/usr/hdp/current/kafka-broker/
config/kafka_client_jaas.conf
```

For more information about the `kafka_client_jaas` file, see "JAAS Configuration File for the Kafka Client" in [Kafka Configuration Options](#).

2. `kinit` with the principal's keytab.

3. Launch `kafka-console-consumer.sh` with the following configuration settings.
(Note: these settings are the same as in previous versions, except for the addition of `--security-protocol SASL_PLAINTEXT`.)

```
./bin/kafka-console-consumer.sh --zookeeper
c6401.ambari.apache.org:2181 --topic test_topic --from-beginning
--security-protocol SASL_PLAINTEXT
```

Consumer Code Example for a Kerberos-Enabled Cluster

The following example shows sample code for a producer in a Kerberos-enabled Kafka cluster. Note that the `SECURITY_PROTOCOL_CONFIG` property is set to `SASL_PLAINTEXT`.

```
package com.hortonworks.example.kafka.consumer;

import org.apache.kafka.clients.CommonClientConfigs;
import org.apache.kafka.clients.consumer.ConsumerConfig;
import org.apache.kafka.clients.consumer.ConsumerRebalanceListener;
import org.apache.kafka.clients.consumer.ConsumerRecord;
import org.apache.kafka.clients.consumer.ConsumerRecords;
import org.apache.kafka.clients.consumer.KafkaConsumer;
import org.apache.kafka.common.TopicPartition;

import java.util.Collection;
import java.util.Collections;
import java.util.Properties;

public class BasicConsumerExample {

    public static void main(String[] args) {

        Properties consumerConfig = new Properties();
        consumerConfig.put(ConsumerConfig.BOOTSTRAP_SERVERS_CONFIG, "kafka.
example.com:6667");

        // specify the protocol for SSL Encryption
        consumerConfig.put(CommonClientConfigs.SECURITY_PROTOCOL_CONFIG,
"SASL_PLAINTEXT");

        consumerConfig.put(ConsumerConfig.GROUP_ID_CONFIG, "my-group");
        consumerConfig.put(ConsumerConfig.AUTO_OFFSET_RESET_CONFIG,
"earliest");
        consumerConfig.put(ConsumerConfig.VALUE_DESERIALIZER_CLASS_CONFIG,
"org.apache.kafka.common.serialization.StringDeserializer");
        consumerConfig.put(ConsumerConfig.KEY_DESERIALIZER_CLASS_CONFIG, "org.
apache.kafka.common.serialization.StringDeserializer");
        KafkaConsumer<byte[], byte[]> consumer = new
KafkaConsumer<>(consumerConfig);
        TestConsumerRebalanceListener rebalanceListener = new
TestConsumerRebalanceListener();
        consumer.subscribe(Collections.singletonList("test-topic"),
rebalanceListener);

        while (true) {
            ConsumerRecords<byte[], byte[]> records = consumer.poll(1000);
            for (ConsumerRecord<byte[], byte[]> record : records) {
```

```
        System.out.printf("Received Message topic =%s, partition =%s,
offset = %d, key = %s, value = %s\n", record.topic(), record.partition(),
record.offset(), record.key(), record.value());
    }

    consumer.commitSync();
}

private static class TestConsumerRebalanceListener implements
ConsumerRebalanceListener {
    @Override
    public void onPartitionsRevoked(Collection<TopicPartition> partitions)
    {
        System.out.println("Called onPartitionsRevoked with partitions:" +
partitions);
    }

    @Override
    public void onPartitionsAssigned(Collection<TopicPartition> partitions)
    {
        System.out.println("Called onPartitionsAssigned with partitions:" +
partitions);
    }
}
```

To run the example, issue the following command:

```
# java -Djava.security.auth.login.config=/usr/hdp/current/kafka-broker/
config/kafka_client_jaas.conf com.hortonworks.example.kafka.consumer.
BasicConsumerExample
```

Troubleshooting

Issue: If you launch the consumer from the command-line interface without specifying the security-protocol option, you will see the following error:

```
2015-07-21 04:14:06,611] ERROR fetching topic metadata for topics
[Set(test_topic)] from broker
[ArrayBuffer(BrokerEndPoint(0,c6401.ambari.apache.org,6667),
BrokerEndPoint(1,c6402.ambari.apache.org,6667))] failed
(kafka.utils.CoreUtils$)
kafka.common.KafkaException: fetching topic metadata for topics
[Set(test_topic)] from broker
[ArrayBuffer(BrokerEndPoint(0,c6401.ambari.apache.org,6667),
BrokerEndPoint(1,c6402.ambari.apache.org,6667))] failed
    at kafka.client.ClientUtils$.fetchTopicMetadata(ClientUtils.scala:73)
Caused by: java.io.EOFException: Received -1 when reading from channel, socket
has likely been closed.
    at kafka.utils.CoreUtils$.read(CoreUtils.scala:193)
    at kafka.network.BoundedByteBufferReceive.
readFrom(BoundedByteBufferReceive.scala:54)
```

Solution: Add `--security-protocol SASL_PLAINTEXT` to the `kafka-console-consumer.sh` runtime options.

2.2.1.6. Authorizing Access when Kerberos is Enabled

Kafka ships with a pluggable Authorizer and an out-of-box authorizer implementation that uses ZooKeeper to store Access Control Lists (ACLs). Authorization can be done via Ranger (see the [Kafka](#) section of the Ranger Install Guide) or with native ACLs.

A Kafka ACL entry has the following general format:

```
Principal P is [Allowed/Denied] Operation O From Host H On
Resource R
```

where

- A principal is any entity that can be authenticated by the system, such as a user account, a thread or process running in the security context of a user account, or security groups of such accounts. `Principal` is specified in the `PrincipalType:PrincipalName` (`user:dev@EXAMPLE.COM`) format. Specify `user:*` to indicate all principals.

`Principal` is a comma-separated list of principals. Specify `*` to indicate all principals. (A principal is any entity that can be authenticated by the system, such as a user account, a thread or process running in the security context of a user account, or security groups of such accounts.)

- `Operation` can be one of: `READ`, `WRITE`, `CREATE`, `DESCRIBE`, or `ALL`.
- `Resource` is a topic name, a consumer group name, or the string “kafka-cluster” to indicate a cluster-level resource (only used with a `CREATE` operation).
- `Host` is the client host IP address. Specify `*` to indicate all hosts.



Note

For more information about ACL structure, including mappings between Operations values and Kafka protocol APIs, see the Apache [KIP-11 Authorization Interface](#) document.

2.2.1.6.1. Kafka Authorization Command Line Interface

The Kafka Authorization CLI script, `kafka-acls.sh`, resides in the `bin` directory.

The following table lists ACL actions supported by the CLI script:

Action Type	Description
<code>--add</code>	Add an ACL.
<code>--remove</code>	Remove an ACL.
<code>--list</code>	List ACLs.

The following table lists additional options for the Authorization CLI:

Option	Description	Default	Option Type
<code>--authorizer</code>	The fully-qualified class name of the authorizer.	<code>kafka.security.auth.SimpleAuthorizer</code>	Configuration
<code>--authorizer-properties</code>	A list of key=value pairs that will be passed to authorizer for initialization. Use this		Configuration

Option	Description	Default	Option Type
	option multiple times to specify multiple properties.		
<code>--cluster</code>	Specifies the cluster as resource.		Resource
<code>--topic <topic-name></code>	Specifies the topic as resource.		Resource
<code>--consumer-group <consumer-group></code>	Specifies the consumer group as resource.		Resource
<code>--allow-principal</code>	<p>These principals will be used to generate an ACL with Allow permission.</p> <p>Specify principal in <code>PrincipalType:name</code> format, such as <code>user:devadmin</code>.</p> <p>To specify more than one principal in a single command, specify this option multiple times. For example:</p> <pre>--allow-principal user: test1@EXAMPLE.COM --allow-principal user:test2@EXAMPLE.COM</pre>		Principal
<code>--deny-principal</code>	<p>These principals will be used to generate an ACL with Deny permission.</p> <p>Principal is in <code>PrincipalType:name</code> format.</p> <p>Multiple principals can be specified (see the <code>--allow-principal</code> option).</p>		Principal
<code>--allow-host</code>	IP address of the host from which the principals listed in <code>--allow-principal</code> will have access. To specify multiple hosts, specify this option multiple times.	if <code>--allow-principal</code> is specified, this defaults to <code>*</code> , which translates to "all hosts"	Host
<code>--deny-host</code>	IP address of the host from which the principals listed in <code>--deny-principal</code> will be denied access. To specify multiple hosts, specify this option multiple times.	if <code>--deny-principal</code> is specified, this defaults to <code>*</code> , which translates to "all hosts"	Host
<code>--operation</code>	<p>An operation that will be allowed or denied based on principal options.</p> <p>Valid values: Read, Write, Create, Delete, Alter, Describe, ClusterAction, All</p>	All	Operation
<code>--producer</code>	Convenience option to add or remove ACLs for		Convenience

Option	Description	Default	Option Type
	the producer role. This will generate ACLs that allow WRITE, DESCRIBE on topic, and CREATE on cluster.		
<code>--consumer</code>	Convenience option to add/remove ACLs for consumer role. This will generate ACLs that allows READ, DESCRIBE on topic, and READ on consumer-group.		Convenience

2.2.1.6.2. Authorization Examples

By default, if a principal does not have an explicit ACL that allows access for an operation to a resource, access requests from the principal will be denied.

The following examples show how to add, remove, and list ACLs.

2.2.1.6.2.1. Grant Read/Write Access to a Topic

To add the following ACL:

"Principals `user:bob` and `user:alice` are allowed to perform Operation Read and Write on Topic `Test-Topic` from Host1 and Host2"

run the CLI with the following options:

```
bin/kafka-acls.sh --add --allow-principal user:bob --allow-principal user:alice --allow-host host1 --allow-host host2 --operation Read --operation Write --topic test-topic
```

2.2.1.6.2.2. Grant Full Access to Topic, Cluster, and Consumer Group

To add ACLs to a topic, specify `--topic <topic-name>` as the resource option. Similarly, to add ACLs to cluster, specify `--cluster`; to add ACLs to a consumer group, specify `--consumer-group <group-name>`.

The following examples grant full access for principal `bob` to topic `test-topic` and consumer group `10`, across the cluster. Substitute your own values for principal name, topic name, and group name.

```
bin/kafka-acls.sh --topic test-topic --add --allow-principal user:bob --operation ALL --config /usr/hdp/current/kafka-broker/config/server.properties
```

```
bin/kafka-acls.sh --consumer-group 10 --add --allow-principal user:bob --operation ALL --config /usr/hdp/current/kafka-broker/config/server.properties
```

```
bin/kafka-acls.sh --cluster --add --allow-principal user:bob --operation ALL --config /usr/hdp/current/kafka-broker/config/server.properties
```

2.2.1.6.2.3. Add a Principal as Producer or Consumer

The most common use case for ACL management is to add or remove a principal as producer or consumer. The following convenience options handle these cases.

To add `user:bob` as a producer of `Test-topic`, run the following command:

```
bin/kafka-acls.sh --add --allow-principal user:bob --producer --
topic test-topic
```

Similarly, to add `user:alice` as a consumer of `test-topic` with consumer group `group-1`, pass the `--consumer` option.



Note

When using the consumer option you must specify the consumer group.

```
bin/kafka-acls.sh --add --allow-principal user:alice --consumer --
topic test-topic --consumer-group group-1
```

2.2.1.6.2.4. Deny Access to a Principal

In rare cases you might want to define an ACL that allows access to all but one or more principals. In this case, use the `--deny-principal` and `--deny-host` options.

For example, to allow all users to read from `test-topic` *except* user `bob` from host `bad-host`:

```
bin/kafka-acls.sh --add --allow-principal user:* --allow-host * --
deny-principal user:bob --deny-host bad-host --operation Read --
topic test-topic
```

2.2.1.6.2.5. Remove Access

Removing ACLs is similar to adding ACLs. The only difference is that you need to specify the `--remove` option instead of the `--add` option.

To remove the ACLs for principals `bob` and `alice` (added in "Grant Read/Write Access to a Topic"), run the CLI with the following options:

```
bin/kafka-acls.sh --remove --allow-principal user:bob --allow-
principal user:alice --allow-host host1 --allow-host host2 --
operation Read --operation Write --topic test-topic
```

Similarly, to remove a principal from a producer or consumer role, specify the `--remove` option instead of `--add`:

```
bin/kafka-acls.sh --remove --allow-principal user:bob --producer
--topic test-topic
```

2.2.1.6.2.6. List ACLs

To list ACLs for any resource, specify the `--list` option with the resource. For example, to list all ACLs for `Test-topic`, run the CLI with following options:

```
bin/kafka-acls.sh --list --topic test-topic
```

2.2.1.6.2.7. Configure Authorizer Settings

To specify which authorizer to use, include the `--authorizer` option. For example:


```
--authorizer kafka.security.auth.SimpleAclAuthorizer ...
```

To specify one or more authorizer initialization settings, include the `--authorizer-properties` option; for example:

```
--authorizer-properties zookeeper.connect=localhost:2181 ...
```

2.2.1.6.3. Troubleshooting Authorizer Settings

Frequently-asked Questions:

When should I use Deny?

By default, all principals that are not explicitly granted permissions get rejected. You should not need to use Deny. (Note: when defined, DENY takes precedence over ALLOW.)

Then why do we have deny?

Deny was introduced into Kafka for advanced use cases where negation was required. Deny should only be used to negate a large allow, where listing all principals or hosts is cumbersome.

Can I define ACLs with principal as user@<realm>?

You can if you are not using `principal.to.local.class`, but if you have set this configuration property you must define your ACL with users without REALM. This is a known issue in HDP 2.3.

I just gave a user CREATE Permissions on a cluster, but the user still can't create topics. Why?

Right now, Kafka create topic is not implemented as an API, but as a script that directly modifies ZooKeeper entries. In a secure environment only the Kafka broker user is allowed to write to ZooKeeper entries. Granting a user CREATE access does not allow that user to modify ZooKeeper entries.

However, if that user makes a producer request to the topic and has `auto.create.topics.enable` set to `true`, a topic will be created at the broker level.

2.2.1.7. Appendix: Kafka Configuration Options

2.2.1.7.1. Server.properties key-value pairs

Ambari configures the following Kafka values during the installation process. Settings are stored as key-value pairs stored in an underlying `server.properties` configuration file.

listeners

A comma-separated list of URIs that Kafka will listen on, and their protocols.

Required property with three parts:

```
<protocol>:<hostname>:<port>
```

Set `<protocol>` to `SASL_PLAINTEXT`, to specify the protocol that server accepts connections. SASL authentication will be used over a plaintext channel. Once SASL

authentication is established between client and server, the session will have the client's principal as an authenticated user. The broker can only accept SASL (Kerberos) connections, and there is no wire encryption applied. (Note: For a non-secure cluster, `<protocol>` should be set to PLAINTEXT.)

Set `hostname` to the hostname associated with the node you are installing. Kerberos uses this value and "principal" to construct the Kerberos service name. Specify `hostname 0.0.0.0` to bind to all interfaces. Leave `hostname` empty to bind to the default interface.

Set `port` to the Kafka service port. When Kafka is installed using Ambari, the default port number is 6667.

Examples of legal listener lists::

```
listeners=SASL_PLAINTEXT://kafka1.host1.com:6667
```

```
listeners=PLAINTEXT://myhost:9092, TRACE://:9091,  
SASL_PLAINTEXT://0.0.0.0:9093
```

advertised.listeners

A list of listeners to publish to ZooKeeper for clients to use, if different than the listeners specified in the preceding section.

In IaaS environments, this value might need to be different from the interface to which the broker binds.

If `advertised.listeners` is not set, the value for `listeners` will be used.

Required value with three parts:

```
<protocol>:<hostname>:<port>
```

Set `protocol` to SASL_PLAINTEXT, to specify the protocol that server accepts connections. SASL authentication will be used over a plaintext channel. Once SASL authentication is established between client and server, the session will have the client's principal as an authenticated user. The broker can only accept SASL (Kerberos) connections, and there is no wire encryption applied. (Note: For a non-secure cluster, `<protocol>` should be set to PLAINTEXT.)

Set `hostname` to the hostname associated with the node you are installing. Kerberos uses this and "principal" to construct the Kerberos service name.

Set `port` to the Kafka service port. When Kafka is installed using Ambari, the default port number is 6667.

For example:

```
advertised.listeners=SASL_PLAINTEXT://kafka1.host1.com:6667
```

security.inter.broker.protocol

Specifies the inter-broker communication protocol. In a Kerberized cluster, brokers are required to communicate over SASL. (This approach supports replication of topic data.) Set the value to `SASL_PLAINTEXT`:

```
security.inter.broker.protocol=SASL_PLAINTEXT
```

authorizer.class.name

Configures the authorizer class.

Set this value to `kafka.security.auth.SimpleAclAuthorizer`:

```
authorizer.class.name=kafka.security.auth.SimpleAclAuthorizer
```

For more information, see "Authorizing Access when Kerberos is Enabled."

principal.to.local.class

Transforms Kerberos principals to their local Unix usernames.

Set this value to `kafka.security.auth.KerberosPrincipalToLocal`:

```
principal.to.local.class=kafka.security.auth.KerberosPrincipalToLocal
```

super.users

Specifies a list of user accounts that will have all cluster permissions. By default, these super users have all permissions that would otherwise need to be added through the `kafka-acls.sh` script. Note, however, that their permissions do not include the ability to create topics through `kafka-topics.sh`, as this involves direct interaction with ZooKeeper.

Set this value to a list of `user:<account>` pairs separated by semicolons. Note that Ambari adds `user:kafka` when Kerberos is enabled.

Here is an example:

```
super.users=user:bob;user:alice
```

2.2.1.7.2. JAAS Configuration File for the Kafka Server

The Java Authentication and Authorization Service (JAAS) API supplies user authentication and authorization services for Java applications.

After enabling Kerberos, Ambari sets up a JAAS login configuration file for the Kafka server. This file is used to authenticate the Kafka broker against Kerberos. The file is stored at:

```
/usr/hdp/current/kafka-broker/config/kafka_server_jaas.conf
```

Ambari adds the following settings to the file. (Note: `serviceName="kafka"` is required for connections from other brokers.)

```
KafkaServer {  
    com.sun.security.auth.module.Krb5LoginModule required
```

```

useKeyTab=true
keyTab="/etc/security/keytabs/kafka.service.keytab"
storeKey=true
useTicketCache=false
serviceName="kafka"
principal="kafka/c6401.ambari.apache.org@EXAMPLE.COM" ;
};

Client { // used for zookeeper connection
com.sun.security.auth.module.Krb5LoginModule required
useKeyTab=true
keyTab="/etc/security/keytabs/kafka.service.keytab"
storeKey=true
useTicketCache=false
serviceName="zookeeper"
principal="kafka/c6401.ambari.apache.org@EXAMPLE.COM" ;
};

```

2.2.1.7.3. Configuration Setting for the Kafka Producer

After enabling Kerberos, Ambari sets the following key-value pair in the `server.properties` file:

```
security.protocol=SASL_PLAINTEXT
```

2.2.1.7.4. JAAS Configuration File for the Kafka Client

After enabling Kerberos, Ambari sets up a JAAS login configuration file for the Kafka client. Settings in this file will be used for any client (consumer, producer) that connects to a Kerberos-enabled Kafka cluster. The file is stored at:

```
/usr/hdp/current/kafka-broker/config/kafka_client_jaas.conf
```

Ambari adds the following settings to the file. (Note: `serviceName=kafka` is required for connections from other brokers.)



Note

For command-line utilities like `kafka-console-producer` and `kafka-console-consumer`, use `kinit`. If you use a long-running process (for example, your own Producer), use `keytab`.

Kafka client configuration *with* keytab, for producers:

```

KafkaClient {
com.sun.security.auth.module.Krb5LoginModule required
useKeyTab=true
keyTab="/etc/security/keytabs/storm.service.keytab"
storeKey=true
useTicketCache=false
serviceName="kafka"
principal="storm@EXAMPLE.COM" ;
};

```

Kafka client configuration *without* keytab, for producers:

```
KafkaClient {
  com.sun.security.auth.module.Krb5LoginModule required
  useTicketCache=true
  renewTicket=true
  serviceName="kafka";
};
```

Kafka client configuration for *consumers*:

```
KafkaClient {
  com.sun.security.auth.module.Krb5LoginModule required
  useTicketCache=true
  renewTicket=true
  serviceName="kafka";
};
```

2.2.2. Configuring Storm for Kerberos Using Ambari

This section describes how to configure Storm for Kerberos security on an Ambari-managed cluster.

2.2.2.1. Prerequisites

Before you enable Kerberos, your cluster must meet the following prerequisites. (Note: Links point to Ambari version 2.2.1.0. If your cluster runs a different version of Ambari, refer to the Ambari document for your version of software.)

Prerequisite	References
Ambari-managed cluster with Storm installed and running. <ul style="list-style-type: none"> Ambari Version 2.2.1.0 or later Stack version HDP 2.4.0 or later 	Installing, Configuring, and Deploying a HDP Cluster in Automated Install with Ambari
Key Distribution Center (KDC) server installed and running.	Installing and Configuring the KDC
JCE installed on all hosts on the cluster (including the Ambari server).	Enabling Kerberos Authentication Using Ambari

When all prerequisites are fulfilled, enable Kerberos security. For more information, see [Running the Kerberos Security Wizard](#).

2.2.2.2. Designating a Storm Client Node

At this point in the configuration process there is no notion of a Storm client node (you won't be able to select "client" via Ambari).

To specify a Storm client node, choose one of the following two approaches, described in the following subsections:

- Dedicate or use an existing independent gateway node as a storm client
- Use one of your existing storm nodes (such as nimbus, supervisors, or drpc) as a client. Choose this option if you prefer not to add a gateway node for Storm.

2.2.2.2.1. Dedicate or Use an Existing Gateway Node

To dedicate or use an existing gateway node (edge node):

1. Install the storm package on the node:

```
sudo yum install storm_<version>
```

For example, for HDP 2.4:

```
sudo yum install storm_2_4*
```

2. Create a file at `/etc/storm/conf/client_jaas.conf`, and add the following entry to it:

```
StormClient {
    com.sun.security.auth.module.Krb5LoginModule required
    useTicketCache=true
    renewTicket=true
    serviceName="nimbus"
};
```

3. Add the following settings to the `/etc/storm/conf/storm.yaml` configuration file:

```
nimbus.seeds: <nimbus-host-array>
nimbus.thrift.port: 6627
java.security.auth.login.config: "/etc/storm/conf/client_jaas.conf"
storm.thrift.transport: "org.apache.storm.security.auth.kerberos.
KerberosSaslTransportPlugin"
```

where `<nimbus-host-array>` is an array of hostnames running Nimbus. (The value should come from `/etc/storm/conf/storm.yaml`.) For example:

```
nimbus.seeds: ["c6401.ambari.apache.org",
"c6402.ambari.apache.org"]
```

2.2.2.2. Use an Existing Storm Node

To use one of your existing Storm nodes (such as nimbus, supervisors, or drpc) as a Storm client node, complete the following steps for every user who requires Storm access (for example, to run Storm commands or deploy topologies):

1. Create a `.storm` directory in the user's home directory. For example, user `john` should have a directory called `/home/john/.storm/`.
2. Add the following settings to the `/etc/storm/conf/storm.yaml` configuration file:

```
nimbus.seeds: <nimbus-host-array>
nimbus.thrift.port: 6627
java.security.auth.login.config: "/etc/storm/conf/client_jaas.conf"
storm.thrift.transport: "org.apache.storm.security.auth.kerberos.
KerberosSaslTransportPlugin"
```

where `<nimbus-host-array>` is an array of hostnames running Nimbus (the value should come from `/etc/storm/conf/storm.yaml`.) For example:

```
nimbus.seeds: ["c6401.ambari.apache.org",
"c6402.ambari.apache.org"]
```

As mentioned earlier, repeat these steps for every user who requires Storm access.

2.2.2.2.3. Running Storm Commands

After configuring the client/gateway node, run `kinit` (with the principal's keytab) before issuing Storm commands.

2.2.2.3. Running Workers as Users

In Storm secure mode, workers can run as the user (owner of the topology) who deployed the topology. To enable, complete the following steps:

1. Make sure all users who are going to deploy topologies have a UNIX account on all of the Storm nodes. Workers will run under the UNIX account for topologies deployed by the user.

Example: For user `testuser1` and principal `testuser1/c6401.ambari.apache.org`, make sure there is a corresponding `testuser1` UNIX account.

2. Add the following configuration under "Custom storm-site" in the Ambari Storm configuration screen:

```
supervisor.run.worker.as.user : true
```

3. Restart Storm components.

2.2.2.4. Accessing the Storm UI

The Storm UI uses SPNEGO AUTH when in Kerberos mode.

Before accessing the UI, configure your browser for SPNEGO authorization, as shown in the following table.

Then `kinit` before accessing the Storm UI.

Table 2.1. Browser Settings for Storm UI

Browser	Configuration
Safari	No changes needed.
Firefox	<ol style="list-style-type: none"> 1. Go to <code>about:config</code> and search for <code>network.negotiate-auth.trusted-uris</code>. 2. Double-click and add the following value: <code>"http://storm-ui-hostname:ui-port"</code> 3. Replace the <code>storm-ui-hostname</code> value with the hostname where your UI is running. 4. Replace the <code>ui-port</code> value with the Storm UI port.
Chrome	From the command line, issue: <pre>google-chrome --auth-server-whitelist="<storm-ui-hostname>" --auth-negotiate-delegate-whitelist="<storm-ui-hostname>"</pre>
Internet Explorer	<ul style="list-style-type: none"> • Configure trusted websites to include <code>"storm-ui-hostname"</code>. • Allow negotiation for the UI website.

2.2.2.5. Accessing the Storm UI (Active Directory Trust Configuration)

If your cluster is configured with Active Directory Trust, use the Active Directory ticket to communicate with MIT KDC for secure negotiation. Here are the additional configuration steps:

1. Make sure UI Kerberos authentication-to-local rules are configured properly. Once a principal from Active Directory is used for negotiation with MIT KDC, you need a rule to translate it to the local account on the Storm UI node. Many times those can be copied from `core-site.xml`.

For example:

```
ui.filter.params:  
  "type": "kerberos"  
  "kerberos.principal": "HTTP/nimbus.host1.com"  
  "kerberos.keytab": "/vagrant/keytabs/http.keytab"  
  "kerberos.name.rules": "RULE:[2:$1@$0]([jt]t@.*EXAMPLE.COM)s/.*/  
$MAPRED_USER/ RULE:[2:$1@$0]([nd]n@.*EXAMPLE.COM)s/.*/$HDFS_USER/DEFAULT"
```

Note: Rules are listed as strings, and are not separated by commas.

2. Create mappings for MIT domain KDC and associated resources used for the domain, in this case Storm UI.

On a Windows workstation, you would run the following commands from the command line:

```
ksetup /AddKDC $DOMAIN $KDC
```

```
ksetup /AddHostToRealmMap $hadoop_resource $Domain
```

Note: this step adds registry entries in `HKLM\System\CurrentControlSet\Control\Lsa\Kerberos\HostToRealm`.

Troubleshooting

To troubleshoot configuration issues, try accessing the Storm UI within the cluster using the `curl` command.

For example:

```
curl -i --negotiate -u:anyUser -b ~/cookiejar.txt -c ~/  
cookiejar.txt http://storm-ui-hostname:8080/api/v1/cluster/summary
```

This will help you determine whether the Kerberos UI configuration is working.

To isolate the issue, use Storm service keytabs and user principals.

Two other important things to check are:

- Make sure that the trust is working properly.
- Make sure that the encryption types match on both KDCs.

2.2.2.6. Storm Security Properties

The following table lists important Storm security properties.

Configuration Property	Description	Example
nimbus.authorizer	This is a pluggable authorizer for a Storm Nimbus node. SimpleACLAuthorizer is the default implementation. Note: Admins can also grant permissions via the Ranger authorizer UI. For more information, see the Ranger User's Guide.	"org.apache.storm.security.auth.authorizer.SimpleACLAuthorizer"
nimbus.admins	Add Nimbus admin users. These users will have super user permissions on all topologies deployed, and will be able to perform other admin operations (such as rebalance, activate, deactivate and kill), even if they are not the owners of the topology. By default, only users who deployed the topologies have access to admin operations such as rebalance, activate, deactivate, and kill.	"John" "Abc"
topology.users:	This and the following config can be added as part of the topology file. The users listed in this setting will have owner privileges for the specified topology.	<pre>Config conf = new Config() conf.put("topology.users", Lists.newArrayList("test_user1", "test_user2")); StormSubmitter submitTopology(topologyName, conf, builder. createTopology());</pre>
topology.groups	Similar to topology.users. Use this to add group-level permissions to a topology.	<pre>Config conf = new Config() conf.put("topology.groups", Lists.newArrayList("test_group1", "test_group2")); StormSubmitter submitTopology(topologyName, conf, builder. createTopology());</pre>

2.2.2.7. Known Issues

Issue: Ambari does not show the security configuration on the Storm configuration tab, so you cannot add users to nimbus.admins.

Workaround: To give permissions to other users, use `topology.users` or `topology.groups`.

Issue: In AD+MIT setup, when trying to access Nimbus on a Kerberized cluster a HTTP 413 full HEAD error is received. ([STORM-633](#))

Workaround: Add `ui.header.buffer.bytes : "65536"` under "Custom storm-site" on the Ambari Storm configuration tab.

Issue: Log viewer. We recommend against creating HTTP principal keytabs for supervisors. This can cause the SPNEGO protocol to fail.

Workaround:

1. Add the HTTP principal for Storm supervisor nodes too. For example:

```
sudo /usr/sbin/kadmin.local -q 'addprinc -randkey HTTP/  
<supervisor-hostname>
```

where

<supervisor-hostname> is your hostname and domain for Kerberos; for example:
supervisor1.host1.com@HOST1.COM

2. Add this principal for all hosts that run supervisor machines.

For example:

```
sudo /usr/sbin/kadmin.local -q "ktadd -k /etc/security/keytabs/  
spnego.service.keytab HTTP/supervisor1.host1.com@HOST1.COM"
```

3. Add the newly created HTTP principals to the `spnego.service.keytab` file.
4. Make sure that the `spnego.service.keytab` file has "storm" user privileges for read operations.
5. Distribute this keytab to all supervisor hosts.
6. On the supervisor node, edit `/etc/storm/conf/storm.yaml`. Change the `ui.filter.parameters` as follows, replacing <supervisor-hostname> with the hostname of your supervisor process:

```
"type": "kerberos"  
  
"kerberos.principal": "HTTP/<supervisor-hostname>"  
  
"kerberos.keytab": "/vagrant/keytabs/http.keytab"
```
7. On each supervisor machine change the `Kerberos.principal` hostname to that supervisor's hostname.
8. Restart the log viewer.
9. Add supervisor hosts to `network.negotiate-auth.trusted-uris` (similar to the steps needed to access the Storm UI).

2.3. Configuring Ambari Authentication with LDAP or AD

2.3.1. Configuring Ambari for LDAP or Active Directory Authentication

By default Ambari uses an internal database as the user store for authentication and authorization. If you want to configure LDAP or Active Directory (AD) external authentication, you need to [collect the following information](#) and [run a setup command](#).

Also, you must [synchronize your LDAP users and groups](#) into the Ambari DB to be able to manage authorization and permissions against those users and groups.



Note

When synchronizing LDAP users and groups, Ambari uses LDAP results paging controls to synchronize large numbers of LDAP objects. Most modern LDAP servers support these control, but for those that do not, such as Oracle Directory Server Enterprise Edition 11g, Ambari introduces a configuration parameter to disable pagination. The `authentication.ldap.pagination.enabled` property can be set to `false` in the `/etc/ambari-server/conf/ambari-properties` file to disable result paging controls. This will limit the maximum number of entities that can be imported at any given time to the maximum result limit of the LDAP server. To work around this, import sets of users or groups using the `-users` and `-groups` options covered in [Specific Set of Users and Groups \[51\]](#).

2.3.1.1. Setting Up LDAP User Authentication

The following table details the properties and values you need to know to set up LDAP authentication.



Note

If you are going to set `bindAnonymously` to `false` (the default), you need to make sure you have an LDAP Manager name and password set up. If you are going to use SSL, you need to make sure you have already set up your certificate and keys.

Ambari Server LDAP Properties

Property	Values	Description
<code>authentication.ldap.primaryUrl</code>	<code>server:port</code>	The hostname and port for the LDAP or AD server. Example: <code>my.ldap.server:389</code>
<code>authentication.ldap.secondaryUrl</code>	<code>server:port</code>	The hostname and port for the secondary LDAP or AD server. Example: <code>my.secondary.ldap.server:389</code> This is an optional value.
<code>authentication.ldap.useSSL</code>	<code>true</code> or <code>false</code>	If <code>true</code> , use SSL when connecting to the LDAP or AD server.
<code>authentication.ldap.usernameAttribute</code>	[LDAP Attribute]	The attribute for username. Example: <code>uid</code>
<code>authentication.ldap.baseDn</code>	[Distinguished Name]	The root Distinguished Name to search in the directory for users. Example: <code>ou=people,dc=hadoop,dc=apache,dc=org</code>
<code>authentication.ldap.referral</code>	[Referral method]	Determines if LDAP referrals should be followed, or ignored.
<code>authentication.ldap.bindAnonymously</code>	<code>true</code> or <code>false</code>	If <code>true</code> , bind to the LDAP or AD server anonymously
<code>authentication.ldap.managerDn</code>	[Full Distinguished Name]	If Bind anonymous is set to <code>false</code> , the Distinguished Name ("DN") for the manager. Example: <code>uid=hdfs,ou=people,dc=hadoop,dc=apache,dc=org</code>
<code>authentication.ldap.managerPassword</code>	[password]	If Bind anonymous is set to <code>false</code> , the password for the manager
<code>authentication.ldap.userObjectClass</code>	[LDAP Object Class]	The object class that is used for users. Example: <code>organizationalPerson</code>
<code>authentication.ldap.groupObjectClass</code>	[LDAP Object Class]	The object class that is used for groups. Example: <code>groupOfUniqueNames</code>

Property	Values	Description
authentication.ldap.groupMembershipAttribute	[LDAP attribute]	The attribute for group membership. Example: uniqueMember
authentication.ldap.groupNameAttribute	[LDAP attribute]	The attribute for group name.

2.3.1.2. Configure Ambari to use LDAP Server



Note

Only if you are using LDAPS, and the LDAPS server certificate is signed by a trusted Certificate Authority, there is no need to import the certificate into Ambari so this section does not apply to you. If the LDAPS server certificate is self-signed, or is signed by an unrecognized certificate authority such as an internal certificate authority, you must import the certificate and create a keystore file. The following example creates a keystore file at `/keys/ldaps-keystore.jks`, but you can create it anywhere in the file system:

Run the LDAP setup command on the Ambari server and answer the prompts, using the information you collected above:

1. `mkdir /etc/ambari-server/keys`

where the keys directory does not exist, but should be created.

2. `$JAVA_HOME/bin/keytool -import -trustcacerts -alias root -file $PATH_TO_YOUR_LDAPS_CERT -keystore /etc/ambari-server/keys/ldaps-keystore.jks`

3. Set a password when prompted. You will use this during `ambari-server setup-ldap`.

`ambari-server setup-ldap`

1. At the `Primary URL*` prompt, enter the server URL and port you collected above. Prompts marked with an asterisk are required values.
2. At the `Secondary URL*` prompt, enter the secondary server URL and port. This value is optional.
3. At the `Use SSL*` prompt, enter your selection. **If using LDAPS**, enter `true`.
4. At the `User object class*` prompt, enter the object class that is used for users.
5. At the `User name attribute*` prompt, enter your selection. The default value is `uid`.
6. At the `Group object class*` prompt, enter the object class that is used for groups.
7. At the `Group name attribute*` prompt, enter the attribute for group name.
8. At the `Group member attribute*` prompt, enter the attribute for group membership.
9. At the `Distinguished name attribute*` prompt, enter the attribute that is used for the distinguished name.

10. At the `Base DN*` prompt, enter your selection.
11. At the `Referral method*` prompt, enter to follow or ignore LDAP referrals.
12. At the `Bind anonymously*` prompt, enter your selection.
13. At the `Manager DN*` prompt, enter your selection if you have set `bind.Anonymously` to false.
14. At the `Enter the Manager Password*` prompt, enter the password for your LDAP manager DN.
15. If you set `Use SSL* = true` in step 3, the following prompt appears: Do you want to provide custom TrustStore for Ambari?

Consider the following options and respond as appropriate.

- **More secure option:** If using a self-signed certificate that you do not want imported to the existing JDK keystore, enter `y`.

For example, you want this certificate used only by Ambari, not by any other applications run by JDK on the same host.

If you choose this option, additional prompts appear. Respond to the additional prompts as follows:

- At the `TrustStore type` prompt, enter `jks`.
- At the `Path to TrustStore file` prompt, enter `/keys/ldaps-keystore.jks` (or the actual path to your keystore file).
- At the `Password for TrustStore` prompt, enter the password that you defined for the keystore.
- **Less secure option:** If using a self-signed certificate that you want to import and store in the existing, default JDK keystore, enter `n`.

- Convert the SSL certificate to X.509 format, if necessary, by executing the following command:

```
openssl x509 -in slapd.pem -out <slapd.crt>
```

Where `<slapd.crt>` is the path to the X.509 certificate.

- Import the SSL certificate to the existing keystore, for example the default jre certificates storage, using the following instruction:

```
/usr/jdk64/jdk1.7.0_45/bin/keytool -import -trustcacerts -file slapd.crt -keystore /usr/jdk64/jdk1.7.0_45/jre/lib/security/cacerts
```

Where Ambari is set up to use JDK 1.7. Therefore, the certificate must be imported in the JDK 7 keystore.

16. Review your settings and if they are correct, select `y`.

17 Start or restart the Server

```
ambari-server restart
```

The users you have just imported are initially granted the Ambari User privilege. Ambari Users can read metrics, view service status and configuration, and browse job information. For these new users to be able to start or stop services, modify configurations, and run smoke tests, they need to be Admins. To make this change, as an Ambari Admin, use `Manage Ambari > Users > Edit`. For instructions, see [Managing Users and Groups](#).

2.3.1.2.1. Example Active Directory Configuration

Directory Server implementations use specific object classes and attributes for storing identities. In this example, configurations specific to Active Directory are displayed as an example. Only those properties that are specific to Active Directory are displayed.

Run `ambari-server setup-ldap` and provide the following information about your Domain.

Prompt	Example AD Values
User object class* (posixAccount)	user
User name attribute* (uid)	sAMAccountName
Group object class* (posixGroup)	group
Group member attribute* (memberUid)	member
Distinguished name attribute* (dn)	distinguishedName

2.3.1.3. Synchronizing LDAP Users and Groups

Run the LDAP synchronize command and answer the prompts to initiate the sync:

```
ambari-server sync-ldap [option]
```



Note

To perform this operation, your Ambari Server must be running.

- When prompted, you must provide credentials for an Ambari Admin.
- When syncing ldap, Local user accounts with matching username will switch to LDAP type, which means their authentication will be against the external LDAP and not against the Local Ambari user store.
- LDAP sync only syncs up-to-1000 users. If your LDAP contains over 1000 users and you plan to import over 1000 users, you must use the `-users` option when syncing and specify a filtered list of users to perform import in batches.

The utility provides three options for synchronization:

- Specific set of users and groups, or
- Synchronize the existing users and groups in Ambari with LDAP, or

- All users and groups

Review log files for failed synchronization attempts, at `/var/log/ambari-server/ambari-server.log` on the Ambari Server host.



Note

When synchronizing LDAP users and groups, Ambari uses LDAP results paging controls to synchronize large numbers of LDAP objects. Most modern LDAP servers support these control, but for those that do not, such as Oracle Directory Server Enterprise Edition 11g, Ambari introduces a configuration parameter to disable pagination. The `authentication.ldap.pagination.enabled` property can be set to `false` in the `/etc/ambari-server/conf/ambari-properties` file to disable result paging controls. This will limit the maximum number of entities that can be imported at any given time to the maximum result limit of the LDAP server. To work around this, import sets of users or groups using the `-users` and `-groups` options covered in [Specific Set of Users and Groups \[51\]](#).

2.3.1.4. Specific Set of Users and Groups

```
ambari-server sync-ldap --users users.txt --groups groups.txt
```

Use this option to synchronize a specific set of users and groups from LDAP into Ambari. Provide the command a text file of comma-separated users and groups. The comma separated entries in each of these files should be based off of the values in LDAP of the attributes chosen during setup. The "User name attribute" should be used for the `users.txt` file, and the "Group name attribute" should be used for the `groups.txt` file. This command will find, import, and synchronize the matching LDAP entities with Ambari.



Note

Group membership is determined using the Group Membership Attribute (`groupMembershipAttr`) specified during `setup-ldap`. User name is determined by using the Username Attribute (`usernameAttribute`) specified during `setup-ldap`.

2.3.1.5. Existing Users and Groups

```
ambari-server sync-ldap --existing
```

After you have performed a synchronization of a [specific set of users and groups](#), you use this option to synchronize only those entities that are in Ambari with LDAP. Users will be removed from Ambari if they no longer exist in LDAP, and group membership in Ambari will be updated to match LDAP.



Note

Group membership is determined using the Group Membership Attribute specified during `setup-ldap`.

2.3.1.6. All Users and Groups



Important

Only use this option if you are sure you want to synchronize all users and groups from LDAP into Ambari. If you only want to synchronize a subset of users and groups, use a [specific set of users and groups](#) option.

```
ambari-server sync-ldap --all
```

This will import all entities with matching LDAP user and group object classes into Ambari.

2.3.2. Configuring Ranger Authentication with UNIX, LDAP, or AD

2.3.2.1. UNIX Authentication Settings

The following figure shows the UNIX authentication settings, and the table below describes each of these properties.

The screenshot shows the 'Add Service Wizard' interface. Under 'Ranger Settings', the 'External URL' is 'http://c6403.ambari.apache.org:6080'. The 'Authentication method' is set to 'UNIX' (selected with a radio button). 'HTTP enabled' is checked. Under 'Unix Authentication Settings', 'Allow remote Login' is set to 'true'. The 'ranger.unixauth.service.hostname' is set to '{{lvsync_host}}' and the 'ranger.unixauth.service.port' is set to '5151'. There are also sections for 'Knox SSO Settings' and 'Advanced ranger-admin-site' which are currently collapsed.

Table 2.2. UNIX Authentication Settings

Configuration Property	Description	Default Value	Example Value	Required?
Allow remote Login	Flag to enable/disable remote login via UNIX Authentication Mode.	TRUE	TRUE	No.

Configuration Property	Description	Default Value	Example Value	Required?
ranger.unixauth.service.host	The FQDN where the ranger-usersync module is running (along with the UNIX Authentication Service).	localhost	myunixhost.domain.com	Yes, if UNIX authentication is selected.
ranger.unixauth.service.port	The port number where the ranger-usersync module is running the UNIX Authentication Service.	5151	5151	Yes, if UNIX authentication is selected.

2.3.2.2. Active Directory Authentication Settings

This section describes how to configure settings for Active Directory authentication.



Note

In addition to these settings, you may also need to configure the Active Directory properties described in Configure Ranger User Sync.

2.3.2.2.1. AD Settings

The following figure shows the Active Directory (AD) authentication settings, and the table below describes each of these properties.

Table 2.3. Active Directory Authentication Settings

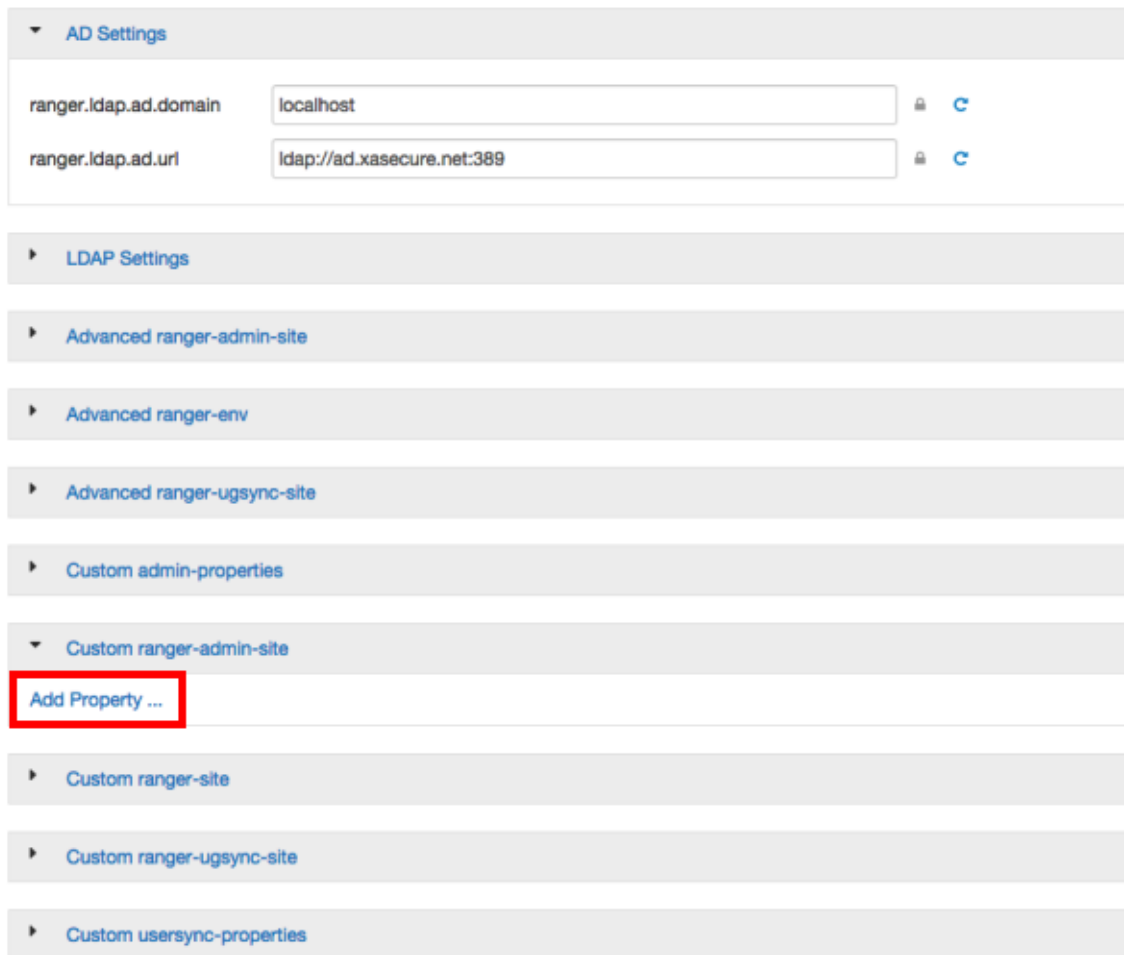
Configuration Property Name	Description	Default Value	Example Value	Required?
ranger ldap.ad.domain	Server domain name (or IP address) where ranger-usersync module is running (along with the AD Authentication Service). The default value of "localhost" must be changed to the domain name.	localhost	example.com	Yes, if Active Directory authentication is selected.
ranger ldap.ad.url	The URL and port number where ranger-usersync module is running the AD Authentication Service. The default value is a placeholder and must be changed to point to the AD server.	ldap://ad.xasecure.net:389	ldap://127.0.0.1:389	Yes, if Active Directory authentication is selected.

2.3.2.2.2. Custom ranger-admin-site Settings for Active Directory (Optional)

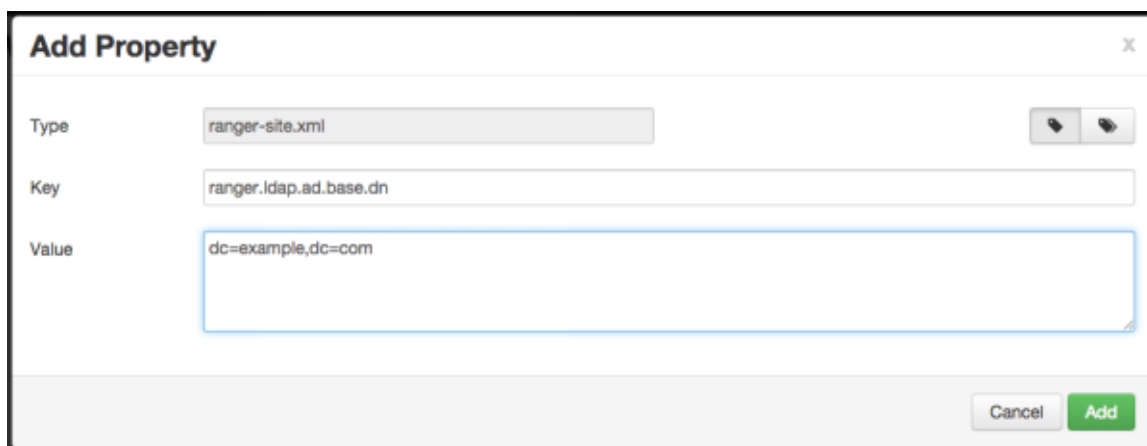
The following Custom ranger-admin-site settings for Active Directory authentication are optional.

To add a Custom ranger-admin-site property:

1. Select **Custom ranger-admin-site**, then click **Add Property**.



2. On the Add Property pop-up, type the property name in the **Key** box, type the property value in the **Value** box, then click **Add**.



The following figure shows the Custom ranger-admin-site settings required for Active Directory (AD) authentication, and the table below describes each of these properties.

▼ Custom ranger-site

ranger.ldap.ad.base.dn	<input type="text" value="dc=example,dc=com"/>	+ -
ranger.ldap.ad.bind.dn	<input type="text" value="cn=adadmin,cn=Users,dc=example,dc=com"/>	+ -
ranger.ldap.ad.bind.password	<input type="text" value="secret123!"/>	+ -
ranger.ldap.ad.referral	<input type="text" value="follow"/>	+ -
Add Property ...		

Table 2.4. Active Directory Custom ranger-admin-site Settings

Custom Property Name	Sample Values for AD Authentication
ranger.ldap.ad.base.dn	dc=example,dc=com
ranger.ldap.ad.bind.dn	cn=adadmin,cn=Users,dc=example,dc=com
ranger.ldap.ad.bind.password	secret123!
ranger.ldap.ad.referral	follow ignore throw

There are three possible values for `ranger.ldap.ad.referral`: `follow`, `throw`, and `ignore`. The recommended setting is `follow`.

When searching a directory, the server might return several search results, along with a few continuation references that show where to obtain further results. These results and references might be interleaved at the protocol level.

- When this property is set to `follow`, the AD service provider processes all of the normal entries first, and then follows the continuation references.
- When this property is set to `throw`, all of the normal entries are returned in the enumeration first, before the `ReferralException` is thrown. By contrast, a "referral" error response is processed immediately when this property is set to `follow` or `throw`.
- When this property is set to `ignore`, it indicates that the server should return referral entries as ordinary entries (or plain text). This might return partial results for the search. In the case of AD, a `PartialResultException` is returned when referrals are encountered while search results are processed.

2.3.2.3. LDAP Authentications Settings

This section describes how to configure LDAP and Advanced ranger-ugsync-site settings for Active Directory authentication.



Note

In addition to these settings, you must also configure the LDAP properties described in [Configure Ranger User Sync](#).

2.3.2.3.1. LDAP Settings

The following figure shows the LDAP authentication settings, and the table below describes each of these properties.

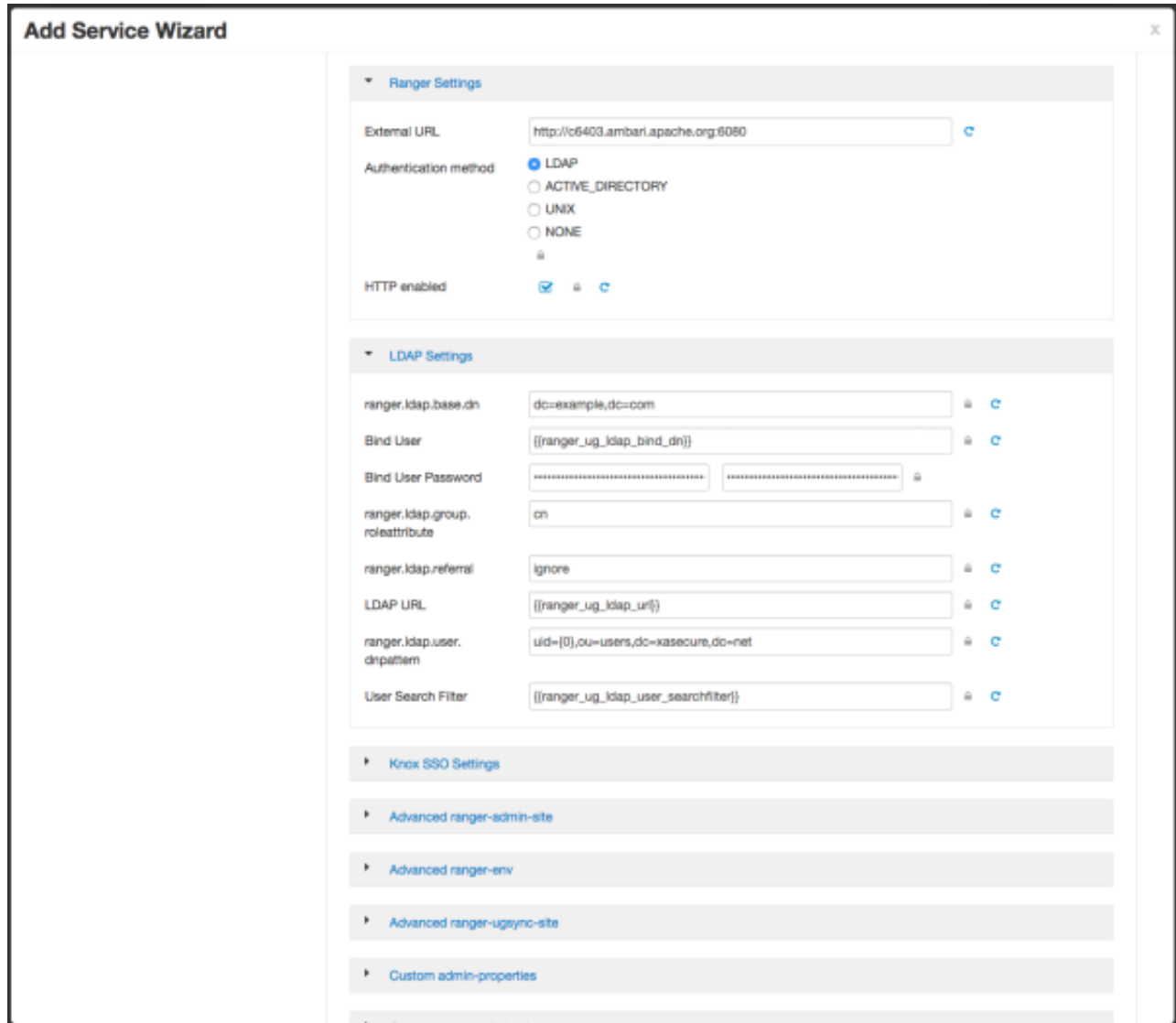


Table 2.5. LDAP Authentication Settings

Configuration Property Name	Description	Default Value	Example Value	Required?
ranger.idap.url	The URL and port number where ranger-usersync module is running the LDAP Authentication Service.	ldap://71.127.43.33:389	ldap://127.0.0.1:389	Yes, if LDAP authentication is selected.

Configuration Property Name	Description	Default Value	Example Value	Required?
ranger.ldap.user.dnpattern	The domain name pattern.	uid={0},ou=users,dc=xasecure,dc=net	cn=ldapadmin,ou=Users,dc=example,dc=com	Yes, if LDAP authentication is selected.
ranger.ldap.group.roleattribute	The LDAP group role attribute.	cn	cn	Yes, if LDAP authentication is selected.

2.3.2.3.2. Custom ranger-admin-site Settings for LDAP (Optional)

The following Custom ranger-admin-site settings for LDAP are optional.

To add a Custom ranger-admin-site property:

1. Select **Custom ranger-admin-site**, then click **Add Property**.

The screenshot shows a configuration interface with several sections. The 'AD Settings' section is expanded, showing two properties: 'ranger.ldap.ad.domain' with the value 'localhost' and 'ranger.ldap.ad.url' with the value 'ldap://ad.xasecure.net:389'. Below this, the 'LDAP Settings' section is collapsed. The 'Advanced ranger-admin-site' section is also collapsed. The 'Custom ranger-admin-site' section is expanded, and the 'Add Property ...' button is highlighted with a red box. Other sections like 'Custom ranger-site', 'Custom ranger-ugsync-site', and 'Custom usersync-properties' are also collapsed.

2. On the Add Property pop-up, type the property name in the **Key** box, type the property value in the **Value** box, then click **Add**.

Add Property

Type: ranger-admin-site.xml

Key: ranger.ldap.base.dn

Value: dc=example,dc=com

Buttons: Cancel, Add

The following figure shows the Custom ranger-admin-site settings required for LDAP authentication, and the table below describes each of these properties.

Custom ranger-site

ranger.ldap.ad.base.dn: dc=example,dc=com

ranger.ldap.ad.bind.dn: cn=adadmin,cn=Users,dc=example,dc=com

ranger.ldap.ad.bind.password: secret123!

ranger.ldap.ad.referral: follow

Add Property ...

Table 2.6. LDAP Custom ranger-admin-site Settings

Custom Property Name	Sample Values for AD or LDAP Authentication
ranger.ldap.base.dn	dc=example,dc=com
ranger.ldap.bind.dn	cn=adadmin,cn=Users,dc=example,dc=com
ranger.ldap.bind.password	secret123!
ranger.ldap.referral	follow ignore throw

There are three possible values for `ranger.ldap.referral`: `follow`, `throw`, and `ignore`. The recommended setting is `follow`.

When searching a directory, the server might return several search results, along with a few continuation references that show where to obtain further results. These results and references might be interleaved at the protocol level.

- When this property is set to `follow`, the LDAP service provider processes all of the normal entries first, and then follows the continuation references.

- When this property is set to `throw`, all of the normal entries are returned in the enumeration first, before the `ReferralException` is thrown. By contrast, a "referral" error response is processed immediately when this property is set to `follow` or `throw`.
- When this property is set to `ignore`, it indicates that the server should return referral entries as ordinary entries (or plain text). This might return partial results for the search.

2.3.2.3.3. Advanced ranger-admin-site Settings

The following Advanced ranger-admin-site properties apply only to LDAP authentication.

Table 2.7. Active Directory Authentication Settings

Property Name	Sample values for LDAP Authentication
<code>ranger.ldap.group.searchbase</code>	<code>dc=example,dc=com</code>
<code>ranger.ldap.group.searchfilter</code>	<code>(member=cn={0},ou=Users,dc=example,dc=com)</code>

2.3.3. Encrypting Database and LDAP Passwords in Ambari

By default the passwords to access the Ambari database and the LDAP server are stored in a plain text configuration file. To have those passwords encrypted, you need to run a special setup command.

Ambari Server should not be running when you do this: either make the edits before you start Ambari Server the first time or bring the server down to make the edits.

1. On the Ambari Server, run the special setup command and answer the prompts:

```
ambari-server setup-security
```

- a. Select Option 2: 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.

- b. Provide a master key for encrypting the passwords. You are prompted to enter the key twice for accuracy.

If your passwords are encrypted, you need access to the master key to start Ambari Server.

- c. You have three options for maintaining the master key:

- Persist it to a file on the server by pressing `y` at the prompt.
- Create an environment variable `AMBARI_SECURITY_MASTER_KEY` and set it to the key.
- Provide the key manually at the prompt on server start up.

- d. Start or restart the Server


```
ambari-server restart
```

2.3.3.1. Reset Encryption

There may be situations in which you want to:

- [Remove Encryption Entirely \[61\]](#)
- [Change the Current Master Key \[61\]](#), either because the key has been forgotten or because you want to change the current key as a part of a security routine.

Ambari Server should not be running when you do this.

2.3.3.2. Remove Encryption Entirely

To reset Ambari database and LDAP passwords to a completely unencrypted state:

1. On the Ambari host, open `/etc/ambari-server/conf/ambari.properties` with a text editor and set this property

```
security.passwords.encryption.enabled=false
```

2. Delete `/var/lib/ambari-server/keys/credentials.jceks`
3. Delete `/var/lib/ambari-server/keys/master`
4. You must now reset the database password and, if necessary, the LDAP password. Run [ambari-server setup](#) and [ambari-server setup-ldap](#) again.

2.3.3.3. Change the Current Master Key

To change the master key:

- If you know the current master key or if the current master key has been persisted:

1. Re-run the encryption setup command and follow the prompts.

```
ambari-server setup-security
```

- a. Select Option 2: 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.

- b. Enter the current master key when prompted if necessary (if it is not persisted or set as an environment variable).

- c. At the `Do you want to reset Master Key` prompt, enter `yes`.

- d. At the prompt, enter the new master key and confirm.

- If you do **not** know the current master key:
 - Remove encryption entirely, as described [here](#).
 - Re-run `ambari-server setup-security` as described [here](#).
 - Start or restart the Ambari Server.

```
ambari-server restart
```

2.4. Configuring LDAP Authentication in Hue

You can configure Hue to authenticate users by using LDAP, including importing users and groups from the LDAP directory service, OpenLDAP, or through Active Directory (AD).

Integrating Hue with LDAP enables users to leverage existing authentication credentials' group permissions directly from their LDAP profiles.

For you to configure LDAP authentication in Hue, Hue must be running and the Hue UI must be accessible.

Each of the following subsections describes a property that needs to be setup for Hue to work with LDAP.

2.4.1. Enabling the LDAP Backend

To use LDAP, you must specify the following Hue values in the Hue configuration file, `/etc/hue/conf/hue.ini`. Specifying this configuration enables Hue to validate login credentials against the LDAP directory service:

```
[desktop]
  [[auth]]
    .....
    .....
    backend=desktop.auth.backend.LdapBackend
    .....
```

2.4.2. Enabling User Authentication with Search Bind

Settings related to LDAP are in the LDAP section of the Hue configuration file, `/etc/hue/conf/hue.ini`:

```
[desktop]
  [[ldap]]
```

There are two ways to authenticate users by using the LDAP directory service in Hue:

- Search Bind (default)

Setting the `search_bind_authentication` property to `true` in `/etc/hue/conf/hue.ini` enables LDAP search using the bind credentials specified for the `bind_dn` and `bind_password` properties.

Search bind performs an LDAP search against the directory service and then binds the results by using the found Distinguished Name (DN) and provided password. The search process starts from the base DN specified for the `base_dn` property and continues to search the base DN for an entry with an attribute that matches the specified in `user_name_attr` of the username provided at login.

You can restrict the results of this search process by using the `user_filter` (default value `objectclass=*`) and `user_name_attr` (default value `sAMAccountName`) properties in the `[desktop] > [[ldap]] > [[[users]]]` section of `/etc/hue/conf/hue.ini`.

If you use the default values of `user_filter` and `user_name_attr`, the LDAP search filter appears as follows, where `<username>` is the user name provided on the Hue login page:

```
(&(objectClass=*)(sAMAccountName=<username>))
```

- Direct Bind

Setting the `search_bind_authentication` property to `false` in `/etc/hue/conf/hue.ini` enables the LDAP direct bind mechanism to authenticate users. Hue binds to the LDAP server using the user name and password provided on the login page.

Depending on the value of the `nt_domain` property, there are two ways that direct bind works:

- If `nt_domain` is specified, the `nt_domain` property is intended to be used only with Active Directory (AD) service.

This property allows Hue to authenticate with AD without having to follow LDAP references to other partitions.

Hue forms the User Principal Name (UPN) as a concatenation of the user name provided on the Hue login page and the `nt_domain` property value: for example, `<username>@<nt_domain>`. The `ldap_username_pattern` property is ignored.

- If `nt_domain` is not specified, the `nt_domain` property is intended to be used for all other directory services.

Without the `nt_domain` property specified, the `ldap_username_pattern` appears as follows, where `<username>` is the user name provided on the Hue login page:

```
uid=<username>,ou=People,dc=mycompany,dc=com
```

2.4.3. Setting the Search Base to Find Users and Groups

You must set the correct `base_dn` value in the `/etc/hue/conf/hue.ini` file to enable Hue to find users and groups:

```
# The search base for finding users and groups
base_dn="DC=mycompany,DC=com"
```

2.4.4. Specifying the URL of the LDAP Server

Specify the following URL in the `/etc/hue/conf/hue.ini` file to specify the endpoint of the corporate LDAP server:

```
# URL of the LDAP server
ldap_url=ldap://auth.mycompany.com
```

2.4.5. Specifying LDAPS and StartTLS Support

LDAPS (secure communication with LDAP) is provided using the SSL/TLS and StartTLS protocols. Using these protocols allows Hue to validate the directory service with which it is going to interact. Optionally, specify the path to the certificate for authentication over TLS in the `/etc/hue/conf/hue.ini` file:

```
# A PEM-format file containing certificates for the CA's that
# Hue will trust for authentication over TLS.
# The certificate for the CA that signed the
# LDAP server certificate must be included among these certificates.
# See more here http://www.openldap.org/doc/admin24/tls.html.
ldap_cert=<path/to/cert/file
use_start_tls=true
```

2.4.6. Specifying Bind Credentials for LDAP Searches

If the LDAP server does not support anonymous searches, you must specify the distinguished name of the user to bind and the bind password in the `/etc/hue/conf/hue.ini` file:

```
# Distinguished name of the user to bind as -- not necessary if the LDAP
server
# supports anonymous searches
bind_dn="CN=ServiceAccount,DC=mycompany,DC=com"

# Password of the bind user -- not necessary if the LDAP server supports
# anonymous searches
bind_password=your_password
```

2.4.7. Synchronizing Users and Groups

You can use the LDAP username pattern to restrict users when performing searches. Using this pattern provides a template for the DN that is sent to the directory service when authenticating. Replace the `<username>` parameter with the user name provided on the Hue login page. Specify this pattern in the `/etc/hue/conf/hue.ini` file:

```
# Pattern for searching for usernames -- Use <username> for the parameter
# For use when using LdapBackend for Hue authentication
ldap_username_pattern="uid=<username>,ou=People,dc=mycompany,dc=com"
```

When performing the authentication, Hue must import users to its database to work properly. In this case, passwords are never imported.

By default, the LDAP authentication backend automatically creates users that do not exist in Hue database. The purpose of disabling the automatic import process is to allow only a predefined list of manually imported users to log in.

```
# Create users in Hue when they try to login with their LDAP credentials
# For use when using LdapBackend for Hue authentication
create_users_on_login = true
```

You can specify that user groups be synchronized when a user logs in (to keep the user permission up to date):

```
# Synchronize a users groups when they login
sync_groups_on_login=false
```

You can configure Hue to ignore username lettercasing or to force lowercasing:

```
# Ignore the case of usernames when searching for existing users in Hue.
ignore_username_case=false

# Force usernames to lowercase when creating new users from LDAP.
force_username_lowercase=false
```

2.4.8. Setting Search Bind Authentication and Importing Users and Groups

Hue uses search bind authentication by default. To enable direct bind authentication, set the `search_bind_authentication` property to `false` in the `/etc/hue/conf/hue.ini` file:

```
# Do not use search bind authentication.
search_bind_authentication=false
```

When you set search bind authentication and you also need to set the kind of subgrouping and referrals for users and groups in the `/etc/hue/conf/hue.ini` file:

```
# Choose which kind of subgrouping to use: nested or subordinate(deprecated).
subgroups=subordinate

# Define the number of levels to search for nested members.
nested_members_search_depth=10

# Whether or not to follow referrals
follow_referrals=false
```

2.4.9. Setting LDAP Users' Filter

You can restrict user access by setting the LDAP filter (`user_filter` attribute) and the username attribute in the LDAP schema (`user_name_attr` attribute) in the `/etc/hue/conf/hue.ini` file. The typical attributes for search in the LDAP schema are `uid` and `sAMAccountName`:

```
[[[users]]]

# Base filter for searching for users
user_filter="objectclass=*"

# The username attribute in the LDAP schema
user_name_attr=sAMAccountName
```

2.4.10. Setting an LDAP Groups Filter

You can restrict the synchronization of LDAP directory groups when using the Hue useradmin application by setting a base filter (`group_filter` attribute) and the username attribute in the LDAP schema (`group_name_attr` attribute) of the `/etc/hue/conf/hue.ini` file:

```
[[[groups]]]

# Base filter for searching for groups
group_filter="objectclass=*"

# The username attribute in the LDAP schema
group_name_attr=cn
```

2.4.11. Setting Multiple LDAP Servers

Hue enables you to configure multiple LDAP servers by providing the multiple server declaration in `/etc/hue/conf/hue.ini`:

```
[[[ldap_servers]]]

[[[[mycompany]]]]

# The search base for finding users and groups
base_dn="DC=mycompany,DC=com"

# URL of the LDAP server
ldap_url=ldap://auth.mycompany.com

# A PEM-format file containing certificates for the CA's that
# Hue will trust for authentication over TLS.
# The certificate for the CA that signed the
# LDAP server certificate must be included among these certificates.
# See more here http://www.openldap.org/doc/admin24/tls.html.
## ldap_cert=
## use_start_tls=true

# Distinguished name of the user to bind as -- not necessary if the LDAP
server
# supports anonymous searches
bind_dn="CN=ServiceAccount,DC=mycompany,DC=com"

# Password of the bind user -- not necessary if the LDAP server supports
# anonymous searches
bind_password=your_password

# Pattern for searching for usernames -- Use <username> for the parameter
```

```
# For use when using LdapBackend for Hue authentication
ldap_username_pattern="uid=<username>,ou=People,dc=mycompany,dc=com"

# Whether or not to follow referrals
## follow_referrals=false

[[[[[users]]]]]

# Base filter for searching for users
user_filter="objectclass=Person"

# The username attribute in the LDAP schema
user_name_attr=sAMAccountName

[[[[[groups]]]]]

# Base filter for searching for groups
group_filter="objectclass=groupOfNames"

# The username attribute in the LDAP schema
group_name_attr=cn
```

2.5. Advanced Security Options for Ambari

This section describes several security options for an Ambari-monitored-and-managed Hadoop cluster.

- [Configuring Ambari for Non-Root \[67\]](#)
- [Optional: Ambari Web Inactivity Timeout \[71\]](#)
- [Optional: Set Up Kerberos for Ambari Server \[72\]](#)
- [Optional: Set Up Two-Way SSL Between Ambari Server and Ambari Agents \[73\]](#)
- [Optional: Configure Ciphers and Protocols for Ambari Server \[73\]](#)
- [Optional: HTTP Cookie Persistence \[73\]](#)

2.5.1. Configuring Ambari for Non-Root

In most secure environments, restricting access to and limiting services that run as root is a hard requirement. For these environments, Ambari can be configured to operate without direct root access. Both Ambari Server and Ambari Agent components allow for non-root operation, and the following sections will walk you through the process.

- [How to Configure Ambari Server for Non-Root \[67\]](#)
- [How to Configure an Ambari Agent for Non-Root \[69\]](#)

2.5.1.1. How to Configure Ambari Server for Non-Root

You can configure the Ambari Server to run as a non-root user.

During the [ambari-server setup](#) process, when prompted to `Customize user account for ambari-server daemon?`, choose `y`.

The setup process prompts you for the appropriate, non-root user to run the Ambari Server as; for example: ambari.



Note

The non-root user you choose to run the Ambari Server should be part of the Hadoop group. This group must match the service Hadoop group accounts referenced in the Custom Services > Misc tab during the Install Wizard configuration step. The default group name is `hadoop` but if you customized this value during cluster install, be sure to make the non-root user a part of that group. See [Customizing HDP Services](#) for more information on service account users and groups.



Note

If Ambari Server is running as a non-root user, such as 'ambari', and you are planning on using Ambari Views, the following properties in **Services > HDFS > Configs > Advanced core-site** must be added:

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

The non-root functionality relies on `sudo` to run specific commands that require elevated privileges as defined in the [Sudoer Configuration - Ambari Server](#). The `sudo` configuration for Ambari Server is split into two sections: [Commands - Ambari Server \[68\]](#), and [Sudo Defaults - Ambari Server \[69\]](#).

2.5.1.1.1. Sudoer Configuration - Ambari Server

The [Commands - Ambari Server \[68\]](#), and [Sudo Defaults - Ambari Server \[69\]](#) sections describe how you should configure `sudo` to enable Ambari to run as a non-root user. Each of the sections includes the specific `sudo` entries that you should place in `/etc/sudoers` by running the `visudo` command.

2.5.1.1.2. Commands - Ambari Server

This section contains the specific commands that must be issued for standard server operations:

```
# Ambari Commands
ambari ALL=(ALL) NOPASSWD:SETENV: /bin/mkdir -p /etc/security/keytabs, /bin/
chmod * /etc/security/keytabs/*.keytab, /bin/chown * /etc/security/keytabs/*.
keytab, /bin/chgrp * /etc/security/keytabs/*.keytab, /bin/rm -f /etc/security/
keytabs/*.keytab, /bin/cp -p -f /var/lib/ambari-server/data/tmp/* /etc/
security/keytabs/*.keytab
ambari ALL=(ALL) NOPASSWD:SETENV: /bin/mkdir -p /var/lib/ambari-server/data/
tmp, /bin/chmod * /var/lib/ambari-server/data/tmp, /bin/chown * /var/lib/
ambari-server/data/tmp, /bin/chgrp * /var/lib/ambari-server/data/tmp, /bin/
rm -rf /var/lib/ambari-server/data/tmp/*, /bin/cp -f /tmp/* /var/lib/ambari-
server/data/tmp/*, /usr/bin/test * *, /bin/stat -c %u %g %a /var/lib/ambari-
server/data/tmp/*
```

To ensure that the configuration has been done properly, you can `su` to the `ambari` user and run `sudo -l`. There, you can double check that there are no warnings, and that the configuration output matches what was just applied.

2.5.1.1.3. Sudo Defaults - Ambari Server

Some versions of sudo have a default configuration that prevents sudo from being invoked from a non-interactive shell. In order for the agent to run its commands non-interactively, some defaults need to be overridden.

```
Defaults exempt_group = ambari
Defaults !env_reset,env_delete-=PATH
Defaults: ambari !requiretty
```



Note

If sudo is not properly set up, the following error will be seen when the "Configure Ambari Identity" stage fails:

```
stderr:
sudo: no tty present and no askpass program specified

stdout:
Server action failed
```

2.5.1.2. How to Configure an Ambari Agent for Non-Root

You can configure the Ambari Agent to run as a non-privileged user as well. That user requires specific sudo access in order to su to Hadoop service accounts and perform specific privileged commands. Configuring Ambari Agents to run as non-root requires that you manually install agents on all nodes in the cluster. For these details, see [Installing Ambari Agents Manually](#). After installing each agent, you must configure the agent to run as the desired, non-root user. In this example we will use the `ambari` user.

Change the `run_as_user` property in the `/etc/ambari-agent/conf/ambari-agent.ini` file, as illustrated below:

```
run_as_user=ambari
```

Once this change has been made, the `ambari-agent` must be restarted to begin running as the non-root user.

The non-root functionality relies on sudo to run specific commands that require elevated privileges as defined in the [Sudoer Configuration](#). The sudo configuration is split into three sections: [Customizable Users - Ambari Agents \[69\]](#), [Commands - Ambari Agents \[70\]](#), and [Sudo Defaults](#).

2.5.1.2.1. Sudoer Configuration - Ambari Agents

The [Customizable Users - Ambari Agents \[69\]](#), [Commands - Ambari Agents \[70\]](#), and [Sudo Defaults - Ambari Agents \[71\]](#) sections will cover how sudo should be configured to enable Ambari to run as a non-root user. Each of the sections includes the specific sudo entries that should be placed in `/etc/sudoers` by running the `visudo` command.

2.5.1.2.2. Customizable Users - Ambari Agents

This section contains the `su` commands and corresponding Hadoop service accounts that are configurable on install:

```
# Ambari Customizable Users
ambari ALL=(ALL) NOPASSWD:SETENV: /bin/su hdfs *,/bin/su ambari-qa *,/bin/su
ranger *,/bin/su zookeeper *,/bin/su Knox *,/bin/su falcon *,/bin/su ams *,
/bin/su flume *,/bin/su hbase *,/bin/su spark *,/bin/su accumulio *,/bin/su
hive *,/bin/su hcat *,/bin/su kafka *,/bin/su mapred *,/bin/su oozie *,/bin/
su sqoop *,/bin/su storm *,/bin/su tez *,/bin/su atlas *,/bin/su yarn *,/bin/
su kms *,/bin/su activity_analyzer *,/bin/su livy *,/bin/su zeppelin *,/bin/su
infra-solr *,/bin/su logsearch *
```



Note

These user accounts must match the service user accounts referenced in the **Customize Services > Misc** tab during the Install Wizard configuration step. For example, if you customize YARN to run as `xyz_yarn`, modify the `su` command above to be `/bin/su xyz_yarn`.

These user accounts must match the service user accounts referenced in the **Customize Services > Misc** tab during the Install Wizard configuration step. For example, if you customize YARN to run as `xyz_yarn`, modify the `su` command above to be `/bin/su xyz_yarn`.

2.5.1.2.3. Commands - Ambari Agents

This section contains the specific commands that must be issued for standard agent operations:

```
# Ambari: Core System Commands
ambari ALL=(ALL) NOPASSWD:SETENV: /usr/bin/yum,/usr/bin/zypper,/usr/bin/apt-
get, /bin/mkdir, /usr/bin/test, /bin/ln, /bin/ls, /bin/chown, /bin/chmod, /
bin/chgrp, /bin/cp, /usr/sbin/setenforce, /usr/bin/test, /usr/bin/stat, /bin/
mv, /bin/sed, /bin/rm, /bin/kill, /bin/readlink, /usr/bin/pgrep, /bin/cat,
/usr/bin/unzip, /bin/tar, /usr/bin/tee, /bin/touch, /usr/bin/mysql, /sbin/
service mysqld *, /usr/bin/dpkg *, /bin/rpm *, /usr/sbin/hst *
```

```
# Ambari: Hadoop and Configuration Commands
ambari ALL=(ALL) NOPASSWD:SETENV: /usr/bin/hdp-select, /usr/bin/conf-select,
/usr/hdp/current/hadoop-client/sbin/hadoop-daemon.sh, /usr/lib/hadoop/bin/
hadoop-daemon.sh, /usr/lib/hadoop/sbin/hadoop-daemon.sh, /usr/bin/ambari-
python-wrap *
```

```
# Ambari: System User and Group Commands
ambari ALL=(ALL) NOPASSWD:SETENV: /usr/sbin/groupadd, /usr/sbin/groupmod, /
usr/sbin/useradd, /usr/sbin/usermod
```

```
# Ambari: Knox Commands
ambari ALL=(ALL) NOPASSWD:SETENV: /usr/bin/python2.6 /var/lib/ambari-agent/
data/tmp/validateKnoxStatus.py *, /usr/hdp/current/knox-server/bin/knoxcli.sh
```

```
# Ambari: Ranger Commands
ambari ALL=(ALL) NOPASSWD:SETENV: /usr/hdp/*/ranger-usersync/setup.sh, /usr/
bin/ranger-usersync-stop, /usr/bin/ranger-usersync-start, /usr/hdp/*/ranger-
admin/setup.sh *, /usr/hdp/*/ranger-knox-plugin/disable-knox-plugin.sh *, /
usr/hdp/*/ranger-storm-plugin/disable-storm-plugin.sh *, /usr/hdp/*/ranger-
hbase-plugin/disable-hbase-plugin.sh *, /usr/hdp/*/ranger-hdfs-plugin/disable-
hdfs-plugin.sh *, /usr/hdp/current/ranger-admin/ranger_credential_helper.py, /
usr/hdp/current/ranger-kms/ranger_credential_helper.py, /usr/hdp/*/ranger-*/
ranger_credential_helper.py
```

```
# Ambari Infra and LogSearch Commands
ambari ALL=(ALL) NOPASSWD:SETENV: /usr/lib/ambari-infra-solr/bin/solr *, /usr/
lib/ambari-logsearch-logfeeder/run.sh *, /usr/sbin/ambari-metrics-grafana *, /
usr/lib/ambari-infra-solr-client/solrCloudCli.sh *
```



Important

Do not modify the command lists, only the usernames in the [Customizable Users - Ambari Agents \[69\]](#) section may be modified.

To re-iterate, you must do this sudo configuration on every node in the cluster. To ensure that the configuration has been done properly, you can su to the ambari user and run sudo -l. There, you can double check that there are no warnings, and that the configuration output matches what was just applied.

2.5.1.2.4. Sudo Defaults - Ambari Agents

Some versions of sudo have a default configuration that prevents sudo from being invoked from a non-interactive shell. In order for the agent to run its commands non-interactively, some defaults need to be overridden.

```
Defaults exempt_group = ambari
Defaults !env_reset,env_delete-=PATH
Defaults: ambari !requiretty
```

To re-iterate, this sudo configuration must be done on every node in the cluster. To ensure that the configuration has been done properly, you can su to the ambari user and run sudo -l. There, you can double-check that there are no warnings, and that the configuration output matches what was just applied.

2.5.2. Optional: Ambari Web Inactivity Timeout

Ambari is capable of automatically logging a user out of Ambari Web after a period of inactivity. After a configurable amount of time, the user's session will be terminated and they will be redirected to the login page.

This capability can be separately configured for Operators and Read-Only users. This allows you to distinguish a read-only user (useful when Ambari Web is used as a monitoring dashboard) from other operators. Alternatively, you can set both inactivity timeout values to be the same so that regardless of the user type, automatic logout will occur after a set period of time.

By default, the Ambari Web inactivity timeout is not enabled (i.e. is set to 0). The following instructions should be used to enable inactivity timeout and set as the amount of time in seconds before users are automatically logged out.

Ensure the Ambari Server is completely stopped before making changes to the inactivity timeout. Either make these changes before you start Ambari Server the first time, or bring the server down before making these changes.

1. On the Ambari Server host, open

```
/etc/ambari-server/conf/ambari.properties with a text editor.
```

2. There are two properties for the inactivity timeout setting. Both are initially set to 0 (which means this capability is disabled).

Property	Description
user.inactivity.timeout.default	Sets the inactivity timeout (in seconds) for all users except Read-Only users.
user.inactivity.timeout.role.readonly.default	Sets the inactivity timeout (in seconds) for all Read-Only users.

3. Modify the values to enable the capability. The values are in seconds.
4. Save changes and restart Ambari Server.
5. After a user logs into Ambari Web, once a period of inactivity occurs, the user will be presented with an Automatic Logout dialog 60 seconds from logout. The user can click to remain logged in or if no activity occurs, Ambari Web will automatically log the user out and redirect the application to the login page.

2.5.3. Optional: Set Up Kerberos for Ambari Server



Note

This section describes how to set up and configure Ambari Server to have a Kerberos principal and keytab. If you performed the [Automated Kerberos Setup](#), these steps are performed automatically (and therefore, you do not need to perform the steps below). If you performed the [Manual Kerberos Setup](#), perform the steps below as well.

When a cluster is enabled for Kerberos, the component REST endpoints (such as the YARN ATS component) require [Enabling SPNEGO Authentication for Hadoop \[74\]](#) authentication.

Depending on the Services in your cluster, Ambari Web needs access to these APIs. As well, views such as the [Tez View](#) need access to ATS. Therefore, the Ambari Server requires a Kerberos principal in order to authenticate via SPNEGO against these APIs. This section describes how to configure Ambari Server with a Kerberos principal and keytab to allow views to authenticate via SPNEGO against cluster components.

1. Create a principal in your KDC for the Ambari Server. For example, using kadmin:

```
addprinc -randkey ambari-server@EXAMPLE.COM
```

2. Generate a keytab for that principal.

```
xst -k ambari.server.keytab ambari-server@EXAMPLE.COM
```

3. Place that keytab on the Ambari Server host. Be sure to set the file permissions so the user running the Ambari Server daemon can access the keytab file.

```
/etc/security/keytabs/ambari.server.keytab
```

4. Stop the ambari server.

```
ambari-server stop
```

5. Run the setup-security command.

```
ambari-server setup-security
```

6. Select 3 for Setup Ambari kerberos JAAS configuration.

7. Enter the Kerberos principal name for the Ambari Server you set up earlier.
8. Enter the path to the keytab for the Ambari principal.
9. Restart Ambari Server.

```
ambari-server restart
```

2.5.4. Optional: Set Up Two-Way SSL Between Ambari Server and Ambari Agents

Two-way SSL provides a way to encrypt communication between Ambari Server and Ambari Agents. By default Ambari ships with Two-way SSL disabled. To enable Two-way SSL:

Ambari Server should not be running when you do this: either make the edits before you start Ambari Server the first time or bring the server down to make the edits.

1. On the Ambari Server host, open `/etc/ambari-server/conf/ambari.properties` with a text editor.
2. Add the following property:

```
security.server.two_way_ssl = true
```

3. Start or restart the Ambari Server.

```
ambari-server restart
```

The Agent certificates are downloaded automatically during Agent Registration.

2.5.5. Optional: Configure Ciphers and Protocols for Ambari Server

Ambari provides control of ciphers and protocols that are exposed via Ambari Server.

1. To disable specific ciphers, you can optionally add a list of the following format to `ambari.properties`. If you specify multiple ciphers, separate each cipher using a vertical bar `|`.

```
security.server.disabled.ciphers=TLS_ECDHE_RSA_WITH_3DES_EDE_CBC_SHA
```

2. To disable specific protocols, you can optionally add a list of the following format to `ambari.properties`. If you specify multiple protocols, separate each protocol using a vertical bar `|`.

```
security.server.disabled.protocols=SSL|SSLv2|SSLv3
```

2.5.6. Optional: HTTP Cookie Persistence

During HTTP authentication, a cookie is dropped. This is a persistent cookie that is valid across browser sessions. For clusters that require enhanced security, it is desirable to have a session cookie that gets deleted when the user closes the browser session.

In HDP-2.3.4 and higher versions, you can use the following property in the `etc/hadoop/conf/core-site.xml` file to specify cookie persistence across browser sessions.

```
<property>
  <name>hadoop.http.authentication.cookie.persistent</name>
  <value>true</value>
</property>
```

The default value for this property is `false`.

2.6. Enabling SPNEGO Authentication for Hadoop

By default, access to the HTTP-based services and UI's for the cluster are not configured to require authentication. Kerberos authentication can be configured for the Web UIs for HDFS, YARN, MapReduce2, HBase, Oozie, Falcon and Storm.

- [Configure Ambari Server for Authenticated HTTP \[74\]](#)
- [Configuring HTTP Authentication for HDFS, YARN, MapReduce2, HBase, Oozie, Falcon and Storm \[74\]](#)

2.6.1. Configure Ambari Server for Authenticated HTTP

In order for Ambari to work with a cluster in which authenticated HTTP access to the Web UIs is required, you must configure the Ambari Server for Kerberos. Refer to [Optional: Set Up Kerberos for Ambari Server \[72\]](#) for more information.

2.6.2. Configuring HTTP Authentication for HDFS, YARN, MapReduce2, HBase, Oozie, Falcon and Storm

1. Create a secret key used for signing authentication tokens. This file should contain random data and be placed on every host in the cluster. It should also be owned by the `hdfs` user and group owned by the `hadoop` group. Permissions should be set to 440. For example:

```
dd if=/dev/urandom of=/etc/security/http_secret bs=1024 count=1
chown hdfs:hadoop /etc/security/http_secret
chmod 440 /etc/security/http_secret
```

2. In Ambari Web, browse to **Services > HDFS > Configs**.
3. Add or modify the following configuration properties to Advanced core-site.

Property	New Value
<code>hadoop.http.authentication.simple.anonymous.allowed</code>	<code>false</code>
<code>hadoop.http.authentication.signature.secret.file</code>	<code>/etc/security/http_secret</code>
<code>hadoop.http.authentication.type</code>	<code>kerberos</code>
<code>hadoop.http.authentication.kerberos.keytab</code>	<code>/etc/security/keytabs/spnego.service.keytab</code>
<code>hadoop.http.authentication.kerberos.principal</code>	<code>HTTP/_HOST@EXAMPLE.COM</code>

Property	New Value
hadoop.http.filter.initializers	org.apache.hadoop.security.AuthenticationFilterInitializer
hadoop.http.authentication.cookie.domain	hortonworks.local



Important

The entries listed in the above table in **bold** and italicized are site-specific. The `hadoop.http.authentication.cookie.domain` property is based off of the fully qualified domain names of the servers in the cluster. For example if the FQDN of your NameNode is `host1.hortonworks.local`, the `hadoop.http.authentication.cookie.domain` should be set to `hortonworks.local`.

- For HBase, you can enable Kerberos-authentication to HBase Web UIs by configuring SPNEGO.
 - In Ambari Web, browse to **Services > HBase > Configs**.
 - Add the following configuration properties to the custom `hbase-site.xml` file.

Property	Value
hbase.security.authentication.ui	kerberos
hbase.security.authentication	kerberos
hbase.security.authentication.spnego.kerberos.principal	HTTP/_HOST@EXAMPLE.COM
hbase.security.authentication.spnego.kerberos.keytab	/etc/security/keytabs/spnego.service.keytab
Hbase.security.authentication.spnego.kerberos.name.rules (Optional)	
Hbase.security.authentication.signature.secret.file(Optional)	

- Save the configuration, then restart the affected services.

2.6.3. Enabling Browser Access to a SPNEGO-enabled Web UI

- Install Kerberos on your local machine (search for instructions on how to install a Kerberos client on your local environment).
- Configure the `krb5.conf` file on your local machine. For testing on a HDP cluster, copy the `/etc/krb5.conf` file from one of the cluster hosts to your local machine at `/etc/krb5.conf`.
- Create your own keytabs and run `kinit`. For testing on a HDP cluster, copy the "ambari_qa" keytab file from `/etc/security/keytabs/smokeuser.headless.keytab` on one of the cluster hosts to your local machine, then run the following command:

```
kinit -kt smokeuser.headless.keytab ambari-qa@EXAMPLE.COM
```

- Use the following steps to enable your web browser with Kerberos SPNEGO.

For Chrome on Mac:

Run the following command from the **same shell** in which you ran the previous `kinit` command to launch Chrome:

```
/Applications/Google\ Chrome.app/Contents/MacOS/Google\ Chrome --auth-server-whitelist="*.hwx.site"
```

- Replace `.hwx.site` with your own domain name.
- If you get the following error, try closing and relaunching all Chrome browser windows.

```
[14617:36099:0810/152439.802775:ERROR:browser_gpu_channel_host_factory.cc(103)] Failed to launch GPU process.
```

For FireFox:

- a. Navigate to the `about:config` URL (type `about:config` in the address box, then press the Enter key).
- b. Scroll down to `network.negotiate-auth.trusted-uris` and change its value to your cluster domain name (For example, `.hwx.site`).
- c. Change the value of `network.negotiate-auth.delegation-uris` to your cluster domain name (For example, `.hwx.site`).

2.7. Setting Up Kerberos Authentication for Non-Ambari Clusters

This section provides information for enabling security for a manually installed version of HDP.

- [Preparing Kerberos \[76\]](#)
- [Configuring HDP for Kerberos \[82\]](#)
- [Configuring HBase and ZooKeeper \[101\]](#)
- [Configuring Phoenix Query Server \[107\]](#)
- [Configuring Hue \[108\]](#)
- [Setting up One-Way Trust with Active Directory \[110\]](#)
- [Configuring Proxy Users \[112\]](#)

2.7.1. Preparing Kerberos

This subsection provides information on setting up Kerberos for an HDP installation.

2.7.1.1. Kerberos Overview

To create secure communication among its various components, HDP uses Kerberos. Kerberos is a third-party authentication mechanism, in which users and services that

users wish to access rely on the Kerberos server to authenticate each to the other. This mechanism also supports encrypting all traffic between the user and the service.

The Kerberos server itself is known as the *Key Distribution Center*, or KDC. At a high level, it has three parts:

- A database of users and services (known as *principals*) and their respective Kerberos passwords
- An *authentication server (AS)* which performs the initial authentication and issues a *Ticket Granting Ticket (TGT)*
- A *Ticket Granting Server (TGS)* that issues subsequent service tickets based on the initial TGT.

A user principal requests authentication from the AS. The AS returns a TGT that is encrypted using the user principal's Kerberos password, which is known only to the user principal and the AS. The user principal decrypts the TGT locally using its Kerberos password, and from that point forward, until the ticket expires, the user principal can use the TGT to get service tickets from the TGS.

Because a service principal cannot provide a password each time to decrypt the TGT, it uses a special file, called a *keytab*, which contains its authentication credentials.

The service tickets allow the principal to access various services. The set of hosts, users, and services over which the Kerberos server has control is called a *realm*.



Note

Because Kerberos is a time-sensitive protocol, all hosts in the realm must be time-synchronized, for example, by using the Network Time Protocol (NTP). If the local system time of a client differs from that of the KDC by as little as 5 minutes (the default), the client will not be able to authenticate.

2.7.1.2. Installing and Configuring the KDC

To use Kerberos with HDP, either use an existing KDC or install a new one for HDP only. The following gives a very high level description of the installation process. For more information, see [RHEL documentation](#), [CentOS documentation](#), [SLES documentation](#), or [Ubuntu and Debian documentation](#).

1. Install the KDC server:

- On RHEL, CentOS, or Oracle Linux, run:

```
yum install krb5-server krb5-libs krb5-auth-dialog krb5-workstation
```

- On SLES, run:

```
zypper install krb5 krb5-server krb5-client
```

- On Ubuntu or Debian, run:

```
apt-get install krb5 krb5-server krb5-client
```



Note

The host on which you install the KDC must itself be secure.

2. When the server is installed you must edit the two main configuration files.

Update the KDC configuration by replacing EXAMPLE.COM with your domain and `kerberos.example.com` with the FQDN of the KDC host. Configuration files are in the following locations:

- On RHEL, CentOS, or Oracle Linux:

```
/etc/krb5.conf  
/var/kerberos/krb5kdc/kdc.conf
```

- On SLES:

```
/etc/krb5.conf  
/var/lib/kerberos/krb5kdc/kdc.conf
```

- On Ubuntu or Debian:

```
/etc/krb5.conf  
/var/kerberos/krb5kdc/kdc.conf
```

3. Copy the updated `krb5.conf` to every cluster node.

2.7.1.3. Creating the Database and Setting Up the First Administrator

1. Use the utility `kdb5_util` to create the Kerberos database:

- On RHEL, CentOS, or Oracle Linux:

```
/usr/sbin/kdb5_util create -s
```

- On SLES:

```
kdb5_util create -s
```

- On Ubuntu or Debian:

```
kdb5_util -s create
```



Note

The `-s` option stores the master server key for the database in a stash file. If the stash file is not present, you must log into the KDC with the master password (specified during installation) each time it starts. This will automatically regenerate the master server key.

2. Set up the KDC Access Control List (ACL):

- On RHEL, CentOS, or Oracle Linux add administrators to `/var/kerberos/krb5kdc/kadm5.acl`.
- On SLES, add administrators to `/var/lib/kerberos/krb5kdc/kadm5.acl`.



Note

For example, the following line grants full access to the database for users with the admin extension: `*/admin@EXAMPLE.COM *`

3. Start **kadmin** for the change to take effect.
4. Create the first user principal. This must be done at a terminal window on the KDC machine itself, while you are logged in as root. Notice the `.local`. Normal `kadmin` usage requires that a principal with appropriate access already exist. The `kadmin.local` command can be used even if no principals exist:

```
/usr/sbin/kadmin.local -q "addprinc $username/admin
```

Now this user can create additional principals either on the KDC machine or through the network. The following instruction assumes that you are using the KDC machine.

5. On the KDC, start Kerberos:

- On RHEL, CentOS, or Oracle Linux:

```
/sbin/service krb5kdc start
/sbin/service kadmin start
```

- On SLES:

```
rckrb5kdc start
rckadmind start
```

- On Ubuntu or Debian:

```
/etc/init.d/krb5-kdc start
/etc/init.d/kadmin start
```

2.7.1.4. Creating Service Principals and Keytab Files for HDP

Each service in HDP must have its own principal. Because services do not login with a password to acquire their tickets, their principal's authentication credentials are stored in a keytab file, which is extracted from the Kerberos database and stored locally with the service principal.

First create the principal, using mandatory naming conventions. Then create the keytab file with that principal's information, and copy the file to the keytab directory on the appropriate service host.

1. To create a service principal you will use the `kadmin` utility. This is a command-line driven utility into which you enter Kerberos commands to manipulate the central database. To start `kadmin`, enter:

```
'kadmin $USER/admin@REALM'
```

To create a service principal, enter the following:

```
kadmin: addprinc -randkey $principal_name/$service-host-FQDN@$hadoop.realm
```

You must have a principal with administrative permissions to use this command. The `randkey` is used to generate the password.

The `$principal_name` part of the name must match the values in the following table.

In the example each service principal's name has appended to it the fully qualified domain name of the host on which it is running. This is to provide a unique principal name for services that run on multiple hosts, like DataNodes and TaskTrackers. The addition of the hostname serves to distinguish, for example, a request from DataNode A from a request from DataNode B.

This is important for two reasons:

- a. If the Kerberos credentials for one DataNode are compromised, it does not automatically lead to all DataNodes being compromised
- b. If multiple DataNodes have exactly the same principal and are simultaneously connecting to the NameNode, and if the Kerberos authenticator being sent happens to have same timestamp, then the authentication would be rejected as a replay request.

Note: The NameNode, Secondary NameNode, and Oozie require two principals each.

If you are configuring High Availability (HA) for a Quorum-based NameNode, you must also generate a principle (`jn/$FQDN`) and keytab (`jn.service.keytab`) for each JournalNode. JournalNode also requires the keytab for its HTTP service. If the JournalNode is deployed on the same host as a NameNode, the same keytab file (`spnego.service.keytab`) can be used for both. In addition, HA requires two NameNodes. Both the active and standby NameNodes require their own principle and keytab files. The service principles of the two NameNodes can share the same name, specified with the `dfs.namenode.kerberos.principal` property in `hdfs-site.xml`, but the NameNodes still have different fully qualified domain names.

Table 2.8. Service Principals

Service	Component	Mandatory Principal Name
HDFS	NameNode	<code>nn/\$FQDN</code>
HDFS	NameNode HTTP	<code>HTTP/\$FQDN</code>
HDFS	SecondaryNameNode	<code>nn/\$FQDN</code>
HDFS	SecondaryNameNode HTTP	<code>HTTP/\$FQDN</code>
HDFS	DataNode	<code>dn/\$FQDN</code>
MR2	History Server	<code>jhs/\$FQDN</code>
MR2	History Server HTTP	<code>HTTP/\$FQDN</code>
YARN	ResourceManager	<code>rm/\$FQDN</code>
YARN	NodeManager	<code>nm/\$FQDN</code>
Oozie	Oozie Server	<code>oozie/\$FQDN</code>
Oozie	Oozie HTTP	<code>HTTP/\$FQDN</code>
Hive	Hive Metastore	<code>hive/\$FQDN</code>
	HiveServer2	
Hive	WebHCat	<code>HTTP/\$FQDN</code>

Service	Component	Mandatory Principal Name
HBase	MasterServer	hbase/\$FQDN
HBase	RegionServer	hbase/\$FQDN
Storm	Nimbus server	nimbus/\$FQDN **
	DRPC daemon	
Storm	Storm UI daemon	storm/\$FQDN **
	Storm Logviewer daemon	
	Nodes running process controller (such as Supervisor)	
Kafka	KafkaServer	kafka/\$FQDN
Zeppelin	Zeppelin Server	zeppelin/\$FQDN
Hue	Hue Interface	hue/\$FQDN
ZooKeeper	ZooKeeper	zookeeper/\$FQDN
JournalNode Server*	JournalNode	jn/\$FQDN
Gateway	Knox	knox/\$FQDN

* Only required if you are setting up NameNode HA.

** For more information, see [Configure Kerberos Authentication for Storm](#).

For example: To create the principal for a DataNode service, issue this command:

```
kadmin: addprinc -randkey dn/$datanode-host@$hadoop.realm
```

2. Extract the related keytab file and place it in the keytab directory of the appropriate respective components. The default directory is `/etc/krb5.keytab`.

```
kadmin: xst -k $keytab_file_name $principal_name/fully.qualified.domain.name
```

You must use the mandatory names for the `$keytab_file_name` variable shown in the following table.

Table 2.9. Service Keytab File Names

Component	Principal Name	Mandatory Keytab File Name
NameNode	nn/\$FQDN	nn.service.keytab
NameNode HTTP	HTTP/\$FQDN	spnego.service.keytab
SecondaryNameNode	nn/\$FQDN	nn.service.keytab
SecondaryNameNode HTTP	HTTP/\$FQDN	spnego.service.keytab
DataNode	dn/\$FQDN	dn.service.keytab
MR2 History Server	jhs/\$FQDN	nm.service.keytab
MR2 History Server HTTP	HTTP/\$FQDN	spnego.service.keytab
YARN	rm/\$FQDN	rm.service.keytab
YARN	nm/\$FQDN	nm.service.keytab
Oozie Server	oozie/\$FQDN	oozie.service.keytab
Oozie HTTP	HTTP/\$FQDN	spnego.service.keytab
Hive Metastore	hive/\$FQDN	hive.service.keytab
HiveServer2		

Component	Principal Name	Mandatory Keytab File Name
WebHCat	HTTP/\$FQDN	spnego.service.keytab
HBase Master Server	hbase/\$FQDN	hbase.service.keytab
HBase RegionServer	hbase/\$FQDN	hbase.service.keytab
Storm	storm/\$FQDN	storm.service.keytab
Kafka	kafka/\$FQDN	kafka.service.keytab
Zeppelin Server	zeppelin/\$FQDN	zeppelin.server.kerberos.keytab
Hue	hue/\$FQDN	hue.service.keytab
ZooKeeper	zookeeper/\$FQDN	zk.service.keytab
Journal Server*	jn/\$FQDN	jn.service.keytab
Knox Gateway**	knox/\$FQDN	knox.service.keytab

* Only required if you are setting up NameNode HA.

** Only required if you are using a Knox Gateway.

For example: To create the keytab files for the NameNode, issue these commands:

```
kadmin: xst -k nn.service.keytab nn/$namenode-host kadmin: xst -k spnego.
service.keytab HTTP/$namenode-host
```

When you have created the keytab files, copy them to the keytab directory of the respective service hosts.

3. Verify that the correct keytab files and principals are associated with the correct service using the klist command. For example, on the NameNode:

```
klist -k -t /etc/security/nn.service.keytab
```

Do this on each respective service in your cluster.

2.7.2. Configuring HDP for Kerberos

Configuring HDP for Kerberos has two parts:

- Creating a mapping between service principals and UNIX usernames.

Hadoop uses group memberships of users at various places, such as to determine group ownership for files or for access control.

A user is mapped to the groups it belongs to using an implementation of the GroupMappingServiceProvider interface. The implementation is pluggable and is configured in `core-site.xml`.

By default Hadoop uses ShellBasedUnixGroupsMapping, which is an implementation of GroupMappingServiceProvider. It fetches the group membership for a username by executing a UNIX shell command. In secure clusters, since the usernames are actually Kerberos principals, ShellBasedUnixGroupsMapping will work only if the Kerberos principals map to valid UNIX usernames. Hadoop provides a feature that lets administrators specify mapping rules to map a Kerberos principal to a local UNIX username.

- Adding information to three main service configuration files.

There are several optional entries in the three main service configuration files that must be added to enable security on HDP.

This section provides information on configuring HDP for Kerberos.

- [Creating Mappings Between Principals and UNIX Usernames \[83\]](#)
- [Adding Security Information to Configuration Files \[84\]](#)
- [Configuring HBase and ZooKeeper \[101\]](#)
- [Configuring Hue \[108\]](#)



Note

You must adhere to the existing upper and lower case naming conventions in the configuration file templates.

2.7.2.1. Creating Mappings Between Principals and UNIX Usernames

HDP uses a rule-based system to create mappings between service principals and their related UNIX usernames. The rules are specified in the `core-site.xml` configuration file as the value to the optional key `hadoop.security.auth_to_local`.

The default rule is simply named `DEFAULT`. It translates all principals in your default domain to their first component. For example, `myusername@APACHE.ORG` and `myusername/admin@APACHE.ORG` both become `myusername`, assuming your default domain is `APACHE.ORG`.

While mapping the Kerberos principals, if the Kerberos principal names are in the `UPPERCASE` or `CaMeLcase`, the names will not be recognized on the Linux machine (as Linux users are always in lower case). You must add the extra switch `"/L"` in the rule definition to force the conversion to lower case.

Creating Rules

To accommodate more complex translations, you can create a hierarchical set of rules to add to the default. Each rule is divided into three parts: base, filter, and substitution.

- **The Base**

The base begins with the number of components in the principal name (excluding the realm), followed by a colon, and the pattern for building the username from the sections of the principal name. In the pattern section `$0` translates to the realm, `$1` translates to the first component, and `$2` to the second component.

For example:

```
[1:$1@$0] translates myusername@APACHE.ORG to myusername@APACHE.ORG
[2:$1] translates myusername/admin@APACHE.ORG to myusername
[2:$1%$2] translates myusername/admin@APACHE.ORG to "myusername%admin"
```

- **The Filter**

The filter consists of a regular expression (regex) in a parentheses. It must match the generated string for the rule to apply.

For example:

```
(.*%admin) matches any string that ends in %admin
(*@SOME.DOMAIN) matches any string that ends in @SOME.DOMAIN
```

- **The Substitution**

The substitution is a sed rule that translates a regex into a fixed string. For example:

```
s/@ACME\.COM// removes the first instance of @ACME.DOMAIN
s/[A-Z]*\\.COM// remove the first instance of @ followed by a name followed
by COM.
s/X/Y/g replace all of X's in the name with Y
```

2.7.2.1.1. Examples

- If your default realm was APACHE.ORG, but you also wanted to take all principals from ACME.COM that had a single component joe@ACME.COM, the following rule would do this:

```
RULE:[1:$1@$0](.@ACME.COM)s/@.//
DEFAULT
```

- To translate names with a second component, you could use these rules:

```
RULE:[1:$1@$0](.@ACME.COM)s/@.//
RULE:[2:$1@$0](.@ACME.COM)s/@.// DEFAULT
```

- To treat all principals from APACHE.ORG with the extension /admin as admin, your rules would look like this:

```
RULE[2:$1%$2@$0](.%admin@APACHE.ORG)s/admin/
DEFAULT
```

- To force username conversion from CaMeLcase or UPPERCASE to lowercase, you could model the following auth_to_local rule examples which have the lowercase switch added:

```
RULE:[1:$1]/L
RULE:[2:$1]/L
RULE:[2:$1;$2](^.*;admin$)s/;admin$//L
RULE:[2:$1;$2](^.*;guest$)s/;guest$//g/L
```

And based on these rules, here are the expected output for the following inputs:

```
"JOE@FOO.COM" to "joe"
"Joe/root@FOO.COM" to "joe"
"Joe/admin@FOO.COM" to "joe"
"Joe/guestguest@FOO.COM" to "joe"
```

2.7.2.2. Adding Security Information to Configuration Files

To enable security on HDP, you must add optional information to various configuration files.

Before you begin, set JSVC_Home in `hadoop-env.sh`.

- For RHEL/CentOS/Oracle Linux:

```
export JSVC_HOME=/usr/libexec/bigtop-utils
```

- For SLES and Ubuntu:

```
export JSVC_HOME=/usr/hdp/current/bigtop-utils
```

2.7.2.2.1. core-site.xml

Add the following information to the `core-site.xml` file on every host in your cluster:

Table 2.10. General core-site.xml, Knox, and Hue

Property Name	Property Value	Description
<code>hadoop.security.authentication</code>	<code>kerberos</code>	Set the authentication type for the cluster. Valid values are: simple or kerberos.
<code>hadoop.rpc.protection</code>	<code>authentication; integrity; privacy</code>	This is an [OPTIONAL] setting. If not set, defaults to authentication. authentication = authentication only; the client and server mutually authenticate during connection setup. integrity = authentication and integrity; guarantees the integrity of data exchanged between client and server as well as authentication. privacy = authentication, integrity, and confidentiality; guarantees that data exchanged between client and server is encrypted and is not readable by a "man in the middle".
<code>hadoop.security.authorization</code>	<code>true</code>	Enable authorization for different protocols.
<code>hadoop.security.auth_to_local</code>	The mapping rules. For example: RULE: [2:\$1@\$0] ([jt]t@.*EXAMPLE.COM)s/./ */ mapred/ RULE:[2:\$1@\$0] ([nd]n@.*EXAMPLE.COM)s/./ */ hdfs/ RULE:[2:\$1@\$0] (hm@.*EXAMPLE.COM)s/./ */ hbase/ RULE:[2:\$1@\$0] (rs@.*EXAMPLE.COM)s/./ */ hbase/ DEFAULT	The mapping from Kerberos principal names to local OS user names. See Creating Mappings Between Principals and UNIX Usernames for more information.

Following is the XML for these entries:

```
<property>
  <name>hadoop.security.authentication</name>
  <value>kerberos</value>
  <description> Set the authentication for the cluster.
  Valid values are: simple or kerberos.</description>
</property>

<property>
  <name>hadoop.security.authorization</name>
  <value>true</value>
```

```

    <description>Enable authorization for different protocols.</description>
  </property>

  <property>
    <name>hadoop.security.auth_to_local</name>
    <value>
      RULE:[2:$1@$0]([jt]t@.*EXAMPLE.COM)s/.* /mapred/
      RULE:[2:$1@$0]([nd]n@.*EXAMPLE.COM)s/.* /hdfs/
      RULE:[2:$1@$0](hm@.*EXAMPLE.COM)s/.* /hbase/
      RULE:[2:$1@$0](rs@.*EXAMPLE.COM)s/.* /hbase/
      DEFAULT
    </value>
    <description>The mapping from kerberos principal names
      to local OS user names.</description>
  </property>

```



Note

Phoenix Query Server sites: See [Configuring Phoenix Query Server](#) for an additional required property setting in the `core-site.xml` file to complete Kerberos security configuration.

When using the Knox Gateway, add the following to the `core-site.xml` file on the master nodes host in your cluster:

Table 2.11. core-site.xml Master Node Settings – Knox Gateway

Property Name	Property Value	Description
hadoop.proxyuser.knox.groups	users	Grants proxy privileges for Knox user.
hadoop.proxyuser.knox.hosts	\$knox_host_FQDN	Identifies the Knox Gateway host.

When using Hue, add the following to the `core-site.xml` file on the master nodes host in your cluster:

Table 2.12. core-site.xml Master Node Settings – Hue

Property Name	Property Value	Description
hue.kerberos.principal.shortname	hue	Group to which all the Hue users belong. Use the wild card character to select multiple groups, for example cli*.
hadoop.proxyuser.hue.groups	*	Group to which all the Hue users belong. Use the wild card character to select multiple groups, for example cli*.
hadoop.proxyuser.hue.hosts	*	
hadoop.proxyuser.knox.hosts	\$hue_host_FQDN	Identifies the Knox Gateway host.

Following is the XML for both Knox and Hue settings:

```

<property>
  <name>hadoop.security.authentication</name>
  <value>kerberos</value>
  <description>Set the authentication for the cluster.
    Valid values are: simple or kerberos.</description>
</property>

<property>
  <name>hadoop.security.authorization</name>

```

```

    <value>true</value>
    <description>Enable authorization for different protocols.
    </description>
  </property>

  <property>
    <name>hadoop.security.auth_to_local</name>
    <value>
      RULE:[2:$1@$0]([jt]t@.*EXAMPLE.COM)s/./mapred/
      RULE:[2:$1@$0]([nd]n@.*EXAMPLE.COM)s/./hdfs/
      RULE:[2:$1@$0](hm@.*EXAMPLE.COM)s/./hbase/
      RULE:[2:$1@$0](rs@.*EXAMPLE.COM)s/./hbase/
      DEFAULT
    </value>
    <description>The mapping from kerberos principal names
    to local OS user names.</description>
  </property>

  <property>
    <name>hadoop.proxyuser.knox.groups</name>
    <value>users</value>
  </property>

  <property>
    <name>hadoop.proxyuser.knox.hosts</name>
    <value>Knox.EXAMPLE.COM</value>
  </property>

```

2.7.2.2.1.1. HTTP Cookie Persistence

During HTTP authentication, a cookie is dropped. This is a persistent cookie that is valid across browser sessions. For clusters that require enhanced security, it is desirable to have a session cookie that gets deleted when the user closes the browser session.

You can use the following `core-site.xml` property to specify cookie persistence across browser sessions.

```

<property>
  <name>hadoop.http.authentication.cookie.persistent</name>
  <value>true</value>
</property>

```

The default value for this property is `false`.

2.7.2.2.2. hdfs-site.xml

To the `hdfs-site.xml` file on every host in your cluster, you must add the following information:

Table 2.13. hdfs-site.xml File Property Settings

Property Name	Property Value	Description
<code>dfs.permissions.enabled</code>	<code>true</code>	If true, permission checking in HDFS is enabled. If false, permission checking is turned off, but all other behavior is unchanged. Switching from one parameter value to the other does not change the mode, owner or group of files or directories.
<code>dfs.permissions.supergroup</code>	<code>hdfs</code>	The name of the group of super-users.

Property Name	Property Value	Description
dfs.block.access.token.enable	true	If true, access tokens are used as capabilities for accessing DataNodes. If false, no access tokens are checked on accessing DataNodes.
dfs.namenode.kerberos.principal	nn/_HOST@EXAMPLE.COM	Kerberos principal name for the NameNode.
dfs.secondary.namenode.kerberos.principal	nn/_HOST@EXAMPLE.COM	Kerberos principal name for the secondary NameNode.
dfs.web.authentication.kerberos.principal	HTTP/_HOST@EXAMPLE.COM	The HTTP Kerberos principal used by Hadoop-Auth in the HTTP endpoint. The HTTP Kerberos principal MUST start with 'HTTP/' per Kerberos HTTP SPNEGO specification.
dfs.web.authentication.kerberos.keytab	/etc/security/keytabs/spnego.service.keytab	The Kerberos keytab file with the credentials for the HTTP Kerberos principal used by Hadoop-Auth in the HTTP endpoint.
dfs.datanode.kerberos.principal	dn/_HOST@EXAMPLE.COM	The Kerberos principal that the DataNode runs as. "_HOST" is replaced by the real host name.
dfs.namenode.keytab.file	/etc/security/keytabs/nn.service.keytab	Combined keytab file containing the NameNode service and host principals.
dfs.secondary.namenode.keytab.file	/etc/security/keytabs/nn.service.keytab	Combined keytab file containing the NameNode service and host principals. <question?>
dfs.datanode.keytab.file	/etc/security/keytabs/dn.service.keytab	The filename of the keytab file for the DataNode.
dfs.https.port	50470	The HTTPS port to which the NameNode binds.
dfs.namenode.https-address	Example: ip-10-111-59-170.ec2.internal:50470	The HTTPS address to which the NameNode binds.
dfs.datanode.data.dir.perm	750	The permissions that must be set on the dfs.data.dir directories. The DataNode will not come up if all existing dfs.data.dir directories do not have this setting. If the directories do not exist, they will be created with this permission.
dfs.cluster.administrators	hdfs	ACL for who all can view the default servlets in the HDFS.
dfs.namenode.kerberos.internal.spnego.principal	\${dfs.web.authentication.kerberos.principal}	
dfs.secondary.namenode.kerberos.internal.spnego.principal	\${dfs.web.authentication.kerberos.principal}	

Following is the XML for these entries:

```
<property>
  <name>dfs.permissions</name>
  <value>true</value>
  <description> If "true", enable permission checking in
HDFS. If "false", permission checking is turned
off, but all other behavior is
unchanged. Switching from one parameter value to the other does
not change the mode, owner or group of files or
directories. </description>
```

```
</property>

<property>
  <name>dfs.permissions.supergroup</name>
  <value>hdfs</value>
  <description>The name of the group of
  super-users.</description>
</property>

<property>
  <name>dfs.namenode.handler.count</name>
  <value>100</value>
  <description>Added to grow Queue size so that more
  client connections are allowed</description>
</property>

<property>
  <name>ipc.server.max.response.size</name>
  <value>5242880</value>
</property>

<property>
  <name>dfs.block.access.token.enable</name>
  <value>true</value>
  <description> If "true", access tokens are used as capabilities
  for accessing datanodes. If "false", no access tokens are checked on
  accessing datanodes. </description>
</property>

<property>
  <name>dfs.namenode.kerberos.principal</name>
  <value>nn/_HOST@EXAMPLE.COM</value>
  <description> Kerberos principal name for the
  NameNode </description>
</property>

<property>
  <name>dfs.secondary.namenode.kerberos.principal</name>
  <value>nn/_HOST@EXAMPLE.COM</value>
  <description>Kerberos principal name for the secondary NameNode.
  </description>
</property>

<property>
  <!--cluster variant -->
  <name>dfs.secondary.http.address</name>
  <value>ip-10-72-235-178.ec2.internal:50090</value>
  <description>Address of secondary namenode web server</description>
</property>

<property>
  <name>dfs.secondary.https.port</name>
  <value>50490</value>
  <description>The https port where secondary-namenode
  binds</description>
</property>

<property>
  <name>dfs.web.authentication.kerberos.principal</name>
  <value>HTTP/_HOST@EXAMPLE.COM</value>
```

```
<description> The HTTP Kerberos principal used by Hadoop-Auth in the HTTP
endpoint.
The HTTP Kerberos principal MUST start with 'HTTP/' per Kerberos HTTP
SPNEGO specification.
</description>
</property>

<property>
  <name>dfs.web.authentication.kerberos.keytab</name>
  <value>/etc/security/keytabs/spnego.service.keytab</value>
  <description>The Kerberos keytab file with the credentials for the HTTP
Kerberos principal used by Hadoop-Auth in the HTTP endpoint.
  </description>
</property>

<property>
  <name>dfs.datanode.kerberos.principal</name>
  <value>dn/_HOST@EXAMPLE.COM</value>
  <description>
The Kerberos principal that the DataNode runs as. "_HOST" is replaced by
the real
host name.
  </description>
</property>

<property>
  <name>dfs.namenode.keytab.file</name>
  <value>/etc/security/keytabs/nn.service.keytab</value>
  <description>
Combined keytab file containing the namenode service and host
principals.
  </description>
</property>

<property>
  <name>dfs.secondary.namenode.keytab.file</name>
  <value>/etc/security/keytabs/nn.service.keytab</value>
  <description>
Combined keytab file containing the namenode service and host
principals.
  </description>
</property>

<property>
  <name>dfs.datanode.keytab.file</name>
  <value>/etc/security/keytabs/dn.service.keytab</value>
  <description>
The filename of the keytab file for the DataNode.
  </description>
</property>

<property>
  <name>dfs.https.port</name>
  <value>50470</value>
  <description>The https port where namenode
binds</description>
</property>

<property>
  <name>dfs.https.address</name>
```

```
<value>ip-10-111-59-170.ec2.internal:50470</value>
<description>The https address where namenode binds</description>
</property>

<property>
  <name>dfs.datanode.data.dir.perm</name>
  <value>750</value>
  <description>The permissions that should be there on
  dfs.data.dir directories. The datanode will not come up if the
  permissions are different on existing dfs.data.dir directories. If
  the directories don't exist, they will be created with this
  permission.</description>
</property>

<property>
  <name>dfs.access.time.precision</name>
  <value>0</value>
  <description>The access time for HDFS file is precise upto this
  value.The default value is 1 hour. Setting a value of 0
  disables access times for HDFS.
  </description>
</property>

<property>
  <name>dfs.cluster.administrators</name>
  <value> hdfs</value>
  <description>ACL for who all can view the default
  servlets in the HDFS</description>
</property>

<property>
  <name>ipc.server.read.threadpool.size</name>
  <value>5</value>
  <description></description>
</property>

<property>
  <name>dfs.namenode.kerberos.internal.spnego.principal</name>
  <value>${dfs.web.authentication.kerberos.principal}</value>
</property>

<property>
  <name>dfs.secondary.namenode.kerberos.internal.spnego.principal</name>
  <value>${dfs.web.authentication.kerberos.principal}</value>
</property>
```

In addition, you must set the user on all secure DataNodes:

```
export HADOOP_SECURE_DN_USER=hdfs
export HADOOP_SECURE_DN_PID_DIR=/grid/0/var/run/hadoop/$HADOOP_SECURE_DN_USER
```

2.7.2.2.3. yarn-site.xml

You must add the following information to the `yarn-site.xml` file on every host in your cluster:

Table 2.14. yarn-site.xml Property Settings

Property	Value	Description
yarn.resourcemanager.principal	yarn/localhost@EXAMPLE.COM	The Kerberos principal for the ResourceManager.
yarn.resourcemanager.keytab	/etc/krb5.keytab	The keytab for the ResourceManager.
yarn.nodemanager.principal	yarn/localhost@EXAMPLE.COM	The Kerberos principal for the NodeManager.
yarn.nodemanager.keytab	/etc/krb5.keytab	The keytab for the NodeManager.
yarn.nodemanager.container-executor.class	org.apache.hadoop.yarn.server.nodemanager.LinuxContainerExecutor	The class that will execute (launch) the containers.
yarn.nodemanager.linux-container-executor.path	hadoop-3.0.0-SNAPSHOT/bin/container-executor	The path to the Linux container executor.
yarn.nodemanager.linux-container-executor.group	hadoop	A special group (e.g., hadoop) with executable permissions for the container executor, of which the NodeManager UNIX user is the group member and no ordinary application user is. If any application user belongs to this special group, security will be compromised. This special group name should be specified for the configuration property.
yarn.timeline-service.principal	yarn/localhost@EXAMPLE.COM	The Kerberos principal for the Timeline Server.
yarn.timeline-service.keytab	/etc/krb5.keytab	The Kerberos keytab for the Timeline Server.
yarn.resourcemanager.webapp.delegation-token-auth-filter.enabled	true	Flag to enable override of the default Kerberos authentication filter with the RM authentication filter to allow authentication using delegation tokens (fallback to Kerberos if the tokens are missing). Only applicable when the http authentication type is Kerberos.
yarn.timeline-service.http-authentication.type	kerberos	Defines authentication used for the Timeline Server HTTP endpoint. Supported values are: simple kerberos \$AUTHENTICATION_HANDLER_CLASSNAME
yarn.timeline-service.http-authentication.kerberos.principal	HTTP/localhost@EXAMPLE.COM	The Kerberos principal to be used for the Timeline Server HTTP endpoint.
yarn.timeline-service.http-authentication.kerberos.keytab	authentication.kerberos.keytab /etc/krb5.keytab	The Kerberos keytab to be used for the Timeline Server HTTP endpoint.

Following is the XML for these entries:

```
<property>
  <name>yarn.resourcemanager.principal</name>
  <value>yarn/localhost@EXAMPLE.COM</value>
</property>

<property>
  <name>yarn.resourcemanager.keytab</name>
  <value>/etc/krb5.keytab</value>
</property>

<property>
```



```
<name>yarn.nodemanager.principal</name>
<value>yarn/localhost@EXAMPLE.COM</value>
</property>

<property>
  <name>yarn.nodemanager.keytab</name>
  <value>/etc/krb5.keytab</value>
</property>

<property>
  <name>yarn.nodemanager.container-executor.class</name>
  <value>org.apache.hadoop.yarn.server.nodemanager.LinuxContainerExecutor</
value>
</property>

<property>
  <name>yarn.nodemanager.linux-container-executor.path</name>
  <value>hadoop-3.0.0-SNAPSHOT/bin/container-executor</value>
</property>

<property>
  <name>yarn.nodemanager.linux-container-executor.group</name>
  <value>hadoop</value>
</property>

<property>
  <name>yarn.timeline-service.principal</name>
  <value>yarn/localhost@EXAMPLE.COM</value>
</property>

<property>
  <name>yarn.timeline-service.keytab</name>
  <value>/etc/krb5.keytab</value>
</property>

<property>
  <name>yarn.resourcemanager.webapp.delegation-token-auth-filter.enabled</
name>
  <value>>true</value>
</property>

<property>
  <name>yarn.timeline-service.http-authentication.type</name>
  <value>kerberos</value>
</property>

<property>
  <name>yarn.timeline-service.http-authentication.kerberos.principal</name>
  <value>HTTP/localhost@EXAMPLE.COM</value>
</property>

<property>
  <name>yarn.timeline-service.http-authentication.kerberos.keytab</name>
  <value>/etc/krb5.keytab</value>
</property>
```

2.7.2.2.4. mapred-site.xml

You must add the following information to the `mapred-site.xml` file on every host in your cluster:

Table 2.15. mapred-site.xml Property Settings

Property Name	Property Value	Description
mapreduce.jobhistory.keytab	/etc/security/keytabs/jhs.service.keytab	Kerberos keytab file for the MapReduce JobHistory Server.
mapreduce.jobhistory.principal	jhs/_HOST@TODO-KERBEROS-DOMAIN	Kerberos principal name for the MapReduce JobHistory Server.
mapreduce.jobhistory.webapp.address	TODO-JOBHISTORYNODE-HOSTNAME:19888	MapReduce JobHistory Server Web UI host:port
mapreduce.jobhistory.webapp.https.address	TODO-JOBHISTORYNODE-HOSTNAME:19889	MapReduce JobHistory Server HTTPS Web UI host:port
mapreduce.jobhistory.webapp.spnego-keytab-file	/etc/security/keytabs/spnego.service.keytab	Kerberos keytab file for the spnego service.
mapreduce.jobhistory.webapp.spnego-principal	HTTP/_HOST@TODO-KERBEROS-DOMAIN	Kerberos principal name for the spnego service.

Following is the XML for these entries:

```
<property>
  <name>mapreduce.jobhistory.keytab</name>
  <value>/etc/security/keytabs/jhs.service.keytab</value>
</property>

<property>
  <name>mapreduce.jobhistory.principal</name>
  <value>jhs/_HOST@TODO-KERBEROS-DOMAIN</value>
</property>

<property>
  <name>mapreduce.jobhistory.webapp.address</name>
  <value>TODO-JOBHISTORYNODE-HOSTNAME:19888</value>
</property>

<property>
  <name>mapreduce.jobhistory.webapp.https.address</name>
  <value>TODO-JOBHISTORYNODE-HOSTNAME:19889</value>
</property>

<property>
  <name>mapreduce.jobhistory.webapp.spnego-keytab-file</name>
  <value>/etc/security/keytabs/spnego.service.keytab</value>
</property>

<property>
  <name>mapreduce.jobhistory.webapp.spnego-principal</name>
  <value>HTTP/_HOST@TODO-KERBEROS-DOMAIN</value>
</property>
```

2.7.2.2.5. hbase-site.xml

For HBase to run on a secured cluster, HBase must be able to authenticate itself to HDFS. Add the following information to the `hbase-site.xml` file on your HBase server. However, include the `phoenix.queryserver.kerberos.principal` and `phoenix.queryserver.kerberos.keytab` property entries only if you will be configuring Kerberos authentication for a Phoenix Query Server.



Note

There are no default values for the property settings. The entries in the "Sample Setting" column are only examples.

Table 2.16. `hbase-site.xml` Property Settings for HBase Server and Phoenix Query Server

Property Name	Sample Setting	Description
<code>hbase.master.keytab.file</code>	<code>/etc/security/keytabs/hbase.service.keytab</code>	The keytab for the HMaster service principal.
<code>hbase.master.kerberos.principal</code>	<code>hbase/_HOST@EXAMPLE.COM</code>	The Kerberos principal name that should be used to run the HMaster process. If <code>_HOST</code> is used as the hostname portion, it will be replaced with the actual hostname of the running instance.
<code>hbase.regionserver.keytab.file</code>	<code>/etc/security/keytabs/hbase.service.keytab</code>	The keytab for the HRegionServer service principal.
<code>hbase.regionserver.kerberos.principal</code>	<code>hbase/_HOST@EXAMPLE.COM</code>	The Kerberos principal name that should be used to run the HRegionServer process. If <code>_HOST</code> is used as the hostname portion, it will be replaced with the actual hostname of the running instance.
<code>hbase.superuser</code>	<code>hbase</code>	A comma-separated list of users or groups that are allowed full privileges, regardless of stored ACLs, across the cluster. Only used when HBase security is enabled.
<code>hbase.coprocessor.region.classes</code>	<i>Setting 1:</i> <code>org.apache.hadoop.hbase.security.token.TokenProvider,</code> <i>Setting 2:</i> <code>org.apache.hadoop.hbase.security.access.SecureBulkLoadEndpoint,</code> <i>Setting 3:</i> <code>org.apache.hadoop.hbase.security.access.AccessController</code>	A comma-separated list of coprocessors that are loaded by default on all tables. For any implemented coprocessor methods, the listed classes will be called in order. After implementing your own coprocessor, add the class to HBase's classpath and add the fully qualified class name here. Coprocessors can also be loaded programmatically using <code>HTableDescriptor</code> .
<code>hbase.coprocessor.master.classes</code>	<code>org.apache.hadoop.hbase.security.access.AccessController</code>	A comma-separated list of MasterObserver coprocessors that are loaded by the active HMaster process. For any implemented coprocessor methods, the listed classes will be called in order. After implementing your own MasterObserver, add the class to HBase's classpath and add the fully qualified class name here.
<code>hbase.coprocessor.regionserver.classes</code>	<code>org.apache.hadoop.hbase.security.access.AccessController</code>	A comma-separated list of RegionServerObserver coprocessors that are loaded by the HRegionServer processes. For any implemented coprocessor methods, the listed classes will be called in order. After implementing your own RegionServerObserver, add the class to the HBase classpath and fully qualified class name here.

Property Name	Sample Setting	Description
phoenix.queryserver.kerberos.principal	HTTP/_HOST@EXAMPLE.COM	The Kerberos principal for the Phoenix Query Server process. The Phoenix Query Server is an optional component; this property only needs to be set when the query server is installed.
phoenix.queryserver.kerberos.keytab	/etc/security/keytabs/spnego.service.keytab	The path to the Kerberos keytab file for the Phoenix Query Server process. The Phoenix Query Server is an optional component; this property only needs to be set when the query server is installed.



Tip

Phoenix Query Server users: See [Configuring Phoenix Query Server](#) for the required setting in the `core-site.xml` file to complete Kerberos setup of the query server.

The following lists the XML for the `hbase-site.xml` file entries:

```
<property>
  <name>hbase.master.keytab.file</name>
  <value>/etc/security/keytabs/hbase.service.keytab</value>
  <description>Full path to the Kerberos keytab file to use for logging
  in the configured HMaster server principal.
  </description>
</property>

<property>
  <name>hbase.master.kerberos.principal</name>
  <value>hm/_HOST@EXAMPLE.COM</value>
  <description>Ex. "hbase/_HOST@EXAMPLE.COM".
  The Kerberos principal name that should be used to run the HMaster
  process. The principal name should be in the form: user/hostname@DOMAIN.
  If "_HOST" is used as the hostname portion, it will be replaced with
  the actual hostname of the running instance.
  </description>
</property>

<property>
  <name>hbase.regionserver.keytab.file</name>
  <value>/etc/security/keytabs/hbase.service.keytab</value>
  <description>Full path to the kerberos keytab file to use for logging
  in the configured HRegionServer server principal.
  </description>
</property>

<property>
  <name>hbase.regionserver.kerberos.principal</name>
  <value>hbase/_HOST@EXAMPLE.COM</value>
  <description>Ex. "hbase/_HOST@EXAMPLE.COM".
  The kerberos principal name that
  should be used to run the HRegionServer process. The
  principal name should be in the form:
  user/hostname@DOMAIN. If _HOST
  is used as the hostname portion, it will be replaced
  with the actual hostname of the running
  instance. An entry for this principal must exist
```

```
        in the file specified in hbase.regionserver.keytab.file
        </description>
    </property>

<!--Additional configuration specific to HBase security -->

<property>
    <name>hbase.superuser</name>
    <value>hbase</value>
    <description>List of users or groups (comma-separated), who are
    allowed full privileges, regardless of stored ACLs, across the cluster.
    Only used when HBase security is enabled.
    </description>
</property>

<property>
    <name>hbase.coprocessor.region.classes</name>
    <value>org.apache.hadoop.hbase.security.token.TokenProvider,
    org.apache.hadoop.hbase.security.access.SecureBulkLoadEndpoint,
    org.apache.hadoop.hbase.security.access.AccessController</value>
    <description>A comma-separated list of coprocessors that are loaded
    by default on all tables. For any override coprocessor method,
    these classes will be called in order. After implementing your
    own coprocessor, just put it in HBase's classpath and add the
    fully qualified class name here. A coprocessor can also be loaded on
    demand by setting HTableDescriptor.
    </description>
</property>

<property>
    <name>hbase.coprocessor.master.classes</name>
    <value>org.apache.hadoop.hbase.security.access.AccessController</value>
    <description>A comma-separated list of MasterObserver coprocessors that
    are loaded by the active HMaster process. For any implemented coprocessor
    methods, the listed classes will be called in order. After implementing
    your
    own MasterObserver, add the class to HBase's classpath and add the fully
    qualified class name here.
    </description>
</property>

<property>
    <name>hbase.coprocessor.regionserver.classes</name>
    <value>org.apache.hadoop.hbase.security.access.AccessController</value>
    <description>A comma-separated list of RegionServerObserver coprocessors
    that are loaded by the HRegionServer processes. For any implemented
    coprocessor methods, the listed classes will be called in order. After
    implementing your own RegionServerObserver, add the class to the HBase
    classpath and fully qualified class name here.
    </description>
</property>

<property>
    <name>phoenix.queryserver.kerberos.principal</name>
    <value>HTTP/_HOST@EXAMPLE.COM</value>
    <description>The Kerberos principal for the Phoenix Query Server
    process. The Phoenix Query Server is an optional component; this
    property only needs to be set when the query server is installed.
    </description>
</property>
```

```
<property>
  <name>phoenix.queryserver.kerberos.keytab</name>
  <value>/etc/security/keytabs/spnego.service.keytab</value>
  <description>The path to the Kerberos keytab file for the
  Phoenix Query Server process. The Phoenix Query Server is an optional
  component; this property only needs to be set when the query server
  is installed.</description>
</property>
```

2.7.2.2.6. hive-site.xml

HiveServer2 supports Kerberos authentication for all clients.

Add the following information to the `hive-site.xml` file on every host in your cluster:

Table 2.17. hive-site.xml Property Settings

Property Name	Property Value	Description
hive.metastore.sasl.enabled	true	If true, the Metastore Thrift interface will be secured with SASL and clients must authenticate with Kerberos.
hive.metastore.kerberos.keytab.file	/etc/security/keytabs/hive.service.keytab	The keytab for the Metastore Thrift service principal.
hive.metastore.kerberos.principal	hive/_HOST@EXAMPLE.COM	The service principal for the Metastore Thrift server. If _HOST is used as the hostname portion, it will be replaced with the actual hostname of the running instance.

Following is the XML for these entries:

```
<property>
  <name>hive.metastore.sasl.enabled</name>
  <value>true</value>
  <description>If true, the metastore thrift interface will be secured with
  SASL.
  Clients must authenticate with Kerberos.</description>
</property>

<property>
  <name>hive.metastore.kerberos.keytab.file</name>
  <value>/etc/security/keytabs/hive.service.keytab</value>
  <description>The path to the Kerberos Keytab file containing the
  metastore thrift server's service principal.
  </description>
</property>

<property>
  <name>hive.metastore.kerberos.principal</name>
  <value>hive/_HOST@EXAMPLE.COM</value>
  <description>The service principal for the metastore thrift server. The
  special string _HOST will be replaced automatically with the correct
  hostname.</description>
</property>
```

2.7.2.2.7. oozie-site.xml

To the `oozie-site.xml` file, add the following information:

Table 2.18. oozie-site.xml Property Settings

Property Name	Property Value	Description
oozie.service.AuthorizationService.security.enabled	true	Specifies whether security (user name/admin role) is enabled or not. If it is disabled any user can manage the Oozie system and manage any job.
oozie.service.HadoopAccessorService.kerberos.enabled	true	Indicates if Oozie is configured to use Kerberos.
local.realm	EXAMPLE.COM	Kerberos Realm used by Oozie and Hadoop. Using local.realm to be aligned with Hadoop configuration.
oozie.service.HadoopAccessorService.keytab.file	/etc/security/keytabs/oozie.service.keytab	The keytab for the Oozie service principal.
oozie.service.HadoopAccessorService.kerberos.principaloozie/_HOSTI@EXAMPLE.COM	oozie/_HOSTI@EXAMPLE.COM	Kerberos principal for Oozie service.
oozie.authentication.type	kerberos	
oozie.authentication.kerberos.principal	HTTP/_HOST@EXAMPLE.COM	Whitelisted job tracker for Oozie service.
oozie.authentication.kerberos.keytab	/etc/security/keytabs/spnego.service.keytab	Location of the Oozie user keytab file.
oozie.service.HadoopAccessorService.nameNode.whitelist		
oozie.authentication.kerberos.name.rules	RULE:[2:\$1@\$0] ([jt]t@.*EXAMPLE.COM)s/.*/ mapred/ RULE:[2:\$1@\$0] ([nd]n@.*EXAMPLE.COM)s/.*/ hdfs/ RULE:[2:\$1@\$0] (hm@.*EXAMPLE.COM)s/.*/ hbase/ RULE:[2:\$1@\$0] (rs@.*EXAMPLE.COM)s/.*/hbase/ DEFAULT	The mapping from Kerberos principal names to local OS user names. See Creating Mappings Between Principals and UNIX Usernames for more information.
oozie.service.ProxyUserService.proxyuser.knox.groups	users	Grant proxy privileges to the Knox user. Note only required when using a Knox Gateway.
oozie.service.ProxyUserService.proxyuser.knox.hosts	\$knox_host_FQDN	Identifies the Knox Gateway. Note only required when using a Knox Gateway.

2.7.2.2.8. webhcat-site.xml

To the `webhcat-site.xml` file, add the following information:

Table 2.19. webhcat-site.xml Property Settings

Property Name	Property Value	Description
templeton.kerberos.principal	HTTP/_HOST@EXAMPLE.COM	
templeton.kerberos.keytab	/etc/security/keytabs/spnego.service.keytab	
templeton.kerberos.secret	secret	
hadoop.proxyuser.knox.groups	users	Grant proxy privileges to the Knox user. Note only required when using a Knox Gateway.
hadoop.proxyuser.knox.hosts	\$knox_host_FQDN	Identifies the Knox Gateway. Note only required when using a Knox Gateway.

2.7.2.2.9. limits.conf

Adjust the Maximum Number of Open Files and Processes

In a secure cluster, if the DataNodes are started as the root user, JSVC downgrades the processing using setuid to hdfs. However, the ulimit is based on the ulimit of the root user, and the default ulimit values assigned to the root user for the maximum number of open files and processes may be too low for a secure cluster. This can result in a “Too Many Open Files” exception when the DataNodes are started.

Therefore, when configuring a secure cluster you should increase the following root ulimit values:

- nofile: The maximum number of open files. Recommended value: 32768
- nproc: The maximum number of processes. Recommended value: 65536

To set system-wide ulimits to these values, log in as root and add the following lines to the `/etc/security/limits.conf` file on every host in your cluster:

```
* - nofile 32768
* - nproc 65536
```

To set only the root user ulimits to these values, log in as root and add the following lines to the `/etc/security/limits.conf` file.

```
root - nofile 32768
root - nproc 65536
```

You can use the `ulimit -a` command to view the current settings:

```
[root@node-1 /]# ulimit -a
core file size (blocks, -c) 0
data seg size (kbytes, -d) unlimited
scheduling priority (-e) 0
file size (blocks, -f) unlimited
pending signals (-i) 14874
max locked memory (kbytes, -l) 64
max memory size (kbytes, -m) unlimited
open files (-n) 1024
pipe size (512 bytes, -p) 8
POSIX message queues (bytes, -q) 819200
real-time priority (-r) 0
stack size (kbytes, -s) 10240
cpu time (seconds, -t) unlimited
max user processes (-u) 14874
virtual memory (kbytes, -v) unlimited
file locks (-x) unlimited
```

You can also use the `ulimit` command to dynamically set these limits until the next reboot. This method sets a temporary value that will revert to the settings in the `/etc/security/limits.conf` file after the next reboot, but it is useful for experimenting with limit settings. For example:

```
[root@node-1 /]# ulimit -n 32768
```

The updated value can then be displayed:

```
[root@node-1 /]# ulimit -n
32768
```


2.7.2.3. Configuring HBase and ZooKeeper

Use the following instructions to set up secure HBase and ZooKeeper:

1. [Configure HBase Master \[101\]](#)
2. [Create JAAS configuration files \[103\]](#)
3. [Start HBase and ZooKeeper services \[104\]](#)
4. [Configure secure client side access for HBase \[105\]](#)
5. [Optional: Configure client-side operation for secure operation - Thrift Gateway \[106\]](#)
6. [Optional: Configure client-side operation for secure operation - REST Gateway \[106\]](#)
7. [Configure HBase for Access Control Lists \(ACL\) \[107\]](#)

2.7.2.3.1. Configure HBase Master

Edit `$HBASE_CONF_DIR/hbase-site.xml` file on your HBase Master server to add the following information (`$HBASE_CONF_DIR` is the directory to store the HBase configuration files. For example, `/etc/hbase/conf`):



Note

There are no default values. The following are all examples.

```
<property>
  <name>hbase.master.keytab.file</name>
  <value>/etc/security/keytabs/hbase.service.keytab</value>
  <description>Full path to the Kerberos keytab file to use
    for logging in the configured HMaster server principal.

  </description>
</property>
```

```
<property>
  <name>hbase.master.kerberos.principal</name>
  <value>hbase/_HOST@EXAMPLE.COM</value>
  <description>Ex. "hbase/_HOST@EXAMPLE.COM".
    The Kerberos principal name that should be used to run the HMaster
    process.
    The principal name should be in the form: user/hostname@DOMAIN.
    If "_HOST" is used as the hostname portion,
    it will be replaced with the actual hostname of the running instance.

  </description>
</property>
```

```
<property>
  <name>hbase.regionserver.keytab.file</name>
  <value>/etc/security/keytabs/hbase.service.keytab</value>
  <description>Full path to the Kerberos keytab file to use for logging
    in the configured HRegionServer server principal.

  </description>
</property>
```

```
<property>
  <name>hbase.regionserver.kerberos.principal</name>
  <value>hbase/_HOST@EXAMPLE.COM</value>
  <description>Ex. "hbase/_HOST@EXAMPLE.COM".
```

The Kerberos principal name that should be used to run the HRegionServer process.

The

principal name should be in the form:

```
user/hostname@DOMAIN.
```

If `_HOST`

is used as the hostname portion, it will be replaced with the actual hostname of the running instance.

An entry for this principal must exist

in the file specified in `hbase.regionserver.keytab.file`

```
</description>
</property>
```

```
<!--Additional configuration specific to HBase security -->
```

```
<property>
  <name>hbase.superuser</name>
  <value>hbase</value>
  <description>List of users or groups (comma-separated), who are
  allowed full privileges, regardless of stored ACLs, across the cluster.
  Only used when HBase security is enabled.
  </description>
</property>
```

```
<property>
  <name>hbase.coprocessor.region.classes</name>
  <value>org.apache.hadoop.hbase.security.token.TokenProvider,
  org.apache.hadoop.hbase.security.access.SecureBulkLoadEndpoint,
  org.apache.hadoop.hbase.security.access.AccessController </value>
  <description>A comma-separated list of Coprocessors that are loaded by
  default on all tables.
  </description>
</property>
```

```
<property>
  <name>hbase.security.authentication</name>
  <value>kerberos</value>
</property>
```

```
<property>
  <name>hbase.rpc.engine</name>
  <value>org.apache.hadoop.hbase.ipc.SecureRpcEngine</value>
</property>
```

```
<property>
  <name>hbase.security.authorization</name>
  <value>true</value>
  <description>Enables HBase authorization.
  Set the value of this property to false to disable HBase authorization.
  </description>
</property>
```

```

<property>
  <name>hbase.coprocessor.master.classes</name>
  <value>org.apache.hadoop.hbase.security.access.AccessController</
value>
</property>

<property>
  <name>hbase.bulkload.staging.dir</name>
  <value>/apps/hbase/staging</value>
  <description>Directory in the default filesystem,
owned by the hbase user, and has permissions(-rwx--x--x, 711) </description>
</property>

```

For more information on bulk loading in secure mode, see [HBase Secure BulkLoad](#). Note that the `hbase.bulkload.staging.dir` is created by HBase.

2.7.2.3.2. Create JAAS configuration files

1. Create the following JAAS configuration files on the HBase Master, RegionServer, and HBase client host machines.

These files must be created under the `$HBASE_CONF_DIR` directory:

where `$HBASE_CONF_DIR` is the directory to store the HBase configuration files. For example, `/etc/hbase/conf`.

- On each machine running an HBase server, create the `hbase-server.jaas` file under the `/etc/hbase/conf` directory. HBase servers include the HMaster and RegionServer. In this file, add the following content:

```

Client {
  com.sun.security.auth.module.Krb5LoginModule required
  useKeyTab=true
  storeKey=true
  useTicketCache=false
  keyTab="/etc/security/keytabs/hbase.service.keytab"
  principal="hbase/$fully.qualified.domain.name";
};

```

- On HBase client machines, create the `hbase-client.jaas` file under the `/etc/hbase/conf` directory and add the following content:

```

Client {
  com.sun.security.auth.module.Krb5LoginModule required
  useKeyTab=false
  useTicketCache=true;
};

```

2. Create the following JAAS configuration files on the ZooKeeper Server and client host machines.

These files must be created under the `$ZOOKEEPER_CONF_DIR` directory, where `$ZOOKEEPER_CONF_DIR` is the directory to store the HBase configuration files. For example, `/etc/zookeeper/conf`:

- On ZooKeeper server host machines, create the `zookeeper-server.jaas` file under the `/etc/zookeeper/conf` directory and add the following content:

```
Server {
  com.sun.security.auth.module.Krb5LoginModule required
  useKeyTab=true
  storeKey=true
  useTicketCache=false
  keyTab="/etc/security/keytabs/zookeeper.service.keytab"
  principal="zookeeper/$ZooKeeper.Server.hostname";
};
```

- On ZooKeeper client host machines, create the `zookeeper-client.jaas` file under the `/etc/zookeeper/conf` directory and add the following content:

```
Client {
  com.sun.security.auth.module.Krb5LoginModule required
  useKeyTab=false
  useTicketCache=true;
};
```

3. Edit the `hbase-env.sh` file on your HBase server to add the following information:

```
export HBASE_OPTS="-Djava.security.auth.login.config=
$HBASE_CONF_DIR/hbase-client.jaas"
export HBASE_MASTER_OPTS="-Djava.security.auth.login.config=
$HBASE_CONF_DIR/hbase-server.jaas"
export HBASE_REGIONSERVER_OPTS="-Djava.security.auth.login.config=
$HBASE_CONF_DIR/hbase-server.jaas"
```

where `HBASE_CONF_DIR` is the HBase configuration directory. For example, `/etc/hbase/conf`.

4. Edit `zoo.cfg` file on your ZooKeeper server to add the following information:

```
authProvider.1=org.apache.zookeeper.server.auth.SASLAuthenticationProvider
jaasLoginRenew=3600000
kerberos.removeHostFromPrincipal=true
kerberos.removeRealmFromPrincipal=true
```

5. Edit `zookeeper-env.sh` file on your ZooKeeper server to add the following information:

```
export SERVER_JVMFLAGS="-Djava.security.auth.login.
config=$ZOOKEEPER_CONF_DIR/zookeeper-server.jaas"
export CLIENT_JVMFLAGS="-Djava.security.auth.login.
config=$ZOOKEEPER_CONF_DIR/zookeeper-client.jaas"
```

where `ZOOKEEPER_CONF_DIR` is the ZooKeeper configuration directory. For example, `/etc/zookeeper/conf`.

2.7.2.3.3. Start HBase and ZooKeeper services

Start the HBase and ZooKeeper services using the instructions provided in the HDP Reference Manual, [Starting HDP Services](#).

If the configuration is successful, you should see the following in your ZooKeeper server logs:

```

11/12/05 22:43:39 INFO zookeeper.Login: successfully logged in.
11/12/05 22:43:39 INFO server.NIOServerCnxnFactory: binding to port 0.0.0.0/0.0.0.0:2181
11/12/05 22:43:39 INFO zookeeper.Login: TGT refresh thread started.
11/12/05 22:43:39 INFO zookeeper.Login: TGT valid starting at:           Mon Dec
05 22:43:39 UTC 2011
11/12/05 22:43:39 INFO zookeeper.Login: TGT expires:                   Tue Dec
06 22:43:39 UTC 2011
11/12/05 22:43:39 INFO zookeeper.Login: TGT refresh sleeping until: Tue Dec 06
18:36:42 UTC 2011
..
11/12/05 22:43:59 INFO auth.SaslServerCallbackHandler:
  Successfully authenticated client: authenticationID=hbase/ip-10-166-175-249.
us-west-1.compute.internal@HADOOP.LOCALDOMAIN;
  authorizationID=hbase/ip-10-166-175-249.us-west-1.compute.internal@HADOOP.
LOCALDOMAIN.
11/12/05 22:43:59 INFO auth.SaslServerCallbackHandler: Setting authorizedID:
  hbase
11/12/05 22:43:59 INFO server.ZooKeeperServer: adding SASL authorization for
  authorizationID: hbase

```

2.7.2.3.4. Configure secure client side access for HBase

HBase configured for secure client access is expected to be running on top of a secure HDFS cluster. HBase must be able to authenticate to HDFS services.

1. Provide a Kerberos principal to the HBase client user using the instructions provided [here](#).

- **Option I:** Provide Kerberos principal to normal HBase clients.

For normal HBase clients, Hortonworks recommends setting up a password to the principal.

- Set `maxrenewlife`.

The client principal's `maxrenewlife` should be set high enough so that it allows enough time for the HBase client process to complete. Client principals are not renewed automatically.

For example, if a user runs a long-running HBase client process that takes at most three days, we might create this user's principal within `kadmin` with the following command:

```
addprinc -maxrenewlife 3days
```

- **Option II:** Provide Kerberos principal to long running HBase clients.

- a. Set-up a keytab file for the principal and copy the resulting keytab files to where the client daemon will execute.

Ensure that you make this file readable only to the user account under which the daemon will run.

2. On every HBase client, add the following properties to the `$HBASE_CONF_DIR/hbase-site.xml` file:

```
<property>
  <name>hbase.security.authentication</name>
  <value>kerberos</value>
</property>
```



Note

The client environment must be logged in to Kerberos from KDC or keytab via the `kinit` command before communication with the HBase cluster is possible. Note that the client will not be able to communicate with the cluster if the `hbase.security.authentication` property in the client- and server-side site files fails to match.

```
<property>
  <name>hbase.rpc.engine</name>
  <value>org.apache.hadoop.hbase.ipc.SecureRpcEngine</value>
</property>
```

2.7.2.3.5. Optional: Configure client-side operation for secure operation - Thrift Gateway

Add the following to the `$HBASE_CONF_DIR/hbase-site.xml` file for every Thrift gateway:

```
<property>
  <name>hbase.thrift.keytab.file</name>
  <value>/etc/hbase/conf/hbase.keytab</value>
</property>
<property>
  <name>hbase.thrift.kerberos.principal</name>
  <value>${USER}/_HOST@HADOOP.LOCALDOMAIN</value>
</property>
```

Substitute the appropriate credential and keytab for `$USER` and `$KEYTAB` respectively.

The Thrift gateway will authenticate with HBase using the supplied credential. No authentication will be performed by the Thrift gateway itself. All client access via the Thrift gateway will use the Thrift gateway's credential and have its privilege.

2.7.2.3.6. Optional: Configure client-side operation for secure operation - REST Gateway

Add the following to the `$HBASE_CONF_DIR/hbase-site.xml` file for every REST gateway:

```
<property>
  <name>hbase.rest.keytab.file</name>
  <value>${KEYTAB}</value>
</property>
<property>
  <name>hbase.rest.kerberos.principal</name>
  <value>${USER}/_HOST@HADOOP.LOCALDOMAIN</value>
</property>
```

Substitute the appropriate credential and keytab for `$USER` and `$KEYTAB` respectively.

The REST gateway will authenticate with HBase using the supplied credential. No authentication will be performed by the REST gateway itself. All client access via the REST gateway will use the REST gateway's credential and have its privilege.

2.7.2.3.7. Configure HBase for Access Control Lists (ACL)

Use the following instructions to configure HBase for ACL:

1. Open `kinit` as HBase user.
 - a. Create a keytab for principal `hbase@REALM` and store it in the `hbase.headless.keytab` file. See instructions provided [Creating Service Principals and Keytab Files for HDP \[79\]](#) for creating principal and keytab file.
 - b. Open `kinit` as HBase user. Execute the following command on your HBase Master:

```
kinit -kt hbase.headless.keytab hbase
```

2. Start the HBase shell. On the HBase Master host machine, execute the following command:

```
hbase shell
```

3. Set ACLs using HBase shell:

```
grant '$USER', '$permissions'
```

where

- `$USER` is any user responsible for create/update/delete operations in HBase.



Note

You must set the ACLs for all those users who will be responsible for create/update/delete operations in HBase.

- `$permissions` is zero or more letters from the set "RWCA": READ('R'), WRITE('W'), CREATE('C'), ADMIN('A').

2.7.2.4. Configuring Phoenix Query Server

The HBase configuration provides most of the settings that enable secure Kerberos environments for Phoenix. However, there are additional configuration properties that complete the setup of Kerberos security for the Phoenix Query Server.

Prerequisite: The value of the `hbase.security.authentication` property in the `$HBASE_CONF_DIR/hbase-site.xml` file must be set to `kerberos`.

1. Provide the Kerberos principal and keytab for the Phoenix Query Server in the `$HBASE_CONF_DIR/hbase-site.xml` file.

```
<property>
```

```

<name>phoenix.queryserver.kerberos.principal</name>
<value>HTTP/_HOST@EXAMPLE.COM</value>
<description>The Kerberos principal name that should be used to run the
Phoenix Query Server process.
  The principal name should be in the form: user/hostname@DOMAIN.  If
  "_HOST" is used as the hostname
  portion, it will be replaced with the actual hostname of the running
  instance.
</description>
</property>

<property>
  <name>phoenix.queryserver.kerberos.keytab</name>
  <value>/etc/security/keytabs/spnego.service.keytab</value>
  <description>Full path to the Kerberos keytab file to use for logging
  in the configured Phoenix Query Server service principal.
  </description>
</property>

```

2. Add the fully-qualified domain name for each host running the Phoenix Query Server to the list of hosts that can impersonate end users in the `$HADOOP_CONF_DIR/core-site.xml` file. Alternatively, insert an asterisk (*) instead of host names if you want to allow all hosts to impersonate end users.

```

<property>
  <name>hadoop.proxyuser.HTTP.hosts</name>
  <value>server1.domain.com,server2.domain.com</value>
  <description>A comma-separated list of fully-qualified
  domain names of hosts running services with the Hadoop
  user "HTTP" that can impersonate end users.
  Alternatively, insert an asterisk (*) instead of
  listing host names if you want to allow all hosts to
  impersonate end users.</description>
</property>

```

2.7.2.5. Configuring Hue

Before you can configure Hue to work with an HDP cluster that is configured for Kerberos, you must refer to and complete the instructions for [Configuring Ambari and Hadoop for Kerberos](#) or [Setting Up Kerberos Security for Manual Installs](#).

To enable Hue to work with an HDP cluster configured for Kerberos, make the following changes to Hue and Kerberos.:

1. Where `$FQDN` is the host name of the Hue server and `EXAMPLE.COM` is the Hadoop realm, create a principal for the Hue server:

```

# kadmin.local
kadmin.local: addprinc -randkey hue/$FQDN@EXAMPLE.COM

```

2. Where `$FQDN` is the host name of the Hue server and `EXAMPLE.COM` is the Hadoop realm, generate a keytab for the Hue principal:

```

kadmin.local: xst -k hue.service.keytab hue/$FQDN@EXAMPLE.COM

```

3. Put the `hue.service.keytab` file on the host where the Hue server is installed, in the directory `/etc/security/keytabs`.

4. Set the permissions and ownership of the `/etc/security/keytabs/hue.service.keytab` file as follows:

```
chown hue:hadoop /etc/security/keytabs/hue.service.keytab
chmod 600 /etc/security/keytabs/hue.service.keytab
```

5. Where `$FQDN` is the host name of the Hue server and `EXAMPLE.COM` is the Hadoop realm, use `kinit` to confirm that the `/etc/security/keytabs/hue.service.keytab` file is accessible to Hue:

```
su - hue kinit -k -t /etc/security/keytabs/hue.service.keytab hue/
$FQDN@EXAMPLE.COM
```

6. Where `$FQDN` is the host name of the Hue server and `EXAMPLE.COM` is the Hadoop realm, add the following to the `[kerberos]` section in the `/etc/hue/conf/hue.ini` configuration file:

```
[[kerberos]]
# Path to Hue's Kerberos keytab file
hue_keytab=/etc/security/keytabs/hue.service.keytab
# Kerberos principal name for Hue
hue_principal=hue/$FQDN@EXAMPLE.COM
```

7. Set the path to `kinit`, based on the OS.

If you do not know the full path to `kinit`, you can find it by issuing the command **where is kinit**.

The following is an example of setting the path to `kinit` for RHEL/CentOS 6.x:

```
# Path to kinit
# For RHEL/CentOS 6.x, kinit_path is /usr/bin/kinit
kinit_path=/usr/kerberos/bin/kinit
```

8. Optionally, for faster performance, you can keep Kerberos credentials cached:

```
ccache_path=/tmp/hue_krb5_ccache
```

9. Edit the `/etc/hue/conf/hue.ini` configuration file and set `security_enabled=true` for every component in the configuration file.

10. Save the `/etc/hue/conf/hue.ini` configuration file.

11. Restart Hue:

```
# /etc/init.d/hue start
```

12. Validate the Hue installation.

- a. To view the current configuration of your Hue server, select **About > Configuration** or http://hue.server:8000/dump_config.

- b. To ensure that Hue server was configured properly, select **About > Check for misconfiguration** or http://hue.server:8000/debug/check_config.

If you detect any potential misconfiguration, fix it and restart Hue.

2.7.3. Setting up One-Way Trust with Active Directory

In environments where users from Active Directory (AD) need to access Hadoop Services, set up one-way trust between Hadoop Kerberos realm and the AD (Active Directory) domain.



Important

Hortonworks recommends setting up one-way trust after fully configuring and testing your Kerberized Hadoop Cluster.

2.7.3.1. Configure Kerberos Hadoop Realm on the AD DC

Configure the Hadoop realm on the AD DC server and set up the one-way trust.

1. Add the Hadoop Kerberos realm and KDC host to the DC:

```
ksetup /addkdc $hadoop.realm $KDC-host
```

2. Establish one-way trust between the AD domain and the Hadoop realm:

```
netdom trust $hadoop.realm /Domain:$AD.domain /add /realm /passwordt:$trust_password
```

3. **(Optional)** If Windows clients within the AD domain need to access Hadoop Services, and the domain does not have a search route to find the services in Hadoop realm, run the following command to create a hostmap for Hadoop service host:

```
ksetup /addhosttorealmmap $hadoop-service-host $hadoop.realm
```



Note

Run the above for each \$hadoop-host that provides services that need to be accessed by Windows clients. For example, Oozie host, WebHCat host, etc.

4. **(Optional)** Define the encryption type:

```
ksetup /SetEncTypeAttr $hadoop.realm $encryption_type
```

Set encryption types based on your security requirements. Mismatched encryption types cause problems.



Note

Run `ksetup /GetEncTypeAttr $krb_realm` to list the available encryption types. Verify that the encryption type is configured for the Hadoop realm in the `krb5.conf`.

2.7.3.2. Configure the AD Domain on the KDC and Hadoop Cluster Hosts

Add the AD domain as a realm to the `krb5.conf` on the Hadoop cluster hosts. Optionally configure encryption types and UDP preferences.

1. Open the krb5.conf file with a text editor and make the following changes:

- To libdefaults, add the following properties.

- Set the Hadoop realm as default:

```
[libdefaults]
default_domain = $hadoop.realm
```

- Set the encryption type:

```
[libdefaults]
default_tkt_enctypes = $encryption_types
default_tgs_enctypes = $encryption_types
permitted_enctypes = $encryption_types
```

where the \$encryption_types match the type supported by your environment.

For example:

```
default_tkt_enctypes = aes256-cts aes128-cts rc4-hmac arcfour-hmac-md5
des-cbc-md5 des-cbc-crc
default_tgs_enctypes = aes256-cts aes128-cts rc4-hmac arcfour-hmac-md5
des-cbc-md5 des-cbc-crc
permitted_enctypes = aes256-cts aes128-cts rc4-hmac arcfour-hmac-md5
des-cbc-md5 des-cbc-crc
```

- If TCP is open on the KDC and AD Server:

```
[libdefaults]
udp_preference_limit = 1
```

- Add a realm for the AD domain:

```
[realms]
$AD.DOMAIN = {
kdc = $AD-host-FQDN
admin_server = $AD-host-FQDN
default_domain = $AD-host-FQDN
}
```

- Save the krb5.conf changes to all Hadoop Cluster hosts.

2. Add the trust principal for the AD domain to the Hadoop MIT KDC:

```
kadmin
kadmin:addprinc krbtgt/$hadoop.realm@$AD.domain
```

This command will prompt you for the trust password. Use the same password as the earlier step.



Note

If the encryption type was defined, then use the following command to configure the AD principal:

```
kadmin:addprinc -e "$encryption_type"krbtgt/$hadoop.realm@$AD.domain
```

When defining encryption, be sure to also enter the encryption type (e.g., 'normal')

2.7.4. Configuring Proxy Users

For information about configuring a superuser account that can submit jobs or access HDFS on behalf of another user, see the following information on the Apache site:

[Proxy user - Superusers Acting on Behalf of Other Users.](#)

2.8. Perimeter Security with Apache Knox

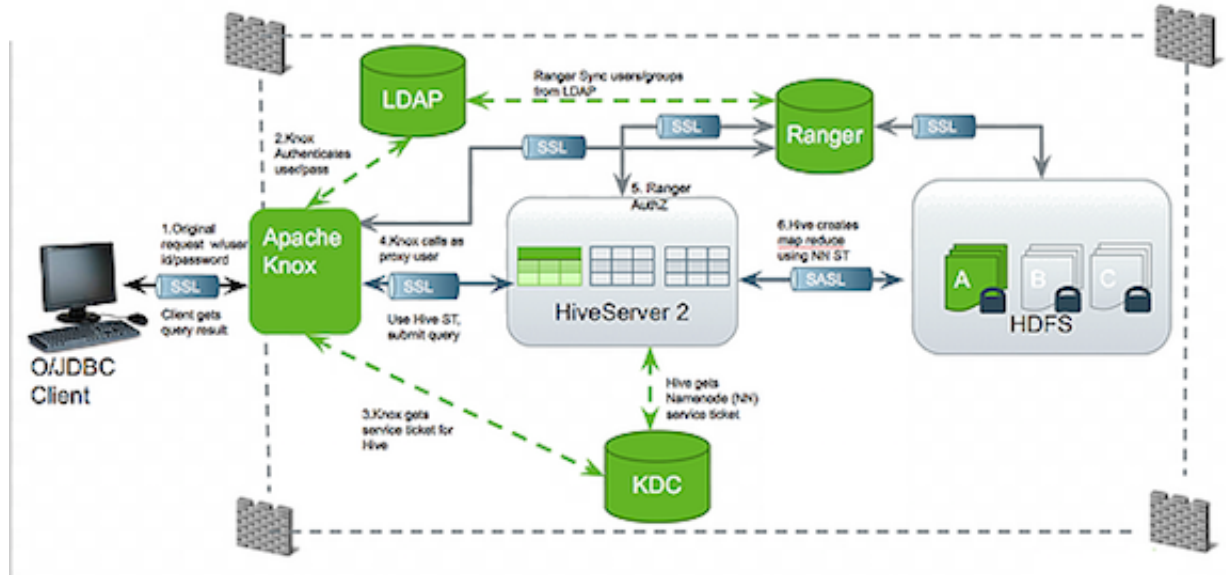
2.8.1. Apache Knox Gateway Overview

The Apache Knox Gateway (“Knox”) is a system to extend the reach of Apache™ Hadoop® services to users outside of a Hadoop cluster without reducing Hadoop Security. Knox also simplifies Hadoop security for users who access the cluster data and execute jobs. The Knox Gateway is designed as a reverse proxy.

Knox integrates with Identity Management and SSO systems used in enterprises and allows identity from these systems be used for access to Hadoop clusters.

Knox Gateways provides security for multiple Hadoop clusters, with these advantages:

- **Simplifies access:** Extends Hadoop’s REST/HTTP services by encapsulating Kerberos to within the Cluster.
- **Enhances security:** Exposes Hadoop’s REST/HTTP services without revealing network details, providing SSL out of the box.
- **Centralized control:** Enforces REST API security centrally, routing requests to multiple Hadoop clusters.
- **Enterprise integration:** Supports LDAP, Active Directory, SSO, SAML and other authentication systems.



Typical Security Flow: Firewall, Routed Through Knox Gateway

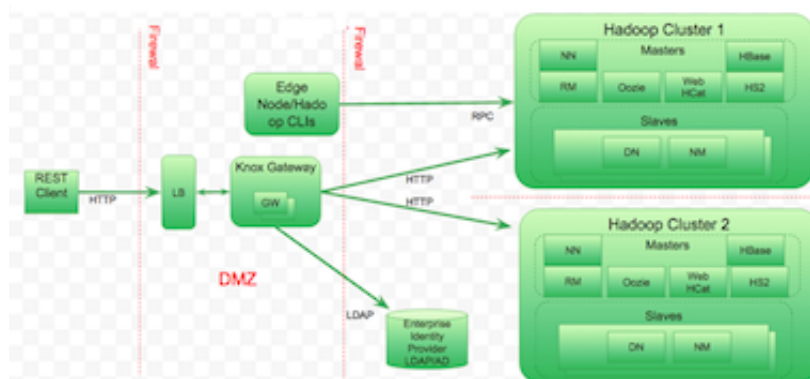
Knox can be used with both unsecured Hadoop clusters, and Kerberos secured clusters. In an enterprise solution that employs Kerberos secured clusters, the Apache Knox Gateway provides an enterprise security solution that:

- Integrates well with enterprise identity management solutions
- Protects the details of the Hadoop cluster deployment (hosts and ports are hidden from end users)
- Simplifies the number of services with which a client needs to interact

2.8.1.1. Knox Gateway Deployment Architecture

Users who access Hadoop externally do so either through Knox, via the Apache REST API, or through the Hadoop CLI tools.

The following diagram shows how Apache Knox fits into a Hadoop deployment.



NN=NameNode, RM=Resource Manager, DN=DataNode, NM=NodeManager

2.8.1.2. Supported Hadoop Services

Apache Knox Gateway supports the following Hadoop services versions in both Kerberized and Non-Kerberized clusters:

Table 2.20. Supported Component APIs: Proxy

Service
YARN
WebHDFS
WebHCat/Templeton
Oozie
HBase/Stargate
Hive (via WebHCat)
Hive (via JDBC)
Ambari
Atlas
Ranger
Zeppelin

Table 2.21. Supported Component UIs: Proxy

Service
Ambari UI
Atlas
Ranger Admin Console
Zeppelin



Note

APIs and UIs in the Apache Knox project that are not listed above are considered Community Features.

Community Features are developed and tested by the Apache Knox community but are not officially supported by Hortonworks. These features are excluded for a variety of reasons, including insufficient reliability or incomplete test case coverage, declaration of non-production readiness by the community at large, and feature deviation from Hortonworks best practices. Do not use these features in your production environments.

2.8.1.3. Knox Gateway Samples

There are a number of different methods you can use to deploy the Knox Gateway in your cluster. Each of these methods consists of different ways you can use to install and configure the Knox Gateway. For more information on these methods, refer to the following Apache documentation:

- <https://knox.apache.org/books/knox-0-12-0/user-guide.html#Gateway+Samples>

2.8.2. Configuring the Knox Gateway

This section describes how to configure the Knox Gateway. This section describes how you can:

- [Create and Secure the Gateway Directories \[115\]](#)
- [Manage the Master Secret \[116\]](#)
- [Manually Redeploy Cluster Topologies \[116\]](#)
- [Manually Start and Stop Apache Knox \[118\]](#)
- [Enable WebSockets \[118\]](#)

2.8.2.1. Create and Secure the Gateway Directories

Installing Knox Gateway with the platform-specific installers creates the following directories:

- HADOOP_NODE_INSTALL_ROOT
- `knox-X.X.X.X.X.X-XXXX` – the \$gateway directory.

For example, `D:/hdp/knox-0.4.0.2.1.1.0-1557` The directory contains the following files:

Table 2.22. Apache Service Gateway Directories

Directory/Filename	Description
conf/topologies	Contains global gateway configuration files.
bin	Contains the executable shell scripts, batch files, and JARs for clients and servers.
deployments	Contains cluster topology descriptor files that define Hadoop clusters.
lib	Contains the JARs for all the components that make up the gateway.
dep	Contains the JARs for all of the component upon which the gateway depends.
ext	A directory where user-supplied extensions JARs can be placed to extends the gateway functionality.
samples	Contains a number of samples that can be used to explore the functionality of the gateway.
templates	Contains default configuration files that can be copied and customized.
README	Provides basic information about the Apache Knox Gateway.
ISSUES	Describes significant known issues.
CHANGES	Enumerates the changes between releases.
LICENSE	Documents the license under which this software is provided.
NOTICE	Documents required attribution notices for included dependencies.
DISCLAIMER	Documents that this release is from a project undergoing incubation at Apache.

- `SystemDrive/hadoop/logs/knox`

This contains the output files from the Knox Gateway.

2.8.2.2. Manage the Master Secret

The master secret is required to start the gateway. The secret protects artifacts used by the gateway instance, such as the keystore, trust stores and credential stores.

You configure the gateway to persist the master secret, which is saved in the `$gateway / data / security / master` file. Ensure that this directory has the appropriate permissions set for your environment. To set the master secret, enter:

```
cd $gateway bin/knoxcli.cmd create-master
```

A warning displays indicating that persisting the secret is less secure than providing it at startup. Knox protects the password by encrypting it with AES 128 bit encryption; where possible, the file permissions are set to be accessible only by the Knox user.



Warning

Ensure that the security directory, `$gateway / data / security`, and its contents are readable and writable only by the Knox user. This is the most important layer of defense for the master secret. **Do not assume that the encryption is sufficient protection.**

Changing the Master Secret

The Master Secret can be changed under dire situations where the Administrator has to redo all the configurations for every gateway instance in a deployment, and no longer knows the Master Secret. Recreating the Master Secret requires not only recreating the master, but also removing all existing keystores and reprovisioning the certificates and credentials.

1. To change the Master Secret:

```
cd $gateway bin/knoxcli.cmd create-master--force
```

2. If there is an existing keystore, update the keystore.

2.8.2.3. Manually Redeploy Cluster Topologies

You are not required to manually redeploy clusters after updating cluster properties. The gateway monitors the topology descriptor files in the `$gateway / conf / topologies` directory and automatically redeploys the cluster if any descriptor changes or a new one is added. (The corresponding deployment is in `$gateway / data / deployments`.)

However, you must redeploy the clusters after changing any of the following gateway properties or gateway-wide settings:

- Time settings on the gateway host
- Implementing or updating Kerberos
- Implementing or updating SSL certificates

- Changing a cluster alias

Redeploying all clusters at the same time

When making gateway-wide changes (such as implementing Kerberos or SSL), or if you change the system clock, you must redeploy all the Cluster Topologies. Do the following:

1. To verify the timestamp on the currently deployed clusters enter: `cd $gatewaydir data/deployments`. The system displays something similar to:

```
Directory of C:/hdp/knox-0.4.0.2.1.1.0-1557/data/deployments

04/17/2014 05:30 PM <DIR> .
04/17/2014 05:30 PM <DIR> ..
04/17/2014 05:30 PM <DIR> cluster.war.145514f4dc8
04/17/2014 05:30 PM <DIR> myCluster.war.145514f4dc8
04/11/2014 08:35 AM <DIR> sandbox.war.145514f4dc8
0 File(s) 0 bytes
5 Dir(s) 9,730,977,792 bytes free
```

2. To redeploy all clusters, enter `/bin/knoxcli.cmd redeploy`.
3. To verify that a new cluster WAR was created, enter `cd $gatewaydir data/deployments`. The system displays something similar to:

```
Directory of C:/hdp/knox-0.4.0.2.1.1.0-1557/data/deployments

04/17/2014 05:34 PM <DIR> .
04/17/2014 05:34 PM <DIR> ..
04/17/2014 05:30 PM <DIR> cluster.war.145514f4dc8
04/17/2014 05:34 PM <DIR> cluster.war.1457241b5dc
04/17/2014 05:30 PM <DIR> myCluster.war.145514f4dc8
04/17/2014 05:34 PM <DIR> myCluster.war.1457241b5dc
04/11/2014 08:35 AM <DIR> sandbox.war.145514f4dc8
04/17/2014 05:34 PM <DIR> sandbox.war.1457241b5dc
0 File(s) 0 bytes
8 Dir(s) 9,730,850,816 bytes free
```

A new file is created for each cluster, with the current timestamp.

Redeploy only specific clusters

When making changes that impact a single cluster, such as changing an alias or restoring from an earlier cluster topology descriptor file, you only redeploy the effected cluster. Do the following:

1. To verify the timestamp on the currently deployed Cluster Topology WAR files, enter: `cd $gatewaydir data/deployments`. The system displays something similar to:

```
Directory of C:/hdp/knox-0.4.0.2.1.1.0-1557/data/deployments

04/17/2014 05:30 PM <DIR> .
04/17/2014 05:30 PM <DIR> ..
04/17/2014 05:30 PM <DIR> cluster.war.145514f4dc8
04/17/2014 05:30 PM <DIR> myCluster.war.145514f4dc8
04/11/2014 08:35 AM <DIR> sandbox.war.145514f4dc8
0 File(s) 0 bytes
5 Dir(s) 9,730,977,792 bytes free
```

2. To redeploy a specific cluster, enter:

```
cd $gateway bin/knoxcli.cmd redeploy --cluster $cluster_name
```

where `$cluster_name` is the name of the cluster topology descriptor (without the `.xml` extension). For example, `myCluster`.

3. To verify that the cluster was deployed, enter: `cd $gatewaydir data/deployments`. The system displays something similar to:

```
Directory of C:/hdp/knox-0.4.0.2.1.1.0-1557/data/deployments
04/17/2014 05:30 PM <DIR> .
04/17/2014 05:30 PM <DIR> ..
04/17/2014 05:30 PM <DIR> cluster.war.145514f4dc8
04/17/2014 05:30 PM <DIR> myCluster.war.145514f4dc8
04/17/2014 05:34 PM <DIR> myCluster.war.1457241b5dc
04/11/2014 08:35 AM <DIR> sandbox.war.145514f4dc8
0 File(s) 0 bytes
5 Dir(s) 9,730,977,792 bytes free
```

You should see that existing cluster war files are unchanged, but the war file for `myCluster` was updated (has a current timestamp).

2.8.2.4. Manually Start and Stop Apache Knox

Except for changes to `../conf/topology/*.xml`, changes to the Knox Gateway global settings in `$gateway /conf/gateway-site.xml` cannot be loaded before the Gateway is restarted.

To manually stop Knox:

```
cd $gateway/bin/gateway.sh stop
```

This is known as a clean shutdown, as the gateway script cleans out all `.out` and `.err` files in the logs directory.

To manually start Knox for the first time, or re-start Knox after a clean shutdown:

```
cd $gateway /bin/gateway.sh start
```

To manually re-start Knox after an unclean shutdown:

```
cd $gateway/bin/gateway.sh clean /bin/gateway.sh start
```

This command eliminates old `.out` and `.err` files in the logs directory.

2.8.2.5. Enable WebSockets

Enabling WebSockets for Knox allows you to proxy applications that use WebSocket connections (e.g., Zeppelin.)

About This Task

WebSocket is a communication protocol that allows full duplex communication over single TCP connection. Knox provides out-of-the-box support for WebSocket protocol, but currently, only text-based messages are supported.

By default, WebSocket functionality is disabled. WebSocket functionality must be enabled for Zeppelin UI (`<role>ZEPPELINUI</role>`) service definition to work.

Task

1. In `/conf/gateway-site.xml`, change `gateway.websocket.feature.enabled` to `true`:

```
<property>
  <name>gateway.websocket.feature.enabled</name>
  <value>true</value>
  <description>Enable/Disable websocket feature.</description>
</property>
```

2. In `/conf/{topology}.xml`, change the topology rule:

```
<service>
  <role>WEBSOCKET</role>
  <url>ws://myhost:9999/ws</url>
</service>
```

3. Restart the gateway:

```
cd $gateway bin/gateway.sh stop bin/gateway.sh start
```

2.8.3. Defining Cluster Topologies

The Knox Gateway supports one or more Hadoop clusters. Each Hadoop cluster configuration is defined in a topology deployment descriptor file in the `$gateway/conf/topologies` directory and is deployed to a corresponding WAR file in the `$gateway/data/deployments` directory. These files define how the gateway communicates with each Hadoop cluster.

The descriptor is an XML file contains the following sections:

- `gateway/provider` – configuration settings enforced by the Knox Gateway while providing access to the Hadoop cluster.
- `service` – defines the Hadoop service URLs used by the gateway to proxy communications from external clients.

The gateway automatically redeploys the cluster whenever it detects a new topology descriptor file, or detects a change in an existing topology descriptor file.

The following table provides an overview of the providers and services:

Table 2.23. Cluster Topology Provider and Service Roles

Type	Role	Description
gateway/provider	hostmap	Maps external to internal node hostnames, replacing the internal hostname with the mapped external name when the hostname is embedded in a response from the cluster.

Type	Role	Description
	authentication	Integrates an LDAP store to authenticate external requests accessing the cluster via the Knox Gateway. Refer to Set Up LDAP Authentication for more information.
	federation	Defines HTTP header authentication fields for an SSO or federation solution provider. Refer to Set up HTTP Header Authentication for Federation/SSO
	identity-assertion	Responsible for the way that the authenticated user's identity is asserted to the service that the request is intended for. Also maps external authenticated users to an internal cluster that the gateway asserts as the current session user or group. Refer to Configure Identity Assertion for more information.
	authorization	Service level authorization that restricts cluster access to specified users, groups, and/or IP addresses. Refer to Configure Service Level Authorization for more information.
	webappspec	Configures a web application security plugin that provides protection filtering against Cross Site Request Forgery Attacks. Refer to Configure Web Application Security for more information.
HA provider	high availability	Syncs all Knox instances to use the same topologies credentials keystores.
service	<code>service_name</code>	Binds a Hadoop service with an internal URL that the gateway uses to proxy requests from external clients to the internal cluster services. Refer to Configure Hadoop Service URLs for more information.

Cluster topology descriptors have the following XML format:

```
<topology>
  <gateway>
    <provider>
      <role></role>
      <name></name>
      <enabled></enabled>
      <param>
        <name></name>
        <value></value>
      </param>
    </provider>
  </gateway>
  <service></service>
</topology>
```

2.8.4. Configuring a Hadoop Server for Knox

The Apache Knox Gateway redirects external requests to an internal Hadoop service using service name and URL of the service definition.

This chapter describes:

- [Setting up Hadoop Service URLs \[121\]](#)
- [Example Service Definitions \[121\]](#)
- [Validating Service Connectivity \[123\]](#)
- [Adding a New Service to the Knox Gateway \[125\]](#)

2.8.4.1. Setting up Hadoop Service URLs

To configure access to an internal Hadoop service through the Knox Gateway:

1. Edit `$gateway/conf/topologies$cluster-name.xml` to add an entry similar to the following, for each Hadoop service:

```
<topology>
  <gateway>
    ...
  </gateway>
  <service>
    <role> $service_name </role>
    <url> $schema://$hostname:$port</url>
  </service>
</topology>
```

where:

- `$service_name` is: AMBARI, AMBARIUI, ATLAS, HIVE, JOBTRACKER, NAMENODE, OOZIE, RANGER, RANGERUI, RESOURCEMANAGER, WEBHBASE, WEBHCAT, WEBHDFS, ZEPPELINUI, or ZEPPELINWS.
- `<url>` is the complete internal cluster URL required to access the service, including:
 - `$schema` – the service protocol
 - `$hostname` – the resolvable internal host name
 - `$port` – the service listening port

2. Save the file.

The gateway creates a new WAR file with modified timestamp in `$gateway/data/deployments`.



Note

It is not necessary to restart the Knox server after making changes to the topology/Hadoop Cluster services.

2.8.4.2. Example Service Definitions

Configure each service that you want to expose externally, being careful to define the internal hostname and service ports of your cluster.

The following example uses the defaults ports and supported service names.

```
<service>
  <role>AMBARI</role>
  <url>http://ambari-host:8080</url>
</service>

<service>
  <role>AMBARIUI</role>
  <url>http://ambari-host:8080</url>
</service>

<service>
  <role>ATLAS</role>
  <url>http://atlas-host:8443</url>
</service>

<service>
  <role>HIVE</role>
  <url>http://hive-host:10001/cliservice</url>
</service>

<service>
  <role>JOBTRACKER</role>
  <url>rpc://jobtracker-host:8050</url>
</service>

<service
  <role>NAMENODE</role>
  <url>hdfs://namenode-host:8020</url>
</service>

<service>
  <role>OOZIE</role>
  <url>http://oozie-host:11000/oozie</url>
</service>

<service>
  <role>RANGER</role>
  <url>http://ranger-host:6080</url>
</service>

<service>
  <role>RANGERUI</role>
  <url>http://ranger-host:6080</url>
</service>

<service>
  <role>RESOURCEMANAGER</role>
  <url>http://hive-host:8088/ws</url>
</service>

<service>
  <role>WEBHBASE</role>
  <url>http://webhbase-host:60080</url>
</service>

<service>
  <role>WEBHCAT</role>
  <url>http://webcat-host:50111/templeton</url>
```

```

</service>

<service>
  <role>WEBHDFS</role>
  <url>http://webhdfs-host:50070/webhdfs</url>
</service>

<service>
  <role>ZEPPELINUI</role>
  <url>http://zeppelin-host:9995</url>
</service>

<service>
  <role>ZEPPELINWS</role>
  <url>http://zeppelin-host:9995/ws</url>
</service>

```

2.8.4.3. Validating Service Connectivity

Use the commands in this section to test connectivity between the gateway host and the Hadoop service, and then test connectivity between an external client to the Hadoop service through the gateway.



Tip

If the communication between the gateway host and an internal Hadoop service fails, telnet to the service port to verify that the gateway is able to access the cluster node. Use the hostname and ports you specified in the service definition.

Testing WebHDFS by getting the home directory

- At the gateway host, enter the following command:

```
curl http://$webhdfs-host:50070/webhdfs/v1?op=GETHOMEDIRECTORY
```

The host displays:

```
{ "Path": "/user/gopher" }
```

- At an external client, enter the following command:

```
curl https://$gateway-host:$gateway_port/$gateway/$cluster_name/
$webhdfs_service_name/v1?op=GETHOMEDIRECTORY
```

The external client displays:

```
{ "Path": "/user/gopher" }
```

Testing WebHCat/Templeton by getting the version

- At the gateway host, enter the following command:

```
curl http://$webhdfs-host:50111/templeton/v1/version
```

The host displays:

```
{ "supportedVersions": [ "v1" ], "version": "v1" }
```

- At an external client, enter the following command:

```
curl https://$gateway-host:$gateway_port/$gateway/$cluster_name/  
$webhcat_service_name/v1/version
```

The external client displays:

```
{"supportedVersions":["v1"],"version":"v1"}
```

Testing Oozie by getting the version

- At the gateway host, enter the following command:

```
curl http://$oozie-host:11000/oozie/v1/admin/build-version
```

The host displays:

```
{"buildVersion":"4.0.0.2.1.1.0-302"}
```

- At an external client, enter the following command:

```
curl https://$gateway-host:$gateway_port/$gateway/$cluster_name/  
$oozie_service_name/v1/admin/build-version
```

The external client displays:

```
{"buildVersion":"4.0.0.2.1.1.0-302"}
```

Testing HBase/Stargate by getting the version

- At the gateway host, enter the following command:

```
curl http://$hbase-host:17000/version
```

The host displays:

```
rest 0.0.2 JVM: Oracle Corporation 1.7.0_51-24.45-b08 OS: Linux 3.8.0-29-  
generic amd64 Server:jetty/6.1.26 Jersey:1.8:
```

- At an external client, enter the following command:

```
curl http://$hbase-host:17000/version
```

The external client displays:

```
rest 0.0.2 JVM: Oracle Corporation 1.7.0_51-24.45-b08 OS: Linux 3.8.0-29-  
generic amd64 Server:jetty/6.1.26 Jersey:1.8
```

Testing HiveServer2

Both of the following URLs return an authentication error, which users can safely ignore.

1. At the gateway host, enter:

```
curl http://$hive-host:10001/cliservice
```

2. At an external client, enter:

```
curl https://$gateway-host:$gateway_port/$gateway/$cluster_name/  
$hive_service_name/cliservice
```


2.8.4.4. Adding a New Service to the Knox Gateway

Services and service additions in the Knox Gateway are defined as extensions to existing Knox Gateway functionality that enable you to extend the gateway's capabilities. You use these services to convert information contained in the topology file to runtime descriptors.

The Knox Gateway supports a declarative way for you to "plug in" a new service into the gateway simply and easily by using the following two files:

- `service.xml`- file that contains the routes (paths) that the service will provide and the rewrite rules to bind these paths.
- `rewrite.xml` – file that contains the rewrite rules for the service.



Note

The `service.xml` file is required, whereas the `rewrite.xml` file is optional.

2.8.4.4.1. Service Directory Structure

The Knox Gateway consists of a directory structure that you should become familiar with before attempting to add a new service to the gateway.

If you navigate to the data directory in your Knox home directory (`{GATEWAY_HOME}/data`), you will see the following directory structure:

```
Services
  Service name
  Version
    service.xml
    rewrite.xml
```

For example, if you were to navigate to the WebHDFS Service directory, you would see the following directory structure:

```
Services
  WebHDFS
  2.4.0
    service.xml
    rewrite.xml
```

2.8.4.4.2. Adding a New Service to the Knox Gateway

Adding a new service to the Knox gateway is a very easy and straightforward process, only requiring you to perform a few simple steps, which are listed below.

1. Navigate to the services directory in your Knox gateway HOME directory (`{GATEWAY_HOME}/data/services`)
2. Add the `service.xml` and `rewrite.xml` files to the directory.



Note

If you want to add the service to the Knox build, then add the `service.xml` and `rewrite` files to the `gateway-services-definitions` module.

2.8.5. Mapping the Internal Nodes to External URLs

Hostmapping is an advanced configuration topic. Generally, it is only required in deployments in virtualized environments, such as Cloud deployments and some development and testing environments.

The isolation of the Hadoop cluster is accomplished through virtualization that will hide the internal networking details (such as IP addresses and/or hostnames) from the outside world, while exposing other IP addresses and/or hostnames for use by clients accessing the cluster from outside of the virtualized environment. The exposed IP addresses and hostnames are available for use in the topology descriptor service definitions. This configuration works great for requests that are initiated from the external clients themselves which only ever use the Knox Gateway exposed endpoints.

Difficulties from these virtualized environments arise when the Hadoop cluster redirects client requests to other nodes within the cluster and indicates the internal hostname locations, rather than those designated to be exposed externally. Since the Hadoop services don't know or care whether a request is coming from an external or internal client, it uses its only view of the cluster, which is the internal details of the virtualized environment.

The Knox Gateway needs to know how to route a request that has been redirected by the Hadoop service to an address that is not actually accessible by the gateway. Hostmapping acts as an adapter that intercepts the redirects from the Hadoop service and converts the indicated internal address to a known external address that Knox will be able to route to once the client resends the request through the client facing gateway endpoint. The gateway uses the hostmap to replace the internal hostname within the routing policy for the particular request with the externally exposed hostname. This enables the dispatching from the Knox Gateway to successfully connect to the Hadoop service within the virtualized environment. Otherwise, attempting to route to an internal-only address will result in connection failures.

A number of the REST API operations require multi-step interactions that facilitate the client's interaction with multiple nodes within a distributed system such as Hadoop. External clients performing multi-step operations use the URL provided by the gateway in the responses to form the next request. Since the routing policy is hidden by the gateway from the external clients, the fact that the subsequent requests in the multi-stepped interaction are mapped to the appropriate externally exposed endpoints is not exposed to the client.

For example, when uploading a file with WebHDFS service:

1. The external client sends a request to the gateway WebHDFS service.
2. The gateway proxies the request to WebHDFS using the service URL.
3. WebHDFS determines which DataNodes to create the file on and returns the path for the upload as a Location header in a HTTP redirect, which contains the datanode host information.
4. The gateway augments the routing policy based on the datanode hostname in the redirect by mapping it to the externally resolvable hostname.
5. The external client continues to upload the file through the gateway.

6. The gateway proxies the request to the datanode by using the augmented routing policy.
7. The datanode returns the status of the upload and the gateway again translates the information without exposing any internal cluster details.

2.8.5.1. Setting Up a Hostmap Provider

Add the `hostmap` provider to the cluster topology descriptor and a parameter for each `DataNode` in the cluster, as follows:

1. Open the cluster topology descriptor file, `$cluster-name.xml`, in a text editor.
2. Add the Hostmap provider to `topology/gateway` using the following format:

```
<provider>
  <role>hostmap</role>
  <name>static</name>
  <enabled>true</enabled>
  <param>
    <name>$external-name</name>
    <value>$internal-dn-host</value>
  </param>
</provider>
```

where:

- `$cluster-name.xml` is the name of the topology descriptor file, located in `$gateway/conf/topologies`.
 - `$external-name` is the value that the gateway uses to replace `$internal_host` host names in responses.
 - `$internal-dn-host` is a comma-separated list of host names that the gateway will replace when rewriting responses.
3. To the `hostmap` provider, add a `param` for each additional `DataNode` in your cluster:

```
<param> <name> $external-name2 </name> <value> $internal-dn2-host </
value> </param>
```

4. Save the file.

Saving the results automatically deploys the topology with the change. The result is the creation of a new WAR file with modified timestamp in `$gateway/data/deployments`.

2.8.5.2. Example of an EC2 Hostmap Provider

In this EC2 example two VMs have been allocated. Each VM has an external hostname by which it can be accessed via the internet. However the EC2 VM is unaware of this external host name, and instead is configured with the internal hostname.

- **External hostnames** - `ec2-23-22-31-165.compute-1.amazonaws.com`,
`ec2-23-23-25-10.compute-1.amazonaws.com`

- **Internal hostnames** - ip-10-118-99-172.ec2.internal, ip-10-39-107-209.ec2.internal

The following shows the Hostmap definition required to allow access external to the Hadoop cluster via the Apache Knox Gateway.

```
<topology>
  <gateway>
    ...
    <provider>
      <role>hostmap</role>
      <name>static</name>
      <enabled>true</enabled>
      <!-- For each host enter a set of parameters -->
      <param>
        <name>ec2-23-22-31-165.compute-1.amazonaws.com</name>
        <value>ip-10-118-99-172.ec2.internal</value>
      </param>
      <param>
        <name>ec2-23-23-25-10.compute-1.amazonaws.com</name>
        <value>ip-10-39-107-209.ec2.internal</value>
      </param>
    </provider>
    ...
  </gateway>
  <service>
    ...
  </service>
  ...
</topology>
```

2.8.5.3. Example of Sandbox Hostmap Provider

Hortonwork's Sandbox 2.x poses a different challenge for hostname mapping. This Sandbox version uses port mapping to make Sandbox appear as though it is accessible via localhost. However, Sandbox is internally configured to consider sandbox.hortonworks.com as the hostname. So from the perspective of a client accessing Sandbox the external host name is localhost.

The following shows the hostmap definition required to allow access to Sandbox from the local machine:

```
<topology>
  <gateway>
    ...
    <provider>
      <role>hostmap</role>
      <name>static</name>
      <enabled>true</enabled>
      <param>
        <name>localhost</name>
        <value>sandbox,sandbox.hortonworks.com</value>
      </param>
    </provider>
    ...
  </gateway>
  ...
</topology>
```

2.8.5.4. Enabling Hostmap Debugging



Warning

Changing the rootLogger value from ERROR to DEBUG generates a large amount of debug logging.

Enable additional logging by editing the `gateway-log4j.properties` file in the directory.

1. Edit the `$gateway /conf/gateway-log4j.properties` file to enable additional logging.
2. Change ERROR to DEBUG on the following line:

```
log4j.rootLogger=ERROR, drfa
```



Warning

Changing the rootLogger value from ERROR to DEBUG generates a large amount of debug logging.

3. Stop and then restart the gateway:

```
cd $gateway bin/gateway.sh stop bin/gateway.sh start
```

2.8.6. Configuring Authentication

Apache Knox Gateway supports authentication using either an LDAP or federation provider for each configured cluster. This section explains how to configure authentication:

- [Authentication Providers \[130\]](#)
- [Setting Up LDAP Authentication \[130\]](#)
- [Setting Up SPNEGO Authentication \[135\]](#)
- [Configuring Advanced LDAP Authentication \[132\]](#)
- [LDAP Authentication Caching \[139\]](#)
- [Example Active Directory Configuration \[141\]](#)
- [Example OpenLDAP Configuration \[144\]](#)
- [Testing an LDAP Provider \[144\]](#)
- [Setting Up HeaderPreAuth Federation Provider \[144\]](#)
- [Example SiteMinder Configuration \[147\]](#)
- [Testing HTTP Header Tokens \[148\]](#)
- [Setting Up 2-Way SSL Authentication \[148\]](#)



Note

For information on how to configure an identity assertion provider, see [Configuring Identity Assertion](#).

2.8.6.1. Authentication Providers

There are two types of providers supported in Knox for establishing a user's identity:

- Authentication Providers
- Federation Providers

Authentication providers directly accept a user's credentials and validates them against some particular user store. Federation providers, on the other hand, validate a token that has been issued for the user by a trusted Identity Provider (IdP).

Authentication Providers

Providers have a name-value based configuration. There are different authentication providers:

- Anonymous
 - Used by Knox to let the proxied service or UI do its own authentication.
- [Setting Up LDAP Authentication \[130\]](#)
 - For LDAP/AD authentication with username and password. No SPNEGO/Kerberos support.
- [Setting Up SPNEGO Authentication \[135\]](#)
 - For SPNEGO/Kerberos authentication with delegation tokens. No LDAP/AD support.
- [Setting up PAM Authentication \[137\]](#)
 - For PAM authentication with username and password, via ShiroProvider.

Federation Providers

There are different authentication providers:

- [Setting Up HeaderPreAuth Federation Provider \[144\]](#)
- [Setting up SSOCookieProvider Federation Provider \[146\]](#)
- [Setting up JWT Federation Provider \[146\]](#)
- [Setting up Pac4j Federation Provider \[146\]](#)

2.8.6.2. Setting Up LDAP Authentication

LDAP authentication is configured by adding a "ShiroProvider" authentication provider to the cluster's topology file. When enabled, the Knox Gateway uses Apache Shiro (`org.apache.shiro.realm.ldap.JndiLdapRealm`) to authenticate users against the configured LDAP store.



Note

Knox Gateway provides HTTP BASIC authentication against an LDAP user directory. It currently supports only a single Organizational Unit (OU) and does not support nested OUs.

Steps

To enable LDAP authentication:

1. Open the cluster topology descriptor file, `$cluster-name.xml`, in a text editor.
2. Add the `ShiroProvider` authentication provider to `/topology/gateway` as follows:

```
<provider>
  <role>authentication</role>
  <name>ShiroProvider</name>
  <enabled>>true</enabled>
  <param>
    <name>main.ldapRealm</name>
    <value>org.apache.shiro.realm.ldap.JndiLdapRealm</value>
  </param>
  <param>
    <name>main.ldapRealm.userDnTemplate</name>
    <value>${USER_DN}</value>
  </param>
  <param>
    <name>main.ldapRealm.contextFactory.url</name>
    <value>${protocol}://${ldaphost}:${port}</value>
  </param>
  <param>
    <name>main.ldapRealm.contextFactory.authenticationMechanism</name>
    <value>simple</value>
  </param>
  <param>
    <name>urls./*</name>
    <value>${auth_type}</value>
  </param>
  <param>
    <name>sessionTimeout</name>
    <value>${minutes}</value>
  </param>
</provider>
```

where:

- `${USER_DN}`

is a comma-separated list of attribute and value pairs that define the User Distinguished Name (DN). The first pair must be set to "`attribute_name={0}`" indicating that the `attribute_name` is equal to the user token parsed from the request. For example, the first attribute in an OpenLdap definition is `UID={0}`. The `main.ldapRealm.userDnTemplate` parameter is only required when authenticating against an LDAP store that requires a full User DN.

- `protocol :// ldaphost : port`

is the URL of the LDAP service, Knox Gateway supports LDAP or LDAPS protocols.

- `$auth_type`

is either `authcBasic`, which provides basic authentication for both secured and non-secured requests, or `SSL authcBasic`, which rejects non-secured requests and provides basic authentication of secured requests.

- `$minutes`

is the session idle time in minutes, the default timeout is 30 minutes.

3. Save the file.

The gateway creates a new WAR file with modified timestamp in `$gateway/data/deployments`.



Note

If the Active Directory and Kerberos names differ in case (e.g. the Active Directory name is in upper case and the Kerberos name is lower case), the Knox Gateway enables you to resolve this conflict using the `auth_to_local` flag.

You can also configure LDAP authentication over SSL by following the steps below.

1. Change the LDAP protocol from `ldap://` to `ldaps://`.
2. If LDAP is using a self-signed certificate, then import the LDAP's certificate into the CACerts file of the Java Virtual Machine (JVM) being used to run the Apache Knox Gateway. To import the LDAP certificate, enter the following commands:

```
%JAVA_HOME%\bin\keytool
-import -trustcerts -alias ldap_ssl -file C:\temp\FileFromLDAP.cert -
keystore %JAVA_HOME%\jre/lib/security/cacerts -storepass "changeit"
```

2.8.6.3. Configuring Advanced LDAP Authentication

The default configuration computes the bind Distinguished Name (DN) for incoming user based on `userDnTemplate`. This does not work in enterprises where users could belong to multiple branches of LDAP tree. You could instead enable advanced configuration that would compute bind DN of incoming user with an LDAP search.

2.8.6.3.1. Using Advanced LDAP Authentication

With advanced LDAP authentication, we find the bind DN of the user by searching LDAP directory instead of interpolating bind DN from `userDNTemplate`.

Example 2.1. Example Search Filter to Find the Client Bind DN

Assuming:

- `ldapRealm.userSearchAttributeName=uid`
- `ldapRealm.userObjectClass=person`

- client specified login id = "guest"

LDAP Filter for doing a search to find the bind DN would be:

```
(&(uid=guest)(objectclass=person))
```

This could find the bind DN to be:

```
uid=guest,ou=people,dc=hadoop,dc=apache,dc=org
```

Please note that the userSearchAttributeName need not be part of bindDN.

For example, you could use

- ldapRealm.userSearchAttributeName=email
- ldapRealm.userObjectClass=person
- client specified login id = "john_doe@gmail.com"
- "

LDAP Filter for doing a search to find the bind DN would be:

```
(&(email=john_doe@gmail.com)(objectclass=person))
```

This could find bind DN to be

```
uid=johnd,ou=contractors,dc=hadoop,dc=apache,dc=org
```

2.8.6.3.2. Advanced LDAP Configuration Parameters

The table below provides a description and sample of the available advanced bind and search configuration parameters:

Parameter	Description	Default	Sample
principalRegex	Parses the principal for insertion into templates via regex.	(.*)	(.*)\\(.*) (e.g. match US\{tom: {0}=US\tom, {1}=US, {2}=tom)
userDnTemplate	Direct user bind DN template.	{0}	cn={2},dc={1},dc=qa,dc=company,dc=com
userSearchBase	Search based template. Used with config below.	none	dc={1},dc=qa,dc=company,dc=com
userSearchAttributeName	Attribute name for simplified search filter.	none	sAMAccountName
userSearchAttributeTemplate	Attribute template for simplified search filter.	{0}	{2}
userSearchFilter	Advanced search filter template. Note & is & in XML.	none	(& (objectclass=person) (sAMAccountName={2}))
userSearchScope	Search scope: subtree, onelevel, object.	subtree	onelevel

2.8.6.3.3. Advanced LDAP Configuration Combinations

There are a limited number of valid combinations of advanced LDAP configuration parameters:

- User DN Template
 - `userDnTemplate` (Required)
 - `principalRegex` (Optional)
- User Search by Attribute
 - `userSearchBase` (Required)
 - `userAttributeName` (Required)
 - `userAttributeTemplate` (Optional)
 - `userSearchScope` (Optional)
 - `principalRegex` (Optional)
- User Search by Filter
 - `userSearchBase` (Required)
 - `userSearchFilter` (Required)
 - `userSearchScope` (Optional)
 - `principalRegex` (Optional)

Advanced LDAP Configuration Precedence

The presence of multiple configuration combinations should be avoided. The rules below clarify which combinations take precedence when present.

- `userSearchBase` takes precedence over `userDnTemplate`
- `userSearchFilter` takes precedence over `userSearchAttributeName`

2.8.6.3.4. Advanced LDAP Authentication Errata

2.8.6.3.4.1. Problem with `userDnTemplate`-Based Authentication

`UserDnTemplate` based authentication uses configuration parameter `ldapRealm.userDnTemplate`. Typical value of `userDnTemplate` would look like `uid={0},ou=people,dc=hadoop,dc=apache,dc=org`.

To compute bind DN of the client, we swap the place holder `{0}` with login id provided by the client. For example, if the login id provided by the client is "guest", the computed bind DN would be `uid=guest,ou=people,dc=hadoop,dc=apache,dc=org`.

This keeps configuration simple.

However, this does not work if users belong to different branches of LDAP DIT. For example, if there are some users under `ou=people,dc=hadoop,dc=apache,dc=org` and some users under `ou=contractors,dc=hadoop,dc=apache,dc=org`,

We can not come up with `userDnTemplate` that would work for all the users.

2.8.6.3.4.2. Special Note on Parameter `main.ldapRealm.contextFactory.systemPassword`

The value for this could have one of the following two formats:

- `plaintextpassword`
- `${ALIAS=ldcSystemPassword}`

The first format specifies the password in plain text in the provider configuration. Use of this format should be limited for testing and troubleshooting.

We strongly recommend using the second format `${ALIAS=ldcSystemPassword}` in production. This format uses an alias for the password stored in credential store. In the example `${ALIAS=ldcSystemPassword}`, `ldcSystemPassword` is the alias for the password stored in credential store.

Assuming the plain text password is "hadoop", and your topology file name is "hdp.xml", you would use following command to create the right password alias in credential store.

```
{GATEWAY_HOME}/bin/knoxcli.sh create-alias ldcSystemPassword --cluster hdp --value hadoop
```

2.8.6.4. Setting Up SPNEGO Authentication

SPNEGO/Kerberos authentication is configured by adding a "HadoopAuth" authentication provider to the cluster's topology file. When enabled, the Knox Gateway uses Kerberos/SPNEGO to authenticate users to Knox.

About This Task

The HadoopAuth authentication provider for Knox integrates the use of the Apache Hadoop module for SPNEGO and delegation token-based authentication. This introduces the same authentication pattern used across much of the Hadoop ecosystem to Apache Knox and allows clients to using the strong authentication and SSO capabilities of Kerberos.

Steps

To enable SPNEGO authentication:

1. Open the cluster topology descriptor file, `$cluster-name.xml`, in a text editor.
2. Add the HadoopAuth authentication provider to `/topology/gateway` as follows:

```
<provider>
  <role>authentication</role>
  <name>HadoopAuth</name>
  <enabled>true</enabled>
  <param>
    <name>config.prefix</name>
    <value>hadoop.auth.config</value>
  </param>
  <param>
    <name>hadoop.auth.config.signature.secret</name>
    <value>knox-signature-secret</value>
  </param>
  <param>
    <name>hadoop.auth.config.type</name>
    <value>kerberos</value>
  </param>
```

```

<param>
  <name>hadoop.auth.config.simple.anonymous.allowed</name>
  <value>>false</value>
</param>
<param>
  <name>hadoop.auth.config.token.validity</name>
  <value>1800</value>
</param>
<param>
  <name>hadoop.auth.config.cookie.domain</name>
  <value>novalocal</value>
</param>
<param>
  <name>hadoop.auth.config.cookie.path</name>
  <value>gateway/default</value>
</param>
<param>
  <name>hadoop.auth.config.kerberos.principal</name>
  <value>HTTP/localhost@LOCALHOST</value>
</param>
<param>
  <name>hadoop.auth.config.kerberos.keytab</name>
  <value>/etc/security/keytabs/spnego.service.keytab</value>
</param>
<param>
  <name>hadoop.auth.config.kerberos.name.rules</name>
  <value>DEFAULT</value>
</param>
</provider>

```

Configuration parameter descriptions:

Name	Description	Default
config.prefix	If specified, all other configuration parameter names must start with the prefix.	none
signature.secret	This is the secret used to sign the delegation token in the hadoop.auth cookie. This same secret needs to be used across all instances of the Knox gateway in a given cluster. Otherwise, the delegation token will fail validation and authentication will be repeated each request.	a simple random number
type	This parameter needs to be set to kerberos.	none, would throw exception
simple.anonymous.allowed	This should always be false for a secure deployment.	true
token.validity	The validity -in seconds- of the generated authentication token. This is also used for the rollover interval when signer.secret.provider is set to random or zookeeper.	36000 seconds
cookie.domain	domain to use for the HTTP cookie that stores the authentication token	null
cookie.path	path to use for the HTTP cookie that stores the authentication token	null
kerberos.principal	The web-application Kerberos principal name. The Kerberos	null

Name	Description	Default
	principal name must start with HTTP/.... For example: HTTP/localhost@LOCALHOST	
kerberos.keytab	The path to the keytab file containing the credentials for the kerberos principal. For example: /Users/lmccay/lmccay.keytab	null
kerberos.name.rules	The name of the ruleset for extracting the username from the kerberos principal.	DEFAULT

3. Save the file.

The gateway creates a new WAR file with modified timestamp in `$gateway/data/deployments`.

REST Invocation

Once a user logs in with kinit, their Kerberos session may be used across client requests with things such as curl. The following curl command can be used to request a directory listing from HDFS while authenticating with SPNEGO via the `-negotiate` flag:

```
curl -k -i --negotiate -u : https://localhost:8443/gateway/sandbox/webhdfs/v1/tmp?op=LISTSTATUS
```

2.8.6.5. Setting up PAM Authentication

PAM authentication is configured by adding a "ShiroProvider" authentication provider to the cluster's topology file with PAM parameters. When enabled, the Knox Gateway uses Apache Shiro and the parameter `org.apache.hadoop.gateway.shirorealm.KnoxPamRealm` to authenticate users against the configured PAM store.

About This Task

There are a large number of pluggable authentication modules available for authenticating access to Hadoop through the Knox Gateway. ShiroProvider, in addition to LDAP support, also includes support for PAM-based authentication for unix-based systems.

This opens up the integration possibilities to many other readily-available authentication mechanisms, as well as other implementations for LDAP-based authentication. More flexibility may be available through various PAM modules for group lookup, more complicated LDAP schemas, or other areas where the KnoxLdapRealm is not sufficient.

The primary motivation for leveraging PAM-based authentication is to provide the ability to use the configuration provided by existing PAM modules that are available in a system's `/etc/pam.d/` directory.

The parameter `main.pamRealm.service` refers to the service located in `/etc/pam.d/login`.

Steps

1. In Ambari, add the ShiroProvider authentication provider to Knox>Configs>Advanced topology as follows:

```

<provider>
  <role>authentication</role>
  <name>ShiroProvider</name>
  <enabled>true</enabled>
  <param>
    <name>sessionTimeout</name>
    <value>30</value>
  </param>
  <param>
    <name>main.pamRealm</name>
    <value>org.apache.hadoop.gateway.shirorealm.KnoxPamRealm</value>
  </param>
  <param>
    <name>main.pamRealm.service</name>
    <value>login</value> </param>
  <param>
    <name>urls./**</name>
    <value>authcBasic</value>
  </param>
</provider>

```

2. Save the file.

Example of a PAM Configuration File

The Shiro configuration above refers to the `login` file contained in `/etc/pam.d`. The configuration of the `login` file can be modified for your deployment:

```

# login: auth account password session
auth      optional      pam_krb5.so use_kcminit
auth      optional      pam_ntlm.so try_first_pass
auth      optional      pam_mount.so try_first_pass
auth      required      pam_opendirectory.so try_first_pass
account   required      pam_nologin.so
account   required      pam_opendirectory.so
password  required      pam_opendirectory.so
session   required      pam_launchd.so
session   required      pam_uwtmp.so
session   optional      pam_mount.so

```

The first four fields are: `service-name`, `module-type`, `control-flag` and `module-filename`. The fifth and greater fields are for optional arguments that are specific to the individual authentication modules.

The second field in the configuration file is the `module-type`, it indicates which of the four PAM management services the corresponding module will provide to the application. Our sample configuration file refers to all four groups:

- `auth`: identifies the PAMs that are invoked when the application calls `pam_authenticate()` and `pam_setcred()`.
- `account`: maps to the `pam_acct_mgmt()` function.
- `session`: indicates the mapping for the `pam_open_session()` and `pam_close_session()` calls.
- `password`: group refers to the `pam_chauthtok()` function.

Generally, you only need to supply mappings for the functions that are needed by a specific application. For example, the standard password changing application, `passwd`, only requires a password group entry; any other entries are ignored.

The third field indicates what action is to be taken based on the success or failure of the corresponding module. Choices for tokens to fill this field are:

- `requisite`: Failure instantly returns control to the application indicating the nature of the first module failure.
- `required`: All these modules are required to succeed for `libpam` to return success to the application.
- `sufficient`: Given that all preceding modules have succeeded, the success of this module leads to an immediate and successful return to the application (failure of this module is ignored).
- `optional`: The success or failure of this module is generally not recorded.

The fourth field contains the name of the loadable module, `pam_*.so`. For the sake of readability, the full pathname of each module is not given. Before Linux-PAM-0.56 was released, there was no support for a default authentication-module directory. If you have an earlier version of Linux-PAM installed, you will have to specify the full path for each of the modules. Your distribution most likely placed these modules exclusively in one of the following directories: `/lib/security/` or `/usr/lib/security/`.

2.8.6.6. LDAP Authentication Caching

You can also configure the Apache Knox Gateway to cache LDAP authentication information by leveraging built-in caching mechanisms that the Shiro EhCache Manager provides. The ability to cache LDAP authentication information is useful in eliminating the need to authenticate against the LDAP server each time you use.



Note

When the authentication information is cached, the Knox gateway will not authenticate the user again until the cache expires.

To enable LDAP authentication caching using the Shiro Provider, follow the steps listed below.

1. Use the `org.apache.hadoop.gateway.shirorealm.knoxLdapRealm` in the Shiro configuration.
2. Set the `main.ldaprealm.authenticationcachingEnabled` property similar to the example shown below.

```
<provider>
  <role>authentication</role>
  <enabled>true</enabled>
  <param>
    <name>main.ldapRealm</name>
    <value>org.apache.hadoop.gateway.shirorealm.KnoxLdapRealm</value>
  </param>
  <param>
    <name>main.ldapGroupContextFactory</name>
```

```

        <value>org.apache.hadoop.gateway.shirorealm.KnoxLdapContextFactory</
value>
    </param>
    <param>
        <name>main.ldapRealm.ContextFactory</name>
        <value>$ldapGroupContextFactory</value>
    </param>
    <param>
        <name>main.ldapRealm.ContextFactory.url</name>
        <value>$ldap://localhost:33389</value>
    </param>
    <param>
        <name>main.ldapRealm.authorizationEnabled</name>
        <value>>true</value>
    </param>
    <param>
        <name>main.ldapRealm.searchBase</name>
        <value>ou-groups,dc=hadoop,dc=apache,dc=org</value>
    </param>
    <param>
        <name>main.cacheManager</name>
        <value>org.apache.knox.gateway.shirorealm.KnoxCacheManager</value>
    </param>
    <param>
        <name>main.securityManager.cacheManager</name>
        <value>$cacheManager</value>
    </param>
    <param>
        <name>main.ldapRealm.authenticationCachingEnabled</name>
        <value>>true</value>
    </param>
    <param>
        <name>main.ldapRealm.memberAttributeValueTemplate</name>
        <value>uid={0}ou=people,dc=hadoop,dc=apache,dc=org</value>
    </param>
    <param>
        <name>main.ldapRealm.contextFactory.systemUsername</name>
        <value>uid=guest,ou=people,dc=hadoop,dc=apache,dc=org</value>
    </param>
    <param>
        <name>main.ldapRealm.contextFactory.systemPassword</name>
        <value>guest=password</value>
    </param>
    <param>
        <name>urls./**</name>
        <value>authBasic</value>
    </param>
</provider>

```

In this example, you need to configure these properties to set the Knox Gateway for LDAP authentication caching. The Knox Gateway also includes several template topology files that you can use to test the caching function. You can locate these template files in the templates directory. To test the caching function, perform the steps listed below.

- a. Navigate to the Knox gateway HOME directory.

```
cd {GATEWAY_HOME}
```

- b. Copy the templates files to your sandbox.


```
cp templates/sandbox.knoxrealm.ehcache.xml
conf.topologies/sandbox.xml
```

- c. Start the LDAP authentication provider.

```
bin/ldap.sh start
```

- d. Start the Knox gateway.

```
bin/gateway.sh start
```

- e. Once the gateway is started, make the following WebHDFS API call:

```
curl -ivk -u tom:tom-password -X GET
https://localhost:8443/gateway/sandbox/webhdfs/v1?op=GETHOMEDIRECTORY
```

- f. To see LDAP authentication caching working, shut down the LDAP authentication provider.

```
bin/ldap.sh stop
```

- g. Run the WebHDFS API call again.

```
curl -ivk -u tom:tom=password -X GET
https://localhost:8443/gateway/sandbox/webhdfs/v1?op=GETHOMEDIRECTORY
```

2.8.6.7. Example Active Directory Configuration

Typically the AD `main.ldapRealm.userDnTemplate` value looks slightly different than OpenLDAP. The value for `main.ldapRealm.userDnTemplate` is only required if AD authentication requires the full User DN.



Note

If Active Directory allows authentication based on the Common Name (CN) and password only, then no value will be required for `main.ldapRealm.userDnTemplate`.

```
<topology>
  <gateway>
    <provider>
      <role>authentication</role>
      <name>ShiroProvider</name>
      <enabled>true</enabled>
      <param>
        <name>sessionTimeout</name>
        <value>30</value>
      </param>
      <param>
        <name>main.ldapRealm</name>
        <value>org.apache.hadoop.gateway.shirorealm.
KnoxLdapRealm</value>
      </param>
    <!-- changes for AD/user sync -->
```

```
<param>
  <name>main.ldapContextFactory</name>
  <value>org.apache.hadoop.gateway.shirorealm.KnoxLdapContextFactory</value>
</param>

<!-- main.ldapRealm.contextFactory needs to be placed before other main.
ldapRealm.contextFactory* entries -->
<param>
  <name>main.ldapRealm.contextFactory</name>
  <value>${ldapContextFactory}</value>
</param>

<!-- AD url -->
<param>
  <name>main.ldapRealm.contextFactory.url</name>
  <value>ldap://ad01.lab.hortonworks.net:389</value>
</param>

<!-- system user -->
<param>
  <name>main.ldapRealm.contextFactory.systemUsername</name>
  <value>cn=ldap-reader,ou=ServiceUsers,dc=lab,dc=hortonworks,dc=net</value>
</param>

<!-- pass in the password using the alias created earlier -->
<param>
  <name>main.ldapRealm.contextFactory.systemPassword</name>
  <value>${ALIAS=knoxLdapSystemPassword}</value>
</param>

      <param>
        <name>main.ldapRealm.contextFactory.
authenticationMechanism</name>
        <value>simple</value>
      </param>
      <param>
        <name>urls./*</name>
        <value>authcBasic</value>
      </param>

<!-- AD groups of users to allow -->
<param>
  <name>main.ldapRealm.searchBase</name>
  <value>ou=CorpUsers,dc=lab,dc=hortonworks,dc=net</value>
</param>
<param>
  <name>main.ldapRealm.userObjectClass</name>
  <value>person</value>
</param>
<param>
  <name>main.ldapRealm.userSearchAttributeName</name>
  <value>sAMAccountName</value>
</param>

<!-- changes needed for group sync-->
<param>
  <name>main.ldapRealm.authorizationEnabled</name>
  <value>>true</value>
</param>
<param>
```

```

    <name>main.ldapRealm.groupSearchBase</name>
    <value>ou=CorpUsers,dc=lab,dc=hortonworks,dc=net</value>
  </param>
  <param>
    <name>main.ldapRealm.groupObjectClass</name>
    <value>group</value>
  </param>
  <param>
    <name>main.ldapRealm.groupIdAttribute</name>
    <value>cn</value>
  </param>

  </provider>

  <provider>
    <role>identity-assertion</role>
    <name>Default</name>
    <enabled>true</enabled>
  </provider>

  <provider>
    <role>authorization</role>
    <name>XASecurePDPKnox</name>
    <enabled>true</enabled>
  </provider>

</gateway>

<service>
  <role>NAMENODE</role>
  <url>hdfs://{{namenode_host}}:{{namenode_rpc_port}}</url>
</service>

<service>
  <role>JOBTRACKER</role>
  <url>rpc://{{rm_host}}:{{jt_rpc_port}}</url>
</service>

<service>
  <role>WEBHDFS</role>
  <url>http://{{namenode_host}}:{{namenode_http_port}}/webhdfs/
url>
</service>

<service>
  <role>WEBHCAT</role>
  <url>http://{{webhcat_server_host}}:{{templeton_port}}/
templeton</url>
</service>

<service>
  <role>OOZIE</role>
  <url>http://{{oozie_server_host}}:{{oozie_server_port}}/
oozie</url>
</service>

<service>
  <role>WEBHBASE</role>
  <url>http://{{hbase_master_host}}:{{hbase_master_port}}</url>

```

```

        </service>

        <service>
            <role>HIVE</role>
            <url>http://{{hive_server_host}}:{{hive_http_port}}/
{{hive_http_path}}</url>
        </service>

        <service>
            <role>RESOURCEMANAGER</role>
            <url>http://{{rm_host}}:{{rm_port}}/ws</url>
        </service>
    </topology>

```

2.8.6.8. Example OpenLDAP Configuration

```

<provider>
  <role>authentication</role>
  <name>ShiroProvider</name>
  <enabled>true</enabled>
  <param>
    <name>main.ldapRealm</name>
    <value>org.apache.hadoop.gateway.shiorealm.KnoxLdapRealm</value>
  </param>
  <param>
    <name>main.ldapContextFactory</name>
    <value>org.apache.hadoop.gateway.shiorealm.KnoxLdapContextFactory</
value>
  </param>
  <param>
    <name>mainLdapRealm.contextFactory</name>
    <value>$ldapContextFactory</value>
  </param>
</provider>

```

2.8.6.9. Testing an LDAP Provider

Using cURL, you can test your LDAP configuration as follows:

1. Open the command line on an external client.



Note

cURL is not a built-in command line utility in Windows.

2. Enter the following command to list the contents of the directory *tmp/test*:

```
curl -i -k -u ldap_user : password -X GET / 'https:// gateway_host :8443/
gateway_path / cluster_name /webhdfs/api/v1/tmp/test?op=LISTSTATUS
```

If the directory exists, a content list displays; if the user cannot be authenticated, the request is rejected with an HTTP status of **401 unauthorized**.

2.8.6.10. Setting Up HeaderPreAuth Federation Provider

The Knox Gateway supports federation solution providers by accepting HTTP header tokens. This section explains how to configure HTTP header fields for SSO or Federation

solutions that have simple HTTP header-type tokens. For further information, see the [Authentication](#) chapter of the Apache Knox 0.12.0 User's Guide.

The gateway extracts the user identifier from the HTTP header field. The gateway can also extract the group information and propagate it to the Identity-Assertion provider.



Important

The Knox Gateway federation plug-in, `HeaderPreAuth`, trusts that the content provided in the authenticated header is valid. Using this provider requires proper network security.

Only use the `HeaderPreAuth` federation provider in environments where the identity system does not allow direct access to the Knox Gateway. Allowing direct access exposes the gateway to identity spoofing. Hortonworks recommends defining the `preauth.ip.addresses` parameter to ensure requests come from a specific IP addresses only.

To configure the HTTP header tokens:

1. Open the cluster topology descriptor file, `$cluster-name.xml`, in a text editor.
2. Add a `HeaderPreAuth` federation provider to `topology/gateway` as follows:

```
<provider>
  <role>federation</role>
  <name>HeaderPreAuth</name>
  <enabled>true</enabled>
  <param>
    <name>preauth.validation.method</name>
    <value>$validation_type</value>
  </param>
  <param>
    <name>preauth.ip.addresses</name>
    <value>$trusted_ip</value>
  </param>
  <param>
    <name>preauth.custom.header</name>
    <value>$user_field</value>
  </param>
  <param>
    <name>preauth.custom.group.header</name>
    <value>$group_field</value>
  </param>
</provider>
```

where the values of the parameters are specific to your environment:

- `$validation_type` (Optional, recommended)

Indicates the type of trust, use either `preauth.ip.validation` indicating to trust only connections from the address defined in `preauth.ip.addresses` OR null (omitted) indicating to trust all IP addresses.

- `$trusted_ip` (Required when the pre-authentication method is set to `preauth.ip.validation`)

A comma-separated list of IP addresses, addresses may contain a wild card to indicate a subnet, such as 10.0.0.*.

- `$user_field`

The name of the field in the header that contains the user name that the gateway extracts. Any incoming request that is missing the field is refused with **HTTP status 401, unauthorized**. If not otherwise specified, the default value is `SM_USER`.

- `$group_field` (Optional)

The name of the field in the header that contains the group name that the gateway extracts. Any incoming request that is missing the field results in no group name being extracted and the connection is allowed.

3. Save the file.

The gateway creates a new WAR file with modified timestamp in `$gateway/data/deployments`.

2.8.6.11. Setting up JWT Federation Provider

For information on the JWT federation provider, see the [Apache Knox documentation](#).

2.8.6.12. Setting up Pac4j Federation Provider

For information on the pac4j federation provider, see the [Apache Knox documentation](#).

2.8.6.13. Setting up SSOCookieProvider Federation Provider

About This Task

The `SSOCookieProvider` enables the federation of the authentication event that occurred through KnoxSSO. KnoxSSO is a typical service provider-initiated webSSO mechanism that sets a cookie to be presented by browsers to participating applications and cryptographically verified.

Knox Gateway needs a pluggable mechanism for consuming these cookies and federating the KnoxSSO authentication event as an asserted identity in its interaction with the Hadoop cluster for REST API invocations. This provider is useful when an application that is integrated with KnoxSSO for authentication also consumes REST APIs through the Knox Gateway.

Steps

To configure the `SSOCookieProvider`:

1. Open the cluster topology descriptor file, `$cluster-name.xml`, in a text editor.
2. Add a `SSOCookieProvider` federation provider to `topology/gateway` as follows:

```

<provider>
  <role>federation</role>
  <name>SSOCookieProvider</name>
  <enabled>>true</enabled>
  <param>
    <name>sso.authentication.provider.url</name>
    <value>https://host:port/gateway/idp/api/v1/websso</value>
  </param>
</provider>

```

where the values of the parameters are specific to your environment:

- <name>sso.authentication.provider.url</name></value>https://host:port/gateway/idp/api/v1/websso</value>

(Required) Indicates the location of the KnoxSSO endpoint and where to redirect the useragent when no SSO cookie is found in the incoming request.

3. Save the file.

Example

```

<topology>
  <gateway>
    <provider>
      <role>federation</role>
      <name>SSOCookieProvider</name>
      <enabled>>true</enabled>
      <param>
        <name>sso.authentication.provider.url</name>
        <value>https://localhost:9443/gateway/idp/api/v1/websso</value>
      </param>
    </provider>
    <provider>
      <role>identity-assertion</role>
      <name>Default</name>
      <enabled>>true</enabled>
    </provider>
  </gateway>
  <service>
    <role>WEBHDFS</role>
    <url>http://localhost:50070/webhdfs</url>
  </service>
  <service>
    <role>WEBHCAT</role>
    <url>http://localhost:50111/templeton</url>
  </service>
</topology>

```

2.8.6.14. Example SiteMinder Configuration

The following example is the bare minimum configuration for SiteMinder (with no IP address validation):

```

<provider>
  <role>federation</role>
  <name>HeaderPreAuth</name>
  <enabled>>true</enabled>
  <param>
    <name>preauth.custom.header</name>
    <value>SM_USER</value>
  </param>
  <param>
    <name>preauth.ip.addresses</name>
    <value>10.10.0.*</value>
  </param>
</provider>

```

2.8.6.15. Testing HTTP Header Tokens

Use following cURL command to request a directory listing from HDFS while passing in the expected header SM_USER, note that the example is specific to sandbox:

```
curl -k -i --header "SM_USER: guest" -v 'https://localhost:8443/gateway/sandbox/webhdfs/v1/tmp?op=LISTSTATUS'
```

Omitting the SM_USER: guest–header: guest” above results in a **HTTP status 401 unauthorized**

2.8.6.16. Setting Up 2-Way SSL Authentication

Mutual authentication with SSL provides the Knox gateway with the means to establish a strong trust relationship with another party. This is especially useful when applications that act on behalf of end-users send requests to Knox. While this feature does establish an authenticated trust relationship with the client application, it does not determine the end-user identity through this authentication. It will continue to look for credentials or tokens that represent the end-user within the request and authenticate or federate the identity accordingly.

To configure your Knox Gateway for 2-way SSL authentication, you must first configure the trust related elements within gateway-site.xml file. The table below lists the different elements that you can configure related to 2-way mutual authentication. Use following cURL command to request a directory listing from HDFS while passing in the expected header SM_USER, note that the example is specific to sandbox:

Table 2.24. gateway-site.xml Configuration Elements

Name	Description	Possible Values	Default Value
gateway.client.auth.needed	Flag used to specify whether authentication is required for client communications to the server.	TRUE/FALSE	FALSE
gateway.truststore.path	The fully-qualified path to the truststore that will be used.		gateway.jks
gateway.truststore.type	The type of keystore used for the truststore.		JKS
gateway.trust.allcerts	Flag used to specify whether certificates passed	TRUE/FALSE	FALSE

Name	Description	Possible Values	Default Value
	by the client should be automatically trusted.		
ssl.include.ciphers	A comma separated list of ciphers to accept for SSL.	See the JSSE Provider docs>The SunJSSE Provider >Cipher Suites for possible ciphers. These can also contain regular expressions as shown in the Jetty documentation .	
ssl.exclude.ciphers	A comma separated list of ciphers to reject for SSL.	See the JSSE Provider docs>The SunJSSE Provider >Cipher Suites for possible ciphers. These can also contain regular expressions as shown in the Jetty documentation .	

Once you have configured the `gateway-site.xml` file, all topologies deployed within the Knox gateway with mutual authentication enabled will require all incoming connections to present trusted client certificates during the SSL handshake process; otherwise, the server will be refuse the connection request.

2.8.7. Configuring Identity Assertion

The Knox Gateway `identity-assertion` provider determines which principal to propagate to the backend cluster service and represent the authenticated user. This allows the Knox Gateway to accept requests from external users and for the internal user to potentially be a product of a mapping, some transformation or other change to disambiguate the user identity within the cluster.

2.8.7.1. Identity Assertion Providers Overview

There are multiple options for Identity Assertion Provider, configured in Ambari under Knox>Configs>Advanced topology. The following `identity-assertion` providers can be used:

Table 2.25. Identity Assertion Providers

Provider	<name>	Use	Example
Default Identity Assertion Provider [152]	<name>Default</name>	The default identity assertion provider enables simple mapping of principal usernames and groups and is responsible for the establishing the identity that gets propagated to the cluster service as the effective user.	<pre><provider> <role>identity-assertion</role> <name>Default</name> <enabled>true</enabled> <param> <name>principal.mapping</name> <value>guest=hdfs;</value> </param> <param> <name>group.principal.mapping</name> <value>*=users;hdfs=admin</value> </param> </provider></pre>
Pseudo (deprecated)	<name>Pseudo</name> (deprecated)		

Provider	<name>	Use	Example
Concat Identity Assertion Provider [153]	<name>Concat</name>	The Concat identity assertion provider allows for composition of a new user principal through the concatenation of optionally configured prefix and/or suffix provider parameters. This is a useful assertion provider for converting an incoming identity into a disambiguated identity within the Hadoop cluster based on what topology is used to access Hadoop.	<pre><provider> <role>identity-assertion</role> <name>Concat</name> <enabled>true</enabled> <param> <name>concat.suffix</name> <value>_domain1</value> </param> </provider></pre>
SwitchCase Identity Assertion Provider [157]	<name>SwitchCase</name>	The SwitchCase identity assertion provider solves issues where down stream ecosystem components require user and group principal names to be a specific case. An example of how this provider is enabled and configured within the <gateway> section of a topology file is shown below.	<pre><provider> <role>identity-assertion</role> <name>SwitchCase</name> <param> <name>principal.case</name> <value>lower</value> </param> <param> <name>group.principal.case</name> <value>upper</value> </param> <enabled>true</enabled> </provider></pre>
Regular Expression Identity Assertion Provider [156]	<name>Regex</name>	The regular expression identity assertion provider allows incoming identities to be translated using a regular expression, template and lookup table. This will probably be most useful in conjunction with the HeaderPreAuth federation provider.	<pre><provider> <role>identity-assertion</role> <name>Regex</name> <enabled>true</enabled> <param> <name>input</name> <value>(.*?)@(.?*)\..*</value> </param> <param> <name>output</name> <value>{1}_{2}</value> </param> <param> <name>lookup</name> <value>us=USA;ca=CANADA</value> </param> </provider></pre>
Hadoop Group Lookup Identity Assertion Provider [154]	<name>HadoopGroupProvider</name>	The Hadoop Group Lookup identity assertion provider looks up the user's 'group membership' for authenticated users using Hadoop's group mapping service (GroupMappingServiceProvider). This allows existing investments in the Hadoop	<pre><provider> <role>identity-assertion</role> <name>HadoopGroupProvider</name> <enabled>true</enabled> <param> <name>hadoop.security.group.mapping</name></pre>

Provider	<name>	Use	Example
		<p>to be leveraged within Knox and used within the access control policy enforcement at the perimeter.</p>	<pre> <value>org. apache.hadoop.security. LdapGroupsMapping</ value> </param> <param> <name>hadoop.security. group.mapping.ldap.bind. user</name> <value>uid= tom,ou=people,dc=hadoop, dc=apache,dc=org</value> </param> <param> <name>hadoop.security. group.mapping.ldap.bind. password</name> <value>tom- password</value> </param> <param> <name>hadoop.security. group.mapping.ldap.url</ name> <value>ldap:// localhost:33389</value> </param> <param> <name>hadoop.security. group.mapping.ldap. base</name> <value></ value> </param> <param> <name>hadoop.security. group.mapping.ldap. search.filter.user</ name> <value>(& amp;((objectclass= person)(objectclass= applicationProcess))(cn= {0}))</value> </param> <param> <name>hadoop.security. group.mapping.ldap. search.filter.group</ name> <value>(objectclass= groupOfNames)</value> </param> <param> <name>hadoop.security. group.mapping.ldap. search.attr.member</ name> <value>member</value> </param> <param> <name>hadoop.security. group.mapping.ldap. search.attr.group.name</ name> </pre>

Provider	<name>	Use	Example
			<pre> <value>cn</ value> </param> </provider> </pre>



Note

You can only use one identity assertion provider at a time.

2.8.7.2. Default Identity Assertion Provider

The default identity assertion provider enables simple mapping of principal usernames and groups and is responsible for the establishing the identity that gets propagated to the cluster service as the effective user.

When you define the `Default identity-assertion` provider without parameters, the authenticated user is asserted as the authenticated user. For example, using simple assertion if a user authenticates as "guest", the user's identity for grouping, authorization, and running the request is "guest". `<name>Pseudo</name>` identity assertion was renamed `<name>Default</name>`, but both are supported in config.

To define a basic identity-assertion provider:

1. Open the cluster topology descriptor file, `$cluster-name.xml`, in a text editor.
2. Add a `Default identity-assertion` provider to `topology/gateway` as follows:

```

<provider>
  <role>identity-assertion</role>
  <name>Default</name>
  <enabled>true</enabled>
</provider>

```

```

<provider> <role>identity-assertion</role> <name>Default</name>
<enabled>true</enabled> </provider>

```

3. Save the file.

The gateway creates a new WAR file with modified timestamp in `$gateway/data/deployments`.

2.8.7.2.1. Mapping Authenticated Users to Other Users

The `principal.mapping` parameter of an `identity-assertion` provider determines the user name that the gateway asserts (uses as the authenticated user) for grouping, authorization, and to run the request on the cluster.



Note

If a user does not match a principal mapping definition, the authenticated user becomes the effective user.

To add user mapping rule to an identity-assertion provider:

1. Open the cluster topology descriptor file, `$cluster-name.xml`, in a text editor.
2. Add a Default identity-assertion provider to `topology/gateway` with the `principal.mapping` parameter as follows:

```
<provider>
  <role>identity-assertion</role>
  <name>Default</name>
  <enabled>true</enabled>
  <param>
    <name>principal.mapping</name>
    <value>$user_ids=$cluster_user;$user_ids=$cluster_user1;...</value>
  </param>
</provider>
```

where the value contains a semi-colon-separated list of external to internal user mappings, and the following variables match the names in your environment:

- `$user_ids`
is a comma-separated list of external users or the wildcard (*) indicates all users.
- `$cluster_user`
is the Hadoop cluster user name the gateway asserts, that is the authenticated user name.

3. Save the file.

The gateway creates a new WAR file with modified timestamp in `$gateway/data/deployments`.

2.8.7.2.2. Mapping Authenticated Users to Groups

The Knox Gateway uses group membership for Service Level Authorization only. The gateway does not propagate the user's group when communicating with the Hadoop cluster.

The `group.principal.mapping` parameter of the identity-assertion provider determines the user's group membership. The gateway evaluates this parameter after the `principal.mapping` parameter using the authenticated user. Unlike `principal.mapping`, the group mapping applies all the matching values. A user is a member of all matching groups.



Note

Although user and group mappings are often used together, the instructions in this section only explain how to add group mappings.

2.8.7.3. Concat Identity Assertion Provider

The Concat identity assertion provider allows for composition of a new user principal through the concatenation of optionally configured prefix and/or suffix provider parameters. This is a useful assertion provider for converting an incoming identity into a

disambiguated identity within the Hadoop cluster based on what topology is used to access Hadoop.

Concat Identity Assertion is a new provider for the Knox Gateway that enables you to map principals by concatenating strings to either the front or the back of a specified username. The Identity Assertion Provider provides the critical function of determining the Identity Principal that you will want to use in your Hadoop cluster to represent the identity that has been authenticated at the gateway. For more information on the Identity Assertion Provider and how it is used in the Knox Gateway, refer to the Identity Assertion chapter in the Apache Knox 0.11.x User Guide. If you would like to convert the user principal into a value that represents an identity from a particular user domain, use a configuration similar to the below example.

```
<provider>
  <role>identity-assertion</role>
  <name>Concat</name>
  <enabled>>true</enabled>
  <param>
    <name>concat.suffix</name>
    <value>domain1</value>
  </param>
</provider>
```

Notice in this example that the `identity-assertion` role has been named `Concat` and has been enabled (`true`) for the Identity Assertion Provider, with the `concat.suffix` parameter given a value of `domain1` and concatenation will occur at the end of the username (`concat.suffix`). You may also use a parameter called `concat.prefix` to indicate a value to concatenate to the front of the username.

2.8.7.4. Hadoop Group Lookup Identity Assertion Provider

The Hadoop Group Lookup identity assertion provider looks up user's 'group membership' for authenticated users using Hadoop's group mapping service (`GroupMappingServiceProvider`).

This allows existing investments in the Hadoop to be leveraged within Knox and used within the access control policy enforcement at the perimeter.

2.8.7.4.1. Using `GroupMappingServiceProvider` to Configure Group Mapping

The 'role' for this provider is 'identity-assertion' and name is 'HadoopGroupProvider':

```
<provider>
  <role>identity-assertion</role>
  <name>HadoopGroupProvider</name>
  <enabled>true</enabled>
  <<param> ... </param>
</provider>
```

Configuration

All the configuration for 'HadoopGroupProvider' resides in the provider section in a gateway topology file. The 'hadoop.security.group.mapping' property determines the implementation. Some of the valid implementations are as follows:

- `org.apache.hadoop.security.JniBasedUnixGroupsMappingWithFallback`

This is the default implementation and will be picked up if 'hadoop.security.group.mapping' is not specified. This implementation will determine if the Java Native Interface (JNI) is available. If JNI is available, the implementation will use the API within Hadoop to resolve a list of groups for a user. If JNI is not available then the shell implementation, `org.apache.hadoop.security.ShellBasedUnixGroupsMapping`, is used, which shells out with the 'bash -c groups' command (for a Linux/Unix environment) or the 'net group' command (for a Windows environment) to resolve a list of groups for a user.

- `org.apache.hadoop.security.LdapGroupsMapping`

This implementation connects directly to an LDAP server to resolve the list of groups. However, this should only be used if the required groups reside exclusively in LDAP, and are not materialized on the Unix servers.

Example 2.2. GroupMappingServiceProvider Example

The following example snippet works with the demo LDAP server that ships with Apache Knox. Replace the existing 'Default' identity-assertion provider with the one below (HadoopGroupProvider):

```
<provider>
  <role>identity-assertion</role>
  <name>HadoopGroupProvider</name>
  <enabled>true</enabled>
  <param>
    <name>hadoop.security.group.mapping</name>
    <value>org.apache.hadoop.security.LdapGroupsMapping</value>
  </param>
  <param>
    <name>hadoop.security.group.mapping.ldap.bind.user</name>
    <value>uid=tom,ou=people,dc=hadoop,dc=apache,dc=org</value>
  </param>
  <param>
    <name>hadoop.security.group.mapping.ldap.bind.password</name>
    <value>tom-password</value>
  </param>
  <param>
    <name>hadoop.security.group.mapping.ldap.url</name>
    <value>ldap://localhost:33389</value>
  </param>
  <param>
    <name>hadoop.security.group.mapping.ldap.base</name>
    <value></value>
  </param>
  <param>
    <name>hadoop.security.group.mapping.ldap.search.filter.user</name>
    <value>(&(|(objectclass=person)(objectclass=
applicationProcess))(cn={0}))</value>
  </param>
  <param>
    <name>hadoop.security.group.mapping.ldap.search.filter.group</
name>
    <value>(objectclass=groupOfNames)</value>
  </param>
  <param>
    <name>hadoop.security.group.mapping.ldap.search.attr.member</name>
```

```

        <value>member</value>
      </param>
    <param>
      <name>hadoop.security.group.mapping.ldap.search.attr.group.name</
name>
      <value>cn</value>
    </param>
  </provider>

```

Here, we are working with the demo LDAP server running at 'ldap://localhost:33389' which populates some dummy users for testing that we will use in this example. This example uses the user 'tom' for LDAP binding. If you have different LDAP/AD settings you will have to update the properties accordingly.

Test the setup using the following command (assuming the gateway is started and listening on localhost:8443). Note that we are using credentials for the user 'sam' along with the command: `curl -i -k -u sam:sam-password -X GET 'https://localhost:8443/gateway/sandbox/webhdfs/v1/?op=LISTSTATUS'`.

The command should be executed successfully and you should see the groups 'scientist' and 'analyst' to which user 'sam' belongs to in gateway-audit.log: `|| a99aa0ab-fc06-48f2-8df3-36e6fe37c230 | audit | WEBHDFS | sam | || identity-mapping | principal | sam | success | Groups: [scientist, analyst]`

2.8.7.5. Regular Expression Identity Assertion Provider

The regular expression identity assertion provider allows incoming identities to be translated using a regular expression, template and lookup table. This will probably be most useful in conjunction with the HeaderPreAuth federation provider.

There are three configuration parameters used to control the behavior of the provider:

Parameter	Description
<i>input</i>	This is a regular expression that will be applied to the incoming identity. The most critical part of the regular expression is the group notation within the expression. In regular expressions, groups are expressed within parenthesis. For example in the regular expression "(.*)@(.*)..*" there are two groups. When this regular expression is applied to "nobody@us.imaginary.tld" group 1 matches "nobody" and group 2 matches "us".
<i>output</i>	This is a template that assembles the result identity. The result is assembled from the static text and the matched groups from the input regular expression. In addition, the matched group values can be looked up in the lookup table. An output value of "{1}_{2}" of will result in "nobody_us".
<i>lookup</i>	This lookup table provides a simple (albeit limited) way to translate text in the incoming identities. This configuration takes the form of "=" separated name values pairs separated by ";". For example a lookup setting is "us=USA;ca=CANADA". The lookup is invoked in the output setting by surrounding the desired group number in square brackets (i.e. []). Putting it all together, output setting of "{1}_{[2]}" combined with input of "(.*)@(.*)..*" and lookup of "us=USA;ca=CANADA" will turn "nobody@us.imaginary.tld" into "nobody@USA".

Within the topology file the provider configuration might look like this:


```

<provider>
  <role>identity-assertion</role>
  <name>Regex</name>
  <enabled>>true</enabled>
  <param>
    <name>input</name>
    <value>(.*)(.*)\..*</value>
  </param>
  <param>
    <name>output</name>
    <value>{1}_{[2]}</value>
  </param>
  <param>
    <name>lookup</name>
    <value>us=USA;ca=CANADA</value>
  </param>
</provider>

```

Using curl with this type of configuration might produce the following results:

```

curl -k --header "SM_USER: nobody@us.imaginary.tld" 'https://localhost:8443/gateway/sandbox/webhdfs/v1?op=GETHOMEDIRECTORY'

{"Path": "/user/member_USA"}

url -k --header "SM_USER: nobody@ca.imaginary.tld" 'https://localhost:8443/gateway/sandbox/webhdfs/v1?op=GETHOMEDIRECTORY'

{"Path": "/user/member_CANADA"}

```

2.8.7.6. SwitchCase Identity Assertion Provider

The SwitchCase identity assertion provider solves issues where down stream ecosystem components require user and group principal names to be a specific case.

An example of how this provider is enabled and configured within the <gateway> section of a topology file is shown below. This particular example will switch user principals names to lower case and group principal names to upper case:

```

<provider>
  <role>identity-assertion</role>
  <name>SwitchCase</name>
  <param>
    <name>principal.case</name>
    <value>lower</value>
  </param>
  <param>
    <name>group.principal.case</name>
    <value>upper</value>
  </param>
  <enabled>>true</enabled>
</provider>

```

These are the configuration parameters used to control the behavior of the provider.

Parameter	Description
<i>principal.case</i>	The case mapping of user principal names. Choices are: lower, upper, none. Defaults to lower.

Parameter	Description
<code>group.principal.case</code>	The case mapping of group principal names. Choices are: lower, upper, none. Defaults to setting of <code>principal.case</code> .

If no parameters are provided the full defaults will result in both user and group principal names being switched to lower case. A setting of “none” or anything other than “upper” or “lower” leaves the case of the principal name unchanged.

2.8.7.7. Configuring Group Mapping

There are two ways to configure group mapping:

- [Mapping Authenticated Users to Groups \[153\]](#)
- [Hadoop Group Lookup Identity Assertion Provider \[154\]](#)

2.8.8. Configuring Service Level Authorization



Note

Group membership is determined by the `identity-assertion` parameter `group.principal.mapping`.

Group membership is determined by the `identity-assertion` parameter `group.principal.mapping`.

2.8.8.1. Setting Up an Authorization Provider

The `ACLAuthz` provider determines who is able to access a service through the Knox Gateway by comparing the authenticated user, group, and originating IP address of the request to the rules defined in the authorization provider.

Configure the `AclsAuthz` provider as follows:

1. Open the cluster topology descriptor file, `$cluster-name.xml`, in a text editor.
2. Add a `AclsAuthz` authorization provider to `topology/gateway` with a parameter for each service as follows:

```
<provider>
  <role>authorization</role>
  <name>AclsAuthz</name>
  <enabled>>true</enabled>
  <param>
    <name>$service_name.acl.mode</name>
    <value>$mode</value>
  </param>
  <param>
    <name>$service_Name.acl</name>
    <value>$cluster_users;$groups_field;IP_field</value>
  </param>
  ...
</provider>
```

where:

- `$service_name` matches the name of a service element. For example, `webhdfs`.
- `$mode` determines how the identity context (the effective user, their associated groups, and the original IP address) is evaluated against the fields as follows:
 - `AND` specifies that the request must match an entry in all three fields of the corresponding `$service_name .acl` parameter.
 - `OR` specifies that the request only needs to match an entry in any field, `$users_field OR $groups_field, OR $IP_field`.



Note

The `$service_name .acl.mode` parameter is optional. When it is not defined, the default mode is `AND`; therefore requests to that service must match all three fields.

- `$cluster_users` is a comma-separated list of authenticated users. Use a wildcard (*) to match all users.
- `$groups_field` is a comma-separated list of groups. Use a wildcard (*) to match all groups.
- `$IP_field` is a comma-separated list of IPv4 or IPv6 addresses. An IP address in the list can contain wildcard at the end to indicate a subnet (for example: `192.168.*`). Use a wildcard (*) to match all addresses.

3. Save the file.

The gateway creates a new WAR file with modified timestamp in `$gateway/data/deployments`.

2.8.8.2. Examples of Authorization

The following examples illustrate how to define authorization rule types to restrict access to requests matching:

- **Only users in a specific group and from specific IP addresses**

The following rule is restrictive. It only allows the guest user in the admin group to access WebHDFS from a system with the IP address of either 127.0.0.2 or 127.0.0.3:

```
<provider>
  <role>authorization</role>
  <name>AclsAuthz</name>
  <enabled>true</enabled>
  <param>
    <name>webhdfs.acl</name>
    <value>guest;admin;127.0.0.2,127.0.0.3</value>
  </param>
</provider>
```

When the parameter `acl.mode` is not defined the default behavior is `ALL`, therefore following rule is the same as the one above:

```

<provider>
  <role>authorization</role>
  <name>AclsAuthz</name>
  <enabled>>true</enabled>
  <param>
    <name>webhdfs.acl.mode</name>
    <value>AND</value>
  </param>
  <param>
    <name>webhdfs.acl</name>
    <value>guest;admin;127.0.0.2,127.0.0.3</value>
  </param>
</provider>

```



Note

If Guest is not in the admin group, the request is denied.

- **Two of the three conditions**

The following rule demonstrates how to require two conditions, user and group but not IP address, using the Wildcard. The rule allows the guest user that belongs to the admin group to send requests from anywhere because the IP field contains an asterisk which matches all IP addresses:

```

<provider>
  <role>authorization</role>
  <name>AclsAuthz</name>
  <enabled>>true</enabled>
  <param>
    <name>webhdfs.acl</name>
    <value>guest;admin;*</value>
  </param>
</provider>

```

- **One of the three conditions**

When the `$service .acl.mode` parameter is set to OR, the request only needs to match one entry in any of the fields. The request fails with HTTP Status 403 unauthorized, if no conditions are met.

The following example allows:

- guest to send requests to WebHDFS from anywhere.
- Any user in the admin group to send requests to WebHDFS from anywhere.
- Any user, in any group, to send a request to WebHDFS from 127.0.0.2 or 127.0.0.3.

```
<provider>
  <role>authorization</role>
  <name>AclsAuthz</name>
  <enabled>true</enabled>
  <param>
    <name>webhdfs.acl.mode</name>
    <value>OR</value>
  </param>
  <param>
    <name>webhdfs.acl</name>
    <value>guest;admin;127.0.0.2,127.0.0.3</value>
  </param>
</provider>
```

- **Allow all requests**

The following rule grants all users, in any group, and from any IP addresses to access WebHDFS:



Note

When a wildcard is used in a field it matches any value. Therefore the Allow all requests example is the same as not defining an ACL.

```
<provider>
  <role>authorization</role>
  <name>AclsAuthz</name>
  <enabled>true</enabled>
  <param>
    <name>webhdfs.acl</name>
    <value>*,*,*</value>
  </param>
</provider>
```

2.8.9. Audit Gateway Activity

The Knox Gateway Audit Facility tracks actions that are executed by Knox Gateway per user request or that are produced by Knox Gateway internal events, such as topology deployments.



Tip

The Knox Audit module is based on the Apache log4j. You can customize the logger by changing the `log4j.appender.auditfile.Layout` property in `$gateway /conf/gateway-log4j.properties` to another class that extends `Log4j`. For detailed information see [Apache's log4j](#).

2.8.9.1. Audit Log Fields

Auditing events on the gateway are informational, the default auditing level is informational (INFO) and it cannot be changed.

The Audit logs located at `C:\hadoop\logs\knox\gateway-audit.log` have the following structure:

EVENT_PUBLISHING_TIMEROOT_REQUEST_ID | PARENT_REQUEST_ID | REQUEST_ID
| LOGGER_NAME | TARGET_SERVICE_NAME | USER_NAME | PROXY_USER_NAME |
SYSTEM_USER_NAME | ACTION | RESOURCE_TYPE | RESOURCE_NAME | OUTCOME |
LOGGING_MESSAGE

where:

- `EVENT_PUBLISHING_TIME` : contains the timestamp when record was written.
- `ROOT_REQUEST_ID` : Reserved, the field is empty.
- `PARENT_REQUEST_ID` : Reserved, the field is empty.
- `REQUEST_ID` : contains a unique value representing the request.
- `LOGGER_NAME` : contains the logger name. For example `audit`.
- `TARGET_SERVICE_NAME` : contains the name of Hadoop service. Empty indicates that the audit record is not linked to a Hadoop service. For example, an audit record for topology deployment.
- `USER_NAME` : contains the ID of the user who initiated session with Knox Gateway.
- `PROXY_USER_NAME` : contains the authenticated user name.
- `SYSTEM_USER_NAME` : Reserved, field is empty.
- `ACTION` : contains the executed action type. The value is either authentication, authorization, redeploy, deploy, undeploy, identity-mapping, dispatch, or access.
- `RESOURCE_TYPE` contains the resource type of the action. The value is either `uri`, `topology`, or `principal`.
- `RESOURCE_NAME` : contains the process name of the resource. For example, `topology` shows the inbound or dispatch request path and `principal` shows the name of mapped user.
- `OUTCOME` contains the action results, `success`, `failure`, or `unavailable`.
- `LOGGING_MESSAGE` contains additional tracking information, such as the HTTP status code.

2.8.9.2. Change Roll Frequency of the Audit Log

Audit records are written to the log file `/var/log/knox/gateway-audit.log` and by default roll monthly. When the log rolls, the date that it rolled is appended to the end of the current log file and a new one is created.

To change the frequency:

1. Open the `$gateway /conf/gateway-log4j.properties` file in a text editor.
2. Change the `log4j.appender.auditfile.DatePattern` as follows:

```
log4j.appender.auditfile.DatePattern = $interval
```

where `$interval` is one of the following:

Setting	Description
yyyy-MM	Rollover at the beginning of each month
yyyy-ww	Rollover at the first day of each week. The first day of the week depends on the locale.
yyyy-MM-dd	Rollover at midnight each day.
yyyy-MM-dd-a	Rollover at midnight and midday of each day.
yyyy-MM-dd-HH	Rollover at the top of every hour.
yyyy-MM-dd-HH-mm	Rollover at the beginning of every minute.



Tip

For more examples, see [Apache log4j: Class DailyRollingFileAppender](#).

3. Save the file.
4. Restart the gateway:

```
cd $gateway bin/gateway.sh stop bin/gateway.sh start
```

2.8.9.3. Configuring Storm Plugin Audit Log to File

When the Storm Ranger plugin sends audit logs to a file via `Log4jAuditProvider`, a specific configuration must be used.

```
<appenders>
....
  <RollingFile name="STORMAUDIT"
    fileName="${sys:storm.log.dir}/ranger_audit.log"
    filePattern="${sys:storm.log.dir}/ranger_audit.log.%i">
    <PatternLayout>
      <pattern>${pattern}</pattern>
    </PatternLayout>
    <Policies>
      <SizeBasedTriggeringPolicy size="100 MB"/> <!-- Or every 100 MB -->
    </Policies>
    <DefaultRolloverStrategy max="9"/>
  </RollingFile>
  <Loggers>
    .....
    <Logger name="xaaudit" level="info">
      <AppenderRef ref="STORMAUDIT"/>
    </Logger>
  </Loggers>
</appenders>
```

Storm uses `log4j2` format for `log4j` configurations. In `log4j.xml`, the name of the logger (in this case, "xaaudit") is needed and not the whole class name with logger name; this is handled by the `<Logger>` tag.

2.8.10. Gateway Security

The Knox Gateway offers the following security features:

- [Implementing Web Application Security \[164\]](#)
- [Configuring Knox With a Secured Hadoop Cluster \[166\]](#)

2.8.10.1. Implementing Web Application Security

The Knox Gateway is a Web API (REST) Gateway for Hadoop clusters. REST interactions are HTTP based, and therefore the interactions are vulnerable to a number of web application security vulnerabilities. The web application security provider allows you to configure protection filter plugins.



Note

The initial vulnerability protection filter is for Cross Site Request Forgery (CSRF). Others will be added in future releases.

2.8.10.2. Configuring Protection Filter Against Cross Site Request Forgery Attacks

A Cross Site Request Forgery (CSRF) attack attempts to force a user to execute functionality without their knowledge. Typically the attack is initiated by presenting the user with a link or image that when clicked invokes a request to another site with which the user already has an established an active session. CSRF is typically a browser based attack.

The only way to create a HTTP request from a browser with a custom HTTP header is to use Javascript XMLHttpRequest or Flash, etc. Browsers have built-in security that prevent web sites from sending requests to each other unless specifically allowed by policy. This means that a website `www.bad.com` cannot send a request to `http://bank.example.com` with the custom header `X-XSRF-Header` unless they use a technology such as a XMLHttpRequest. That technology would prevent such a request from being made unless the `bank.example.com` domain specifically allowed it. This then results in a REST endpoint that can only be called via XMLHttpRequest (or similar technology).



Note

After enabling CSRF protection within the gateway, a custom header is required for all clients that interact with it, not just browsers.

To add a CSRF protection filter:

1. Open the cluster topology descriptor file, `$cluster-name.xml`, in a text editor.
2. Add a `WebAppSec` `webappsec` provider to `topology/gateway` with a parameter for each service as follows:

```
<provider>
  <role>webappsec</role>
  <name>WebAppSec</name>
  <enabled>true</enabled>
```



```

<param>
  <name>csrf.enabled</name>
  <value>${csrf_enabled}</value>
</param>
<param><!-- Optional -->
  <name>csrf.customHeader</name>
  <value>${header_name}</value>
</param>
<param><!-- Optional -->
  <name>csrf.methodsToIgnore</name>
  <value>${HTTP_methods}</value>
</param>
</provider>

```

where:

- `${csrf_enabled}` is either true or false.
- `${header_name}` when the optional parameter `csrf.customHeader` is present the value contains the name of the header that determines if the request is from a trusted source. The default, X-XSRF-Header, is described by the NSA in its guidelines for dealing with CSRF in REST.

`${http_methods}` when the optional parameter `csrf.methodsToIgnore` is present the value enumerates the HTTP methods to allow without the custom HTTP header. The possible values are GET, HEAD, POST, PUT, DELETE, TRACE, OPTIONS, CONNECT, or PATCH. For example, specifying GET allows GET requests from the address bar of a browser.

3. Save the file.

The gateway creates a new WAR file with modified timestamp in `${gateway}/data/deployments`.



Note

Make sure you have properly set your `$JAVA_HOME` variable in your user environment.

2.8.10.3. Validate CSRF Filtering

The following curl command can be used to request a directory listing from HDFS while passing in the expected header X-XSRF-Header.

```
curl -k -i --header "X-XSRF-Header: valid" -v -u guest:guest-password https://localhost:8443/gateway/sandbox/webhdfs/v1/tmp?op=LISTSTATUS
```



Note

The above LISTSTATUS request only works if you remove the GET method from the `csrf.methodsToIgnore` list.

Omitting the `--header "X-XSRF-Header: valid"` above results in an **HTTP 400 bad_request**. Disabling the provider, by setting `csrf.enabled` to false allows a request that is missing the header.

2.8.10.4. Configuring Knox With a Secured Hadoop Cluster

Once you have a Hadoop cluster that uses Kerberos for authentication, you must configure Knox to work with that cluster.

To enable the Knox Gateway to interact with a Kerberos-protected Hadoop cluster, add a Knox user and Knox Gateway properties to the cluster.

Do the following:

1. Find the fully-qualified domain name of the host running the gateway:

```
hostname -f
```

If the Knox host does not have a static IP address, you can define the Knox host as `*` for local developer testing.

2. At every Hadoop Master:

- Create a UNIX account for Knox:

```
useradd -g hadoop Knox
```

- Edit `core-site.xml` to include the following lines (near the end of the file):

```
<property>
  <name>hadoop.proxyuser.knox.groups</name>
  <value>users</value>
</property>

<property>
  <name>hadoop.proxyuser.knox.hosts</name>
  <value>${knox-host}</value>
</property>
```

where `${knox-host}` is the fully-qualified domain name of the host running the gateway.

- Edit `webhcat-site.xml` to include the following lines (near the end of the file):

```
<property>
  <name>hadoop.proxyuser.knox.groups</name>
  <value>users</value>
</property>

<property>
  <name>hadoop.proxyuser.knox.hosts</name>
  <value>${knox-host}</value>
</property>
```

where `${knox_host}` is the fully-qualified domain name of the host running the gateway.

3. At the Oozie host, edit `oozie-site.xml` to include the following lines (near the end of the file):

```
<property>
  <name>oozie.service.ProxyUserService.proxyuser.knox.groups</name>
  <value>users</value>
</property>

<property>
  <name>oozie.service.ProxyUserService.proxyuser.knox.hosts</name>
  <value>$knox-host</value>
</property>
```

where `$knox-host` is the fully-qualified domain name of the host running the gateway.

4. At each node running HiveServer2, edit `hive-site.xml` to include the following properties and values:

```
<property>
  <name>hive.server2.enable.doAs</name>
  <value>>true</value>
</property>

<property>
  <name>hive.server2.allow.user.substitution</name>
  <value>>true</value>
</property>

<property>
  <name>hive.server2.transport.mode</name>
  <value>http</value>
  <description>Server transport mode. "binary" or "http".</description>
</property>

<property>
  <name>hive.server2.thrift.http.port</name>
  <value>10001</value>
  <description>Port number when in HTTP mode.</description>
</property>

<property>
  <name>hive.server2.thrift.http.path</name>
  <value>cliservice</value>
  <description>Path component of URL endpoint when in HTTP mode.</
description>
</property>
```

2.8.11. Setting Up Knox Services for HA

This chapter describes how to set up the Knox Gateway for HA (high availability). Knox provides connectivity based failover functionality for service calls that can be made to more than one server instance in a cluster. Knox supports HA for HBase, Hive, Oozie, WebHCat, and WebHDFS.

Example:

```
<provider>
  <role>ha</role>
  <name>HaProvider</name>
  <enabled>>true</enabled>
```

```

    <param>
      <name>OOZIE</name>
      <value>maxFailoverAttempts=3;failoverSleep=1000;enabled=true</value>
    </param>
    <param>
      <name>HBASE</name>
      <value>maxFailoverAttempts=3;failoverSleep=1000;enabled=true</value>
    </param>
    <param>
      <name>WEBHCAT</name>
      <value>maxFailoverAttempts=3;failoverSleep=1000;enabled=true</value>
    </param>
    <param>
      <name>WEBHDFS</name>
      <value>maxFailoverAttempts=3;failoverSleep=1000;maxRetryAttempts=
300;retrySleep=1000;enabled=true</value>
    </param>
    <param>
      <name>HIVE</name>
      <value>maxFailoverAttempts=3;failoverSleep=1000;enabled=
true;zookeeperEnsemble=machine1:2181,machine2:2181,machine3:2181;
zookeeperNamespace=hiveserver2</value>
    </param>
  </provider>

<service>
  <role>OOZIE</role>
  <url>http://sandbox1:11000/oozie</url>
  <url>http://sandbox2:11000/oozie</url>
</service>
<service>
  <role>HBASE</role>
  <url>http://sandbox3:22000/hbase</url>
  <url>http://sandbox4:22000/hbase</url>
</service>
<service>
  <role>WEBHCAT</role>
  <url>http://sandbox5:33000/webhcat</url>
  <url>http://sandbox6:33000/webhcat</url>
</service>
<service>
  <role>WEBHDFS</role>
  <url>http://sandbox7:44000/webhdfs</url>
  <url>http://sandbox8:44000/webhdfs</url>
</service>
<service>
  <role>HIVE</role>
</service>

```

2.8.11.1. Prerequisites

Add the following configuration to the Knox>Configs>Advanced>Topology file:

```

<provider>
  <role>ha</role>
  <name>HaProvider</name>
  <enabled>true</enabled>

```

2.8.11.2. Configure WebHDFS for Knox

REST API access to HDFS in a Hadoop cluster is provided by [WebHDFS](#). The [WebHDFS REST API](#) documentation is available online. The following properties for Knox WebHDFS must be enabled in the `/etc/hadoop/conf/hdfs-site.xml` configuration file. The example values shown in these properties are from an installed instance of the Hortonworks Sandbox.

```
<property>
  <name>dfs.webhdfs.enabled</name>
  <value>>true</value>
</property>
<property>
  <name>dfs.namenode.rpc-address</name>
  <value>sandbox.hortonworks.com:8020</value>
</property>
<property>
  <name>dfs.namenode.http-address</name>
  <value>sandbox.hortonworks.com:50070</value>
</property>
<property>
  <name>dfs.https.namenode.https-address</name>
  <value>sandbox.hortonworks.com:50470</value>
</property>
```

The values above must be reflected in each topology descriptor file deployed to the gateway. The gateway by default includes a sample topology descriptor file located at `{GATEWAY_HOME}/deployments/sandbox.xml`. The values in the following sample are also configured to work with an installed Hortonworks Sandbox VM.

```
<service>
  <role>NAMENODE</role>
  <url>hdfs://localhost:8020</url>
</service>
<service>
  <role>WEBHDFS</role>
  <url>http://localhost:50070/webhdfs</url>
</service>
```

The URL provided for the NAMENODE role does not result in an endpoint being exposed by the gateway. This information is only required so that other URLs can be rewritten that reference the Name Node's RPC address. This prevents clients from needing to be aware of the internal cluster details.

2.8.11.3. Configure Knox for HA

Knox provides basic failover and retry functionality for REST API calls made to a service when service HA has been configured and enabled.

To enable HA functionality in Knox the following configuration must be added to the topology file:

Service	Parameter
WebHDFS	<pre><param> <name>WEBHDFS</name> <value>maxFailoverAttempts=3;failoverSleep= 1000;maxRetryAttempts=300;retrySleep=1000;enabled= true</value> </param></pre>

Service	Parameter
HBase	<pre><param> <name>HBASE</name> <value>maxFailoverAttempts= 3;failoverSleep=1000;enabled=true</value> </param></pre>
Hive	<pre><param> <name>HIVE</name> <value>maxFailoverAttempts= 3;failoverSleep=1000;enabled= true;zookeeperEnsemble=machine1:2181, machine2:2181,machine3:2181; zookeeperNamespace=hiveserver2</value> </param></pre>
Oozie	<pre><param> <name>OOZIE</name> <value>maxFailoverAttempts= 3;failoverSleep=1000;enabled=true</value> </param></pre>
WebHCat	<pre><param> <name>WEBHCAT</name> <value>maxFailoverAttempts= 3;failoverSleep=1000;enabled=true</value> </param></pre>

The various configuration parameters are described below:

- `maxFailoverAttempts` – The maximum number of times a failover will be attempted. The current failover strategy is very simplistic in that the next URL in the list of URLs provided for the service is used, and the one that failed is put at the bottom of the list. If the list is exhausted and the maximum number of attempts has not been reached, the first URL that failed will be tried again (the list will start again from the original top entry).
- `failoverSleep` – The amount of time in milliseconds that the process will wait or sleep before attempting to failover.
- `maxRetryAttempts` – The maximum number of times that a retry request will be attempted. Unlike failover, the retry is done on the same URL that failed. This is a special case in HDFS when the node is in safe mode. The expectation is that the node will come out of safe mode, so a retry is desirable here as opposed to a failover.
- `retrySleep` – The amount of time in milliseconds that the process will wait or sleep before a retry is issued.
- `enabled` - Flag to turn the particular service on or off for HA.

The additional configuration parameters for Hive are described below:

- `zookeeperEnsemble` – A comma separated list of host names (or IP addresses) of the zookeeper hosts that consist of the ensemble that the Hive servers register their information with. This value can be obtained from Hive's config file `hive-site.xml` as the value for the parameter `'hive.zookeeper.quorum'`.
- `zookeeperNamespace` – This is the namespace under which HiveServer2 information is registered in the ZooKeeper ensemble. This value can be obtained from Hive's config file `hive-site.xml` as the value for the parameter `'hive.server2.zookeeper.namespace'`.

For the service configuration itself, the additional URLs for standby nodes should be added to the list. The active URL (at the time of configuration) should ideally be added at the top of the list. Example for HBase, Oozie, WebHCat, and WebHDFS:

```
<service>
  <role>{COMPONENT}</role>
  <url>http://{host1}:50070/{component}</url>
  <url>http://{host2}:50070/{component}</url>
</service>
```

Example for Hive:

```
<service>
  <role>HIVE</role>
</service>
```

Please note that there is no `<url>` tag specified here as the URLs for the Hive servers are obtained from ZooKeeper.

2.8.12. Knox CLI Testing Tools

This section describes how to use the Knox CLI (Command Line Interface) to run diagnostic tests.

The Knox CLI is a command line utility that can be used to manage and test various aspects of a Knox deployment.

The `knoxcli.sh` command line utility script is located in the `{GATEWAY_HOME}/bin` directory.

2.8.12.1. Knox CLI LDAP Authentication and Authorization Testing

You can use the following command format to authenticate a user name and password against LDAP.

```
bin/knoxcli.sh user-auth-test [--cluster c] [--u username] [--p password] [--g] [--d] [--help]
```

This command will test a topology's ability to connect, authenticate, and authorize a user with an LDAP server. The only required argument is the `--cluster` argument to specify the name of the topology you wish to use. The topology must be valid (passes a `validate-topology` command). If the `-u` and `-p` arguments are not specified, you will be prompted for a user name and password.

If authentication is successful, the command will attempt to use the topology to do an LDAP group lookup. The topology must be configured correctly to do this. If it is not, groups will not be returned and no errors will be printed unless the `--g` argument is specified. Currently this command only works if a topology supports the use of ShiroProvider for authentication.

Table 2.26. LDAP Authentication and Authorization Arguments

Argument	Description	Required?
<code>--cluster</code>	The name of the cluster to authenticate.	Yes
<code>-u</code>	The user name to authenticate with.	No
<code>-p</code>	The password to authenticate with.	No
<code>-g</code>	Specifies that you want to return a user's groups. If not specified, group lookup errors will not be returned.	No

Argument	Description	Required?
-d	Print extra debug information for a failed authentication.	No

2.9. Knox SSO

Authentication of the Hadoop component UIs, and those of the overall ecosystem, is usually limited to Kerberos (which requires SPNEGO to be configured for the user's browser) and simple/psuedo. This often results in the UIs not being secured - even in secured clusters. This is where KnoxSSO provides value by providing WebSSO capabilities to the Hadoop cluster.

By leveraging the hadoop-auth module in Hadoop common, we have introduced the ability to consume a common SSO cookie for web UIs while retaining the non-web browser authentication through Kerberos/SPNEGO. We do this by extending the `AltKerberosAuthenticationHandler` class which provides the useragent-based multiplexing.

The flexibility of the Apache Knox authentication and federation providers allows KnoxSSO to provide normalization of authentication events through token exchange. resulting in a common JWT (JSON WebToken)-based token.

KnoxSSO provides an abstraction for integrating any number of authentication systems and SSO solutions, and enables participating web applications to scale to those solutions more easily. Without the token exchange capabilities offered by KnoxSSO, each component UI would need to integrate with each desired solution on its own. With KnoxSSO, they only need to integrate with the single solution and common token.

Table 2.27. Supported Component UIs: SSO

UI
Ambari
Atlas
Ranger Admin Console

2.9.1. Identity Providers (IdP)

Knox has two identity providers: form-based and SAML. It requires that LDAP authentication be configured for Ambari and that it be the same LDAP server as Knox SSO is using for form-based IdP.

2.9.1.1. Form-based Identity Provider (IdP)

The form-based identity provider (IdP) is the default identity provider for KnoxSSO out of the box and is installed by Ambari.

The installed configuration of the provider leverages the Shiro provider which attempts to authenticate a request by looking for HTTP basic credentials.

Instead of responding with an HTTP basic challenge, however, the browser is redirected to the KnoxAuth application to present the user with a form in which to submit username and password.

Example 2.3. knosso.xml with Shiro provider

The following knosso.xml topology file illustrates the use of the Shiro provider, the hosting of the knoaxauth application, and the configuration of the KNOXSSO service itself.

The typical Shiro provider configuration is augmented with new parameters for achieving the behavior described above.

The `restrictedCookies` parameter is used to add the WWW-Authenticate header in order to suppress the HTTP basic challenge.

The `redirectToUrl` parameter is used to indicate where to redirect the browser rather, than issuing an HTTP basic challenge.

Note, also, the `application` element which is used to indicate that a given application is to be hosted by the Knox gateway and how it relates to the `redirectToUrl` parameter in the Shiro provider.

The `knosso.xml` topology describes the manner in which a client acquires a KnoxSSO websso cookie/token. The Shiro provider allows the integration LDAP/AD with HTTP Basic Auth credentials.

```
<topology>
  <gateway>
    <provider>
      <role>webappsec</role>
      <name>WebAppSec</name>
      <enabled>true</enabled>
      <param>
        <name>xframe.options.enabled</name>
        <value>true</value>
      </param>
    </provider>
    <provider>
      <role>authentication</role>
      <name>ShiroProvider</name>
      <enabled>true</enabled>
      <param>
        <name>sessionTimeout</name>
        <value>30</value>
      </param>
      <param>
        <name>redirectToUrl</name>
        <value>/gateway/knosso/knoaxauth/login.html</value>
      </param>
      <param>
        <name>restrictedCookies</name>
        <value>rememberme,WWW-Authenticate</value>
      </param>
      <param>
        <name>main.ldapRealm</name>
        <value>org.apache.hadoop.gateway.shirorealm.KnoxLdapRealm</
value>
      </param>
      <param>
        <name>main.ldapContextFactory</name>
        <value>org.apache.hadoop.gateway.shirorealm.
KnoxLdapContextFactory</value>
      </param>
```

```

    <param>
      <name>main.ldapRealm.contextFactory</name>
      <value>$ldapContextFactory</value>
    </param>
    <param>
      <name>main.ldapRealm.userDnTemplate</name>
      <value>uid={0},ou=people,dc=hadoop,dc=apache,dc=org</value>
    </param>
    <param>
      <name>main.ldapRealm.contextFactory.url</name>
      <value>ldap://localhost:33389</value>
    </param>
    <param>
      <name>main.ldapRealm.authenticationCachingEnabled</name>
      <value>>false</value>
    </param>
    <param>
      <name>main.ldapRealm.contextFactory.authenticationMechanism</
name>
      <value>simple</value>
    </param>
    <param>
      <name>urls./*</name>
      <value>authcBasic</value>
    </param>
  </provider>
  <provider>
    <role>identity-assertion</role>
    <name>Default</name>
    <enabled>>true</enabled>
  </provider>
  <provider>
    <role>hostmap</role>
    <name>static</name>
    <enabled>>true</enabled>
    <param><name>localhost</name><value>sandbox,sandbox.hortonworks.
com</value></param>
  </provider>
</gateway>

<application>
  <name>knoxauth</name>
</application>

<service>
  <role>KNOXSSO</role>
  <param>
    <name>knoxsso.cookie.secure.only</name>
    <value>>true</value>
  </param>
  <param>
    <name>knoxsso.token.ttl</name>
    <value>30000</value>
  </param>
  <param>
    <name>knoxsso.redirect.whitelist.regex</name>
    <value>^https?:\\/(c64\d\d\.ambari\.apache\.org|localhost|127\.0\.
0\.1|0:0:0:0:0:0:1|::1):[0-9]*$</value>
  </param>
</service>

```

```
</topology>
```

2.9.1.2. SAML-based Identity Provider (IdP)

Apache Knox with KnoxSSO + pac4j provider enables the use of a number of new authentication and SSO solutions for accessing and developing KnoxSSO-enabled applications (including Ambari, Ranger, Hadoop UIs and custom built applications that utilize REST APIs through Knox.)

This section illustrates the integration of the Okta identity service offering by leveraging the pac4j provider SAML capabilities in Apache Knox. A similar flow to what is described below would be available for Ambari and Ranger, or any KnoxSSO participating application.

As opposed to the KnoxSSO form-based IdP, where the actual form is hosted by a Knox hosted authentication app, SAML IdPs need to have KnoxSSO instances configured within the SAML solution as participating in the SAML SSO flow as a service provider. This generally requires the creation of an "Application" in Okta and other providers which will contain the required endpoints and metadata required for verifying requests for login and redirecting users back to KnoxSSO after the authentication event is successful.

[Okta information on SAML-based SSO](#). KnoxSSO is the Service Provider, not the end application that is participating with KnoxSSO.

[Configuring the SAML application using the Okta SAML App Wizard](#).

Example 2.4. knoxsso.xml with Okta

The knoxsso.xml topology file will need to be changed from the form-based IdP configuration to the SAML-based IdP by swapping the Shiro provider with the pac4j provider for Knox.

The knoxsso.xml topology describes the manner in which a client acquires a KnoxSSO websso cookie/token. The pac4j federation provider allows the integration of a number of authentication solutions. In this case, the openid connect capability is being leveraged to integrate the cloud-based PrivaKey identity service.

The following topology file is an example for use with Okta.

```
<topology>
  <gateway>
    <provider>
      <role>federation</role>
      <name>pac4j</name>
      <enabled>true</enabled>
      <param>
        <name>pac4j.callbackUrl</name>
        <value>https://www.local.com:8443/gateway/knoxssso/api/v1/websso</value>
      </param>

      <param>
        <name>clientName</name>
        <value>SAML2Client</value>
      </param>

      <param>
        <name>saml.identityProviderMetadataPath</name>
```

```

        <value>https://dev-122415.oktapreview.com/app/exk5nc5z1xbFKb7nH0h7/
sso/saml/metadata</value>
    </param>

    <param>

        <name>saml.serviceProviderMetadataPath</name>
        <value>/tmp/sp-metadata.xml</value>
    </param>

    <param>
        <name>saml.serviceProviderEntityId</name>
        <value>https://www.local.com:8443/gateway/knoxssso/api/v1/websso?
pac4jCallback=true&amp;client_name=SAML2Client</value>
    </param>
</provider>
<provider>
    <role>identity-assertion</role>
    <name>Default</name>
    <enabled>true</enabled>
    <param>
        <name>principal.mapping</name>
        <value>guest@example.com=guest;</value>
    </param>
</provider>
</gateway>

<service>
    <role>KNOXSSO</role>
    <param>
        <name>knoxssso.cookie.secure.only</name>
        <value>true</value>
    </param>
    <param>
        <name>knoxssso.token.ttl</name>
        <value>100000</value>
    </param>
    <param>
        <name>knoxssso.redirect.whitelist.regex</name>
        <value>^https?:\:\/\/(www\.local\.com|localhost|127\.0\.0\.1|
0:0:0:0:0:0:1|::1):[0-9].*$</value>
    </param>
</service>
</topology>

```



Note

You must encode the ampersand within the `saml.serviceProviderEntityId` parameter as `&` and include a value for the `saml.serviceProviderMetadataPath` - the file location here doesn't need to exist. There is a bug that will throw an NPE if `saml.serviceProviderMetadataPath` is not included even though the actual metadata will be served up to the IdP via request.

In the above example, we have configured the Okta application to assert the user's ID as their email address. In order to leverage this as the identity, we need to map it to a username that will be valid within the Hadoop cluster. The identity assertion provider above does a simple mapping from a known email address to a known username. More

appropriate assertion provider usage would likely be to use the regex assertion provider that would allow you to extract the username from the email address.

We currently do not have support for establishing groups from the SAML assertion and must rely on the participating applications to do a group lookup based on the username.

2.9.2. Setting up Knox SSO for Ambari

This section describes how to configure Ambari to use Knox SSO (Single Sign-on) to authenticate users. With this configuration, unauthenticated users who try to access Ambari are redirected to the Knox SSO login page for authentication.

Use the following steps to configure Knox SSO for Ranger:

1. Log in as the root user
2. Run the following command:


```
ambari-server setup-sso
```
3. When prompted, enter **y**.
4. For the provider URL, enter: `https://<hostname>:8443/gateway/knoxssso/api/v1/webssso`.
5. Run the following CLI command to export the Knox certificate:

```
JAVA_HOME/bin/keytool -export -alias gateway-identity -rfc -file <cert.pem>
-keystore /usr/hdp/current/knox-server/data/security/keystores/gateway.jks
```

- When prompted, enter the Knox master password.
 - Note the location where you save the `cert.pem` file.
6. When prompted to configure advanced properties, enter **n**.
 7. Leave `JWT Cookie name (hadoop-jwt)` and `JWT audiences list` empty.

The prompt returns `Ambari Server 'setup-sso' completed successfully`.

8. Restart the Ambari Server: `ambari-server restart`.

Example 2.5. Example Knox SSO for Ambari

```
ambari-server setup-sso
Setting up SSO authentication properties...
Do you want to configure SSO authentication [y/n] (y)?y
Provider URL [URL] (http://example.com):https://c6402.ambari.apache.org:8443/
gateway/knoxssso/api/v1/webssso
Public Certificate pem (empty) (empty line to finish input):
MIICYTCCAcqgAwIBAgIIHd3j94bX9IMwDQYJKoZIhvcNAQEFBQAwczELMAkGA1UEBhMCVVMxDTAL
BgNVBAGTBFRlc3QxDTAALBgNVBACTBFRlc3QxDzANBgNVBAoTBkhZG9vcDENMAsGA1UECxMEVGVz
dDEmMCQGA1UEAxMda25veHNzby1za29uZXJ1LTItMi5ub3ZhbG9jYWwHhcNMTYwMzAxMTEzMTQ0
WWhcNMTcwMzAxMTEzMTQ0WjBzMQswCQYDVQQGEWJVUzENMAsGA1UECBMEVGVzZDENMAsGA1UEBxME
VGVzZDENMAsGA1UEChMGSGFkb29wMQ0wCwYDVQQLEwRUZXN0MSYwJAYDVQQDExlrbm94c3NvLXNrc
```

```
b251cnUtMi0yIm5vdmFsb2NhbDCBnzANBgkqhkiG9w0BAQEFAAOBjQAwgYkCgYEA1V0Jtd8zmzVZ
UZRBqxXvK9MV5OYIOWTX9/FMthwr99eClHp3JdZ1x3utYr9nwdZ6fjZaUIihzu8a8SGoipbW2ZVU
TShGZ/5VKtu96YcSAoB3VTyc3WWRDGERRs7aKAlEqnURDkQz7KR52tvItJpBBjrTXZpHKFTOecL4
hCkaalUCAwEAATANBgkqhkiG9w0BAQUFAAOBgQAqvPfl4fivozd+4QI4ZBohFHHvflz4Y7+DxlY7
iNAnjnau4W3wgwTt6CQ1B9fSx3zVTlhu2PfdJwvumBbuKuth/M+KXpG28AbKIojrL2Odlv+cftrJ
YeJC6Qjee+5Pf2P9G2wd9fahWF+aQpr50YlMZSU+VMiTO2a2FSAXvOdvA==
```

```
Do you want to configure advanced properties [y/n] (n) ?y
JWT Cookie name (hadoop-jwt):
JWT audiences list (comma-separated), empty for any ():
Ambari Server 'setup-ss0' completed successfully.

ambari-server restart
```

2.9.3. Setting up Knox SSO for Ranger Web UI

This section describes how to configure Ranger to use Knox SSO (Single Sign-on) to authenticate users on an Ambari cluster. With this configuration, unauthenticated users who try to access Ranger are redirected to the Knox SSO login page for authentication.



Note

- This task describes Knox SSO as only applied to web UI users. To enable Knox SSO for Ranger REST APIs, see [Setting up the Knox Token Service for Ranger APIs](#).
- Internal Ranger users have the option to bypass Knox SSO and log in to the Ranger UI directly using the "locallogin" URL: `http://<ranger_host>:6080/locallogin`.

Use the following steps to configure Knox SSO for Ranger:

1. Install Ambari with HDP-2.5 or higher. Install Knox along with the other services.
2. [Install Ranger using Ambari](#).
3. The Knox SSO topology settings are preconfigured in **Knox > Configs > Advanced knoxsso-topology**.
4. Run the following CLI command to export the Knox certificate:

```
JAVA_HOME/bin/keytool -export -alias gateway-identity -rfc -file <cert.pem>
-keystore /usr/hdp/current/knox-server/data/security/keystores/gateway.jks
```

- When prompted, enter the Knox master password.
 - Note the location where you save the `cert.pem` file.
5. Select **Ranger > Configs > Advanced > Knox SSO Settings** and set the following properties:
 - **Enable Ranger SSO** – Select this check box to enable Ranger SSO.
 - **SSO provider url** – `https://<knox_host>:8443/gateway/knoxss0/api/v1/webss0`

- **SSO public key** – Paste in the contents of the `cert.pem` certificate file exported from Knox.

When you paste the contents, exclude the header and footer.

- **SSO browser useragent** – Preconfigured with `Mozilla,chrome`.

6. Click **Save** to save the new configuration, then click through the confirmation pop-ups.
7. Restart Ranger. Select **Actions > Restart All Required** to restart all other services that require a restart.
8. Knox SSO should now be enabled. Users who try to access Ranger are redirected to the Knox SSO login page for authentication.

2.9.4. Setting up the Knox Token Service for Ranger APIs

About This Task

Once logged into Knox SSO, the UI service uses a cookie named `hadoop-jwt`. The Knox Token Service enables clients to acquire this same JWT token to use for accessing REST APIs. By acquiring the token and setting it as a bearer token on a request, a client is able to access REST APIs that are protected with the [Setting up JWT Federation Provider \[146\]](#).

Steps

To configure the Knox Token Service for Ranger APIs:

1. The Knox Token Service configuration can be configured in any topology. For example, from Ambari>Knox>Configs>Advanced knoxsso-topology, add:

```
<service>
  <role>KNOXTOKEN</role>
  <param>
    <name>knox.token.ttl</name>
    <value>numeric_value_in_miliseconds</value>
  </param>
  <param>
    <name>knox.token.audiences</name>
    <value>tokenbased</value>
  </param>
  <param>
    <name>knox.token.target.url</name>
    <value>https://host:port/gateway/tokenbased</value>
  </param>
</service>
```

where the values of the parameters are specific to your environment:

Parameter	Description	Optional/Required	Default
knox.token.ttl	The lifespan of the token in milliseconds. Once it expires, a new token must be acquired from KnoxToken service.	Required	30000 (30 seconds)
knox.token.audiences	Comma separated list of audiences to add to the JWT token. Used to ensure that a token received by a participating application knows that the token was intended for use with that application. In the event that an endpoint has expected audiences, and they are not present, the token must be rejected. In the event where the token has audiences, and the endpoint has none expected, then the token is accepted.	Optional	
knox.token.target.url	Indicates the intended endpoint for which the token may be used. The KnoxShell token credential collector can pull this URL from a knoxtokencache file to be used in scripts. Eliminates the need to prompt for or hard-code endpoints in your scripts.	Optional	

Example

From Ambari>Knox>Configs>Advanced knoxsso-topology, add:


```
<service>
  <role>KNOXTOKEN</role>
  <param>
    <name>knox.token.ttl</name>
    <value>3600000</value>
  </param>
  <param>
    <name>knox.token.audiences</name>
    <value>tokenbased</value>
  </param>
  <param>
    <name>knox.token.target.url</name>
    <value>https://localhost:8443/gateway/tokenbased</value>
  </param>
</service>
```

Next Steps

Acquire a token from the Knox Token service as configured in the sandbox topology

```
curl -ivku guest:guest-password https://localhost:8443/gateway/sandbox/
knoxtoken/api/v1/token
```

Resulting in a JSON response that contains the token, the expiration and the optional target endpoint:

```
`{"access_token": "eyJhbGciOiJSUzI1NiJ9.
eyJzdWIiOiJndWVzdCIsImF1ZCI6InRva2VuYmFzZWQpLCJpc3MiOiJLTk9YU1NPIiwiaXNjaXhwIjoxNDg5OTQyMTg4fQ.
bcqSK7zMnABEM_HVsm3oWnDrQ_ei7PcMI4AtZEERY9LaPo9dzugOg3PA5JH2BRF-
lXM3tuEYuzPaZVf8PenzjtBbuQsCg9VVImuu2r1YNVJlcTQ7OV-
eW50L60TI0uzfyrFwX6C7jVhf7d7YR1NNxs4eVbXpS1TZ5fDIRSFU3MU" ,
"target_url": "https://localhost:8443/gateway/tokenbased", "token_type": "Bearer
", "expires_in": 1489942188233}`
```

The following curl example shows how to add a bearer token to an Authorization header:

```
curl -ivk -H "Authorization: Bearer eyJhbGciOiJSUzI1NiJ9.
eyJzdWIiOiJndWVzdCIsImF1ZCI6InRva2VuYmFzZWQpLCJpc3MiOiJLTk9YU1NPIiwiaXNjaXhwIjoxNDg5OTQyMTg4fQ.
bcqSK7zMnABEM_HVsm3oWnDrQ_ei7PcMI4AtZEERY9LaPo9dzugOg3PA5JH2BRF-
lXM3tuEYuzPaZVf8PenzjtBbuQsCg9VVImuu2r1YNVJlcTQ7OV-
eW50L60TI0uzfyrFwX6C7jVhf7d7YR1NNxs4eVbXpS1TZ5fDIRSFU3MU" https://
localhost:8443/gateway/tokenbased/webhdfs/v1/tmp?op=LISTSTATUS
```

2.9.5. Setting up Knox SSO for Apache Atlas

This section describes how to configure Apache Atlas to use Knox SSO (Single Sign-on) to authenticate users on an Ambari cluster. With this configuration, unauthenticated users who try to access Atlas are redirected to the Knox SSO login page for authentication.



Note

- Atlas SSO is only applied to web UI users.
- Internal Atlas users have the option to bypass Knox SSO and log in to the Atlas UI directly using the "login.jsp" URL: `http://<atlas_host>:21000/login.jsp`.

Use the following steps to configure Knox SSO for Atlas:

1. Install Ambari with HDP-2.6 or higher. Install Knox along with the other services.
2. [Install Atlas using Ambari](#).
3. The Knox SSO topology settings are preconfigured in **Knox > Configs > Advanced knoxsso-topology**.
4. Run the following CLI command to export the Knox certificate:

```
JAVA_HOME/bin/keytool -export -alias gateway-identity -rfc -file knox-pub-key.cert -keystore /usr/hdp/current/knox-server/data/security/keystores/gateway.jks
```

- When prompted, enter the Knox master password.
 - Note the location where you save the `cert.pem` file.
5. Select **Atlas > Configs > Authentication**.
 6. Select the **Enable Knox SSO** check box, then set the following properties:

- **atlas.sso.knox.providerurl** – Enter the Knox SSO Provider URL:

```
https://<knox_host>:8443/gateway/knoxsso/api/v1/websso
```

- **atlas.sso.knox.publicKey** – Paste in the contents of the `cert.pem` certificate file exported from Knox, excluding the header and footer.
- **atlas.sso.knox.browser.useragent** – Enter the browsers to use with Knox SSO, for example:

```
Mozilla,chrome
```

The screenshot shows the configuration page for Atlas authentication in the Hortonworks Data Platform. The interface includes a sidebar with navigation options like Pig, Oozie, ZooKeeper, Storm, Ambari Infra, Ambari Metrics, Atlas (selected), Kafka, Knox, Log Search, SmartSense, Spark, Zeppelin Notebook, and Slider. The main content area is titled 'Authentication' and has a sub-tab 'Advanced'. It features three sections: 'Authentication Methods' with checkboxes for 'Enable File Authentication' (checked), 'Enable LDAP Authentication' (unchecked), and 'Enable Atlas Knox SSO' (checked); 'File' with a text input field for 'atlas.authentication.method.file.filename' containing '[[conf_dir]]/users-credentials.properties'; and 'Atlas Knox SSO' with text input fields for 'atlas.sso.knox.providerurl' (https://dh-a25h26.field.hortonworks.com:8443/gateway/knoxso/api/v1/webso), 'atlas.sso.knox.publicKey' (MIIEpAIBAAKCAQEAAuGIXDcMTL2S8ixAlrycqn+a3ZPkfgzNoM7dHBBWLv0VdrlgBnxEnngjBG Wqmq), and 'atlas.sso.knox.browser.useragent' (Mozilla,chrome). A top bar shows 'admin authored on Thu, Dec 01, 2016 13:48' and buttons for 'Discard' and 'Save'.

7. Click **Save** to save the new configuration, then click through the confirmation pop-ups.
8. Restart Atlas. Select **Actions > Restart All Required** to restart all other services that require a restart.
9. Knox SSO should now be enabled. Users who try to access Atlas are redirected to the Knox SSO login page for authentication.

3. Configuring Authorization in Hadoop

3.1. Installing Ranger Using Ambari

3.1.1. Overview

Apache Ranger can be installed either manually using the Hortonworks Data Platform (HDP) or the Ambari User Interface (UI). Unlike the manual installation process, which requires you to perform a number of installation steps, installing Ranger using the Ambari UI is simpler and easier. The Ranger service option will be made available through the Add Service wizard after the HDP cluster is installed using the installation wizard.

Once Ambari has been installed and configured, you can use the Add Service wizard to install the following components:

- Ranger Admin
- Ranger UserSync
- [Ranger Key Management Service](#)

After these components are installed and started, you can enable Ranger plugins by navigating to each individual Ranger service (HDFS, HBase, Hiveserver2, Storm, Knox, YARN, and Kafka) and modifying the configuration under *advanced ranger-<service>-plugin-properties*.

Note that when you enable a Ranger plugin, you will need to restart the component.



Note

Enabling Apache Storm or Apache Kafka requires you to enable Kerberos. To enable Kerberos on your cluster, see [Enabling Kerberos Authentication Using Ambari](#).



Note

On fresh HDP-2.6 installs (on Ambari 2.5.0), the Ranger DB setup is performed the first time Ranger starts up. (In previous versions, the Ranger DB setup was done during installation). This means that Ranger may take longer than previously to start up the first time, but subsequent restarts should be as fast as before.

3.1.2. Installation Prerequisites

Before you install Ranger, make sure your cluster meets the following requirements:



Important

As of HDP-2.5, Audit to DB is no longer supported. If you previously used Audit to DB, you can migrate the logs to Solr using the instructions in [Migrating Audit Logs from DB to Solr in Ambari Clusters](#).

- It is recommended that you store audits in both HDFS and Solr. The default configuration for Ranger Audits to Solr uses the shared Solr instance provided under the [Ambari Infra](#) service. For more information about Audits to Solr, see [Ranger Audit Settings](#) and [Using Apache Solr for Ranger Audits](#).
- To ensure that LDAP/AD group level authorization is enforced in Hadoop, you should [Setting Up Hadoop Group Mapping for LDAP/AD \[185\]](#) for LDAP.
- A MySQL, Oracle, PostgreSQL, or Amazon RDS database instance must be running and available to be used by Ranger.

The Ranger installation will create two new users (default names: rangeradmin and rangerlogger) and two new databases (default names: ranger and ranger_audit).

- Configuration of the database instance for Ranger is described in the following sections for some of the databases supported by Ranger.
 - [Configuring MySQL for Ranger \[188\]](#)
 - [Configuring PostgreSQL for Ranger \[189\]](#)
 - [Configuring Oracle for Ranger \[190\]](#)
 - [Amazon RDS Requirements \[191\]](#)
- If you choose not to provide system Database Administrator (DBA) account details to the Ambari Ranger installer, you can use the `dba_script.py` Python script to create Ranger DB database users without exposing DBA account information to the Ambari Ranger installer. You can then run the normal Ambari Ranger installation without specifying a DBA user name and password. For more information see [Setting up Database Users Without Sharing DBA Credentials](#).

3.1.2.1. Setting Up Hadoop Group Mapping for LDAP/AD

To ensure that LDAP/AD group level authorization is enforced in Hadoop, you should set up Hadoop group mapping for LDAP/AD.

Prerequisites: Access to LDAP and the connection details. Note that LDAP settings can vary depending on what LDAP implementation you are using.

There are three ways to set up Hadoop group mapping:

- [Configure Hadoop Group Mapping for LDAP/AD Using SSSD \(Recommended\) \[185\]](#)
- [Configure Hadoop Group Mapping in core-site.xml \[186\]](#)
- [Manually Create the Users and Groups in the Linux Environment \[187\]](#)

3.1.2.1.1. Configure Hadoop Group Mapping for LDAP/AD Using SSSD (Recommended)

The recommended method for group mapping is to use [SSSD](#) or one of the following services to connect the Linux OS with LDAP:

- Centrify

- NSLCD
- Winbind
- SAMBA

Note that most of these services allow you to not only look up a user and enumerate their groups, but also allow you to perform other actions on the host. None of these features are required for LDAP group mapping on Hadoop – all that is required is the ability to lookup (or "validate") a user within LDAP and enumerate their groups. Therefore, when evaluating these services, take the time to understand the difference between the NSS module (which performs user/group resolution) and the PAM module (which performs user authentication). NSS is required. PAM is not required, and may represent a security risk.

3.1.2.1.2. Configure Hadoop Group Mapping in `core-site.xml`

You can use the following steps to configure Hadoop to use LDAP-based group mapping in `core-site.xml`.

1. Add the properties shown in the example below to the `core-site.xml` file. You will need to provide the value for the bind user, the bind password, and other properties specific to your LDAP instance, and make sure that object class, user, and group filters match the values specified in your LDAP instance.

```
<property>
<name>hadoop.security.group.mapping</name>
<value>org.apache.hadoop.security.LdapGroupsMapping</value>
</property>

<property>
<name>hadoop.security.group.mapping.ldap.bind.user</name>
<value>cn=Manager,dc=hadoop,dc=apache,dc=org</value>
</property>

<!--
<property>
<name>hadoop.security.group.mapping.ldap.bind.password.file</name>
<value>/etc/hadoop/conf/ldap-conn-pass.txt</value>
</property>
-->

<property>
<name>hadoop.security.group.mapping.ldap.bind.password</name>
<value>hadoop</value>
</property>

<property>
<name>hadoop.security.group.mapping.ldap.url</name>
<value>ldap://localhost:389/</value>
</property>

<property>
<name>hadoop.security.group.mapping.ldap.base</name>
<value></value>
</property>

<property>
```

```
<name>hadoop.security.group.mapping.ldap.search.filter.user</name>
<value>(&amp;(|(objectclass=person)(objectclass=applicationProcess))(cn=
{0}))</value>
</property>

<property>
<name>hadoop.security.group.mapping.ldap.search.filter.group</name>
<value>(objectclass=groupOfNames)</value>
</property>

<property>
<name>hadoop.security.group.mapping.ldap.search.attr.member</name>
<value>member</value>
</property>

<property>
<name>hadoop.security.group.mapping.ldap.search.attr.group.name</name>
<value>cn</value>
</property>
```

2. Depending on your configuration, you may be able to refresh user and group mappings using the following HDFS and YARN commands:

```
hdfs dfsadmin -refreshUserToGroupsMappings
yarn rmadmin -refreshUserToGroupsMappings
```

If a restart is required, you can use the applicable instructions on [this page](#) to re-start the HDFS NameNode and the YARN ResourceManager.

3. Verify LDAP group mapping by running the `hdfs groups` command. This command will fetch groups from LDAP for the current user. Note that with LDAP group mapping configured, the HDFS permissions can leverage groups defined in LDAP for access control.

3.1.2.1.3. Manually Create the Users and Groups in the Linux Environment

You can also [manually create users and groups](#) in your Linux environment.

3.1.2.2. Configuring a Database Instance for Ranger

A MySQL, Oracle, PostgreSQL, or Amazon RDS database instance must be running and available to be used by Ranger. The Ranger installation will create two new users (default names: `rangeradmin` and `rangerlogger`) and two new databases (default names: `ranger` and `ranger_audit`).

Choose from the following:

- [Configuring MySQL for Ranger \[188\]](#)
- [Configuring PostgreSQL for Ranger \[189\]](#)
- [Configuring Oracle for Ranger \[190\]](#)

If you are using Amazon RDS, there are additional requirements:

- [Amazon RDS Requirements \[191\]](#)

3.1.2.2.1. Configuring MySQL for Ranger

Prerequisites

When using MySQL, the storage engine used for the Ranger admin policy store tables MUST support transactions. InnoDB is an example of engine that supports transactions. A storage engine that does not support transactions is not suitable as a policy store.

Steps

If you are using Amazon RDS, see the [Amazon RDS Requirements](#).

1. The MySQL database administrator should be used to create the Ranger databases.

The following series of commands could be used to create the `rangerdba` user with password `rangerdba`.

- a. Log in as the root user, then use the following commands to create the `rangerdba` user and grant it adequate privileges.

```
CREATE USER 'rangerdba'@'localhost' IDENTIFIED BY 'rangerdba';
GRANT ALL PRIVILEGES ON *.* TO 'rangerdba'@'localhost';
CREATE USER 'rangerdba'@'%' IDENTIFIED BY 'rangerdba';
GRANT ALL PRIVILEGES ON *.* TO 'rangerdba'@'%';
GRANT ALL PRIVILEGES ON *.* TO 'rangerdba'@'localhost' WITH GRANT OPTION;
GRANT ALL PRIVILEGES ON *.* TO 'rangerdba'@'%' WITH GRANT OPTION;
FLUSH PRIVILEGES;
```

- b. Use the `exit` command to exit MySQL.
- c. You should now be able to reconnect to the database as `rangerdba` using the following command:

```
mysql -u rangerdba -prangerdba
```

After testing the `rangerdba` login, use the `exit` command to exit MySQL.

2. Use the following command to confirm that the `mysql-connector-java.jar` file is in the Java share directory. This command must be run on the server where Ambari server is installed.

```
ls /usr/share/java/mysql-connector-java.jar
```

If the file is not in the Java share directory, use the following command to install the MySQL connector .jar file.

RHEL/CentOS/Oracle Linux

```
yum install mysql-connector-java*
```

SLES


```
zypper install mysql-connector-java*
```

3. Use the following command format to set the `jdbc/driver/path` based on the location of the MySQL JDBC driver `.jar` file. This command must be run on the server where Ambari server is installed.

```
ambari-server setup --jdbc-db={database-type} --jdbc-driver={/jdbc/driver/path}
```

For example:

```
ambari-server setup --jdbc-db=mysql --jdbc-driver=/usr/share/java/mysql-connector-java.jar
```

3.1.2.2.2. Configuring PostgreSQL for Ranger

If you are using Amazon RDS, see the [Amazon RDS Requirements](#).

1. On the PostgreSQL host, install the applicable PostgreSQL connector.

RHEL/CentOS/Oracle Linux

```
yum install postgresql-jdbc*
```

SLES

```
zypper install -y postgresql-jdbc
```

2. Confirm that the `.jar` file is in the Java share directory.

```
ls /usr/share/java/postgresql-jdbc.jar
```

3. Change the access mode of the `.jar` file to 644.

```
chmod 644 /usr/share/java/postgresql-jdbc.jar
```

4. The PostgreSQL database administrator should be used to create the Ranger databases.

The following series of commands could be used to create the `rangerdba` user and grant it adequate privileges.

```
echo "CREATE DATABASE $dbname;" | sudo -u $postgres psql -U postgres
echo "CREATE USER $rangerdba WITH PASSWORD '$passwd';" | sudo -u $postgres
psql -U postgres
echo "GRANT ALL PRIVILEGES ON DATABASE $dbname TO $rangerdba;" | sudo -u
$postgres psql -U postgres
```

Where:

- `$postgres` is the Postgres user.
- `$dbname` is the name of your PostgreSQL database

5. Use the following command format to set the `jdbc/driver/path` based on the location of the PostgreSQL JDBC driver `.jar` file. This command must be run on the server where Ambari server is installed.

```
ambari-server setup --jdbc-db={database-type} --jdbc-driver={/jdbc/driver/path}
```

For example:

```
ambari-server setup --jdbc-db=postgres --jdbc-driver=/usr/share/java/postgresql-jdbc.jar
```

6. Run the following command:

```
export HADOOP_CLASSPATH=${HADOOP_CLASSPATH}:${JAVA_JDBC_LIBS}:/connector.jar
```

7. Add Allow Access details for Ranger users:

- change `listen_addresses='localhost'` to `listen_addresses='*' ('*' = any)` to listen from all IPs in `postgresql.conf`.
- Make the following changes to the Ranger db user and Ranger audit db user in the `pg_hba.conf` file.

```
# TYPE DATABASE USER CIDR-ADDRESS METHOD
# "local" is for Unix domain socket connections only
local all postgres,rangeradmin,rangerlogger trust
# IPv4 local connections:
host all postgres,rangeradmin,rangerlogger 0.0.0.0/0 trust
# IPv6 local connections:
host all postgres,rangeradmin,rangerlogger ::/0 trust
"/var/lib/pgsql/data/pg_hba.conf" 74L, 3445C
```

8. After editing the `pg_hba.conf` file, run the following command to refresh the PostgreSQL database configuration:

```
sudo -u postgres /usr/bin/pg_ctl -D $PGDATA reload
```

For example, if the `pg_hba.conf` file is located in the `/var/lib/pgsql/data` directory, the value of `$PGDATA` is `/var/lib/pgsql/data`.

3.1.2.2.3. Configuring Oracle for Ranger

If you are using Amazon RDS, see the [Amazon RDS Requirements](#).

1. On the Oracle host, install the appropriate JDBC .jar file.

- Download the Oracle JDBC (OJDBC) driver from <http://www.oracle.com/technetwork/database/features/jdbc/index-091264.html>.
- For **Oracle Database 11g**: select Oracle Database 11g Release 2 drivers > `ojdbc6.jar`.
- For **Oracle Database 12c**: select Oracle Database 12c Release 1 driver > `ojdbc7.jar`.
- Copy the .jar file to the Java share directory. For example:

```
cp ojdbc7.jar /usr/share/java/
```



Note

Make sure the .jar file has the appropriate permissions. For example:

```
chmod 644 /usr/share/java/ojdbc7.jar
```

2. The Oracle database administrator should be used to create the Ranger databases.

The following series of commands could be used to create the RANGERDBA user and grant it permissions using SQL*Plus, the Oracle database administration utility:

```
# sqlplus sys/root as sysdba
CREATE USER $RANGERDBA IDENTIFIED BY $RANGERDBAPASSWORD;
GRANT SELECT_CATALOG_ROLE TO $RANGERDBA;
GRANT CONNECT, RESOURCE TO $RANGERDBA;
QUIT;
```

3. Use the following command format to set the `jdbc/driver/path` based on the location of the Oracle JDBC driver `.jar` file. This command must be run on the server where Ambari server is installed.

```
ambari-server setup --jdbc-db={database-type} --jdbc-driver={jdbc/driver/path}
```

For example:

```
ambari-server setup --jdbc-db=oracle --jdbc-driver=/usr/share/java/ojdbc6.jar
```

3.1.2.2.4. Amazon RDS Requirements

Ranger requires a relational database as its policy store. There are additional prerequisites for Amazon RDS-based databases due to how Amazon RDS is set up and managed:

- [MySQL/MariaDB Prerequisite \[191\]](#)
- [PostgreSQL Prerequisite \[192\]](#)
- [Oracle Prerequisite \[192\]](#)

3.1.2.2.4.1. MySQL/MariaDB Prerequisite

You must change the variable `log_bin_trust_function_creators` to 1 during Ranger installation.

From RDS Dashboard>Parameter group (on the left side of the page):

1. Set the MySQL Server variable `log_bin_trust_function_creators` to 1.
2. (Optional) After Ranger installation is complete, reset `log_bin_trust_function_creators` to its original setting. The variable is only required to be set to 1 during Ranger installation.

For more information, see:

- [Stratalux: Why You Should Always Use a Custom DB Parameter Group When Creating an RDS Instance](#)
- [AWS Documentation>Amazon RDS DB Instance Lifecycle » Working with DB Parameter Groups](#)
- [MySQL 5.7 Reference Manual >Binary Logging of Stored Programs](#)

3.1.2.2.4.2. PostgreSQL Prerequisite

The Ranger database user in Amazon RDS PostgreSQL Server should be created before installing Ranger and should be granted an existing role which must have the role CREATEDB.

1. Using the master user account, log in to the Amazon RDS PostgreSQL Server from master user account (created during RDS PostgreSQL instance creation) and execute following commands:

- a. `CREATE USER $rangerdbuser WITH LOGIN PASSWORD 'password'`
- b. `GRANT $rangerdbuser to $postgresroot`

Where `$postgresroot` is the RDS PostgreSQL master user account (for example: postgresroot) and `$rangerdbuser` is the Ranger database user name (for example: rangeradmin).

2. If you are using Ranger KMS, execute the following commands:

- a. `CREATE USER $rangerkmsuser WITH LOGIN PASSWORD 'password'`
- b. `GRANT $rangerkmsuser to $postgresroot`

Where `$postgresroot` is the RDS PostgreSQL master user account (for example: postgresroot) and `$rangerkmsuser` is the Ranger KMS user name (for example: rangerkms).

3.1.2.2.4.3. Oracle Prerequisite

Due to [limitations in Amazon RDS](#), the Ranger database user and tablespace must be created manually and the required privileges must be manually granted to the Ranger database user.

1. Log in to the RDS Oracle Server from the master user account (created during RDS Oracle instance creation) and execute following commands:

```
create user $rangerdbuser identified by "password";
GRANT CREATE SESSION,CREATE PROCEDURE,CREATE TABLE,CREATE VIEW,CREATE
SEQUENCE,CREATE PUBLIC SYNONYM,CREATE ANY SYNONYM,CREATE TRIGGER,UNLIMITED
Tablespace TO $rangerdbuser;
create tablespace $rangerdb datafile size 10M autoextend on;
alter user $rangerdbuser DEFAULT Tablespace $rangerdb;
```

Where `$rangerdb` is a actual Ranger database name (for example: ranger) and `$rangerdbuser` is Ranger database username (for example: rangeradmin).

2. If you are using Ranger KMS, execute the following commands:

```
create user $rangerdbuser identified by "password";
GRANT CREATE SESSION,CREATE PROCEDURE,CREATE TABLE,CREATE VIEW,CREATE
SEQUENCE,CREATE PUBLIC SYNONYM,CREATE ANY SYNONYM,CREATE TRIGGER,UNLIMITED
Tablespace TO $rangerkmsuser;
create tablespace $rangerkmsdb datafile size 10M autoextend on;
alter user $rangerkmsuser DEFAULT Tablespace $rangerkmsdb;
```

Where *\$rangerkmsdb* is a actual Ranger database name (for example: rangerkms) and *\$rangerkmsuser* is Ranger database username (for example: rangerkms).

3.1.3. Ranger Installation

To install Ranger using Ambari:

1. [Start the Installation \[193\]](#)
2. [Customize Services \[198\]](#)
3. [Complete the Ranger Installation \[228\]](#)

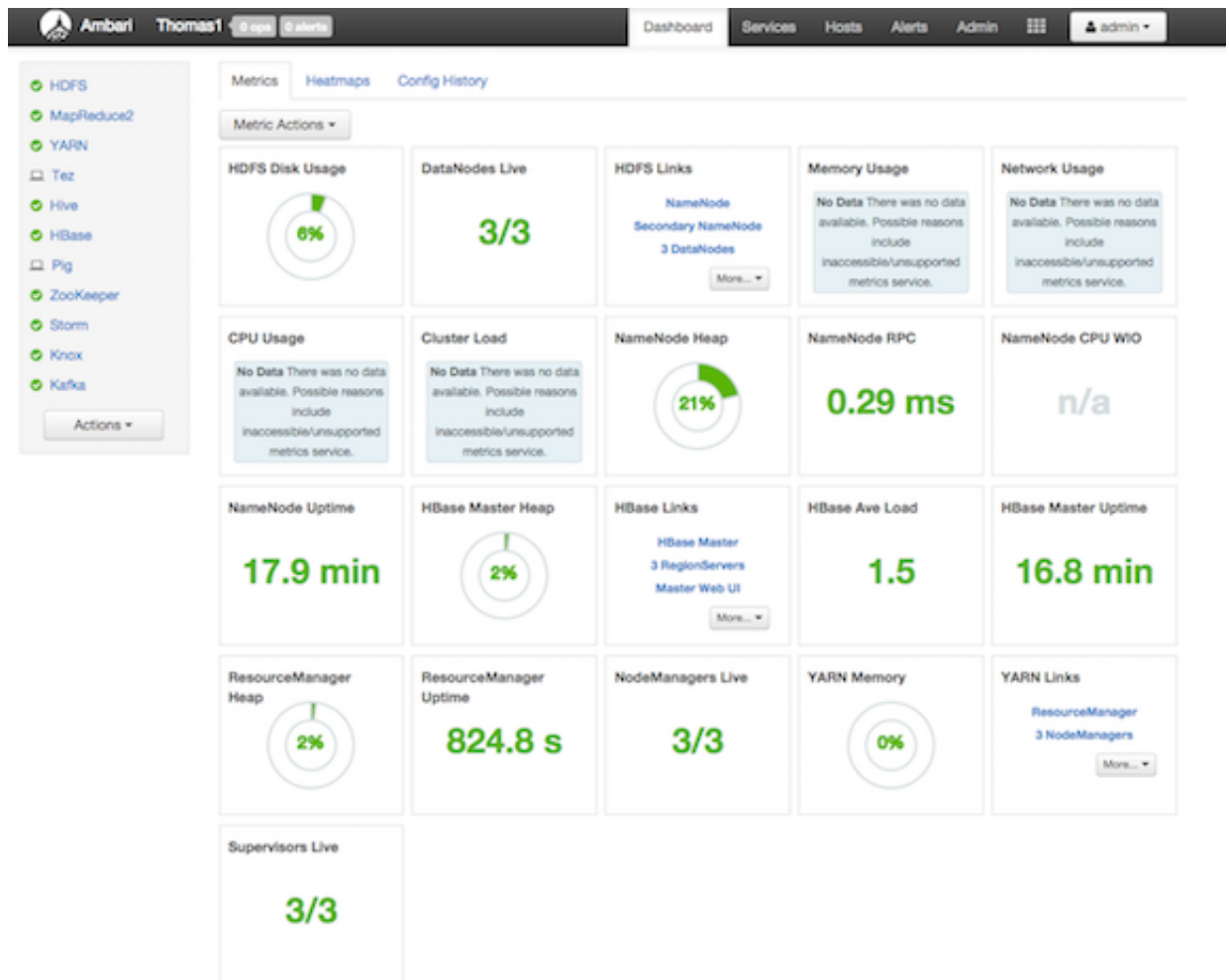
Related Topics

- [Setting up Database Users Without Sharing DBA Credentials \[232\]](#)
- [Updating Ranger Admin Passwords \[233\]](#)

3.1.3.1. Start the Installation

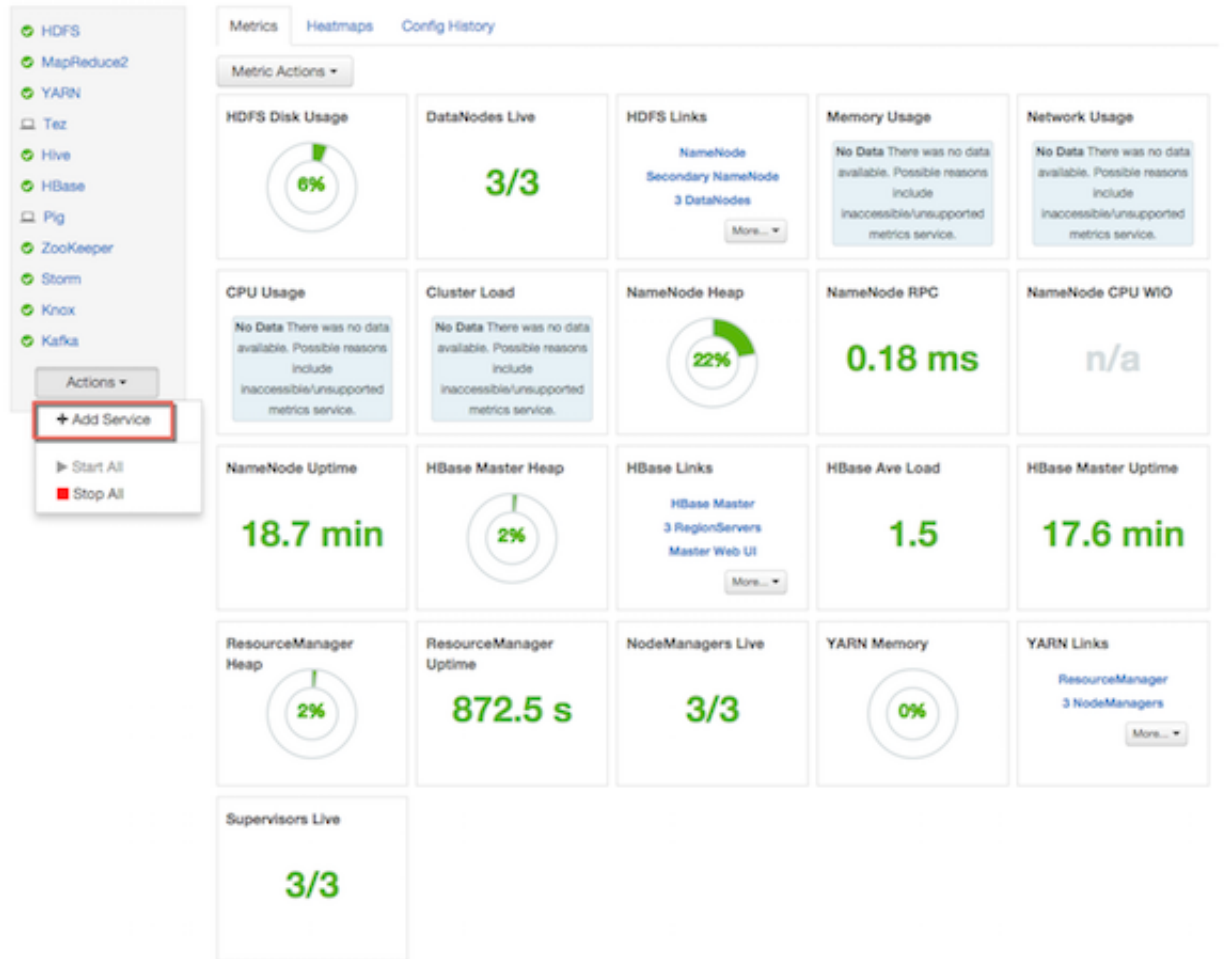
1. Log into your Ambari cluster with your designated user credentials. The main Ambari Dashboard page will be displayed.

Figure 3.1. Installing Ranger - Main Dashboard View



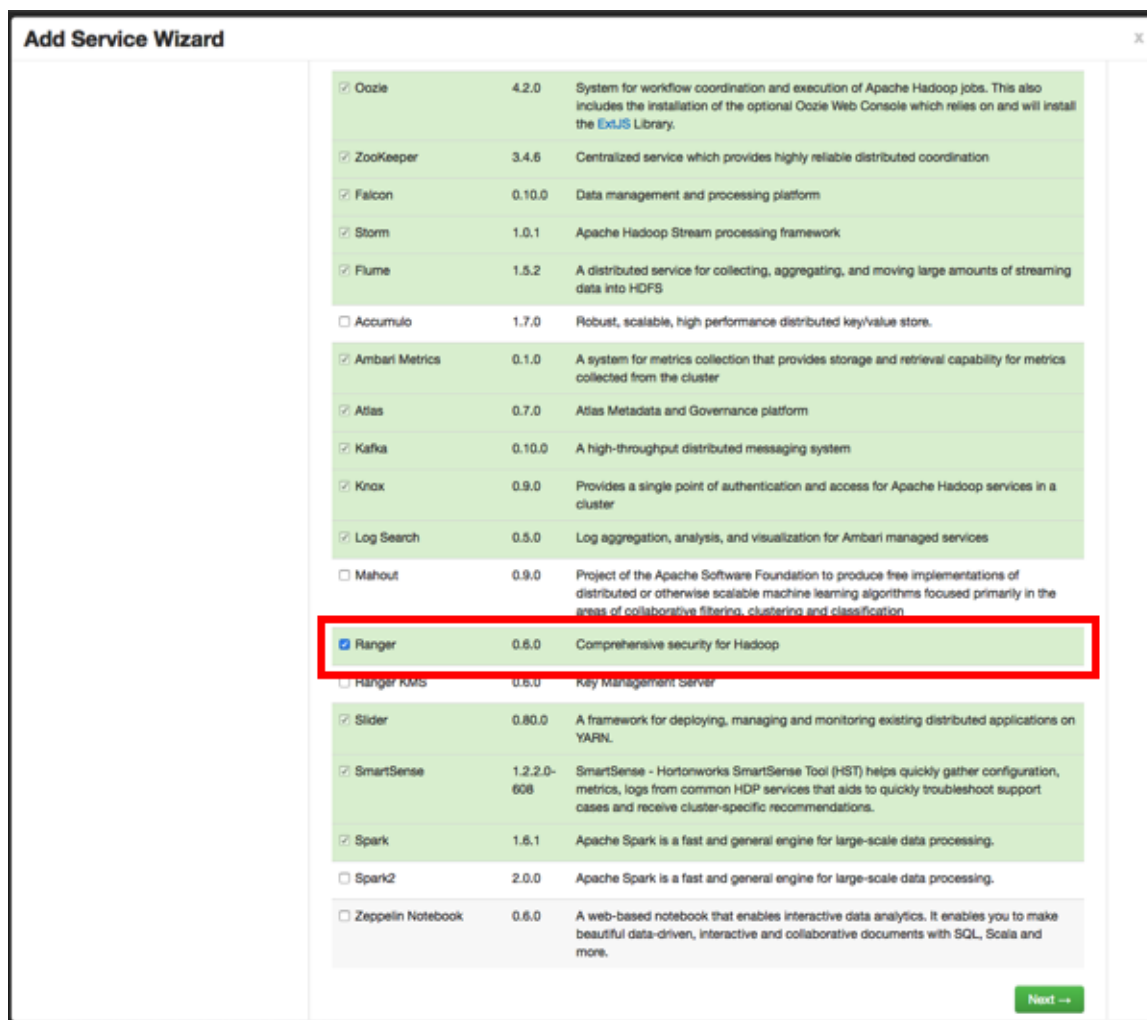
2. In the left navigation menu, click **Actions**, then select **Add Service**.

Figure 3.2. Installing Ranger - Add Service



3. On the Choose Services page, select **Ranger**, then click **Next**.

Figure 3.3. Installing Ranger - Choose Service



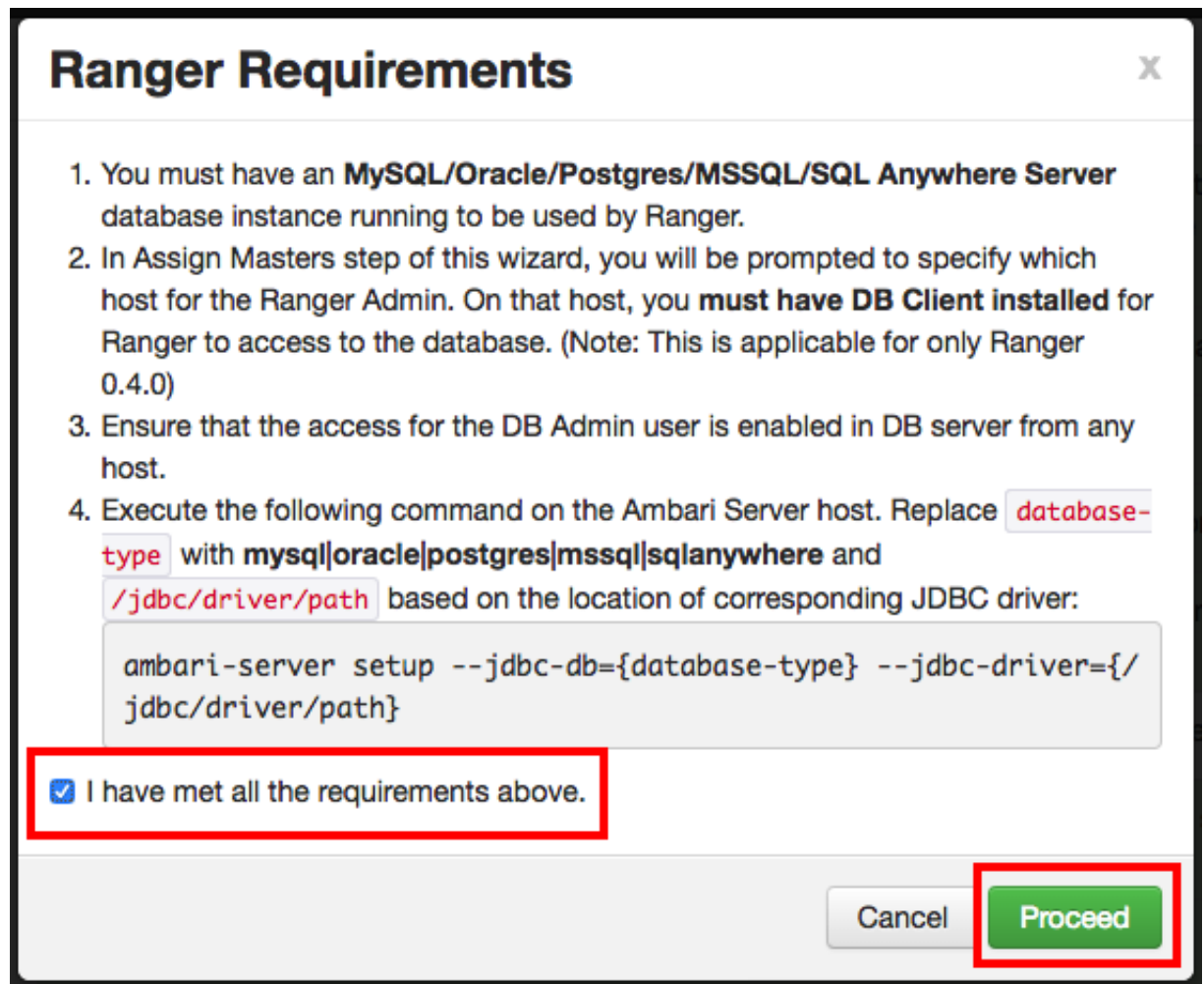
The screenshot shows the 'Add Service Wizard' interface with a list of services. The 'Ranger' service is highlighted with a red box. The services listed are:

Service	Version	Description
<input checked="" type="checkbox"/> Oozie	4.2.0	System for workflow coordination and execution of Apache Hadoop jobs. This also includes the installation of the optional Oozie Web Console which relies on and will install the ExtJS Library.
<input checked="" type="checkbox"/> ZooKeeper	3.4.6	Centralized service which provides highly reliable distributed coordination
<input checked="" type="checkbox"/> Falcon	0.10.0	Data management and processing platform
<input checked="" type="checkbox"/> Storm	1.0.1	Apache Hadoop Stream processing framework
<input checked="" type="checkbox"/> Flume	1.5.2	A distributed service for collecting, aggregating, and moving large amounts of streaming data into HDFS
<input type="checkbox"/> Accumulo	1.7.0	Robust, scalable, high performance distributed key/value store.
<input checked="" type="checkbox"/> Ambari Metrics	0.1.0	A system for metrics collection that provides storage and retrieval capability for metrics collected from the cluster
<input checked="" type="checkbox"/> Atlas	0.7.0	Atlas Metadata and Governance platform
<input checked="" type="checkbox"/> Kafka	0.10.0	A high-throughput distributed messaging system
<input checked="" type="checkbox"/> Knox	0.9.0	Provides a single point of authentication and access for Apache Hadoop services in a cluster
<input checked="" type="checkbox"/> Log Search	0.5.0	Log aggregation, analysis, and visualization for Ambari managed services
<input type="checkbox"/> Mahout	0.9.0	Project of the Apache Software Foundation to produce free implementations of distributed or otherwise scalable machine learning algorithms focused primarily in the areas of collaborative filtering, clustering and classification
<input checked="" type="checkbox"/> Ranger	0.6.0	Comprehensive security for Hadoop
<input type="checkbox"/> Ranger KMS	0.6.0	Key Management Server
<input checked="" type="checkbox"/> Slider	0.80.0	A framework for deploying, managing and monitoring existing distributed applications on YARN.
<input checked="" type="checkbox"/> SmartSense	1.2.2.0-608	SmartSense - Hortonworks SmartSense Tool (HST) helps quickly gather configuration, metrics, logs from common HDP services that aids to quickly troubleshoot support cases and receive cluster-specific recommendations.
<input checked="" type="checkbox"/> Spark	1.6.1	Apache Spark is a fast and general engine for large-scale data processing.
<input type="checkbox"/> Spark2	2.0.0	Apache Spark is a fast and general engine for large-scale data processing.
<input type="checkbox"/> Zeppelin Notebook	0.6.0	A web-based notebook that enables interactive data analytics. It enables you to make beautiful data-driven, interactive and collaborative documents with SQL, Scala and more.

Next -->

4. The Ranger Requirements page appears. Ensure that you have met all of the installation requirements, then select the "I have met all the requirements above" check box and click **Proceed**.

Figure 3.4. Installing Ranger - Ranger Requirements



5. You are then prompted to select the host where Ranger Admin will be installed. This host should have DB admin access to the Ranger DB host and User Sync. Notice in the figure below that both the Ranger Admin and Ranger User Sync services will be installed on the primary node in the cluster (c6401.ambari.apache.org in the example shown below).

Make a note of the Ranger Admin host for use in subsequent installation steps. Click **Next** when finished to continue with the installation.

Figure 3.5. Installing Ranger Assign Masters

6. The Customize Services page appears. These settings are described in the next section.

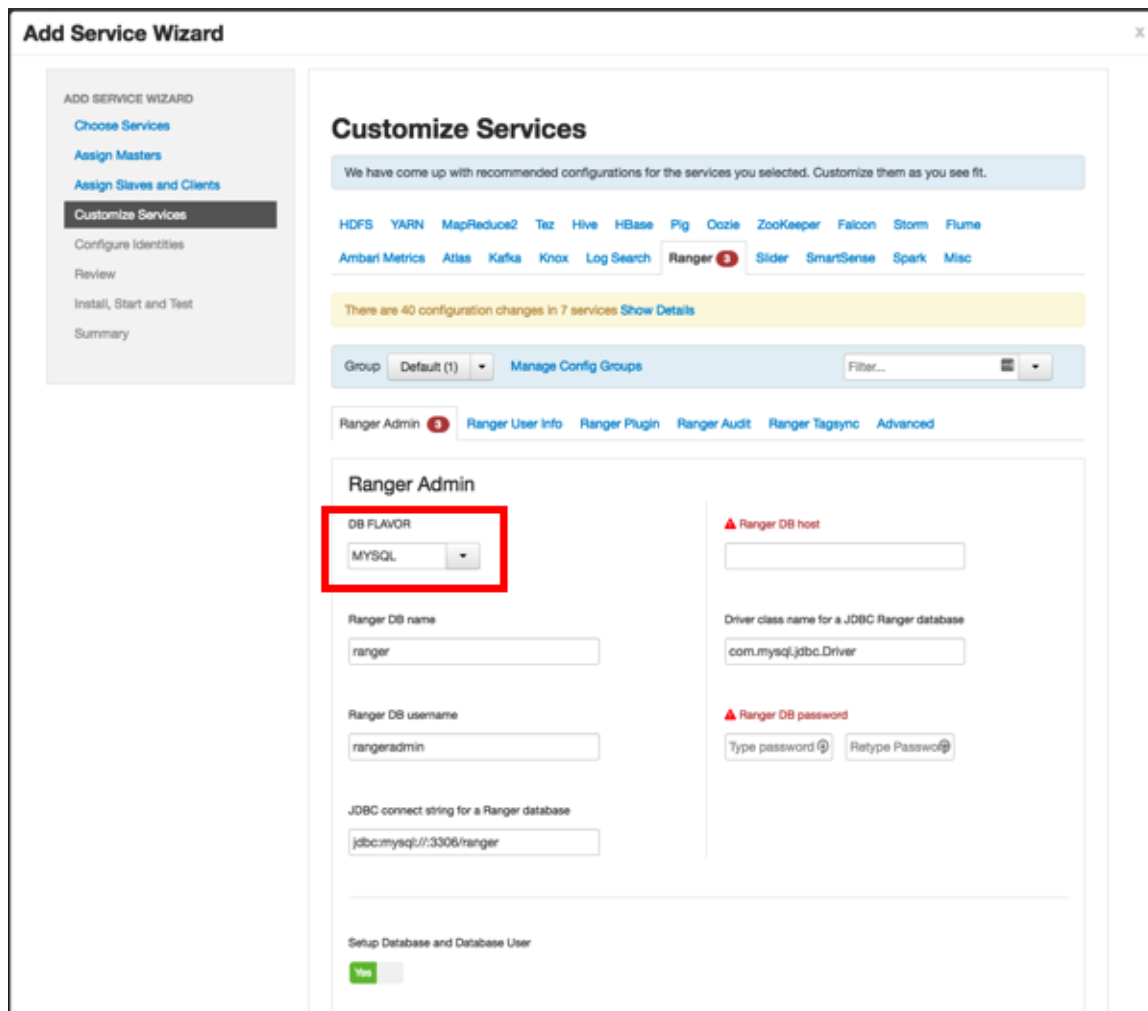
3.1.3.2. Customize Services

The next step in the installation process is to specify Ranger settings on the Customize Services page.

- [Ranger Admin Settings \[199\]](#)
- [Ranger Audit Settings \[208\]](#)
- [Configure Ranger User Sync \[210\]](#)
- [Configure Ranger Authentication \[220\]](#)

3.1.3.2.1. Ranger Admin Settings

1. On the Customize Services page, select the Ranger Admin tab, then use the **DB Flavor** drop-down to select the database type that you are using with Ranger.



2. Enter the database server address in the **Ranger DB Host** box.

Table 3.1. Ranger DB Host

DB Flavor	Host	Example
MySQL	<HOST[:PORT]>	c6401.ambari.apache.org or c6401.ambari.apache.org:3306
	<HOST:PORT:SID> <HOST:PORT/Service>	c6401.ambari.apache.org:1521:ORCL c6401.ambari.apache.org:1521/XE
PostgreSQL	<HOST[:PORT]>	c6401.ambari.apache.org or c6401.ambari.apache.org:5432

DB Flavor	Host	Example
MS SQL	<HOST[:PORT]>	c6401.ambari.apache.org or c6401.ambari.apache.org:1433
SQLA	<HOST[:PORT]>	c6401.ambari.apache.org or c6401.ambari.apache.org:2638

3. **Ranger DB name** – The name of the Ranger Policy database, i.e. ranger_db. Please note that if you are using Oracle, you must specify the Oracle tablespace name here.
4. **Driver class name for a JDBC Ranger database** – the driver class name is automatically generated based on the selected DB Flavor. The table below lists the default driver class settings. Currently Ranger does not support any third party JDBC driver.

Table 3.2. Driver Class Name

DB Flavor	Driver class name for a JDBC Ranger database
MySQL	com.mysql.jdbc.Driver
Oracle	oracle.jdbc.driver.OracleDriver
PostgreSQL	org.postgresql.Driver
MS SQL	com.microsoft.sqlserver.jdbc.SQLServerDriver
SQLA	sap.jdbc4.sqlanywhere.IDriver

5. **Ranger DB username and Ranger DB Password** – Enter the user name and passwords for your Ranger database server. The following table describes these settings in more detail. You can use the MySQL database that was installed with Ambari, or an external MySQL, Oracle, PostgreSQL, MS SQL or SQL Anywhere database.

Table 3.3. Ranger DB Username Settings

Property	Description	Default Value	Example Value	Required?
Ranger DB username	The username for the Policy database.	rangeradmin	rangeradmin	Yes
Ranger DB password	The password for the Ranger Policy database user.		PassWORD	Yes

6. JDBC connect string



Important

Currently the Ambari installer generates the JDBC connect string using the `jdbc:oracle:thin:@//host:port/db_name` format. You must replace the connection string as described in the following table:

Table 3.4. JDBC Connect String

DB Flavor	Syntax	Example Value
MySQL	jdbc:mysql://DB_HOST:PORT/ db_name	jdbc:mysql:// c6401.ambari.apache.org:3306/ ranger_db
Oracle	For Oracle SID: jdbc:oracle:thin:@DB_HOST:PORT:SID	jdbc:oracle:thin:@c6401.ambari.apache.org:1521:ORCL
	For Oracle Service Name: jdbc:oracle:thin:@//DB_HOST[:PORT] [/ServiceName]	jdbc:oracle:thin:@// c6401.ambari.apache.org:1521/XE
PostgreSQL	jdbc:postgresql://DB_HOST/ db_name	jdbc:postgresql:// c6401.ambari.apache.org:5432/ ranger_db
MS SQL	jdbc:sqlserver:// DB_HOST;databaseName=db_name	jdbc:sqlserver:// c6401.ambari.apache.org:1433;databaseName=ranger_db
SQLA	jdbc:sqlanywhere:host=DB_HOST;data	jdbc:sqlanywhere:host=c6401.ambari.apache.org:2638;data

7. Setup Database and Database User

- If set to **Yes** – The Database Administrator (DBA) user name and password will need to be provided as described in the next step.



Note

Ranger does not store the DBA user name and password after setup. Therefore, you can clear these values in the Ambari UI after the Ranger setup is complete.

- If set to **No** – A **No** indicates that you do not wish to provide Database Administrator (DBA) account details to the Ambari Ranger installer. Setting this to No continues the Ranger installation process without providing DBA account details. In this case, you must perform the system database user setup as described in [Setting up Database Users Without Sharing DBA Credentials](#), and then proceed with the installation.



Note

If **No** is selected and the UI still requires you to enter a user name and password in order to proceed, you can enter any value – the values do not need to be the actual DBA user name and password.

8. **Database Administrator (DBA) username and Database Administrator (DBA) password** – The DBA username and password are set when the database server is installed. If you do not have this information, contact the database administrator who installed the database server.

Table 3.5. DBA Credential Settings

Property	Description	Default Value	Example Value	Required?
Database Administrator (DBA) username	The Ranger database user that has administrative	root	root	Yes

Property	Description	Default Value	Example Value	Required?
	privileges to create database schemas and users.			
Database Administrator (DBA) password	The root password for the Ranger database user.		root	Yes

If the Oracle DB root user Role is SYSDBA, you must also specify that in the **Database Administrator (DBA) username** parameter. For example, if the DBA user name is `orcl_root` you must specify `orcl_root AS SYSDBA`.



Note

As mentioned in the note in the previous step, if **Setup Database and Database User** is set to **No**, a placeholder DBA username and password may still be required in order to continue with the Ranger installation.

The following images show examples of the DB settings for each Ranger database type.



Note

To test the DB settings, click **Test Connection**. If a Ranger database has not been pre-installed, **Test Connection** will fail even for a valid configuration.

MySQL

Ranger Admin [Ranger User Info](#) [Ranger Plugin](#) [Ranger Audit](#) [Ranger Tagsync](#) [Advanced](#)

Ranger Admin

DB FLAVOR
MYSQL

Ranger DB name
ranger

Ranger DB username
rangeradmin

JDBC connect string for a Ranger database
jdbc:mysql://c6402.ambari.apache.org:330

Ranger DB host
c6402.ambari.apache.org

Driver class name for a JDBC Ranger database
com.mysql.jdbc.Driver

Ranger DB password

Setup Database and Database User
 Yes

Database Administrator (DBA) username
rangerdba

Database Administrator (DBA) password

JDBC connect string for root user
jdbc:mysql://c6402.ambari.apache.org:330

Oracle – if the Oracle instance is running with a Service name.

Ranger Admin [Ranger User Info](#) [Ranger Plugin](#) [Ranger Audit](#) [Ranger Tagsync](#) [Advanced](#)

Ranger Admin

DB FLAVOR
ORACLE ▾

Ranger DB name
ranger

Ranger DB username
rangeradmin

JDBC connect string for a Ranger database
//c6402.ambari.apache.org:1521/XE/ranger

Ranger DB host
c6402.ambari.apache.org:1521/XE

Driver class name for a JDBC Ranger database
oracle.jdbc.driver.OracleDriver

Ranger DB password
..... ⓘ ⓘ

Setup Database and Database User
 Yes

Database Administrator (DBA) username
rangerdba

Database Administrator (DBA) password
..... ⓘ ⓘ

JDBC connect string for root user
cthin:@//c6402.ambari.apache.org:1521/XE

Oracle – if the Oracle instance is running with a SID.

Ranger Admin [Ranger User Info](#) [Ranger Plugin](#) [Ranger Audit](#) [Ranger Tagsync](#) [Advanced](#)

Ranger Admin

DB FLAVOR
ORACLE

Ranger DB name
ranger

Ranger DB username
rangeradmin

JDBC connect string for a Ranger database
jdbc:oracle:thin:@//c6402.ambari.apache.org:1521:ORCL

Ranger DB host
c6402.ambari.apache.org:1521:ORCL

Driver class name for a JDBC Ranger database
oracle.jdbc.driver.OracleDriver

Ranger DB password

Setup Database and Database User
 Yes

Database Administrator (DBA) username
rangerdba

Database Administrator (DBA) password

JDBC connect string for root user
jdbc:oracle:thin:@//c6402.ambari.apache.org:1521:ORCL

PostgreSQL

Ranger Admin [Ranger User Info](#) [Ranger Plugin](#) [Ranger Audit](#) [Ranger Tagsync](#) [Advanced](#)

Ranger Admin

DB FLAVOR
POSTGRES

Ranger DB name
ranger

Ranger DB username
rangeradmin

JDBC connect string for a Ranger database
jdbc://c6402.ambari.apache.org:5432/ranger

Ranger DB host
c6402.ambari.apache.org:5432

Driver class name for a JDBC Ranger database
org.postgresql.Driver

Ranger DB password

Setup Database and Database User
 Yes

Database Administrator (DBA) username
rangerdba

Database Administrator (DBA) password

JDBC connect string for root user
jdbc://c6402.ambari.apache.org:5432/postgres

MS SQL

Ranger Admin [Ranger User Info](#) [Ranger Plugin](#) [Ranger Audit](#) [Ranger Tagsync](#) [Advanced](#)

Ranger Admin

DB FLAVOR
MSSQL

Ranger DB name
ranger

Ranger DB username
rangeradmin

JDBC connect string for a Ranger database
ari.apache.org:1433;databaseName=ranger

Ranger DB host
c6402.ambari.apache.org:1433

Driver class name for a JDBC Ranger database
com.microsoft.sqlserver.jdbc.SQLServerDriver

Ranger DB password

Setup Database and Database User
 Yes

Database Administrator (DBA) username
rangerdba

Database Administrator (DBA) password

JDBC connect string for root user
sqlserver://c6402.ambari.apache.org:1433;

SQL Anywhere

Ranger Admin
Ranger User Info
Ranger Plugin
Ranger Audit
Ranger Tagsync
Advanced

Ranger Admin

DB FLAVOR

SQL Anywhere ▼

Ranger DB name

ranger

Ranger DB username

rangeradmin

JDBC connect string for a Ranger database

!ambari.apache.org:2638;database=ranger

Ranger DB host

c6402.ambari.apache.org:2638

Driver class name for a JDBC Ranger database

sap.jdbc4.sqlanywhere.IDriver

Ranger DB password

***** 🔒 ***** 🔒

Setup Database and Database User

Yes 🔗 🔒

Database Administrator (DBA) username

rangerdba

JDBC connect string for root user

vhere:host=c6402.ambari.apache.org:2638;

Database Administrator (DBA) password

***** 🔒 ***** 🔒

3.1.3.2.2. Ranger Audit Settings



Important

As of HDP-2.5, Audit to DB is no longer supported. If you previously used Audit to DB, you can migrate the logs to Solr using the instructions in [Migrating Audit Logs from DB to Solr in Ambari Clusters](#).

Apache Ranger uses Apache Solr to store audit logs and provides UI searching through the audit logs. Solr must be installed and configured before installing Ranger Admin or any of the Ranger component plugins. The default configuration for Ranger Audits to Solr uses

the shared Solr instance provided under the [Ambari Infra](#) service. Solr is both memory and CPU intensive. If your production system has high volume of access requests, make sure that the Solr host has adequate memory, CPU, and disk space.

SolrCloud is the preferred setup for production usage of Ranger. [SolrCloud](#), which is deployed with the [Ambari Infra](#) service, is a scalable architecture that can run as a single node or multi-node cluster. It has additional features such as replication and sharding, which is useful for high availability (HA) and scalability. You should plan your deployment based on your cluster size. Because audit records can grow dramatically, plan to have at least 1 TB of free space in the volume on which Solr will store the index data. Solr works well with a minimum of 32 GB of RAM. You should provide as much memory as possible to the Solr process. It is highly recommended to use SolrCloud with at least two Solr nodes running on different servers with [replication](#) enabled. SolrCloud also requires Apache ZooKeeper.

1. On the Customize Services page, select the Ranger Audit tab.

It is recommended that you store audits in Solr and HDFS. Both of these options are set to ON by default. Solr provides the capability to index and search on the most recent logs while HDFS is used as the more permanent or longer term store. By default, Solr is used to index the preceding 30 days of audit logs.

2. Under Audit to Solr, click **OFF** under SolrCloud to enable SolrCloud. The button label will change to ON, and the SolrCloud configuration settings will be loaded automatically.

The screenshot shows the 'Add Service Wizard' interface for configuring Ranger Audit. The 'Ranger Audit' tab is selected, and the 'Audit to Solr' section is active. The 'Audit to Solr' section has a toggle switch set to 'ON'. Below it, the 'SolrCloud' toggle switch is set to 'OFF'. The 'ranger.audit.solr.url' field is populated with 'c5402.ambari.apache.org:2181/solr/hanger_auc'. The 'ranger.audit.solr.username' field is populated with 'ranger_solr'. The 'ranger.audit.solr.password' field is masked with dots. The 'Audit to HDFS' section has a toggle switch set to 'ON' and a 'Destination HDFS Directory' field populated with 'c5402.ambari.apache.org:8000'. At the bottom, a green message bar states 'All configurations have been addressed.' and there are 'Back' and 'Next' buttons.

3.1.3.2.3. Configure Ranger User Sync

This section describes how to configure Ranger User Sync for either UNIX or LDAP/AD.

- [Test Run Ranger Usersync \[210\]](#)
- [Configuring Ranger User Sync for UNIX \[210\]](#)
- [Configuring Ranger User Sync for LDAP/AD \[211\]](#)
- [Automatically Assign ADMIN/KEYADMIN Role for External Users \[219\]](#)

3.1.3.2.3.1. Test Run Ranger Usersync

Steps

Before committing to usersync changes, it is recommended that you test-run that users and groups are being retrieved as intended.

To test-run loading User and Group data into Ranger before committing to the changes:

1. Set `ranger.usersync.policymanager.mockrun=true`. This parameter can be found in Ambari > Ranger > Configs > Advanced > Advanced ranger-ugsync-site.
2. View the Users and Groups that will be loaded into Ranger: `tail -f /var/log/ranger/usersync/usersync.log`.
3. After confirming that the users and groups are retrieved as intended, set `ranger.usersync.policymanager.mockrun=false` and restart Ranger Usersync.

This will sync the users shown in the usersync log to the Ranger database.

3.1.3.2.3.2. Configuring Ranger User Sync for UNIX

Before you begin

Before committing to usersync changes, it is recommended that you test-run that users and groups are being retrieved as intended: [Test Run Ranger Usersync \[210\]](#).

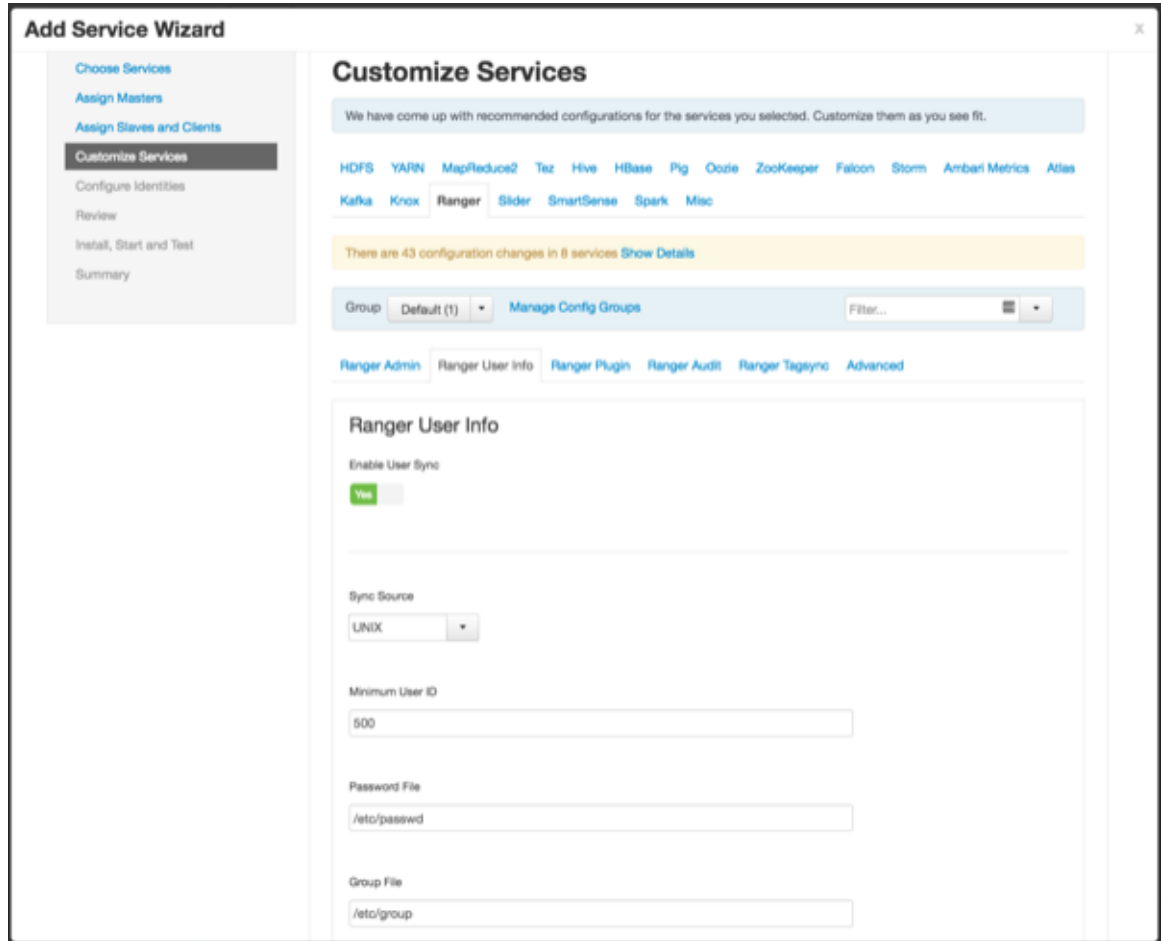
Steps

Use the following steps to configure Ranger User Sync for UNIX.

1. On the Customize Services page, select the Ranger User Info tab.
2. Click **Yes** under Enable User Sync.
3. Use the Sync Source drop-down to select UNIX, then set the following properties.

Table 3.6. UNIX User Sync Properties

Property	Description	Default Value
Sync Source	Only sync users above this user ID.	500
Password File	The location of the password file on the Linux server.	/etc/passwd
Group File	The location of the groups file on the Linux server.	/etc/group



3.1.3.2.3.3. Configuring Ranger User Sync for LDAP/AD



Important

To ensure that LDAP/AD group level authorization is enforced in Hadoop, you should [Setting Up Hadoop Group Mapping for LDAP/AD \[185\]](#).



Note

You can use the [LDAP Connection Check tool](#) to determine User Sync settings for LDAP/AD.

Before you begin

Before committing to usersync changes, it is recommended that you test-run that users and groups are being retrieved as intended: [Test Run Ranger Usersync \[210\]](#).

Steps

Use the following steps to configure Ranger User Sync for LDAP/AD.

1. On the Customize Services page, select the Ranger User Info tab.

2. Click **Yes** under Enable User Sync.
3. Use the Sync Source drop-down to select LDAP/AD.
4. Set the following properties on the Common Configs tab.

Table 3.7. LDAP/AD Common Configs

Property	Description	Default Value	Sample Values
LDAP/AD URL	Add URL depending upon LDAP/AD sync source	ldap://{host}:{port}	ldap:// ldap.example.com:389 or ldaps:// ldap.example.com:636
Bind Anonymous	If Yes is selected, the Bind User and Bind User Password are not required.	NO	
Bind User	The location of the groups file on the Linux server.	The full distinguished name (DN), including common name (CN), of an LDAP/AD user account that has privileges to search for users. The LDAP bind DN is used to connect to LDAP and query for users and groups.	cn=admin,dc=example,dc=com or admin@example.com
Bind User Password	The password of the Bind User.		
Incremental Sync	<p>If Yes is selected, Ranger Usersync saves the latest timestamp of all the objects that are synced previously and uses that timestamp to perform the next sync. Usersync then uses a polling mechanism to perform incremental sync by using LDAP attributes uSNChanged (for AD) or modifytimestamp (for LDAP).</p> <p>Enabling Incremental Sync for the first time will initiate a full sync; subsequent sync operations will be incremental.</p> <p>When Incremental Sync is enabled, Group Sync (under the Group Configs tab) is mandatory.</p> <p>Recommended for large deployments.</p>	<p>For upgrade installations: No</p> <p>For new installations: Yes</p>	Yes

Ranger Admin **Ranger User Info** Ranger Plugin Ranger Audit Ranger Tagsync Advanced

Ranger User Info

Enable User Sync

Sync Source
LDAP/AD

Common Configs **User Configs** Group Configs

LDAP/AD URL
ldap://c6401.ambari.apache.org:389

Bind User
cn=admin,dc=example,dc=com or admin@example.com

Bind User Password
[password field] [password field]

Incremental Sync

5. Set the following properties on the User Configs tab.

Table 3.8. LDAP/AD User Configs

Property	Description	Default Value	Sample Values
Group User Map Sync	Sync specific groups for users.	Yes	Yes
Username Attribute	The LDAP user name attribute.		sAMAccountName for AD, uid or cn for OpenLDAP
User Object Class	Object class to identify user entries.	person	top, person, organizationalPerson, user, or posixAccount
User Search Base	Search base for users.		cn=users,dc=example,dc=com;ou=example1,ou=e

Property	Description	Default Value	Sample Values
	Ranger can search multiple OUs in AD. Ranger UserSync module performs a user search on each configured OU and adds all the users into single list. Once all the OUs are processed, a user's group membership is computed based on the group search.		
User Search Filter	Optional additional filter constraining the users selected for syncing.		Sample filter to retrieve all the users: cn= Sample filter to retrieve all the users who are members of groupA or groupB: ((memberof=CN=GroupA,OU=groups,DC=example (memberof=CN=GroupB,OU=groups,DC=example
User Search Scope	This value is used to limit user search to the depth from search base.	sub	base, one, or sub
User Group Name Attribute	Attribute from user entry whose values would be treated as group values to be pushed into the Access Manager database. You can provide multiple attribute names separated by commas.	memberof,ismemberof	memberof, ismemberof, or gidNumber
Enable User Search	This option is available only when the "Enable Group Search First" option is selected.	No	Yes

Ranger User Info

Enable User Sync

Sync Source
 ▼

[Common Configs](#) [User Configs](#) [Group Configs](#)

Username Attribute

User Object Class

User Search Base

User Search Filter

User Search Scope

User Group Name Attribute

Group User Map Sync

6. Set the following properties on the Group Configs tab.

Table 3.9. LDAP/AD Group Configs


Property	Description	Default Value	Sample Values
Enable Group Sync	If Enable Group Sync is set to No, the group names the users belong to are derived	No	Yes


Property	Description	Default Value	Sample Values
	<p>from "User Group Name Attribute". In this case no additional group filters are applied.</p> <p>If Enable Group Sync is set to Yes, the groups the users belong to are retrieved from LDAP/AD using the following group-related attributes.</p> <p>Enabled by default if "Incremental Sync" is enabled under the Common Configs tab.</p>		
Group Member Attribute	The LDAP group member attribute name.		member
Group Name Attribute	The LDAP group name attribute.		distinguishedName for AD, cn for OpenLdap
Group Object Class	LDAP Group object class.		group, groupofnames, or posixGroup
Group Search Base	<p>Search base for groups.</p> <p>Ranger can search multiple OUs in AD. Ranger UserSync module performs a user search on each configured OU and adds all the users into single list. Once all the OUs are processed, a user's group membership is computed based on the group search configuration. Each OU segment needs to be separated by a ; (semi-colon).</p>		ou=groups,DC=example,DC=com;ou=group1;ou=
Group Search Filter	Optional additional filter constraining the groups selected for syncing.		<p>Sample filter to retrieve all groups: cn=*</p> <p>Sample filter to retrieve only the groups whose cn is Engineering or Sales: (&(cn=Engineering)(cn=Sales))</p>
Enable Group Search First	<p>When Enable Group Search First is selected, there are two possible ways of retrieving users:</p> <ul style="list-style-type: none"> If Enable User Search is not selected: users are retrieved from the "member" attribute of the group. If Enable User Search is selected: user membership is computed by performing an LDAP search based on the user configuration. 	No	Yes


Property	Description	Default Value	Sample Values
Sync Nested Groups	<p>Enables nested group memberships in Ranger so that the policies configured for parent groups are applied for all the members in the subgroups.</p> <p>If a group itself is a member of another group, the users belonging to the member group are part of the parent group as well.</p> <p>Group Hierarchy Levels determines evaluated nested group depth.</p> <p>If you do not see the Sync Nested Groups flag, upgrade to Ambari 2.6.0+.</p>	No	Yes, No
Group Hierarchy Levels	<p>Determines how many nested groups to evaluate in support of Sync Nested Groups.</p> <p>If Group Hierarchy Levels is greyed out, enable Sync Nested Groups.</p> <p>Set to any integer >0.</p>	0	2


[Common Configs](#) [User Configs](#) [Group Configs](#)


Enable Group Sync
 Yes


Group Member Attribute
 


Group Name Attribute
 


Group Object Class
 

Group Search Base
 

Group Search Filter
 

Enable Group Search First
 Yes 

Sync Nested Groups
 Yes 

Group Hierarchy Levels
 

3.1.3.2.4. Automatically Assign ADMIN/KEYADMIN Role for External Users

About this task

You can use usersync to mark specific external users, or users in a specific external group, with ADMIN or KEYADMIN role within Ranger. This is useful in cases where internal users are not allowed to login to Ranger.

Steps

1. From Ambari>Ranger>Configs>Advanced>Custom ranger-ugsync-site, select Add Property.

2. Add the following properties:

- `ranger.usersync.role.assignment.list.delimiter = &`

The default value is `&`.

- `ranger.usersync.users.groups.assignment.list.delimiter = :`

The default value is `:`.

- `ranger.usersync.username.groupname.assignment.list.delimiter = ,`

The default value is `,`.

- `ranger.usersync.group.based.role.assignment.rules =`

`ROLE_SYS_ADMIN:u:userName1,userName2&ROLE_SYS_ADMIN:g:groupName1,groupName2&RO`

3. Click Add.

4. Restart Ranger.

Example

```
ranger.usersync.role.assignment.list.delimiter = &
ranger.usersync.users.groups.assignment.list.delimiter = :
ranger.usersync.username.groupname.assignment.list.delimiter = ,
ranger.usersync.group.based.role.assignment.rules : &
ROLE_SYS_ADMIN:u:ldapuser_12,ldapuser2
```

3.1.3.2.5. Configure Ranger Tagsync

To configure Ranger Tagsync, select Ranger Tagsync on the Customize Services page, then specify a Tagsync source. You can use Atlas, AtlasREST, or a file as the Tagsync source.

Table 3.10. Atlas Tag Source Properties

Property	Description
Atlas Source: Kafka endpoint	The Kafka endpoint: <code><kafka_server_url>:6667</code>
Atlas Source: ZooKeeper endpoint	The ZooKeeper endpoint: <code><zookeeper_server_url>:2181</code>

Property	Description
Atlas Source: Kafka consumer group	The Ranger entities consumer.

Table 3.11. AtlasREST Source Properties

Property	Description
AtlasREST Source: Atlas endpoint	The AtlasREST source endpoint: <atlas_host_url>:21000
AtlasREST Source: Atlas source download interval	The AtlasREST source download interval (milliseconds).

Table 3.12. File Tag Source Properties

Property	Description
File Source: File update polling interval	The file update polling interval (milliseconds).
File Source: Filename	The tag source file name.

Add Service Wizard

Ranger Admin Ranger User Info Ranger Plugin Ranger Audit Ranger Tagsync **Advanced**

Atlas Tag Source

Enable Atlas Tag Source

Atlas Source: Kafka endpoint
o6402.ambari.apache.org:6607

Atlas Source: Zookeeper endpoint
o6402.ambari.apache.org:2181

Atlas Source: Kafka consumer group
ranger_entities_consumer

AtlasRest Tag Source

Enable AtlasRest Tag Source

AtlasREST Source: Atlas endpoint
[Input Field]

AtlasREST Source: Atlas source download interval
[Input Field]

File Tag Source

Enable File Tag Source

File Source: File update polling interval
[Input Field]

File Source: Filename
[Input Field]

All configurations have been addressed.

3.1.3.2.6. Configure Ranger Authentication

This section describes how to configure Ranger authentication for UNIX, LDAP, and AD.

- [Configuring Ranger UNIX Authentication \[221\]](#)
- [Configuring Ranger LDAP Authentication \[222\]](#)
- [Configuring Ranger Active Directory Authentication \[225\]](#)

3.1.3.2.6.1. Configuring Ranger UNIX Authentication

Use the following steps to configure Ranger authentication for UNIX.

1. Select the Advanced tab on the Customize Services page.
2. Under Ranger Settings, specify the Ranger Access Manager/Service Manager host address in the **External URL** box in the format `http://<your_ranger_host>:6080`.
3. Under Ranger Settings, select **UNIX**.

HTTP is enabled by default – if you disable HTTP, only HTTPS is allowed.

4. Under UNIX Authentication Settings, set the following properties.

Table 3.13. UNIX Authentication Settings

Property	Description	Default Value	Example Value
Allow remote Login	Flag to enable/disable remote login. Only applies to UNIX authentication.	true	true
ranger.unixauth.service.hostname	The address of the host where the UNIX authentication service is running.	{{ugsync_host}}	{{ugsync_host}}
ranger.unixauth.service.port	The port number on which the UNIX authentication service is running.	5151	5151



Note

Properties with value `{{xyz}}` are macro variables that are derived from other specified values in order to streamline the configuration process. Macro variables can be edited if required – if you need to restore the original value, click the Set Recommended symbol at the right of the property box.

3.1.3.2.6.2. Configuring Ranger LDAP Authentication



Note

You can use the [LDAP Connection Check tool](#) to determine authentication settings for LDAP.

Use the following steps to configure Ranger authentication for LDAP.

1. Select the Advanced tab on the Customize Services page.
2. Under Ranger Settings, specify the Ranger Access Manager/Service Manager host address in the **External URL** box in the format `http://<your_ranger_host>:6080`.
3. Under Ranger Settings, select **LDAP**.
4. Under LDAP Settings, set the following properties.

Table 3.14. LDAP Authentication Settings

Property	Description	Default Value	Example Value
ranger.ldap.base.dn	The Distinguished Name (DN) of the starting point for directory server searches.	dc=example,dc=com	dc=example,dc=com
Bind User	The full Distinguished Name (DN), including Common Name (CN) of	{{ranger_ug_ldap_bind_dn}}ranger_ug_ldap_bind_dn}}	

Property	Description	Default Value	Example Value
	an LDAP user account that has privileges to search for users. This is a macro variable value that is derived from the Bind User value from Ranger User Info > Common Configs .		
Bind User Password	Password for the Bind User. This is a macro variable value that is derived from the Bind User Password value from Ranger User Info > Common Configs .		
ranger.ldap.group.roleattribute	The LDAP group role attribute.	cn	cn
ranger.ldap.referral	See description below.	ignore	follow ignore throw
LDAP URL	The LDAP server URL. This is a macro variable value that is derived from the LDAP/AD URL value from Ranger User Info > Common Configs .	{{ranger_ug_ldap_url}}	{{ranger_ug_ldap_url}}
ranger.ldap.user.dnpattern	The user DN pattern is expanded when a user is being logged in. For example, if the user "ldapadmin" attempted to log in, the LDAP Server would attempt to bind against the DN "uid=ldapadmin,ou=users,dc=example,dc=com" using the	uid={0},ou=users,dc=xasecure,dc=net	cn=ldapadmin,ou=Users,dc=example,dc=com

Property	Description	Default Value	Example Value
	password the user provided>		
User Search Filter	The search filter used for Bind Authentication. This is a macro variable value that is derived from the User Search Filter value from Ranger User Info > User Configs .	{{ranger_ug_ldap_user_searchfilter}}	{{ranger_ug_ldap_user_searchfilter}}



Note

Properties with value `{{xyz}}` are macro variables that are derived from other specified values in order to streamline the configuration process. Macro variables can be edited if required – if you need to restore the original value, click the Set Recommended symbol at the right of the property box.

There are three possible values for `ranger.ldap.referral`: `follow`, `throw`, and `ignore`. The recommended setting is `follow`.

When searching a directory, the server might return several search results, along with a few continuation references that show where to obtain further results. These results and references might be interleaved at the protocol level.

- When this property is set to `follow`, the LDAP service provider processes all of the normal entries first, and then follows the continuation references.
- When this property is set to `throw`, all of the normal entries are returned in the enumeration first, before the `ReferralException` is thrown. By contrast, a "referral" error response is processed immediately when this property is set to `follow` or `throw`.
- When this property is set to `ignore`, it indicates that the server should return referral entries as ordinary entries (or plain text). This might return partial results for the search.

3.1.3.2.6.3. Configuring Ranger Active Directory Authentication



Note

You can use the [LDAP Connection Check tool](#) to determine authentication settings for Active Directory.

Use the following steps to configure Ranger authentication for Active Directory.

1. Select the Advanced tab on the Customize Services page.
2. Under Ranger Settings, specify the Ranger Access Manager/Service Manager host address in the **External URL** box in the format `http://<your_ranger_host>:6080`.
3. Under Ranger Settings, select **ACTIVE_DIRECTORY**.
4. Under AD Settings, set the following properties.

Table 3.15. AD Settings

Property	Description	Default Value	Example Value
ranger.ldap.ad.base.dn	The Distinguished Name (DN) of the starting point for directory server searches.	dc=example,dc=com	dc=example,dc=com
ranger.ldap.ad.bind.dn	The full Distinguished Name (DN), including Common Name (CN) of an LDAP user account that has privileges to search for users. This is a macro variable value that is derived from the Bind User value from Ranger User Info > Common Configs .	{{ranger_ug_ldap_bind_dn}}	{{ranger_ug_ldap_bind_dn}}
ranger.ldap.ad.bind.password	Password for the bind.dn. This is a macro variable value that is derived from the Bind User Password value from Ranger User Info > Common Configs .		
Domain Name (Only for AD)	The domain name of the AD Authentication service.		dc=example,dc=com
ranger.ldap.ad.referral	See description below.	ignore	follow ignore throw
ranger.ldap.ad.url	The AD server URL. This is a macro variable value that is derived from the LDAP/AD URL value from Ranger User Info > Common Configs .	{{ranger_ug_ldap_url}}	{{ranger_ug_ldap_url}}
ranger.ldap.ad.user.searchfilter	The search filter used for Bind Authentication. This is a macro variable value that is derived from the User Search Filter value from Ranger User Info > User Configs .	{{ranger_ug_ldap_user_searchfilter}}	{{ranger_ug_ldap_user_searchfilter}}



Note

Properties with value `{{xyz}}` are macro variables that are derived from other specified values in order to streamline the configuration process. Macro variables can be edited if required – if you need to restore the original value, click the Set Recommended symbol at the right of the property box.

There are three possible values for `ranger.ldap.ad.referral`: `follow`, `throw`, and `ignore`. The recommended setting is `follow`.

When searching a directory, the server might return several search results, along with a few continuation references that show where to obtain further results. These results and references might be interleaved at the protocol level.

- When this property is set to `follow`, the AD service provider processes all of the normal entries first, and then follows the continuation references.

- When this property is set to `throw`, all of the normal entries are returned in the enumeration first, before the `ReferralException` is thrown. By contrast, a "referral" error response is processed immediately when this property is set to `follow` or `throw`.
- When this property is set to `ignore`, it indicates that the server should return referral entries as ordinary entries (or plain text). This might return partial results for the search. In the case of AD, a `PartialResultException` is returned when referrals are encountered while search results are processed.

Add Service Wizard

Ranger Settings

External URL:

Authentication method: LDAP ACTIVE_DIRECTORY UNIX NONE

HTTP enabled:

AD Settings

ranger.kdap.ad.base.dn:

ranger.kdap.ad.bind.dn:

ranger.kdap.ad.bind.password:

Domain Name (Only for AD):

ranger.kdap.ad.referral:

ranger.kdap.ad.uri:

ranger.kdap.ad.user.searchfilter:

Knox SSO Settings

Advanced ranger-admin-site

When you have finished configuring all of the Customize Services Settings, click **Next** at the bottom of the page to continue with the installation.

5. When you save the authentication method as Active Directory, a Dependent Configurations pop-up may appear recommending that you set the authentication method as LDAP. This recommended configuration should not be applied for AD, so you should clear (un-check) the **ranger.authentication.method** check box, then click **OK**.

Dependent Configurations

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

<input type="checkbox"/>	Property	Service	Config Group	File Name	Current Value	Recommended Value
<input type="checkbox"/>	ranger.authentication.method	Ranger	Default	ranger-admin-site	UNIX	LDAP

Cancel OK

3.1.3.3. Complete the Ranger Installation

1. On the Review page, carefully review all of your settings and configurations. If everything looks good, click **Deploy** to install Ranger on the Ambari server.

Add Service Wizard

ADD SERVICE WIZARD

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

Review

Please review the configuration before installation

Admin Name : admin
Cluster Name : Thomas1
Total Hosts : 3 (0 new)

Repositories:

```

redhat5 (HDP-2.2):
http://public-repo-1.hortonworks.com/HDP/centos5/2.x/updates/2.2.6.0
redhat5 (HDP-UTILS-1.1.0.20):
http://public-repo-1.hortonworks.com/HDP-UTILS-1.1.0.20/repos/centos5
redhat6 (HDP-2.2):
http://public-repo-1.hortonworks.com/HDP/centos6/2.x/updates/2.2.6.0
redhat6 (HDP-UTILS-1.1.0.20):
http://public-repo-1.hortonworks.com/HDP-UTILS-1.1.0.20/repos/centos6
suse11 (HDP-2.2):
http://public-repo-1.hortonworks.com/HDP/suse11sp3/2.x/updates/2.2.6.0
suse11 (HDP-UTILS-1.1.0.20):
http://public-repo-1.hortonworks.com/HDP-UTILS-1.1.0.20/repos/suse11sp3
ubuntu12 (HDP-2.2):
http://public-repo-1.hortonworks.com/HDP/ubuntu12/2.x/updates/2.2.6.3

```

← Back Print **Deploy** →

2. When you click **Deploy**, Ranger is installed on the specified host on your Ambari server. A progress bar displays the installation progress.

Add Service Wizard

ADD SERVICE WIZARD

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

Install, Start and Test

Please wait while the selected services are installed and started.

24 % overall

Show: **All (3)** | In Progress (0) | Warning (0) | Success (0) | Fail (0)

Host	Status	Message
c6401.ambari.apache.org	8%	Installing Ranger Admin
c6402.ambari.apache.org	33%	Install complete (Waiting to start)
c6403.ambari.apache.org	33%	Install complete (Waiting to start)

3 of 3 hosts showing - Show All Show: 25 | 1 - 3 of 3

Next →

3. When the installation is complete, a Summary page displays the installation details. You may need to restart services for cluster components after installing Ranger.



Note

If the installation fails, you should complete the installation process, then reconfigure and reinstall Ranger.

3.1.3.4. Advanced Usersync Settings

To access Usersync settings, select the Advanced tab on the Customize Service page. Usersync pulls in users from UNIX, LDAP, or AD and populates Ranger's local user tables with these users.



Important

To ensure that LDAP/AD group level authorization is enforced in Hadoop, you must first [Setting Up Hadoop Group Mapping for LDAP/AD \[185\]](#) for LDAP.



Note

Before committing to usersync changes, it is recommended that you test-run that users and groups are being retrieved as intended: [Test Run Ranger Usersync \[210\]](#).

3.1.3.4.1. UNIX Usersync Settings

If you are using UNIX authentication, the default values for the Advanced ranger-ugsync-site properties are the settings for UNIX authentication.

▼ Advanced ranger-ugsync-site

ranger.usersync.idap.bindkeystore	<input type="text"/>	🔒	➕
ranger.usersync.idap.idapbindpassword	Type password <input type="password"/> Retype Password <input type="password"/>	🔒	
ranger.usersync.group.memberattributename	<input type="text"/>	🔒	➕
ranger.usersync.group.nameattribute	<input type="text"/>	🔒	➕
ranger.usersync.group.objectclass	<input type="text"/>	🔒	➕
ranger.usersync.group.searchbase	<input type="text"/>	🔒	➕
ranger.usersync.group.searchenabled	false	🔒	➕
ranger.usersync.group.searchfilter	<input type="text"/>	🔒	➕
ranger.usersync.group.searchscope	<input type="text"/>	🔒	➕
ranger.usersync.group.usermapsyncenabled	false	🔒	➕
ranger.usersync.idap.searchBase	dc=hadoop,dc=apache,dc=org	🔒	➕
ranger.usersync.source.impl.class	org.apache.ranger.unixusersync.process.UnixUserGroupBuilder	🔒	➕
ranger.usersync.credstore.filename	/usr/hdp/current/ranger-usersync/conf/ugsync.jceks	🔒	➕
ranger.usersync.enabled	true	🔒	➕
ranger.usersync.filesource.file	/tmp/usergroup.txt	🔒	➕
ranger.usersync.filesource.text.delimiter	,	🔒	➕
ranger.usersync.keystore.file	/usr/hdp/current/ranger-usersync/conf/unixauthservice.jks	🔒	➕

3.1.3.4.2. Required LDAP and AD Usersync Settings

If you are using LDAP authentication, you must update the following Advanced ranger-ugsync-site properties.



Note

Before committing to usersync changes, it is recommended that you test-run that users and groups are being retrieved as intended: [Test Run Ranger Usersync \[210\]](#).

Table 3.16. LDAP Advanced ranger-ugsync-site Settings

Property Name	LDAP Value
ranger.usersync.ldap.bindkeystore	Set this to the same value as the <code>ranger.usersync.credstore.filename</code> property, i.e, the default value is <code>/usr/hdp/current/ranger-usersync/conf/ugsync.jceks</code>
ranger.usersync.ldap.bindalias	ranger.usersync.ldap.bindalias
ranger.usersync.source.impl.class	ldap

Table 3.17. AD Advanced ranger-ugsync-site Settings

Property Name	LDAP Value
ranger.usersync.source.impl.class	ldap

3.1.3.4.3. Additional LDAP and AD Usersync Settings

If you are using LDAP or Active Directory authentication, you may need to update the following properties, depending upon your specific deployment characteristics.



Note

Before committing to usersync changes, it is recommended that you test-run that users and groups are being retrieved as intended: [Test Run Ranger Usersync \[210\]](#).

Table 3.18. Advanced ranger-ugsync-site Settings for LDAP and AD

Property Name	LDAP ranger-ugsync-site Value	AD ranger-ugsync-site Value
ranger.usersync.ldap.url	ldap://127.0.0.1:389	ldap://ad-conrowoller-hostname:389
ranger.usersync.ldap.binddn	cn=ldapadmin,ou=users,dc=example,dc=com	cn=adadmin,cn=Users,dc=example,dc=com
ranger.usersync.ldap.ldapbindpassword	secret	secret
ranger.usersync.ldap.searchBase	dc=example,dc=com	dc=example,dc=com
ranger.usersync.source.impl.class	org.apache.ranger.ladpusersync.process.LdapUserGroupBuilder	
ranger.usersync.ldap.user.searchbase	ou=users, dc=example, dc=com	dc=example,dc=com
ranger.usersync.ldap.user.searchscope	sub	sub

Property Name	LDAP ranger-ugsync-site Value	AD ranger-ugsync-site Value
ranger.usersync.ldap.user.objectclass	person	person
ranger.usersync.ldap.user.searchfilter	Set to single empty space if no value. Do not leave it as "empty"	(objectcategory=person)
ranger.usersync.ldap.user.nameattribute	uid or cn	sAMAccountName
ranger.usersync.ldap.user.groupnameattribute	memberof,ismemberof	memberof,ismemberof
ranger.usersync.ldap.username.caseconversion	none	none
ranger.usersync.ldap.groupname.caseconversion	none	none
ranger.usersync.group.searchenabled *	false	false
ranger.usersync.group.usermapsyncenabled *	false	false
ranger.usersync.group.searchbase *	ou=groups, dc=example, dc=com	dc=example,dc=com
ranger.usersync.group.searchscope *	sub	sub
ranger.usersync.group.objectclass *	groupofnames	groupofnames
ranger.usersync.group.searchfilter *	needed for AD authentication	(member=CN={0}, OU=MyUsers, DC=AD-HDP, DC=COM)
ranger.usersync.group.nameattribute *	cn	cn
ranger.usersync.group.memberattributename *	member	member
ranger.usersync.pagedresultsenabled *	true	true
ranger.usersync.pagedresultssize *	500	500
ranger.usersync.user.searchenabled *	false	false
ranger.usersync.group.search.first.enabled *	false	false

* Only applies when you want to filter out groups.

After you have finished specifying all of the settings on the Customize Services page, click **Next** at the bottom of the page to continue with the installation.

3.1.3.5. Configuring Ranger for LDAP SSL

You can use the following steps to configure LDAP SSL using self-signed certs in the default Ranger User Sync TrustStore.

1. The default location is `/usr/hdp/current/ranger-usersync/conf/mytruststore.jks` for the `ranger.usersync.truststore.file` property.
2. Alternatively, copy and edit the self-signed ca certs.
3. Set the `ranger.usersync.truststore.file` property to that new cacert file.

```
cd /usr/hdp/<version>/ranger-usersync
service ranger-usersync stop
service ranger-usersync start
```

Where `cert.pem` has the LDAPS cert.

3.1.3.6. Setting up Database Users Without Sharing DBA Credentials

If do not wish to provide system Database Administrator (DBA) account details to the Ambari Ranger installer, you can use the `dba_script.py` Python script to create Ranger

DB database users without exposing DBA account information to the Ambari Ranger installer. You can then run the normal Ambari Ranger installation without specifying a DBA user name and password.

To create Ranger DB users using the `dba_script.py` script:

1. Download the Ranger rpm using the yum install command.

```
yum install ranger-admin
```

2. You should see one file named `dba_script.py` in the `/usr/hdp/current/ranger-admin` directory.
3. Get the script reviewed internally and verify that your DBA is authorized to run the script.
4. Execute the script by running the following command:

```
python dba_script.py
```

5. Pass all values required in the argument. These should include `db flavor`, `JDBC jar`, `db host`, `db name`, `db user`, and other parameters.

- If you would prefer not to pass runtime arguments via the command prompt, you can update the `/usr/hdp/current/ranger-admin/install.properties` file and then run:

```
python dba_script.py -q
```

When you specify the `-q` option, the script will read all required information from the `install.properties` file

- You can use the `-d` option to run the script in "dry" mode. Running the script in dry mode causes the script to generate a database script.

```
python dba_script.py -d /tmp/generated-script.sql
```

Anyone can run the script, but it is recommended that the system DBA run the script in dry mode. In either case, the system DBA should review the generated script, but should only make minor adjustments to the script, for example, change the location of a particular database file. No major changes should be made that substantially alter the script – otherwise the Ranger install may fail.

The system DBA must then run the generated script.

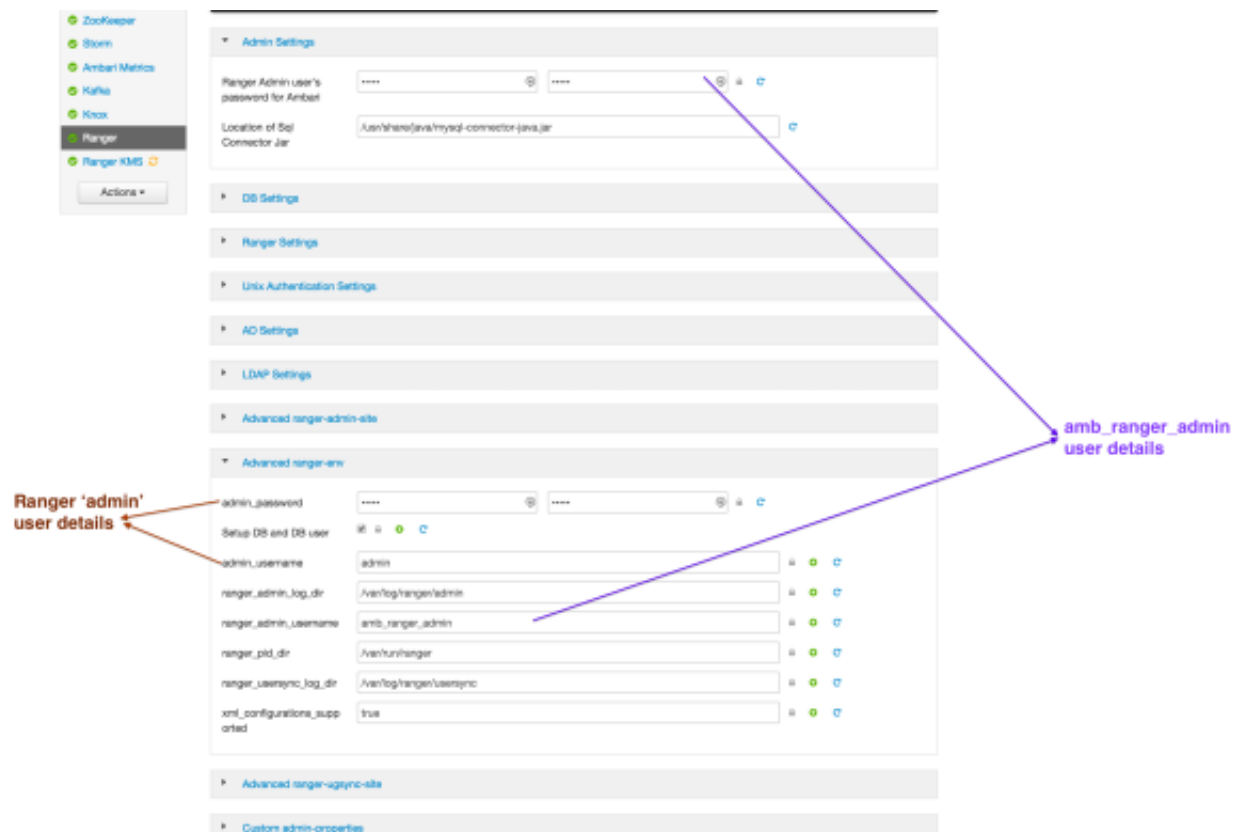
6. Run the Ranger Ambari install procedure, but set **Setup Database and Database User** to **No** in the Ranger Admin section of the Customize Services page.

3.1.3.7. Updating Ranger Admin Passwords

For the following users, if you update the passwords on the Ranger Configs page, you must also update the passwords on the Configs page of each Ambari component that has the Ranger plugin enabled. Individual Ambari component configurations are not automatically updated – the service restart will fail if you do not update these passwords on each component.

- Ranger Admin user – The credentials for this user are set in **Configs > Advanced ranger-env** in the fields labeled **admin_username** (default value: `admin`) and **admin_password** (default value: `admin`).
- Admin user used by Ambari to create repo/policies – The user name for this user is set in **Configs > Admin Settings** in the field labeled **Ranger Admin username for Ambari** (default value: `amb_ranger_admin`). The password for this user is set in the field labeled **Ranger Admin user's password for Ambari**. This password is specified during the Ranger installation.

The following image shows the location of these settings on the Ranger Configs page:



3.1.4. Enabling Ranger Plugins

Ranger plugins can be enabled for several HDP services. This section describes how to enable each of these plugins. For performance reasons, it is recommended that you store audits in Solr and HDFS, and not in a database.

If you are using a Kerberos-enabled cluster, there are a number of additional steps you must follow to ensure that you can use the Ranger plugins on a Kerberos cluster.

The following Ranger plugins are available:

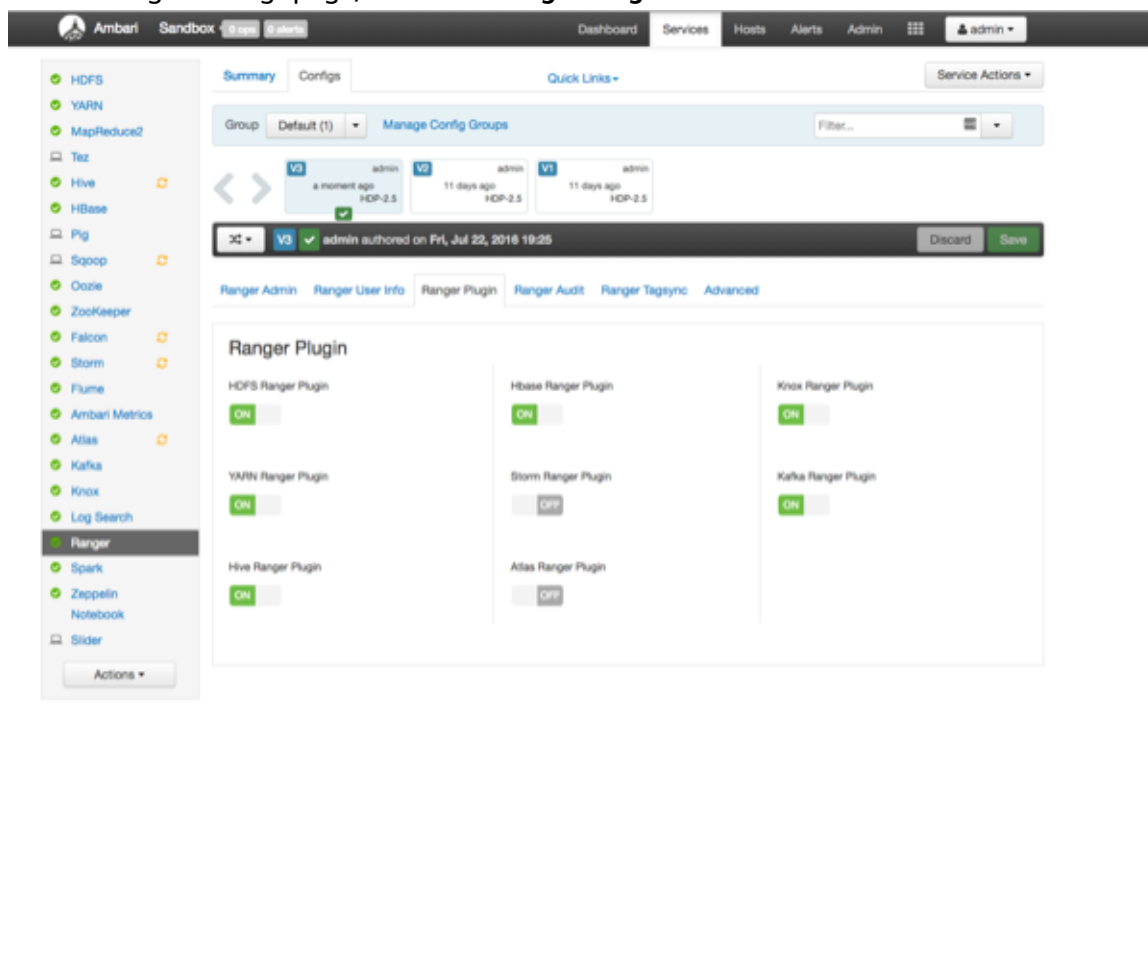
- [HDFS \[235\]](#)
- [Hive \[239\]](#)

- [HBase \[244\]](#)
- [Kafka \[248\]](#)
- [Knox \[252\]](#)
- [YARN \[256\]](#)
- [Storm \[260\]](#)
- [Atlas \[265\]](#)

3.1.4.1. HDFS

Use the following steps to enable the Ranger HDFS plugin.

1. On the Ranger Configs page, select the **Ranger Plugin** tab.

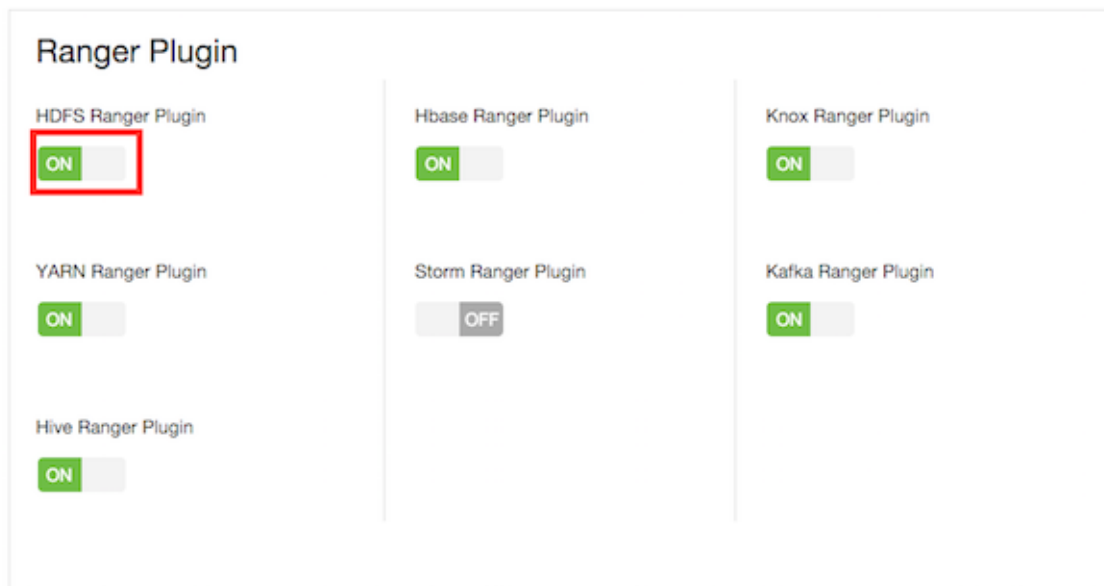


The screenshot shows the Ambari Ranger Admin interface. The top navigation bar includes 'Dashboard', 'Services', 'Hosts', 'Alerts', 'Admin', and a user profile 'admin'. The left sidebar lists various services, with 'Ranger' highlighted. The main content area is titled 'Ranger Plugin' and contains several toggle switches for different plugins:

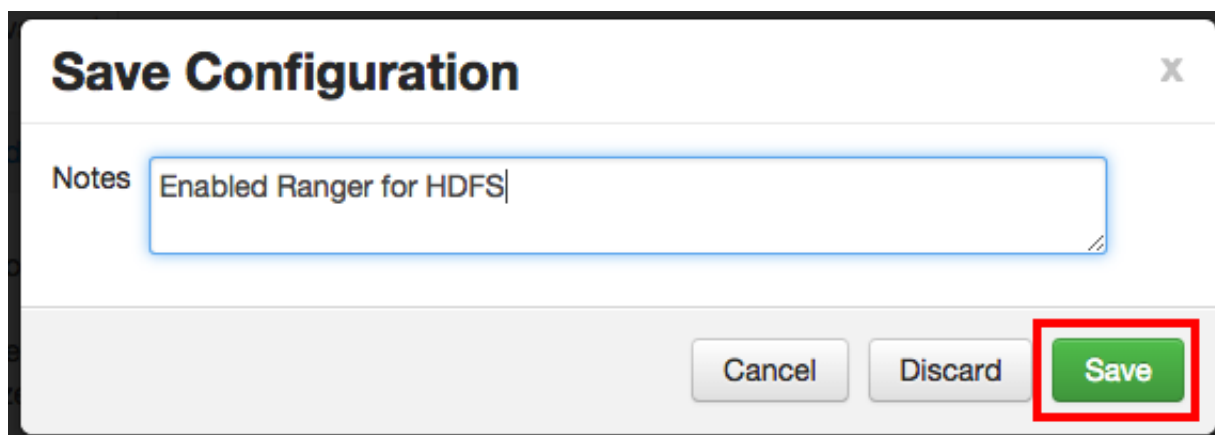
- HDFS Ranger Plugin: ON
- Hbase Ranger Plugin: ON
- Knox Ranger Plugin: ON
- YARN Ranger Plugin: ON
- Storm Ranger Plugin: OFF
- Kafka Ranger Plugin: ON
- Hive Ranger Plugin: ON
- Atlas Ranger Plugin: OFF

A black menu bar at the bottom of the configuration area contains 'Discard' and 'Save' buttons. The 'Save' button is highlighted in green.

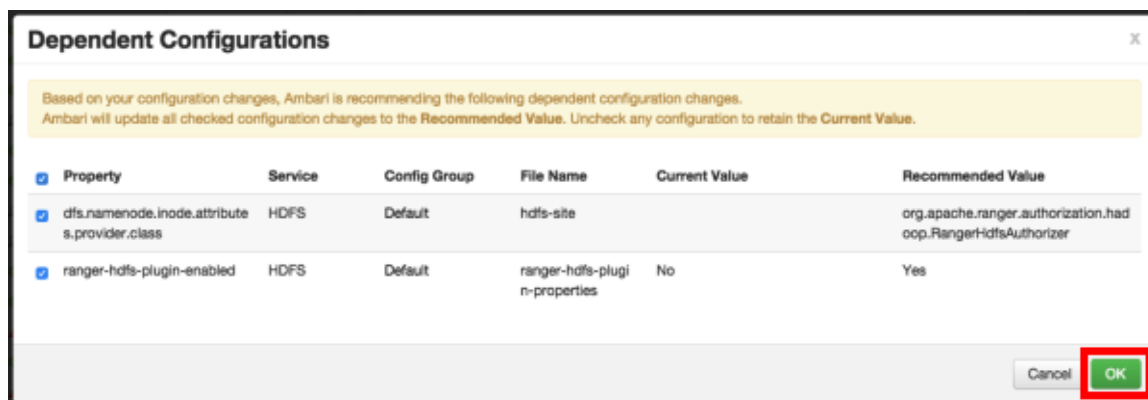
2. Under HDFS Ranger Plugin, select **On**, then click **Save** in the black menu bar.



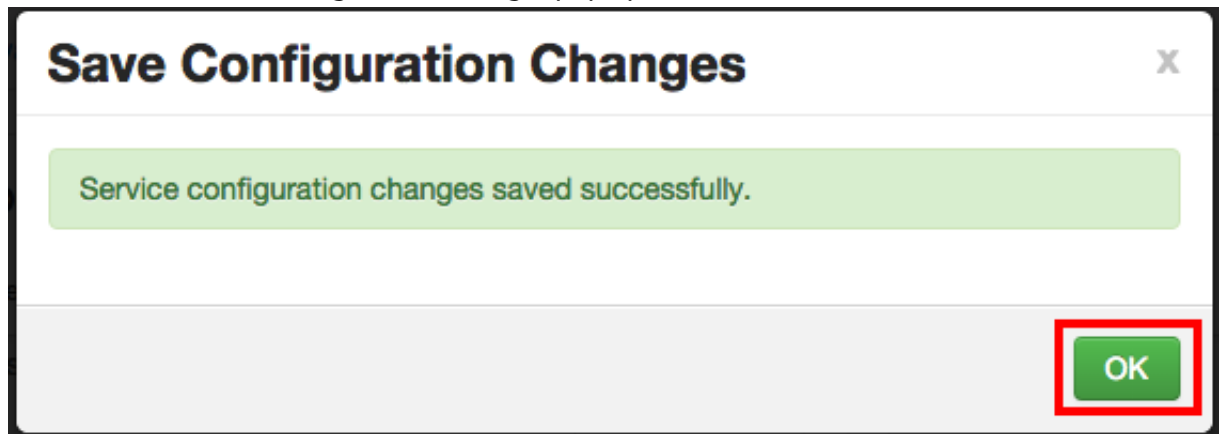
- 3. A Save Configuration pop-up appears. Type in a note describing the changes you just made, then click **Save**.



- 4. A Dependent Configuration pop-up appears. Click **OK** to confirm the configuration updates.



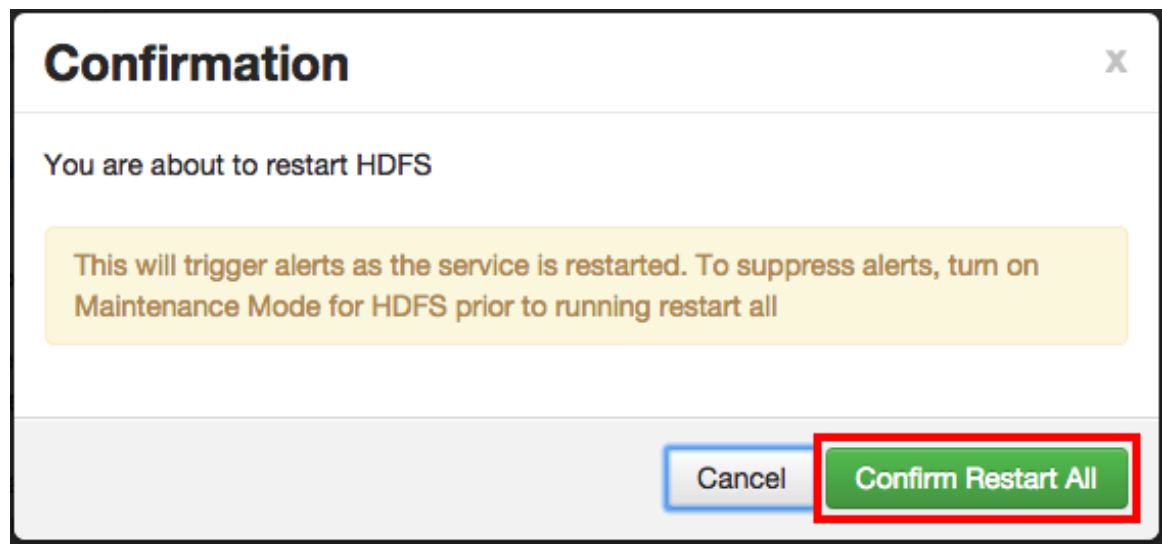
5. Click **OK** on the Save Configuration Changes pop-up.



6. Select **HDFS** in the navigation menu, then select **Restart > Restart All Affected** to restart the HDFS service and load the new configuration.

The screenshot shows the Ambari web interface for a cluster named 'test_cluster'. The top navigation bar includes 'Dashboard', 'Services', 'Hosts 1', 'Alerts', and 'Admin'. A red notification banner at the top right says 'Restart Required: 4 Components on 1 Host'. Below this, a 'Restart' button is visible, and a 'Restart All Affected' button is highlighted with a red rectangle. The left sidebar lists various services like HDFS, MapReduce2, YARN, Tez, Hive, HBase, Pig, Sqoop, Oozie, ZooKeeper, Falcon, Storm, Flume, Accumulo, Ambari Metrics, Atlas, Kafka, Knox, Mahout, Ranger, Slider, SmartSense, and Spark. The main content area shows the 'Advanced' settings for HDFS, with sections for 'NameNode' and 'DataNode' configurations. The 'NameNode' section includes fields for directories, Java heap size (set to 1GB), server threads (set to 25), and minimum replicated blocks (% set to 100%). The 'DataNode' section includes fields for directories, failed disk tolerance (set to 0), maximum Java heap size (set to 1GB), and max data transfer threads (set to 10004).

7. Click **Confirm Restart All** on the confirmation pop-up to confirm the HDFS restart.



8. After HDFS restarts, the Ranger plugin for HDFS will be enabled. Other components may also require a restart.

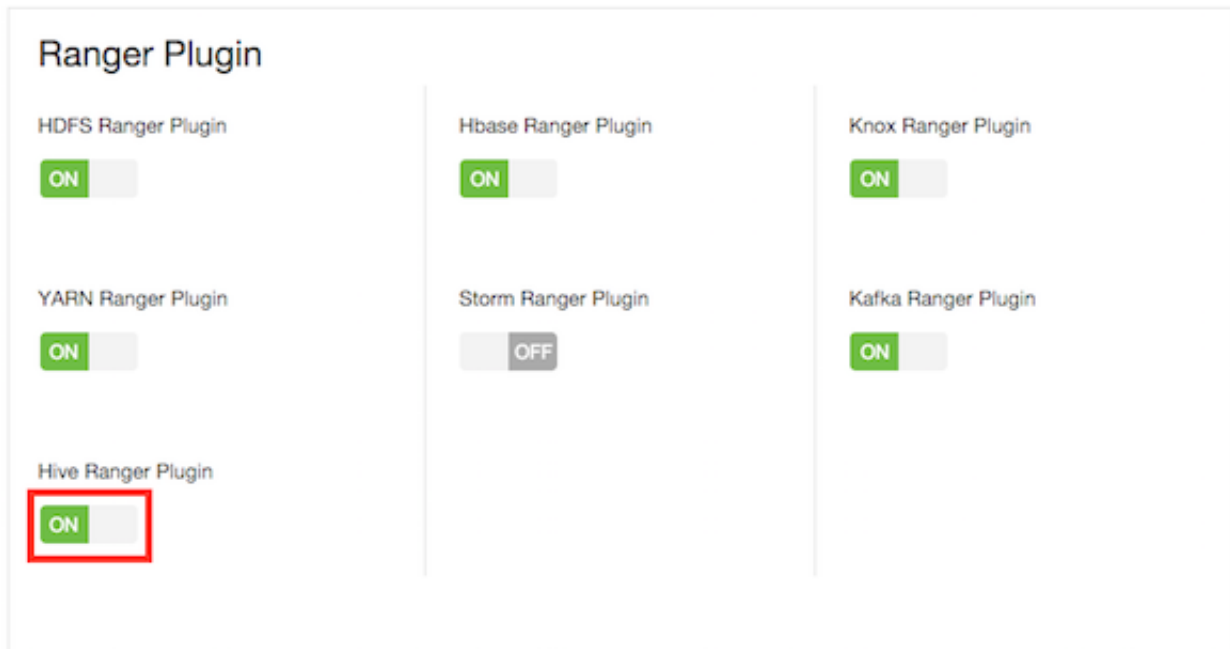
3.1.4.2. Hive

Use the following steps to enable the Ranger Hive plugin.

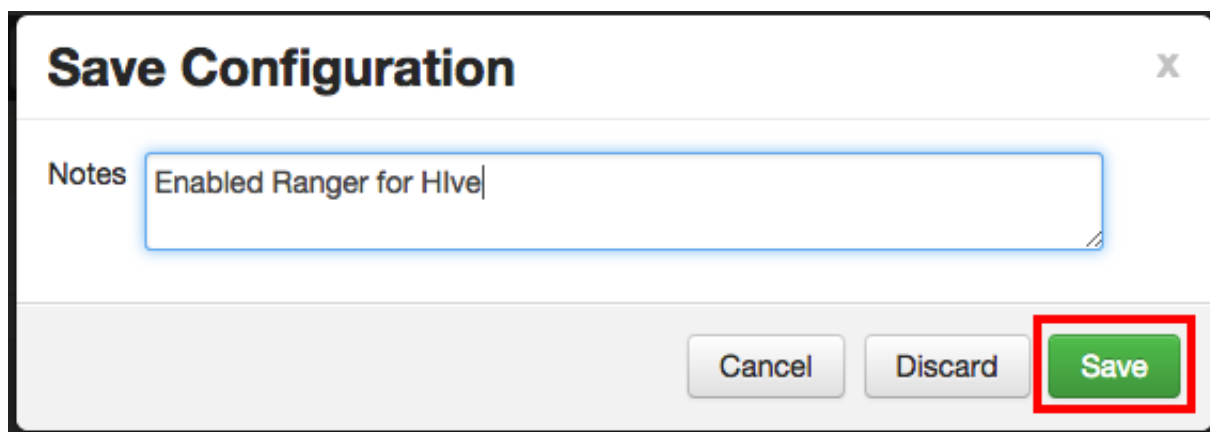
1. On the Ranger Configs page, select the **Ranger Plugin** tab.

The screenshot displays the Ambari Ranger Admin interface. The top navigation bar includes 'Ambari', 'Sandbox', 'Dashboard', 'Services', 'Hosts', 'Alerts', 'Admin', and a user profile 'admin'. The left sidebar lists various services, with 'Ranger' selected. The main content area shows the 'Ranger Admin' section, specifically the 'Ranger Plugin' configuration page. This page features a grid of toggle switches for various plugins: HDFS Ranger Plugin (ON), Hbase Ranger Plugin (ON), Knox Ranger Plugin (ON), YARN Ranger Plugin (ON), Storm Ranger Plugin (OFF), Kafka Ranger Plugin (ON), Hive Ranger Plugin (ON), and Atlas Ranger Plugin (OFF). A black menu bar at the top of the configuration area contains 'V3', a checkmark, and the text 'admin authored on Fri, Jul 22, 2016 19:25', along with 'Discard' and 'Save' buttons.

2. Under Hive Ranger Plugin, select **On**, then click **Save** in the black menu bar.



3. A Save Configuration pop-up appears. Type in a note describing the changes you just made, then click **Save**.



4. A Dependent Configuration pop-up appears. Click **OK** to confirm the configuration updates.

Dependent Configurations

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

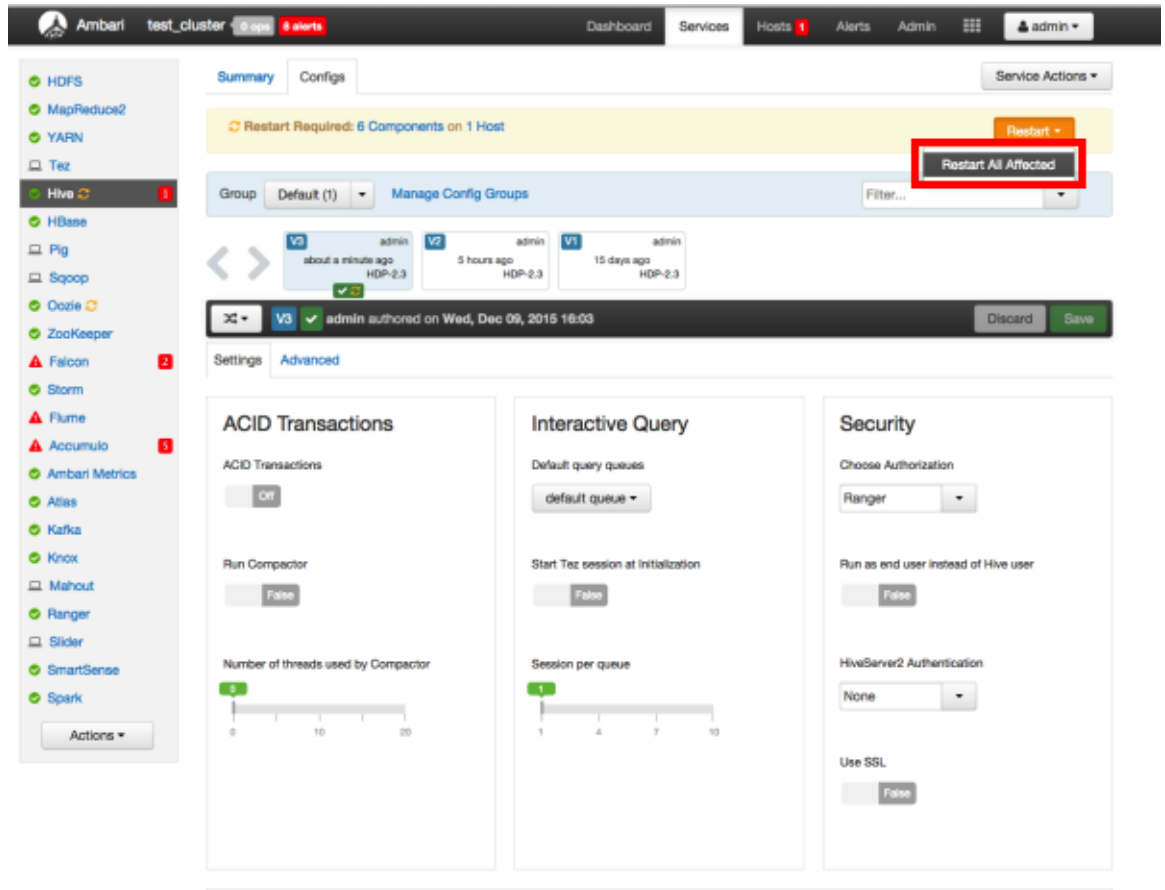
<input checked="" type="checkbox"/> Property	Service	Config Group	File Name	Current Value	Recommended Value
<input checked="" type="checkbox"/> hive.security.authorization	Hive	Default	hive-env	None	Ranger
<input checked="" type="checkbox"/> hive.security.authorization.enabled	Hive	Default	hive-site	false	true
<input checked="" type="checkbox"/> hive.server2.enable.doAs	Hive	Default	hive-site	true	false
<input checked="" type="checkbox"/> hive.security.authorization.enabled	Hive	Default	hiveserver2-site	false	true
<input checked="" type="checkbox"/> hive.conf.restricted.list	Hive	Default	hiveserver2-site		hive.security.authorization.enabled,hive.security.authorization.manager,hive.security.authenticator.manager
<input checked="" type="checkbox"/> hive.security.authenticator.manager	Hive	Default	hiveserver2-site		org.apache.hadoop.hive.ql.security.SessionStateUserAuthenticator
<input checked="" type="checkbox"/> hive.security.authorization.manager	Hive	Default	hiveserver2-site		org.apache.ranger.authorization.hive.authorizer.RangerHiveAuthorizerFactory

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

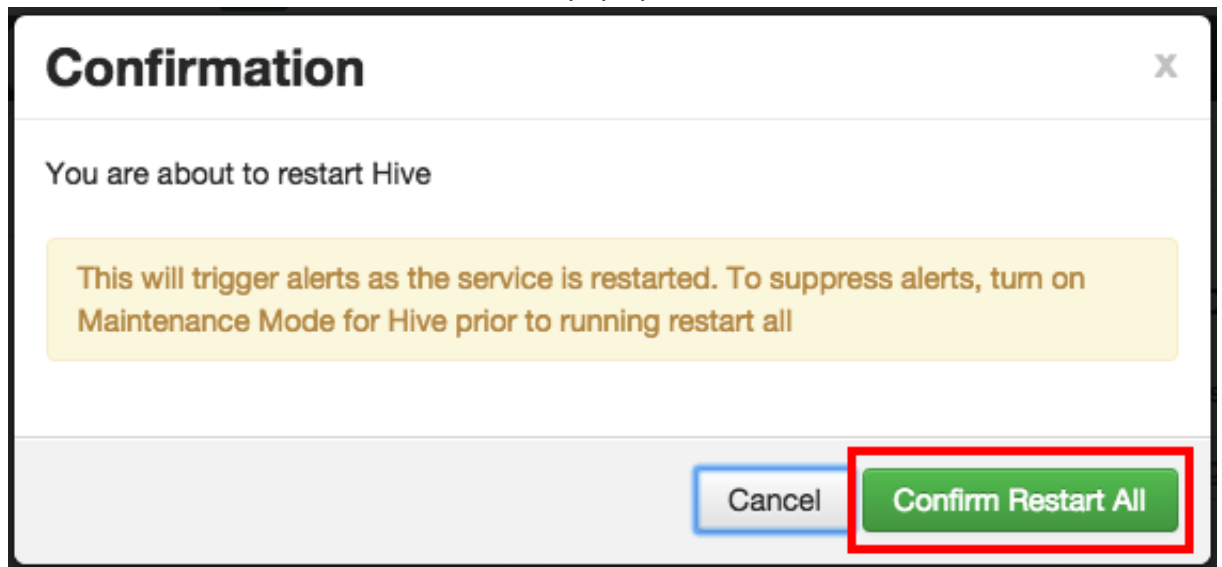
Save Configuration Changes

Service configuration changes saved successfully.

- Select **Hive** in the navigation menu, then select **Restart > Restart All Affected** to restart the Hive service and load the new configuration.



7. Click **Confirm Restart All** on the confirmation pop-up to confirm the Hive restart.



8. After Hive restarts, the Ranger plugin for Hive will be enabled.

3.1.4.3. HBase



Note

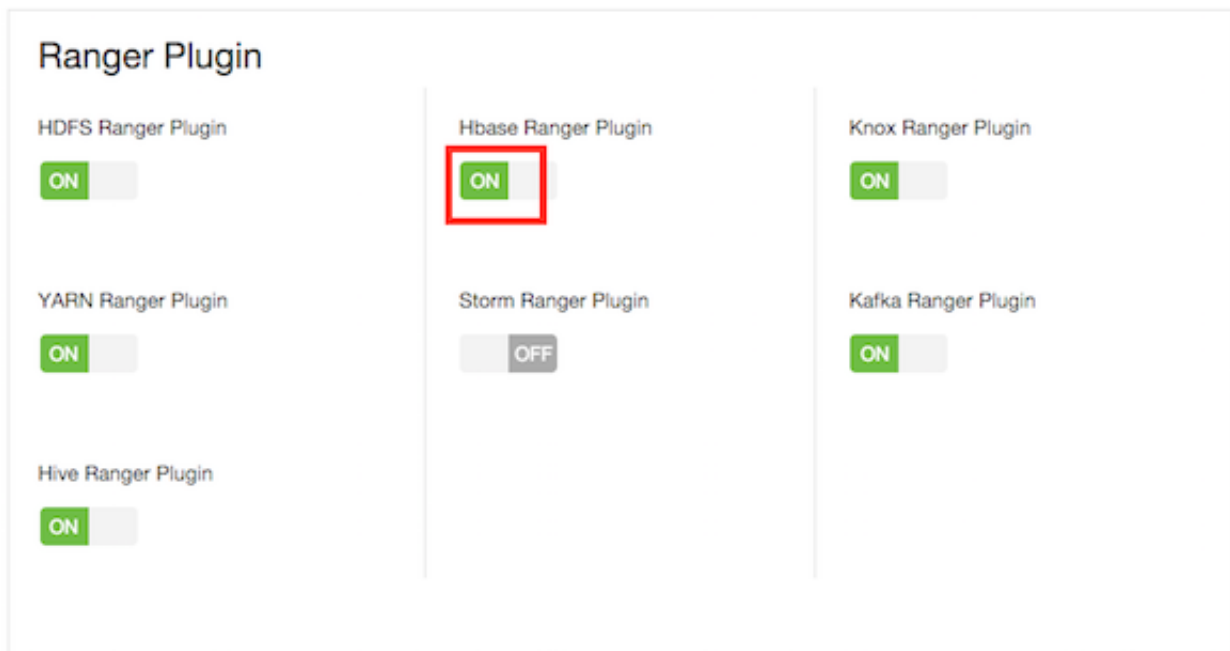
When HBase is configured with Ranger, and specifically XASecure Authorizer, you may only grant and revoke privileges.

Use the following steps to enable the Ranger HBase plugin.

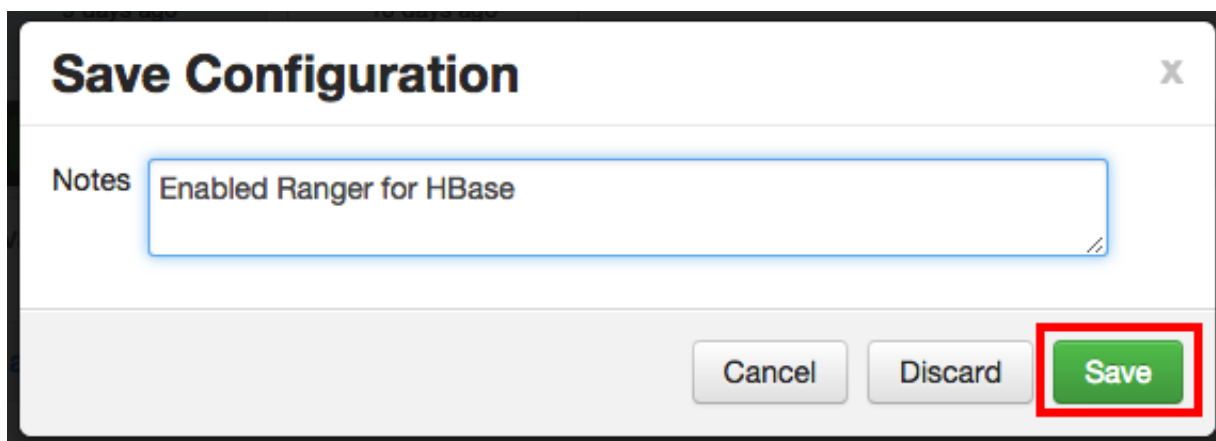
1. On the Ranger Configs page, select the **Ranger Plugin** tab.

The screenshot shows the Ambari Ranger Admin interface. The top navigation bar includes 'Ambari', 'Sandbox', 'Dashboard', 'Services', 'Hosts', 'Alerts', 'Admin', and a user profile 'admin'. The left sidebar lists various services: HDFS, YARN, MapReduce2, Tez, Hive, HBase, Pig, Sqoop, Cozie, ZooKeeper, Falcon, Storm, Flume, Ambari Metrics, Atlas, Kafka, Knox, Log Search, Ranger (selected), Spark, Zeppelin Notebook, and Slider. The main content area is titled 'Ranger Admin' and has several tabs: 'Ranger Admin', 'Ranger User Info', 'Ranger Plugin' (selected), 'Ranger Audit', 'Ranger Tagsync', and 'Advanced'. The 'Ranger Plugin' tab displays a grid of toggle switches for various plugins: HDFS Ranger Plugin (ON), YARN Ranger Plugin (ON), Hive Ranger Plugin (ON), Hbase Ranger Plugin (ON), Storm Ranger Plugin (OFF), and Atlas Ranger Plugin (OFF). A black menu bar at the bottom of the configuration area shows 'V3' and 'admin authored on Fri, Jul 22, 2016 19:25', with 'Discard' and 'Save' buttons.

2. Under HBase Ranger Plugin, select **On**, then click **Save** in the black menu bar.



3. A Save Configuration pop-up appears. Type in a note describing the changes you just made, then click **Save**.



4. A Dependent Configuration pop-up appears. Click **OK** to confirm the configuration updates.

Dependent Configurations

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

<input checked="" type="checkbox"/>	Property	Service	Config Group	File Name	Current Value	Recommended Value
<input checked="" type="checkbox"/>	hbase.security.authorization	HBase	Default	hbase-site	false	true
<input checked="" type="checkbox"/>	hbase.coprocessor.regionserver.classes	HBase	Default	hbase-site		org.apache.ranger.authorization.hbase.RangerAuthorizationCoprocesor
<input checked="" type="checkbox"/>	hbase.coprocessor.master.classes	HBase	Default	hbase-site		org.apache.ranger.authorization.hbase.RangerAuthorizationCoprocesor
<input checked="" type="checkbox"/>	hbase.coprocessor.region.classes	HBase	Default	hbase-site	org.apache.hadoop.hbase.security.access.SecureBulkLoadEndpoint	org.apache.hadoop.hbase.security.access.SecureBulkLoadEndpoint, or g.apache.ranger.authorization.hbase.RangerAuthorizationCoprocesor
<input checked="" type="checkbox"/>	ranger-hbase-plugin-enabled	HBase	Default	ranger-hbase-plugin-properties	No	Yes

Cancel **OK**

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

Save Configuration Changes

Service configuration changes saved successfully.

OK

6. Select **HBase** in the navigation menu, then select **Restart > Restart All Affected** to restart the HBase service and load the new configuration.

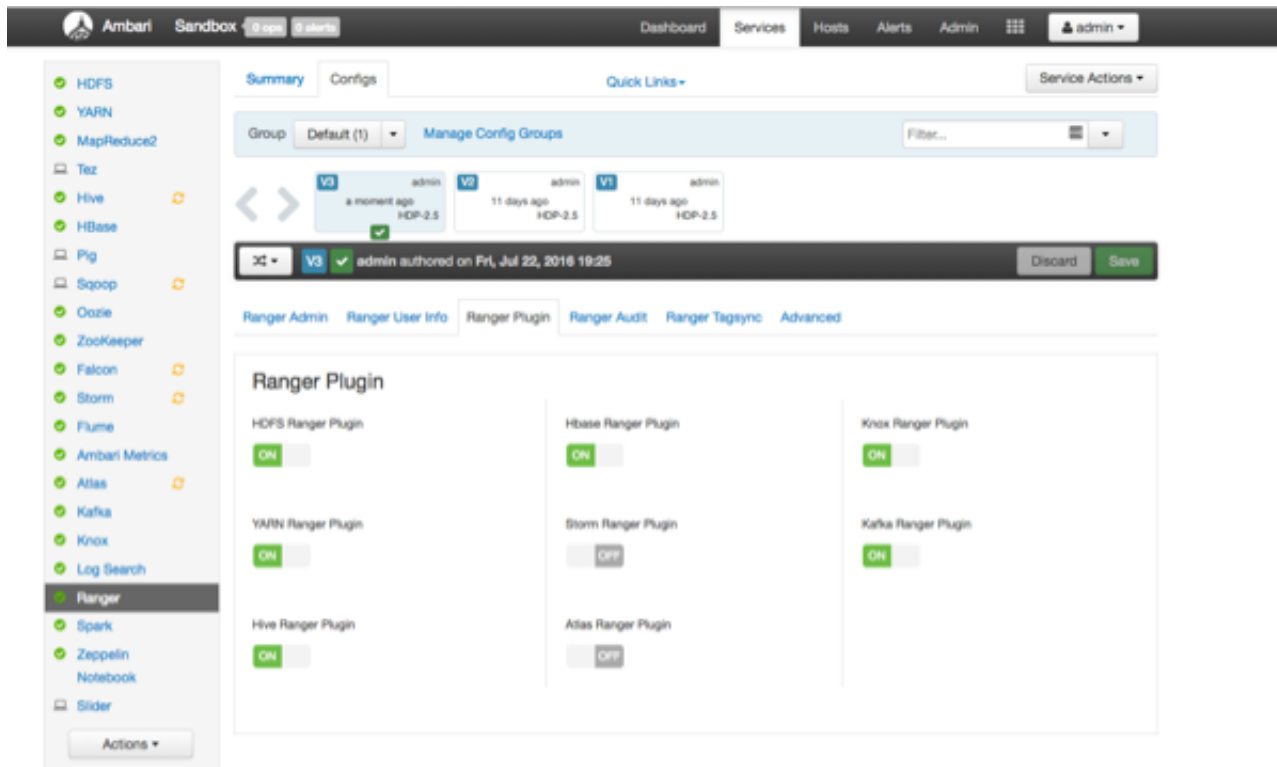
7. Click **Confirm Restart All** on the confirmation pop-up to confirm the HBase restart.

8. After HBase restarts, the Ranger plugin for HBase will be enabled.

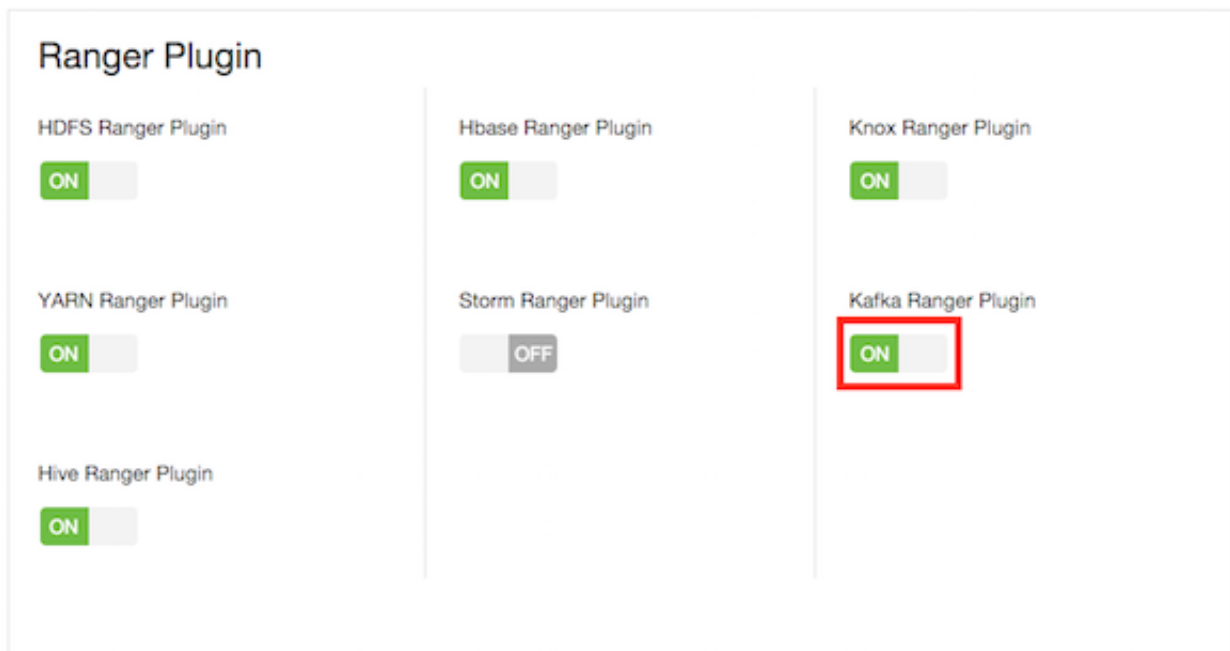
3.1.4.4. Kafka

Use the following steps to enable the Ranger Kafka plugin.

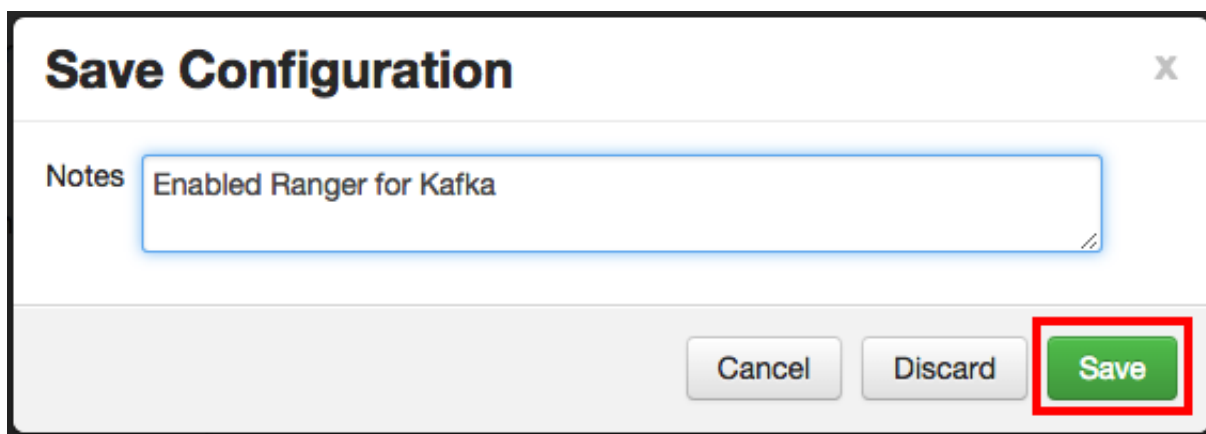
1. On the Ranger Configs page, select the **Ranger Plugin** tab.



2. Under Kafka Ranger Plugin, select **On**, then click **Save** in the black menu bar.



3. A Save Configuration pop-up appears. Type in a note describing the changes you just made, then click **Save**.



4. A Dependent Configuration pop-up appears. Click **OK** to confirm the configuration updates.

Dependent Configurations

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

<input checked="" type="checkbox"/>	Property	Service	Config Group	File Name	Current Value	Recommended Value
<input checked="" type="checkbox"/>	authorizer.class.name	Kafka	Default	kafka-broker		org.apache.ranger.authorization.kafka.authorizer.RangerKafkaAuthorizer
<input checked="" type="checkbox"/>	content	Kafka	Default	kafka-log4j	##### Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements. See the NOTICE file distributed with this work for additional information regarding copyright ownership. The ASF licenses this file to you under the Apache License, Version 2.0 (the "License"); you may not use this file except in compliance with the License. You may obtain a copy of the License at http://www.apache.org/licenses/LICENSE-2.0 Unless required by applicable law or agreed to in writing, software distributed under the License is distributed on an "AS IS" BASIS, WITHOUT WARRANTIES OR CONDITIONS OF ANY KIND, either express or implied. See the License for the specific language governing permissions and limitations under the License. ###	##### Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements. See the NOTICE file distributed with this work for additional information regarding copyright ownership. The ASF licenses this file to you under the Apache License, Version 2.0 (the "License"); you may not use this file except in compliance with the License. You may obtain a copy of the License at http://www.apache.org/licenses/LICENSE-2.0 Unless required by applicable law or agreed to in writing, software distributed under the License is distributed on an "AS IS" BASIS, WITHOUT WARRANTIES OR CONDITIONS OF ANY KIND, either express or implied. See the License for the specific language governing permissions and limitations under the License. ###

Cancel **OK**

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

Save Configuration Changes

Service configuration changes saved successfully.

OK

6. Select **Kafka** in the navigation menu, then select **Restart > Restart All Affected** to restart the Kafka service and load the new configuration.

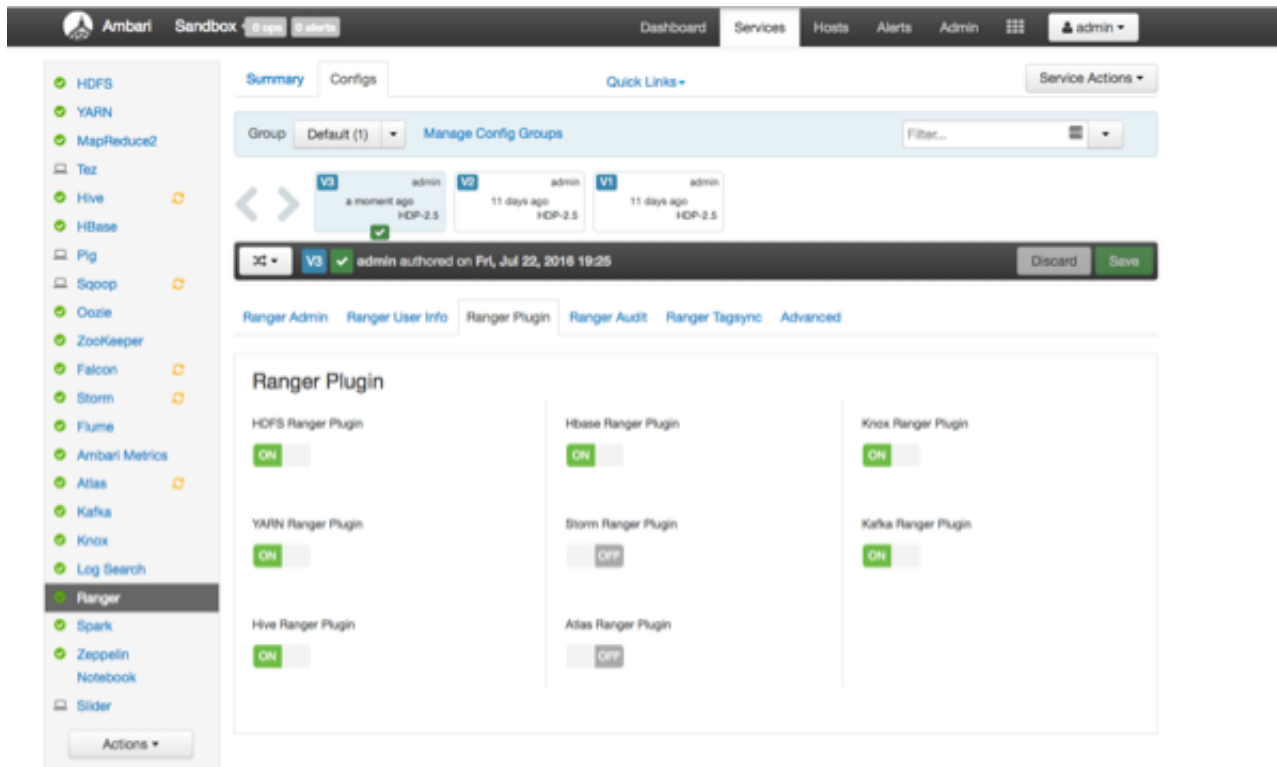
7. Click **Confirm Restart All** on the confirmation pop-up to confirm the Kafka restart.

8. After Kafka restarts, the Ranger plugin for Kafka will be enabled.

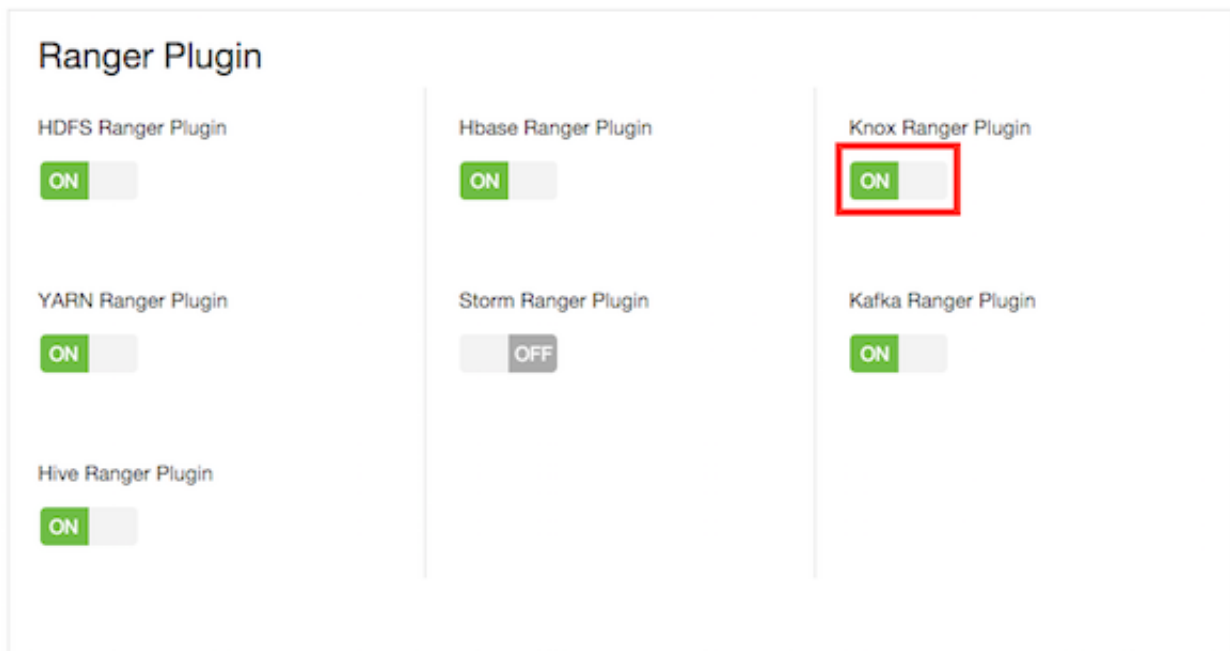
3.1.4.5. Knox

Use the following steps to enable the Ranger Knox plugin.

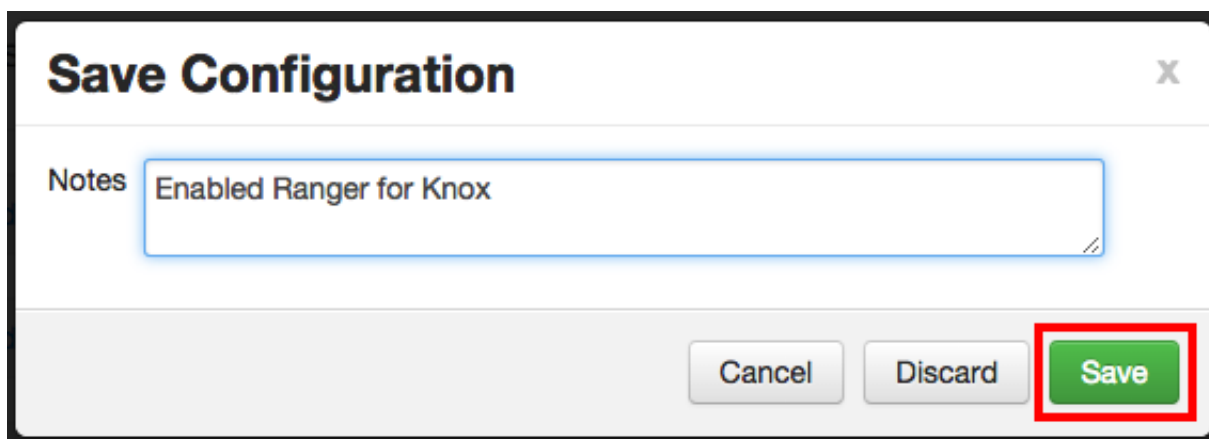
1. On the Ranger Configs page, select the **Ranger Plugin** tab.



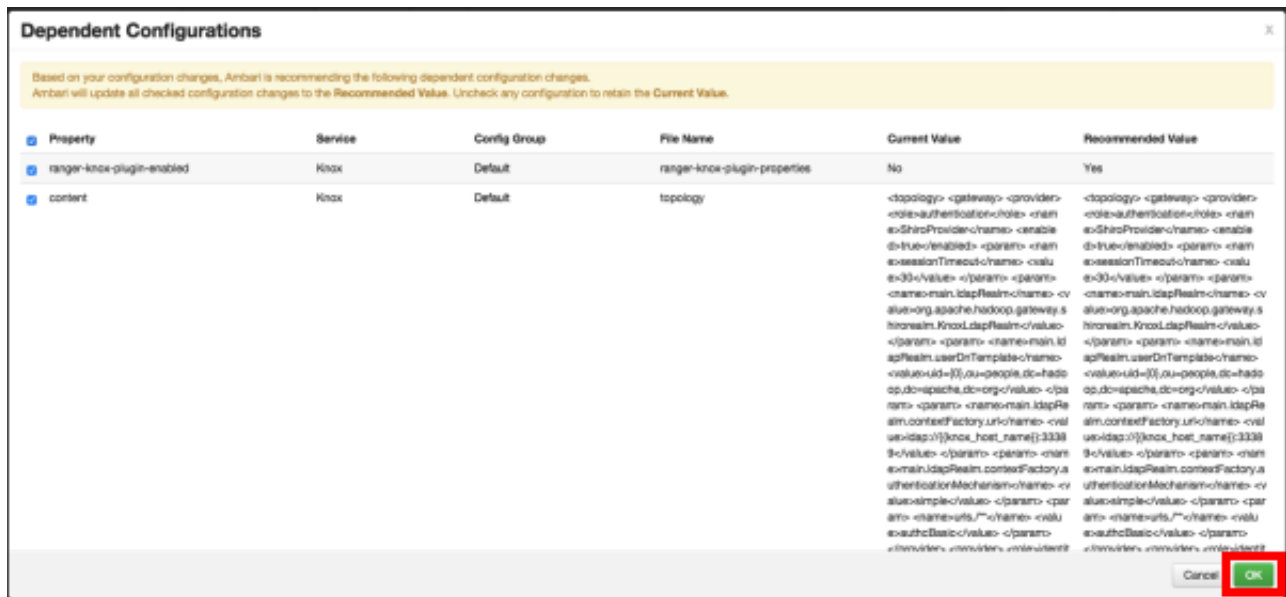
2. Under Knox Ranger Plugin, select **On**, then click **Save** in the black menu bar.



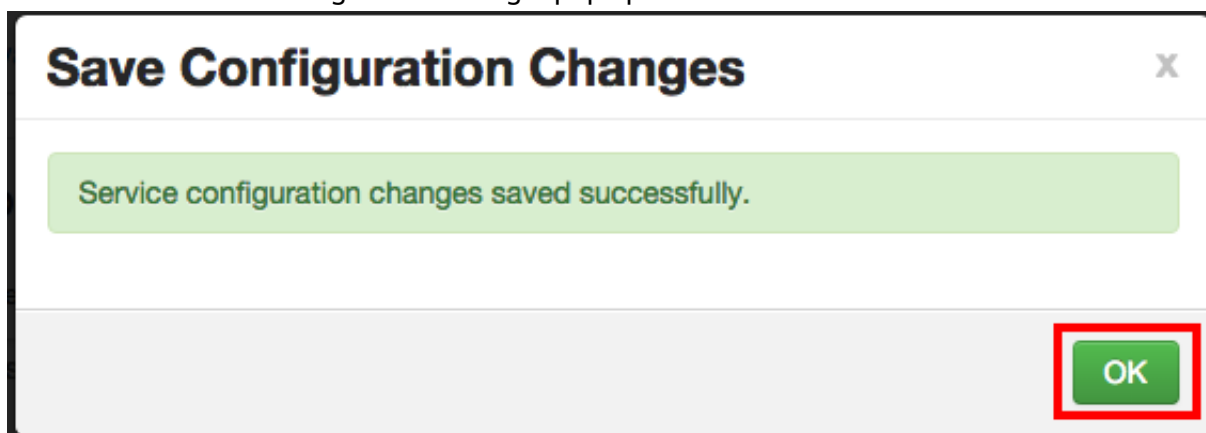
3. A Save Configuration pop-up appears. Type in a note describing the changes you just made, then click **Save**.



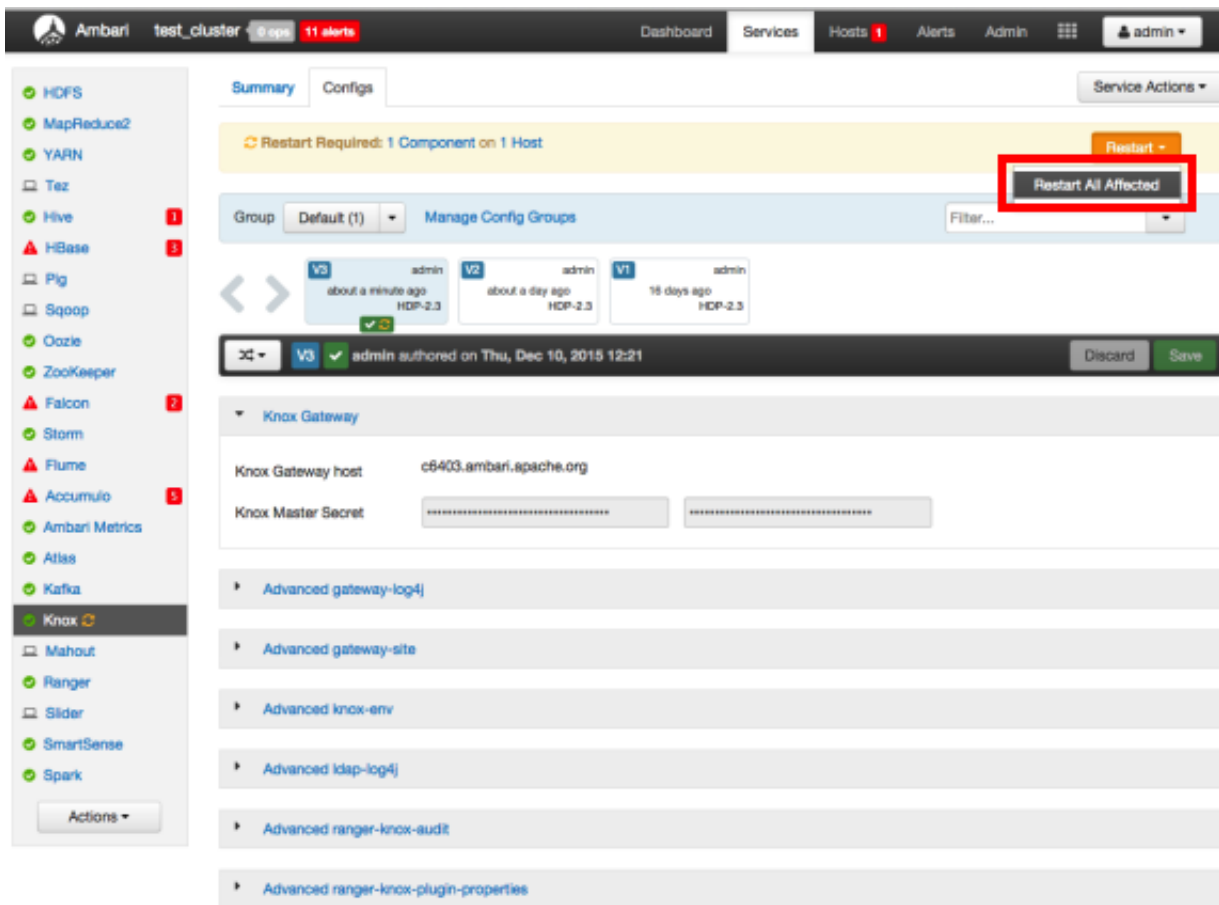
4. A Dependent Configuration pop-up appears. Click **OK** to confirm the configuration updates.



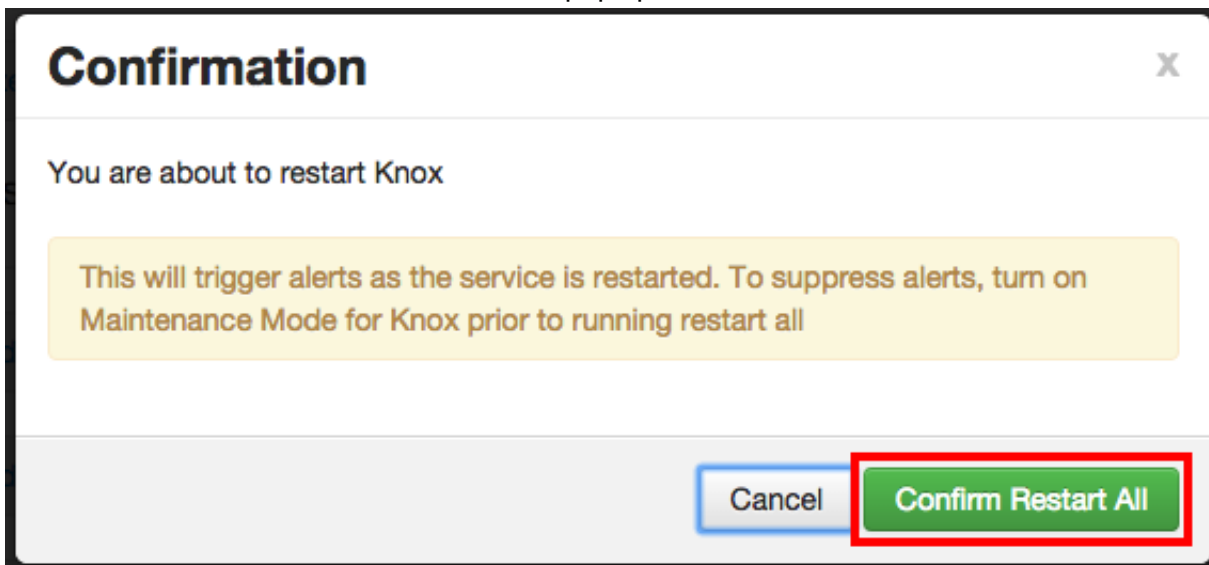
5. Click **OK** on the Save Configuration Changes pop-up.



6. Select **Knox** in the navigation menu, then select **Restart > Restart All Affected** to restart the Knox service and load the new configuration.



7. Click **Confirm Restart All** on the confirmation pop-up to confirm the Knox restart.

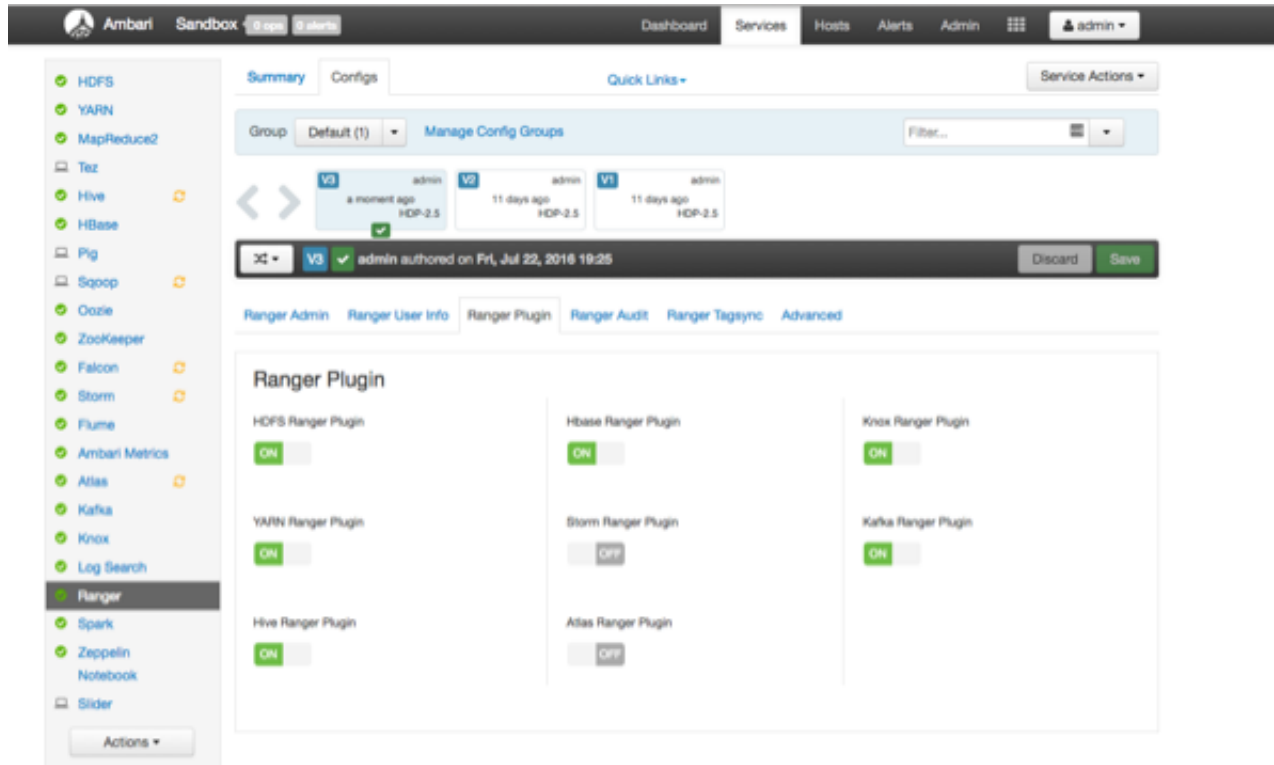


8. After Knox restarts, the Ranger plugin for Knox will be enabled.

3.1.4.6. YARN

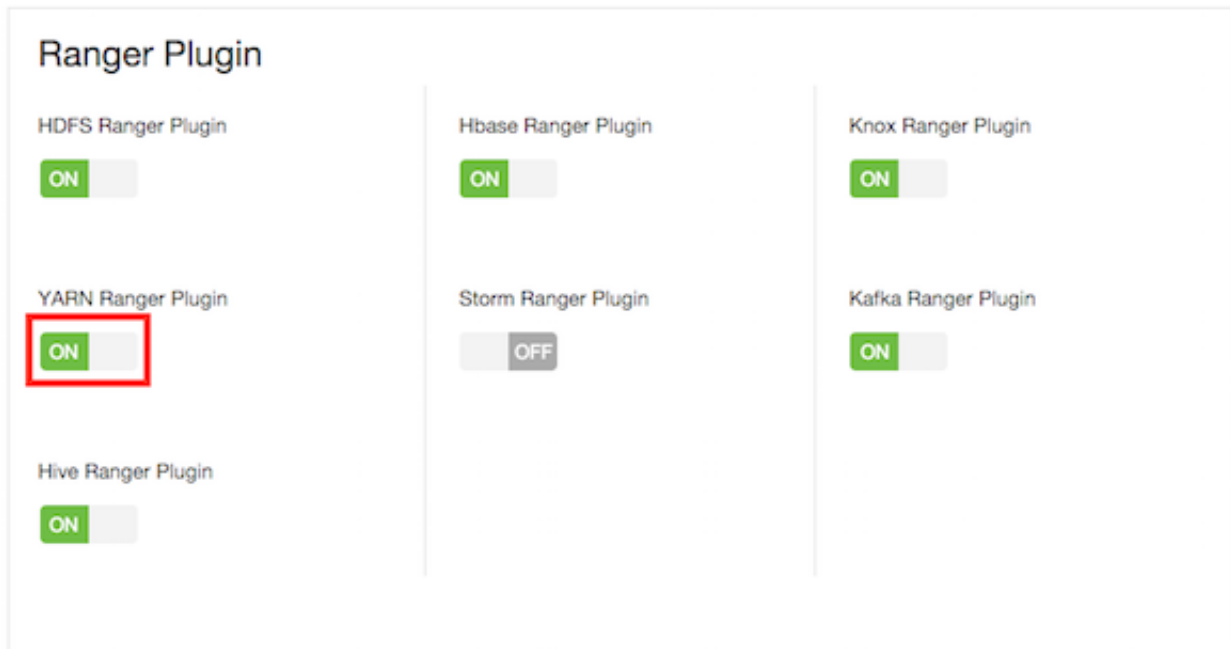
Use the following steps to enable the Ranger YARN plugin.

1. On the Ranger Configs page, select the **Ranger Plugin** tab.

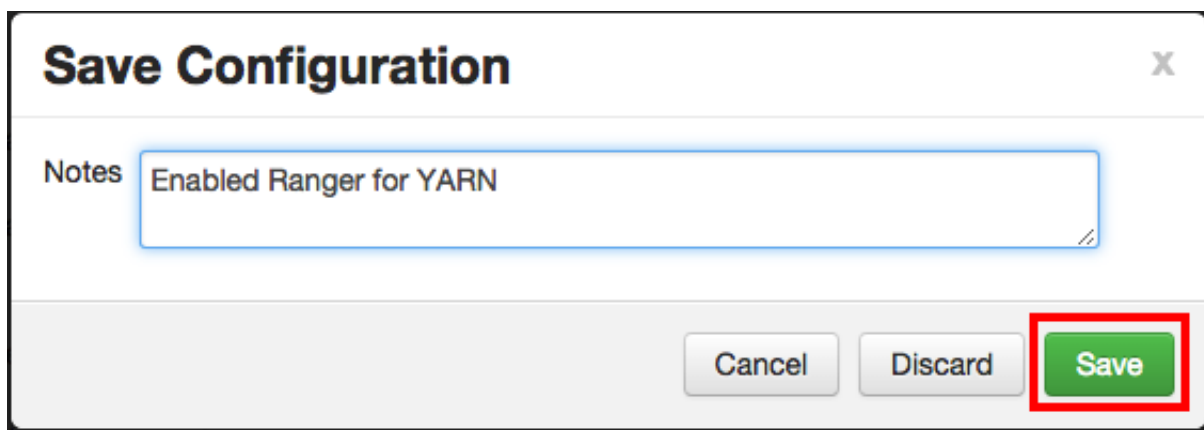


The screenshot displays the Ambari Ranger Admin interface. The top navigation bar includes 'Ambari', 'Sandbox', 'Dashboard', 'Services', 'Hosts', 'Alerts', 'Admin', and a user profile 'admin'. The left sidebar lists various services, with 'Ranger' selected. The main content area shows the 'Ranger Plugin' configuration page. The 'Ranger Plugin' section contains several toggle switches for enabling or disabling plugins: HDFS Ranger Plugin (ON), YARN Ranger Plugin (ON), Hive Ranger Plugin (ON), Hbase Ranger Plugin (ON), Storm Ranger Plugin (OFF), Atlas Ranger Plugin (OFF), and Kafka Ranger Plugin (ON). A black menu bar at the bottom of the configuration area contains 'Discard' and 'Save' buttons.

2. Under YARN Ranger Plugin, select **On**, then click **Save** in the black menu bar.



- 3. A Save Configuration pop-up appears. Type in a note describing the changes you just made, then click **Save**.



- 4. A Dependent Configuration pop-up appears. Click **OK** to confirm the configuration updates.

Ambari test1 7 alerts Dashboard Services Hosts 1 Alerts Admin admin

Summary Heatmaps Configs Quick Links Service Actions

Restart Required: 4 Components on 1 Host Restart

Restart All Affected

Restart NodeManagers

Group: YARN Default (1) Manage Config Groups Filter

V3 admin a moment ago HDP-2.3 V2 admin 9 days ago HDP-2.3 V1 admin 10 days ago HDP-2.3

admin authored on Thu, Sep 10, 2015 18:33 Discard Save

Settings Advanced

Memory

Node: Memory allocated for all YARN containers on a node. Slider: 0 MB to 256 MB, set to 612 MB.

Container: Minimum Container Size (Memory). Slider: 0 MB to 512 MB, set to 128 MB.

Maximum Container Size (Memory). Slider: 0 MB to 512 MB, set to 612 MB.

YARN Features

Node Labels: Disabled

Pre-emption: Disabled

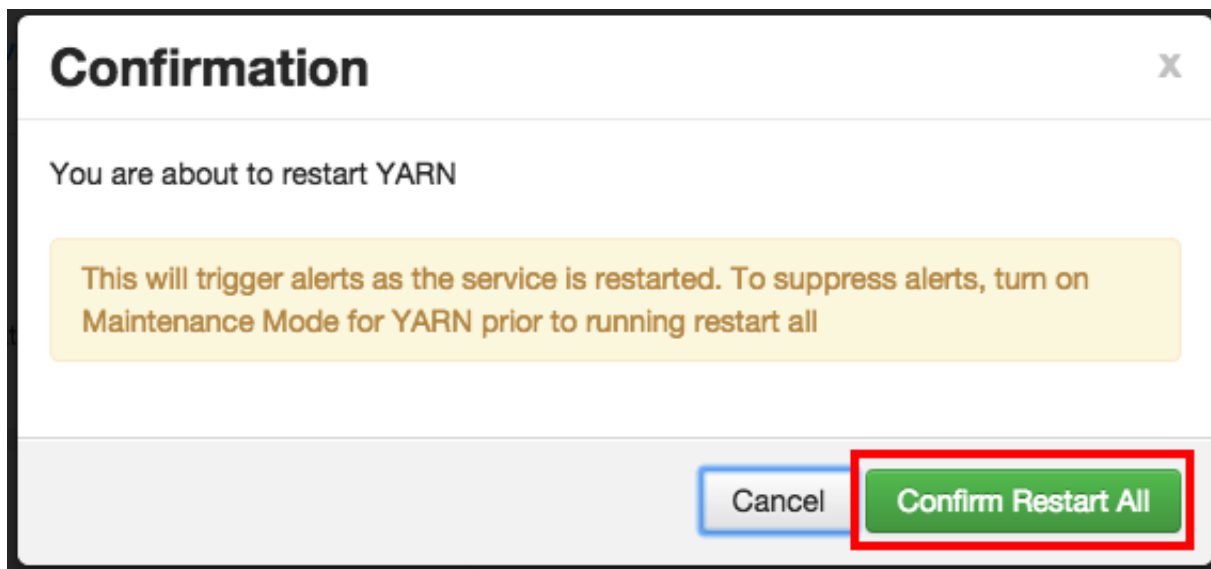
CPU

Node: CPU Scheduling: Disabled. CPU Isolation: Disabled.

Container: Minimum Container Size (VCores). Slider: 0 to 1, set to 1.

Maximum Container Size (VCores). Slider: 0 to 1, set to 1.

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



8. After YARN restarts, the Ranger plugin for YARN will be enabled. Other components may also require a restart.

3.1.4.7. Storm

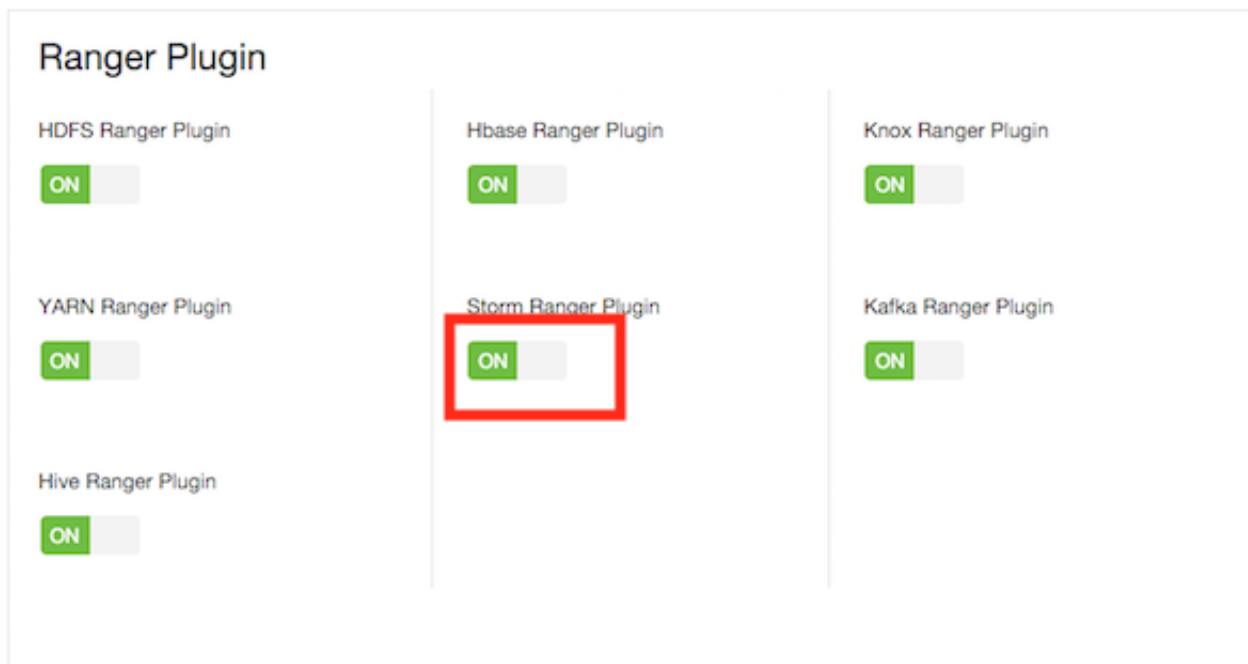
Before you can use the Storm plugin, you must first enable Kerberos on your cluster. To enable Kerberos on your cluster, see [Enabling Kerberos Authentication Using Ambari](#).

Use the following steps to enable the Ranger Storm plugin.

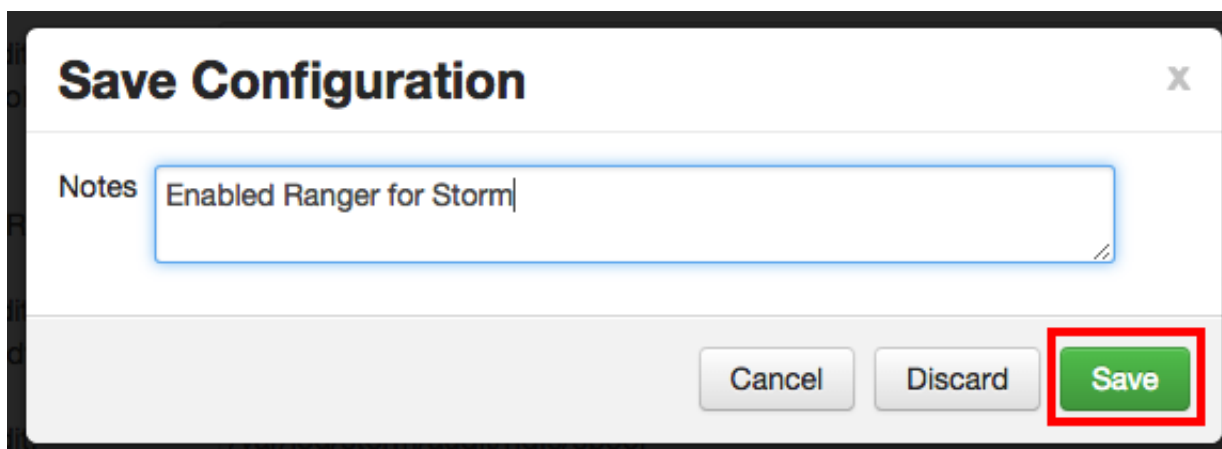
1. On the Ranger Configs page, select the **Ranger Plugin** tab.

The screenshot displays the Ambari Ranger Admin interface. The top navigation bar includes 'Ambari', 'Sandbox', 'Dashboard', 'Services', 'Hosts', 'Alerts', 'Admin', and a user profile 'admin'. The left sidebar lists various services, with 'Ranger' selected. The main content area shows the 'Ranger Admin' section, specifically the 'Ranger Plugin' configuration page. This page features a grid of toggle switches for various Ranger plugins: HDFS Ranger Plugin (ON), Hbase Ranger Plugin (ON), Knox Ranger Plugin (ON), YARN Ranger Plugin (ON), Storm Ranger Plugin (OFF), Kafka Ranger Plugin (ON), Hive Ranger Plugin (ON), and Atlas Ranger Plugin (OFF). A black menu bar at the top of the configuration area contains a 'Save' button. The interface also shows a 'Summary' tab and a 'Manage Config Groups' section with a filter and a list of configurations.

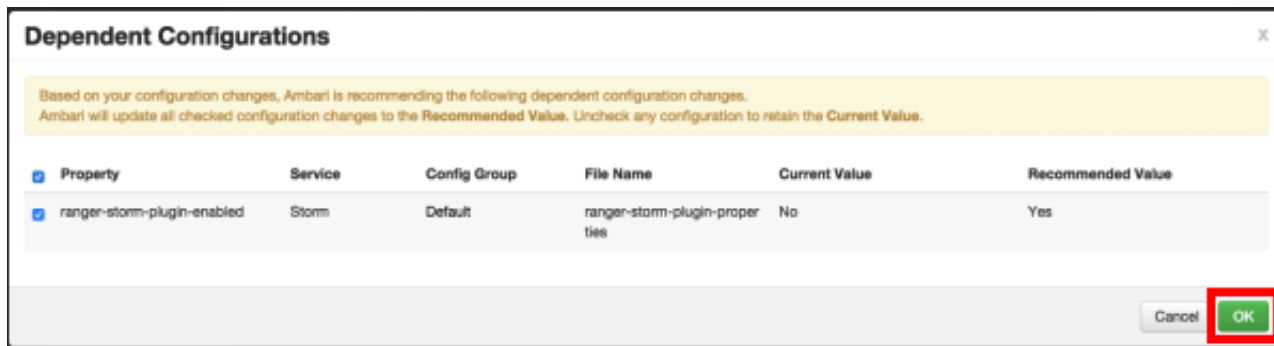
2. Under Storm Ranger Plugin, select **On**, then click **Save** in the black menu bar.



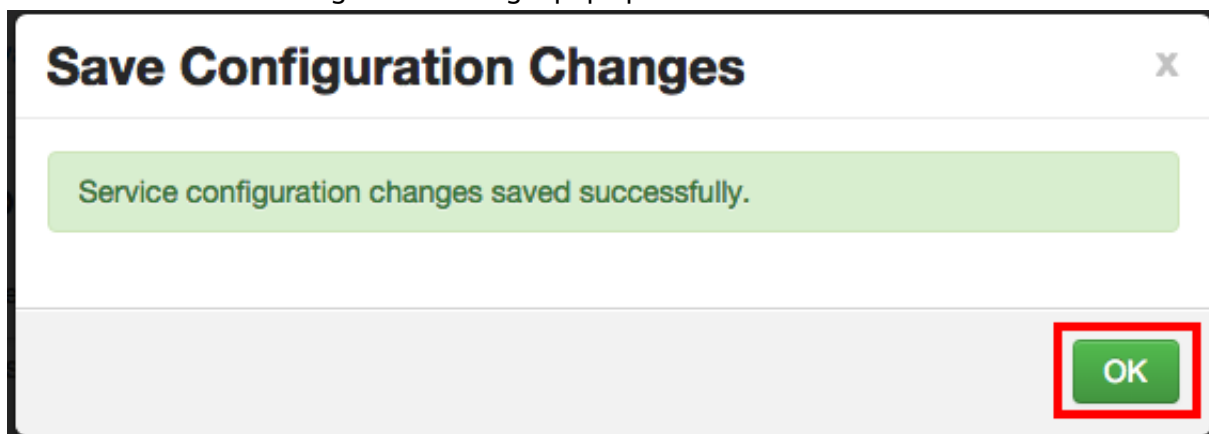
3. A Save Configuration pop-up appears. Type in a note describing the changes you just made, then click **Save**.



4. A Dependent Configuration pop-up appears. Click **OK** to confirm the configuration updates.



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



6. Select **Storm** in the navigation menu, then select **Restart > Restart All Affected** to restart the Storm service and load the new configuration.

Ambari test1 7 alerts Dashboard Services Hosts 1 Alerts Admin admin

Summary Configs Quick Links Service Actions

Restart Required: 4 Components on 1 Host Restart

Restart All Affected

Group Storm Default (1) Manage Config Groups Filter...

V5 admin a moment ago HDP-2.3 V4 admin about a minute ago HDP-2.3 V3 admin 2 minutes ago HDP-2.3 V2 admin 10 days ago HDP-2.3 V1 admin 11 days ago HDP-2.3

V5 admin authored on Fri, Sep 11, 2015 13:55 Discard Save

Nimbus

nimbus.reassign

nimbus.childopts `-Xmx1024m -_JAAS_PLACEHOLDER -javaagent:/usr/hdp/current/storm-nimbus/contrib/storm-jmxtric/lib/jmxtric-1.0.4.jar-host=localhost,port=8648,wireformat31x=true,mode=multicast,config=/usr/hdp/current/storm-nimbus/contrib/storm-jmxtric/conf/jmxtric-`

nimbus.cleanup.inbox.freq.secs 600 seconds

nimbus.file.copy.expiration.secs 600 seconds

nimbus.inbox.jar.expiration.secs 3600 seconds

nimbus.monitor.freq.secs 10 seconds

nimbus.supervisor.timeout.secs 60 seconds

nimbus.task.launch.secs 120 seconds

nimbus.task.timeout.secs 30 seconds

nimbus.thrift 1048576 bytes

7. Click **Confirm Restart All** on the confirmation pop-up to confirm the Storm restart.

Confirmation

You are about to restart Storm

This will trigger alerts as the service is restarted. To suppress alerts, turn on Maintenance Mode for Storm prior to running restart all

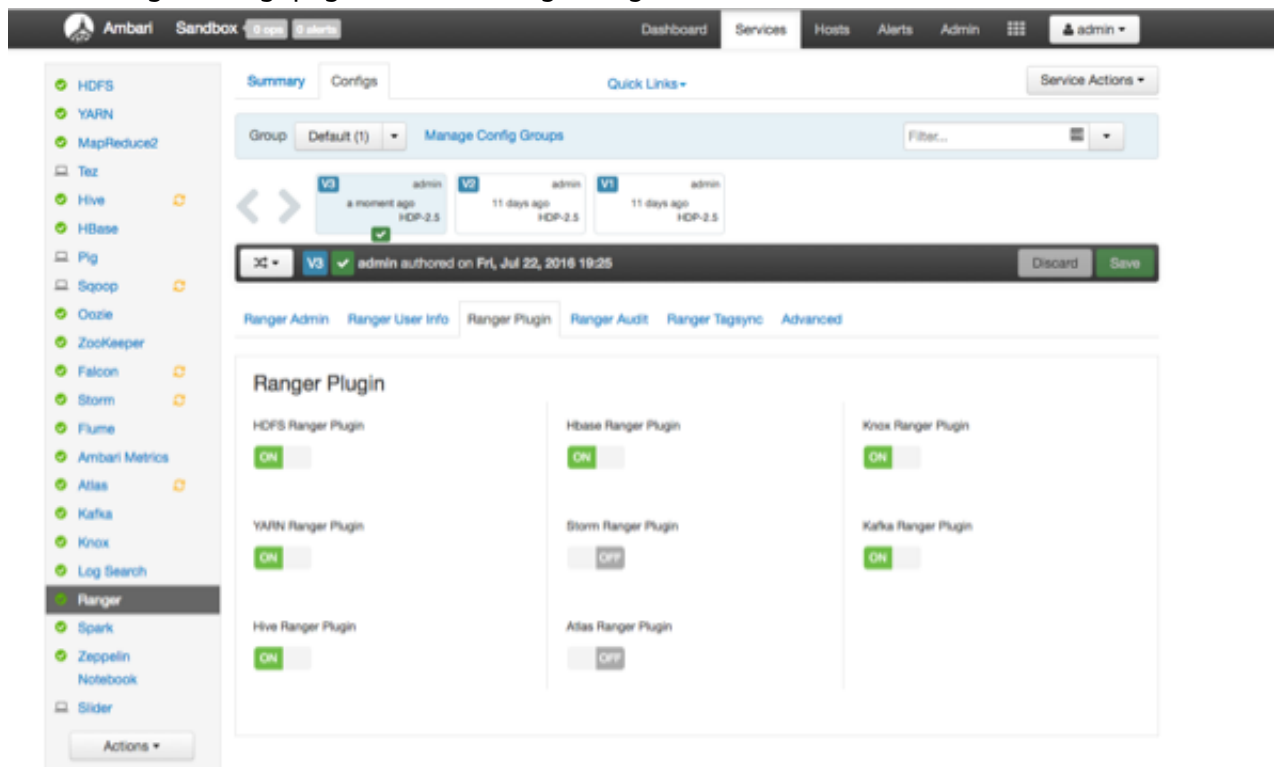
Cancel Confirm Restart All

8. After Storm restarts, the Ranger plugin for Storm will be enabled.

3.1.4.8. Atlas

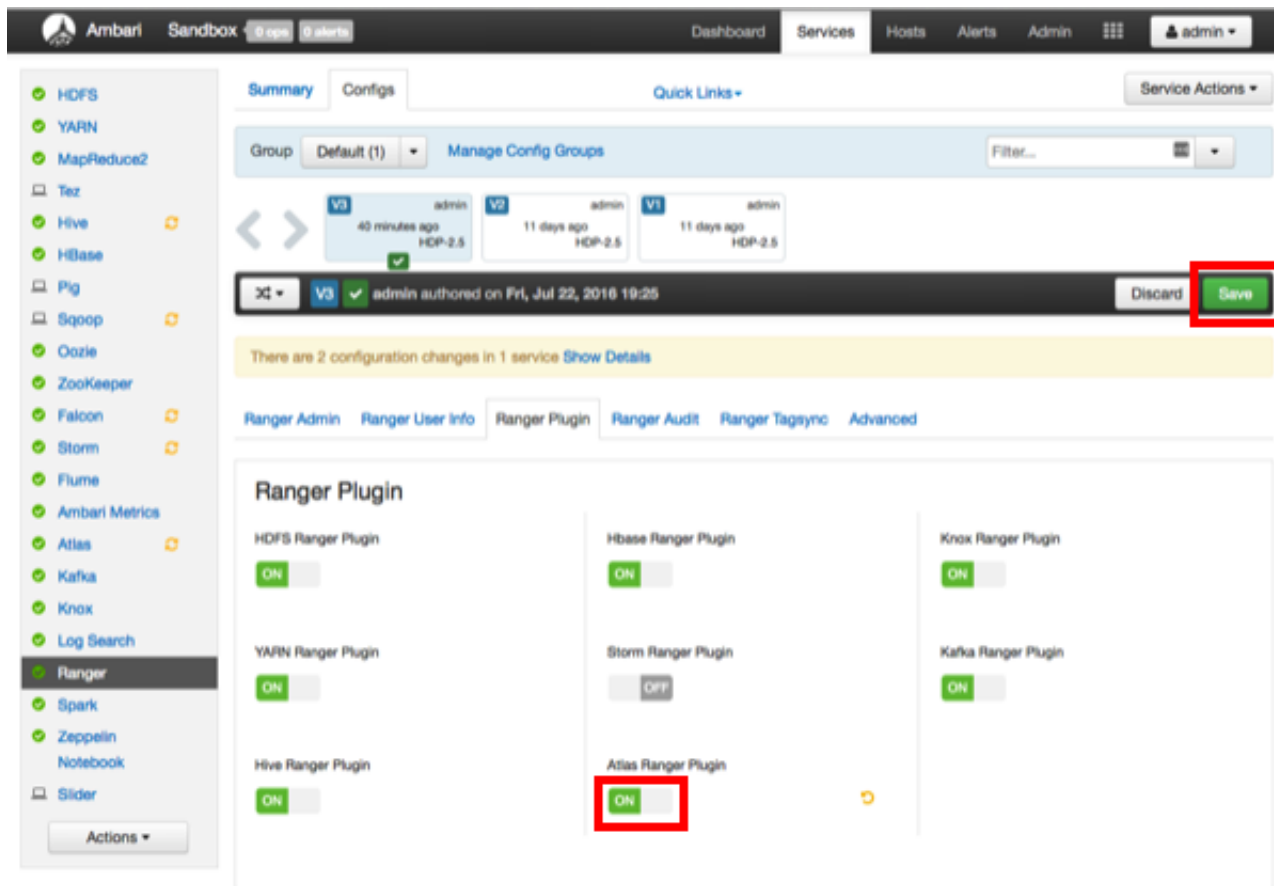
Use the following steps to enable the Ranger Atlas plugin.

1. On the Ranger Configs page, select the **Ranger Plugin** tab.

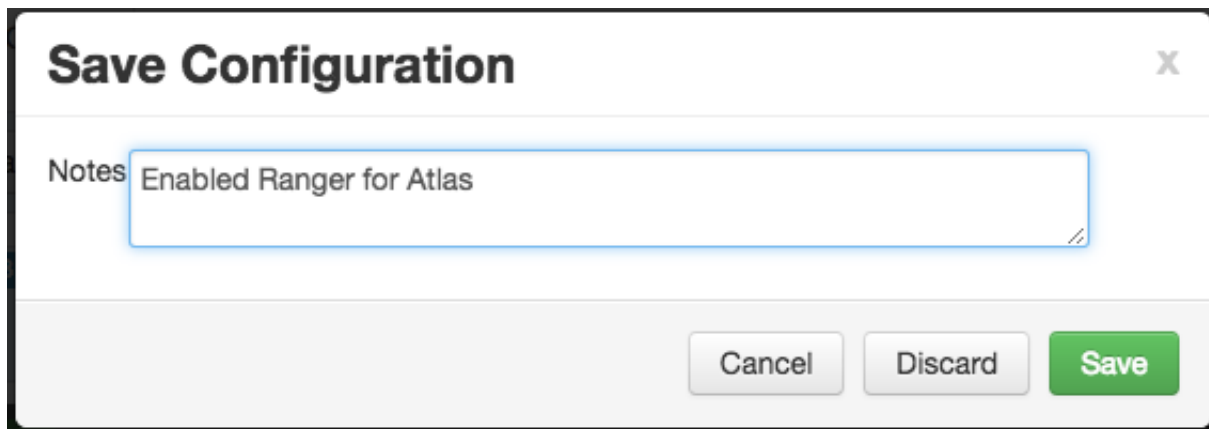


The screenshot displays the Ambari interface for configuring Ranger. The top navigation bar includes 'Ambari', 'Sandbox', 'Dashboard', 'Services', 'Hosts', 'Alerts', 'Admin', and a user profile 'admin'. The left sidebar lists various services, with 'Ranger' highlighted. The main content area is titled 'Ranger Configs' and shows a 'Summary' tab. Below this, there are configuration groups for 'HDP-2.5'. A black menu bar at the bottom of the configuration area contains 'V3', a checkmark, and the text 'admin authored on Fri, Jul 22, 2016 19:25', with 'Discard' and 'Save' buttons. The 'Ranger Plugin' tab is selected, showing a grid of toggle switches for various plugins: HDFS Ranger Plugin (ON), Hbase Ranger Plugin (ON), Knox Ranger Plugin (ON), YARN Ranger Plugin (ON), Storm Ranger Plugin (OFF), Kafka Ranger Plugin (ON), Hive Ranger Plugin (ON), and Atlas Ranger Plugin (OFF).

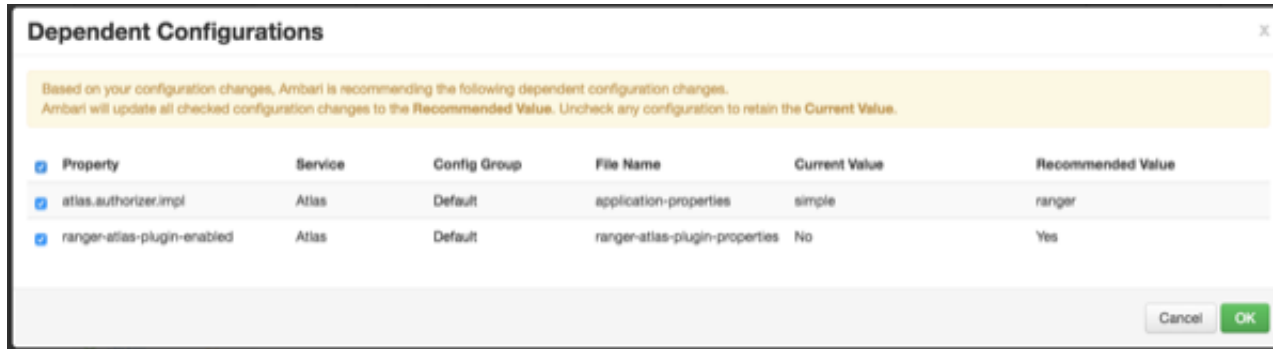
2. Under Atlas Ranger Plugin, select **On**, then click **Save** in the black menu bar.



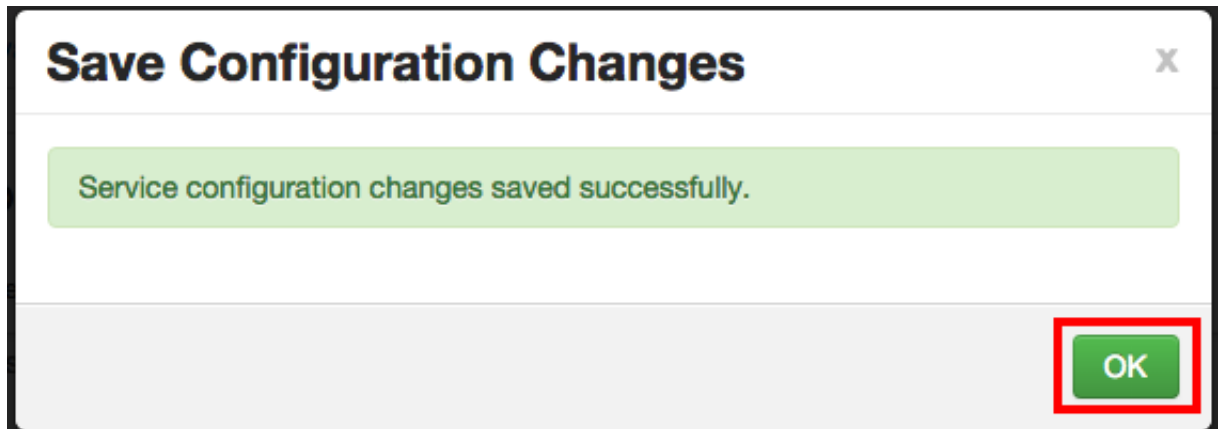
- 3. A Save Configuration pop-up appears. Type in a note describing the changes you just made, then click **Save**.



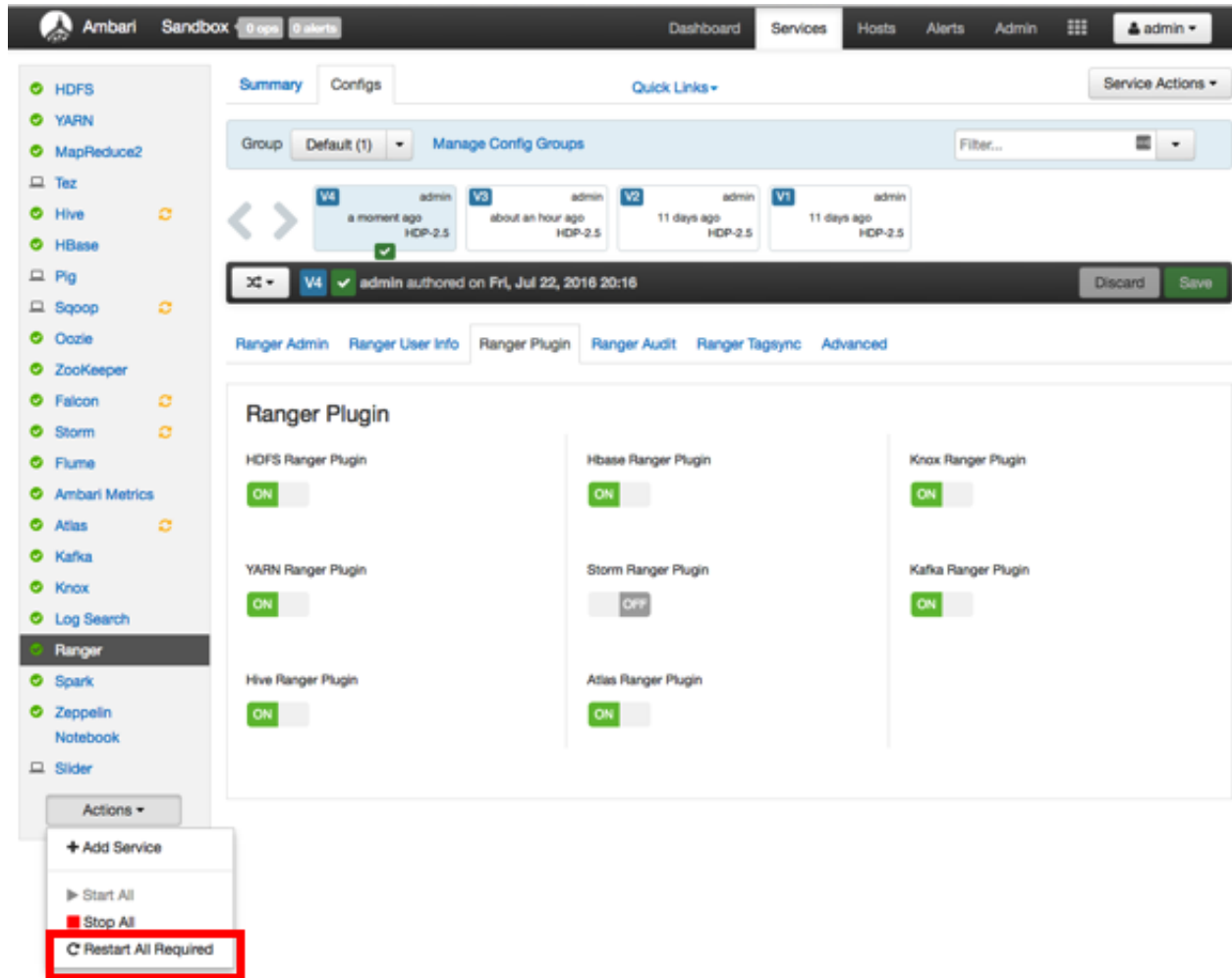
- 4. A Dependent Configurations pop-up appears. Click **OK** to confirm the configuration updates.



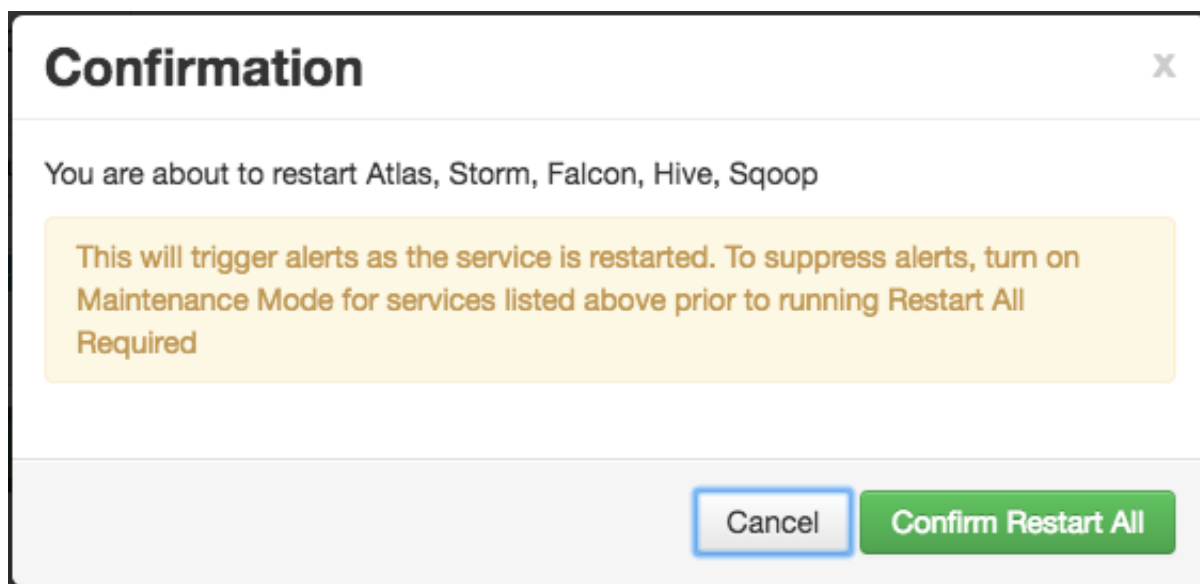
5. Click **OK** on the Save Configuration Changes pop-up.



6. Select **Actions** in the navigation menu, then select **Restart All Required** to restart all services that require a restart.



7. Click **Confirm Restart All** on the confirmation pop-up to confirm the Storm restart.



8. After the services restart, the Ranger plugin for Atlas will be enabled.

3.1.5. Ranger Plugins - Kerberos Overview

If you are using a Kerberos-enabled cluster, there are a number of steps you need to follow in order use the following Ranger plugins on a Kerberos cluster:

1. [HDFS \[269\]](#)
2. [Hive \[270\]](#)
3. [HBase \[271\]](#)
4. [Knox \[271\]](#)



Note

These procedures assume that you have already [enabled Ranger plugins](#).

3.1.5.1. HDFS

To enable the Ranger HDFS plugin on a Kerberos-enabled cluster, perform the steps described below.

1. Create the system (OS) user `rangerhdfslookup`. Make sure this user is synced to Ranger Admin (under *Settings>Users/Groups* tab in the Ranger Admin User Interface).
2. Create a Kerberos principal for `rangerhdfslookup` by entering the following command:

- `kadmin.local -q 'addprinc -pw rangerhdfslookup rangerhdfslookup@example.com'`



Note

A single user/principal (e.g., rangerrepouser) can also be created and used across services.

3. Navigate to the HDFS service.
4. Click the **Config** tab.
5. Navigate to *advanced ranger-hdfs-plugin-properties* and update the properties listed in the table shown below.

Advanced ranger-hdfs-plugin-properties

Enable Ranger for HDFS

Audit to HDFS

Audit to DB

policy User for HDFS

Ranger repository config password

Ranger repository config user

common.name.for.certificate

hadoop.rpc.protection

SSL_KEYSTORE_FILE_PATH

Table 3.19. HDFS Plugin Properties

Configuration Property Name	Value
Ranger repository config user	rangerhdfslookup@example.com
Ranger repository config password	rangerhdfslookup
common.name.for.certificate	blank

6. After updating these properties, click **Save** and restart the HDFS service.

3.1.5.2. Hive

To enable the Ranger Hive plugin on a Kerberos-enabled cluster, perform the steps described below.

1. Create the system (OS) user `rangerhivelookup`. Make sure this user is synced to Ranger Admin (under *Settings>Users/Groups* tab in the Ranger Admin UI).
2. Create a Kerberos principal for `rangerhivelookup` by entering the following command:
 - `kadmin.local -q 'addprinc -pw rangerhivelookup rangerhivelookup@example.com'`
3. Navigate to the Hive service.

- Click the **Config** tab and navigate to *advanced ranger-hive-plugin-properties*.
- Update the following properties with the values listed in the table below.

Table 3.20. Hive Plugin Properties

Configuration Property Name	Value
Ranger service config user	rangerhivelookup@example.com
Ranger service config password	rangerhivelookup
common.name.for.certificate	blank

- After updating these properties, click **Save** and then restart the Hive service.

3.1.5.3. HBase

To enable the Ranger HBase plugin on a Kerberos-enabled cluster, perform the steps described below.

- Create the system (OS) user `rangerhbaselookup`. Make sure this user is synced to Ranger Admin (under *users/groups* tab in the Ranger Admin UI).
- Create a Kerberos principal for `rangerhbaselookup` by entering the following command:
 - `kadmin.local -q 'addprinc -pw rangerhbaselookup rangerhbaselookup@example.com'`
- Navigate to the HBase service.
- Click the **Config** tab and go to *advanced ranger-hbase-plugin-properties*.
- Update the following properties with the values listed in the table below.

Table 3.21. HBase Plugin Properties

Configuration Property Name	Value
Ranger service config user	rangerhbaselookup@example.com
Ranger service config password	rangerhbaselookup
common.name.for.certificate	blank

- After updating these properties, click **Save** and then restart the HBase service.

3.1.5.4. Knox

To enable the Ranger Knox plugin on a Kerberos-enabled cluster, perform the steps described below.

- Create the system (OS) user `rangerknoxlookup`. Make sure this user is synced to Ranger Admin (under *Settings>Users/Groups* tab in the Ranger Admin UI).
- Create a Kerberos principal for `rangerknoxlookup` by entering the following command:
 - `kadmin.local -q 'addprinc -pw rangerknoxlookup rangerknoxlookup@example.com'`

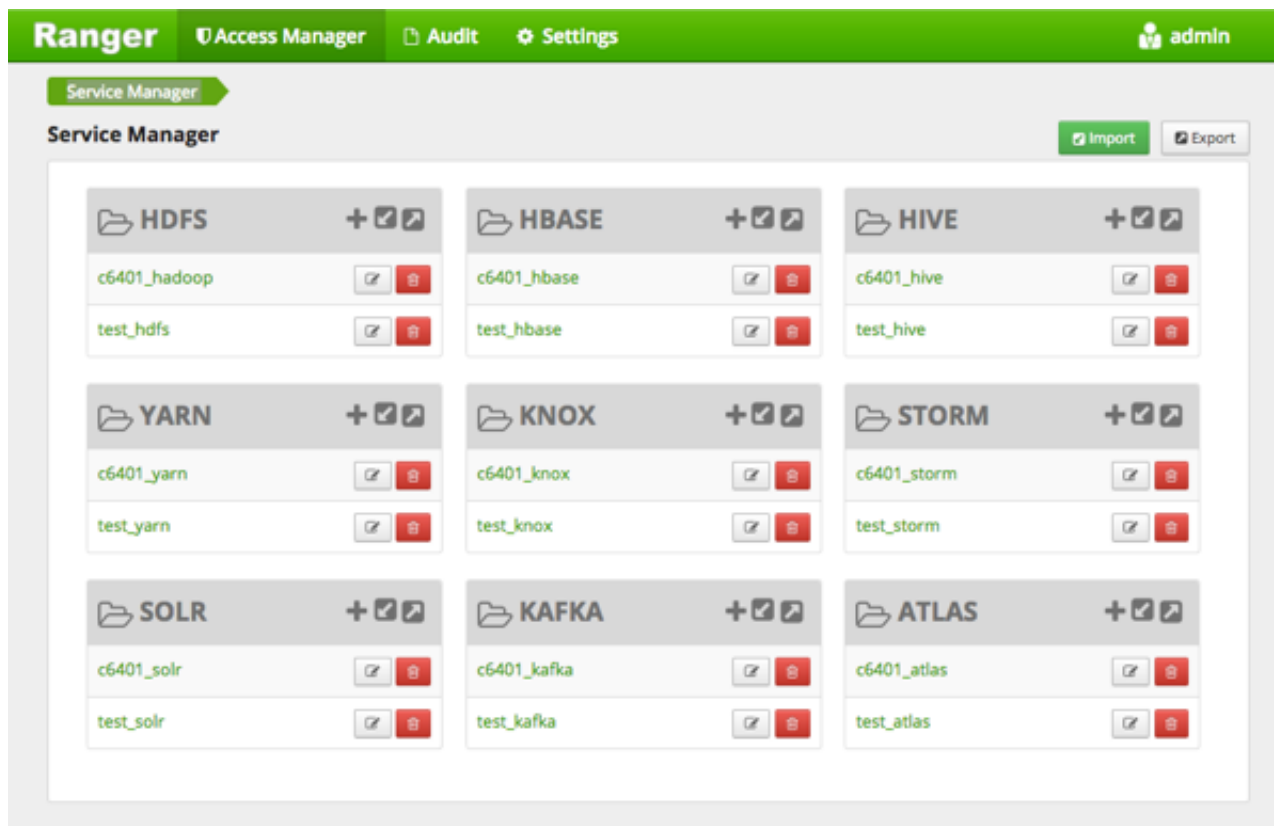
3. Navigate to the Knox service.
4. Click the **Config** tab and navigate to *advanced ranger-knox-plugin-properties*.
5. Update the following properties with the values listed in the table below.

Table 3.22. Knox Plugin Properties

Configuration Property Name	Value
Ranger service config user	rangerknoxlookup@example.com
Ranger service config password	rangerknoxlookup
common.name.for.certificate	blank

6. After updating these properties, click **Save** and then restart the Knox service.
7. Open the Ranger Admin UI by entering the following information:
 - `http://ranger-host>:6080`
 - **username/password** - `admin/admin`. or use `username` as shown in *advanced ranger-env* under the **Config** tab of the Ranger service, and `password` as shown in **Admin Settings**.
8. After you have successfully logged into the system, you will be redirected to the Access Manager page.

Figure 3.6. Service Manager



9. Click the repository (clusterName_hadoop) **Edit** option under the HDFS box.

The screenshot shows the Ranger console interface for editing a repository. The 'Config Properties' section is visible, with several input fields. The 'fs.default.name' field contains the value 'hdfs://mangeramban-muaradur-113march-hdfs-4-c1-hdsw.internal'. The 'hadoop.rpc.protection' and 'Common Name for Certificate' fields are currently empty and highlighted with red boxes. A 'Test Connection' button is located at the bottom of the form.

10. Update the following properties listed in the table below under the Config Properties section:

Table 3.23. Knox Configuration Properties

Configuration Property Name	Value
fs.default.name	hdfs
hadoop.rpc.protection	blank
common.name.for.certificate	blank

11. Click **Named Test Connection**. You should see a *Connected Successfully* dialog box appears.
12. Click **Save**.

3.2. Using Ranger to Provide Authorization in Hadoop

Once a user has been authenticated, their access rights must be determined. Authorization defines user access rights to resources. For example, a user may be allowed to create a policy and view reports, but not allowed to edit users and groups. You can use Ranger to set up and manage access to Hadoop services.

Ranger enables you to create services for specific Hadoop resources (HDFS, HBase, Hive, etc.) and add access policies to those services. You can also create tag-based services and add access policies to those services. Using tag-based policies enables you to control access to resources across multiple Hadoop components without creating separate services and policies in each component. You can also use Ranger TagSync to synchronize the Ranger tag store with an external metadata service such as Apache Atlas.

- [Using the Ranger Console \[279\]](#)
- [Configuring Resource-Based Services \[284\]](#)
- [Resource-Based Policy Management \[301\]](#)

- [Users/Groups and Permissions Administration \[369\]](#)
- [Reports Administration \[381\]](#)

For more information on Ranger authorization, see the [Authorization](#) overview.

3.2.1. About Ranger Policies

3.2.1.1. Ranger Resource-Based Policies

Ranger enables you to [Configuring Resource-Based Services \[284\]](#) (HDFS, HBase, Hive, etc.) and [add access policies](#) to those services.

3.2.1.2. Ranger Tag-Based Policies

Ranger also enables you to create tag-based services and add access policies to those services.

- An important feature of Ranger tag-based authorization is the separation of resource-classification from access-authorization. For example, resources (HDFS file/directory, Hive database/table/column etc.) containing sensitive data such as social security numbers, credit card numbers, or sensitive health care data can be tagged with PII/PCI/PHI – either as the resource enters the Hadoop ecosystem or at a later time. Once a resource is tagged, the authorization for the tag would be automatically enforced, thus eliminating the need to create or update policies for the resource.
- Using tag-based policies also enables you to control access to resources across multiple Hadoop components without creating separate services and policies in each component.
- Tag details are stored in a tag store. Ranger TagSync can be used to synchronize the tag store with an external metadata service such as Apache Atlas.

3.2.1.2.1. Tag Store

Details of tags associated with resources are stored in a tag store. Apache Ranger plugins retrieve the tag details from the tag store for use during policy evaluation. To minimize the performance impact during policy evaluation (in finding tags for resources), Apache Ranger plugins cache the tags and periodically poll the tag store for any changes. When a change is detected, the plugins update the cache. In addition, the plugins store the tag details in a local cache file – just as the policies are stored in a local cache file. On component restart, the plugins will use the tag data from the local cache file if the tag store is not reachable.

Apache Ranger plugins download the tag details from the store managed by Ranger Admin. Ranger Admin persists the tag details in its policy store and provides a REST interface for the plugins to download the tag details.

3.2.1.2.2. TagSync

Ranger TagSync is used to synchronize the tag store with an external metadata service such as Apache Atlas. TagSync is a daemon process similar to the Ranger UserSync process.

Ranger TagSync receives tag details from Apache Atlas via change notifications. As tags are added to, updated, or deleted from resources in Apache Atlas, Ranger TagSync receives notifications and updates the tag store.

3.2.1.2.3. Tags

Ranger Tags can have attributes. Tag attribute values can be used in Ranger tag-based policies to influence the authorization decision.

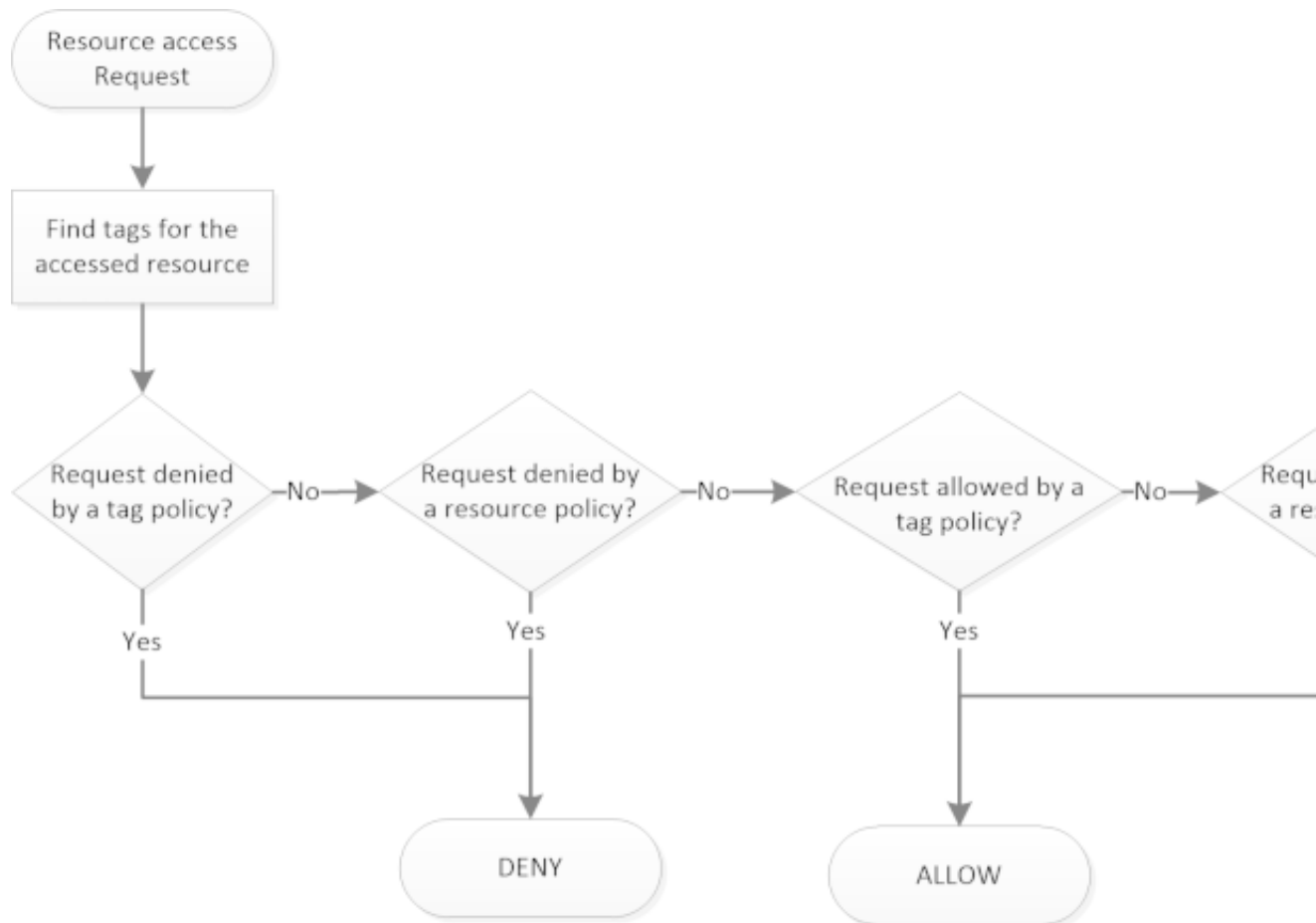
For example, to deny access to a resource after a specific date:

1. Add the EXPIRES_ON tag to the resource.
2. Add an `expiry_date` tag attribute and set its value to the expiry date.
3. Create a Ranger policy for the EXPIRES_ON tag.
4. Add a condition in this policy to deny access when the date specified in the `expiry_date` tag attribute is later than the current date.

Note that the EXPIRES_ON tag policy is created as the default policy in tag service instances.

3.2.1.3. Tags and Policy Evaluation

When authorizing an access request, an Apache Ranger plugin evaluates applicable Ranger policies for the resource being accessed. The following diagram shows the details of the policy evaluation flow. More details on the steps in this workflow are provided in the subsequent sections.



Apache Ranger Policy Evaluation Flow with T

3.2.1.3.1. Finding Tags

Apache Ranger supports a service to register [context enrichers](#), which are used to update context data to the access request.

The Ranger Tag service, which is part of the tag-based policies feature, adds a context enricher named RangerTagEnricher. This context enricher is responsible for finding tags for the requested resource and adding the tag details to the request context. This context enricher keeps a cache of the available tags; while processing an access request, it finds the tags applicable for the requested resource and adds the tags to the request context. The context enricher keeps the cache updated by periodically polling Ranger Admin for changes.

3.2.1.3.2. Evaluating Tag-Based Policies

Once the list of tags for the requested resource is found, the Apache Ranger policy engine evaluates the tag-based policies applicable to the tags. If a policy for one of these tag

results in a deny, access will be denied. If none of the tags are denied, and if a policy allows for one of the tags, access will be allowed. If there is no result for any tag, or if there are no tags for the resource, the policy engine will evaluate the resource-based policies to make the authorization decision.

3.2.1.3.3. Using Tags in Conditions

Apache Ranger allows the use of custom conditions while evaluating authorization policies. The Apache Ranger policy engine makes various request details – such as user, groups, resource, and context – available to the conditions. Tags in the request context, which are added by the enricher, are available to the conditions and can be used to influence the authorization decision.

The default policy in tag service instances, the EXPIRES_ON tag, uses such condition to check to see if the request date is later than the value specified in tag attribute expiry_date. This default policy does not work unless an EXPIRES_ON tag has been created in Atlas.

3.2.1.4. Apache Ranger Access Conditions

The Apache Ranger access policy model consists of two major components:

- Specification of the resources a policy is applied to, such as HDFS files and directories, Hive databases, tables, and columns, HBase tables, column-families, and columns, and so on.
- Specification of access conditions for specific users and groups.

3.2.1.4.1. Allow, Deny, and Exclude Conditions

Apache Ranger supports the following access conditions:

- Allow
- Exclude from Allow
- Deny
- Exclude from Deny

These access conditions enable you to set up fine-grained access control policies.

For example, you can allow access to a "finance" database to all users in the "finance" group, but deny access to all users in the "interns" group. Let's say that one of the members of the "interns" group, "scott", needs to work on an assignment that requires access to the "finance" database. In that case, you can add an Exclude from Deny condition that will allow user "scott" to access the "finance" database. The following image shows how this policy would be set up in Apache Ranger:

Policy Details :

Policy ID **13**

Policy Name * enabled

Hive Database * include

table * include

Hive Column * include

Description

Audit Logging YES

Allow Conditions :

Select Group	Select User	Permissions	Delegate Admin	
<input type="text" value="finance"/>	<input type="text" value="Select User"/>	<input checked="" type="checkbox"/> All	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>

Exclude from Allow Conditions :

Deny Conditions :

Select Group	Select User	Permissions	Delegate Admin	
<input type="text" value="interns"/>	<input type="text" value="Select User"/>	<input checked="" type="checkbox"/> All	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>

Exclude from Deny Conditions :

Select Group	Select User	Permissions	Delegate Admin	
<input type="text" value="Select Group"/>	<input type="text" value="scott"/>	<input checked="" type="checkbox"/> select	<input type="checkbox"/>	<input checked="" type="checkbox"/>

If **Deny Conditions** does not appear on your **Policy Details** page, you must first [Enable Deny Conditions for Policies](#).

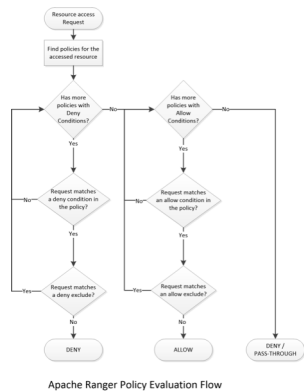
3.2.1.4.1.1. Enable Deny Conditions for Policies

The deny condition in policies is disabled by default and must be enabled for use.

1. From Ambari>Ranger>Configs>Advanced>Custom ranger-admin-site, add `ranger.servicedef.enableDenyAndExceptionsInPolicies=true`.
2. Restart Ranger.

3.2.1.4.2. Policy Evaluation of Access Conditions

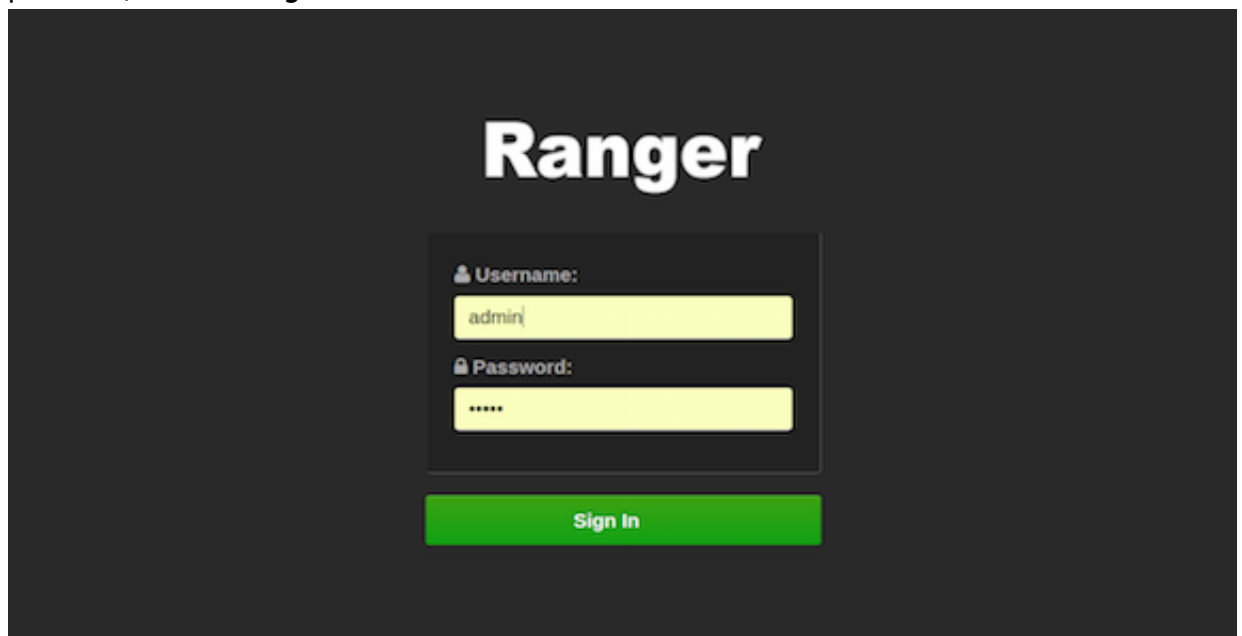
Apache Ranger policies are evaluated in a specific order to ensure predictable results (if there is no access policy that allows access, the authorization request will typically be denied). The following diagram shows the policy evaluation work-flow:



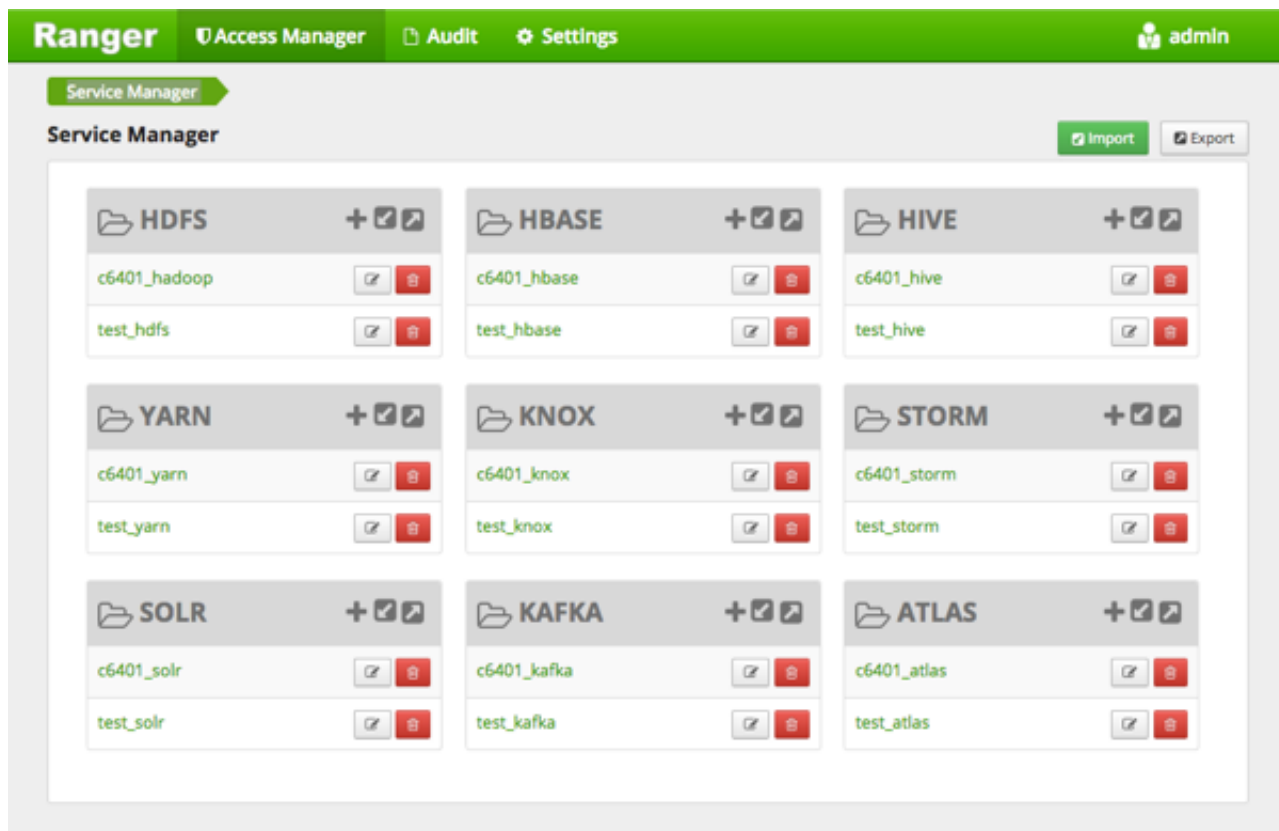
3.2.2. Using the Ranger Console

3.2.2.1. Opening and Closing the Ranger Console

To open the Ranger Console, log in to the Ranger portal at `http://<your_ranger_server_address>:6080`. To log in, enter your user name and password, then click **Sign In**.

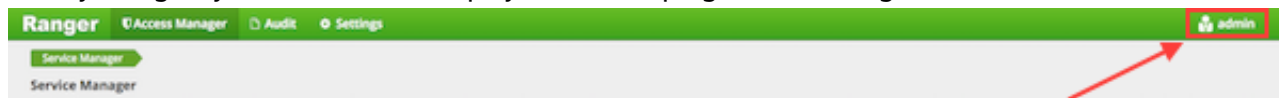


Ranger Console Home Page



Ranger Login Console

After you log in, your user name is displayed at the top right of the Ranger Console.

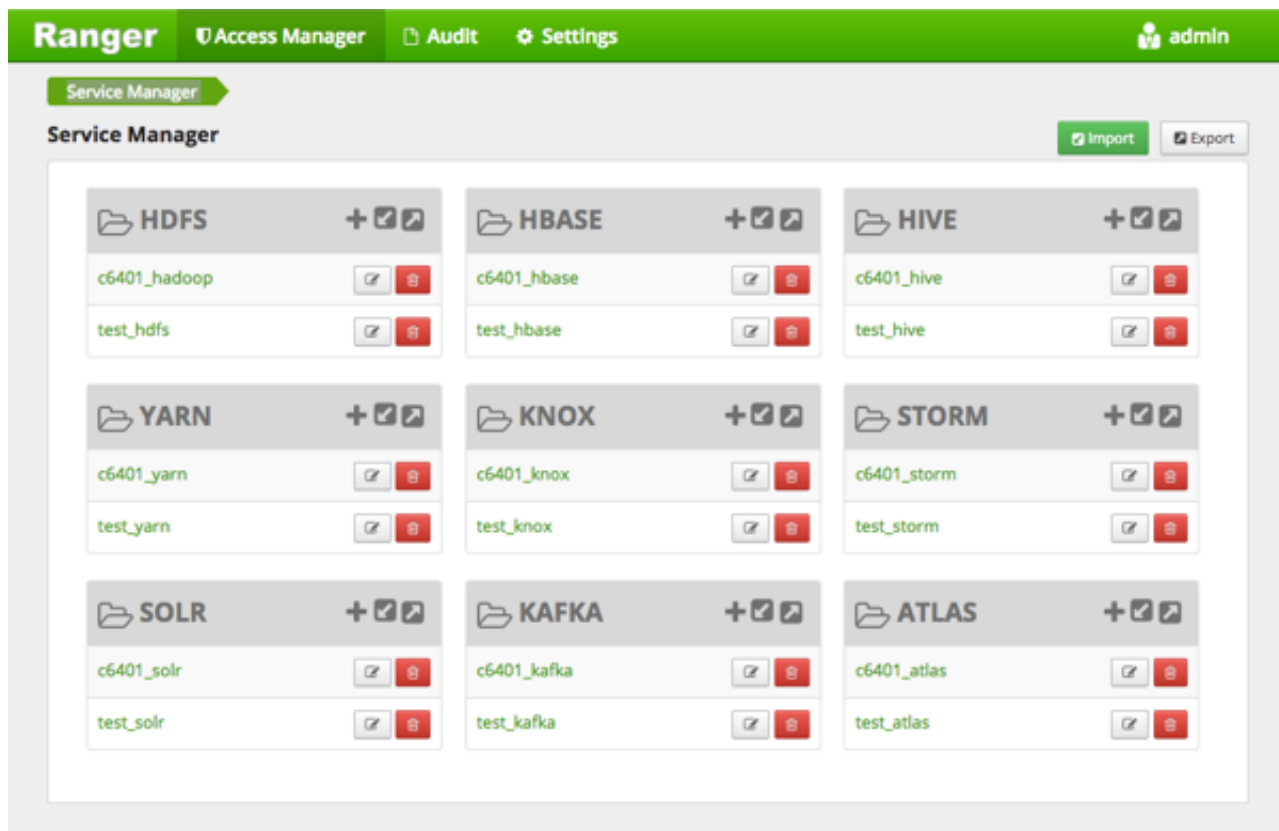


To log out of the Ranger Console, click your user name, then select **Log Out**.

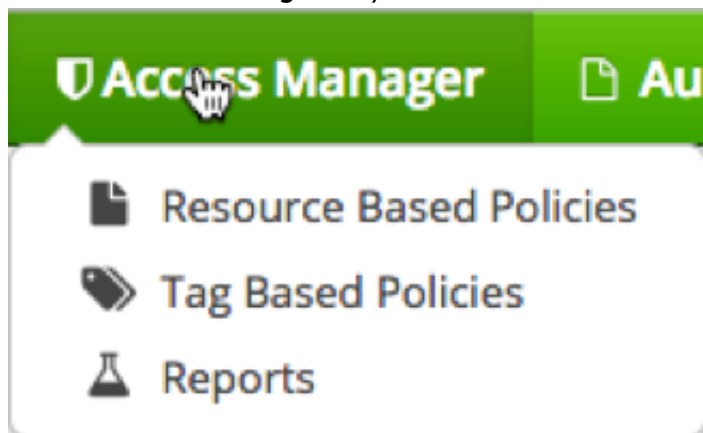


3.2.2.2. Ranger Console Navigation

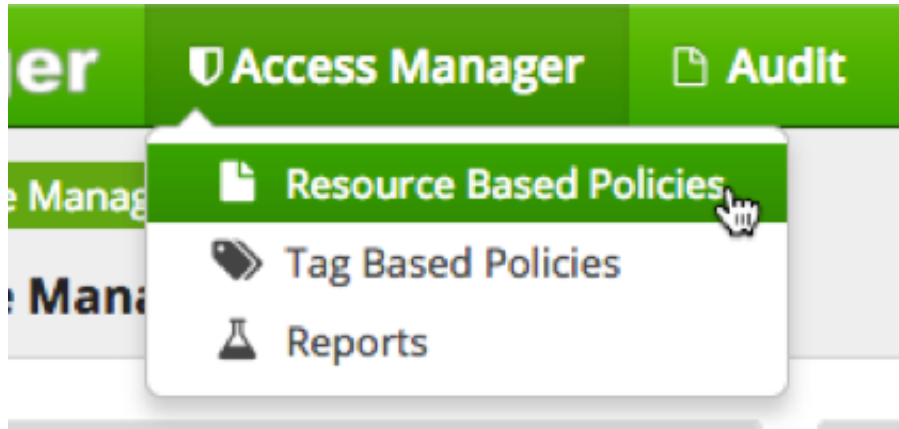
- The Service Manager for Resource Based Policies page is displayed when you log in to the Ranger Console. You can use this page to create services for Hadoop resources (HDFS, HBase, Hive, etc.) and add access policies to those resources.



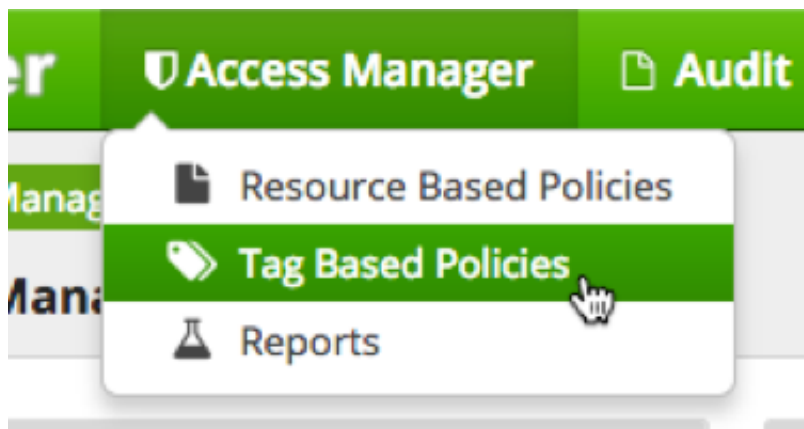
Clicking **Access Manager** in the top menu opens the Service Manager for Resource Based Policies page, and also displays a submenu with links to Resource Based Policies, Tag Based Policies, and Reports (this submenu is also displayed when you pass the mouse over the **Access Manager** link).



- **Access Manager > Resource Based Policies** – Opens the Service Manager for Resource Based Policies page. You can use this page to create services for resources (HDFS, HBase, Hive, etc.) and add access policies to those services.

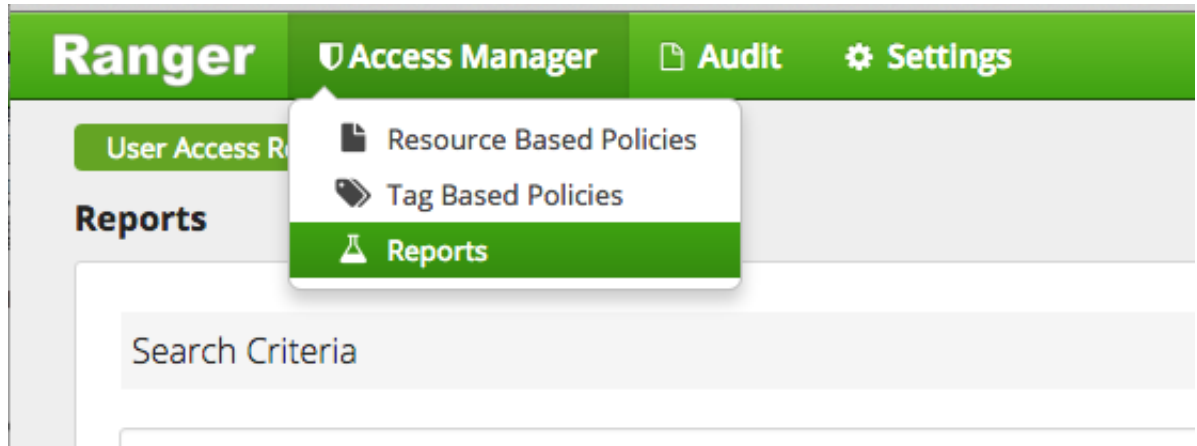


- **Access Manager > Tag Based Policies** – Opens the Service Manager for Tag Based Policies page. You can use this page to create tag-based services and add access policies to those services. Using tag-based policies enables you to control access to resources across multiple Hadoop components without creating separate services and policies in each component.

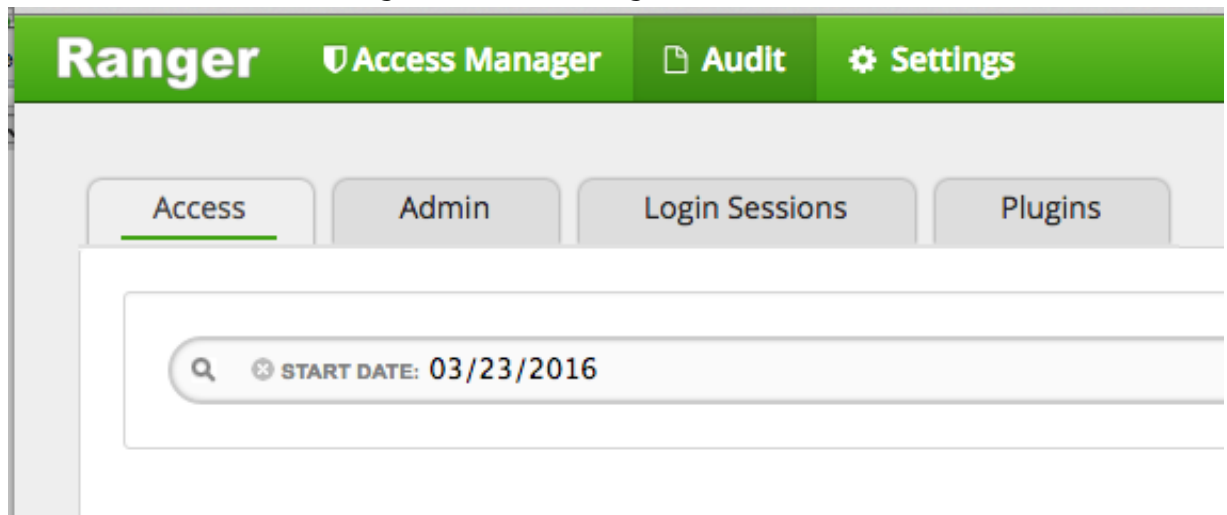


Shows Ranger > Access Manager menu expanded with Tag Based Policies highlighted.

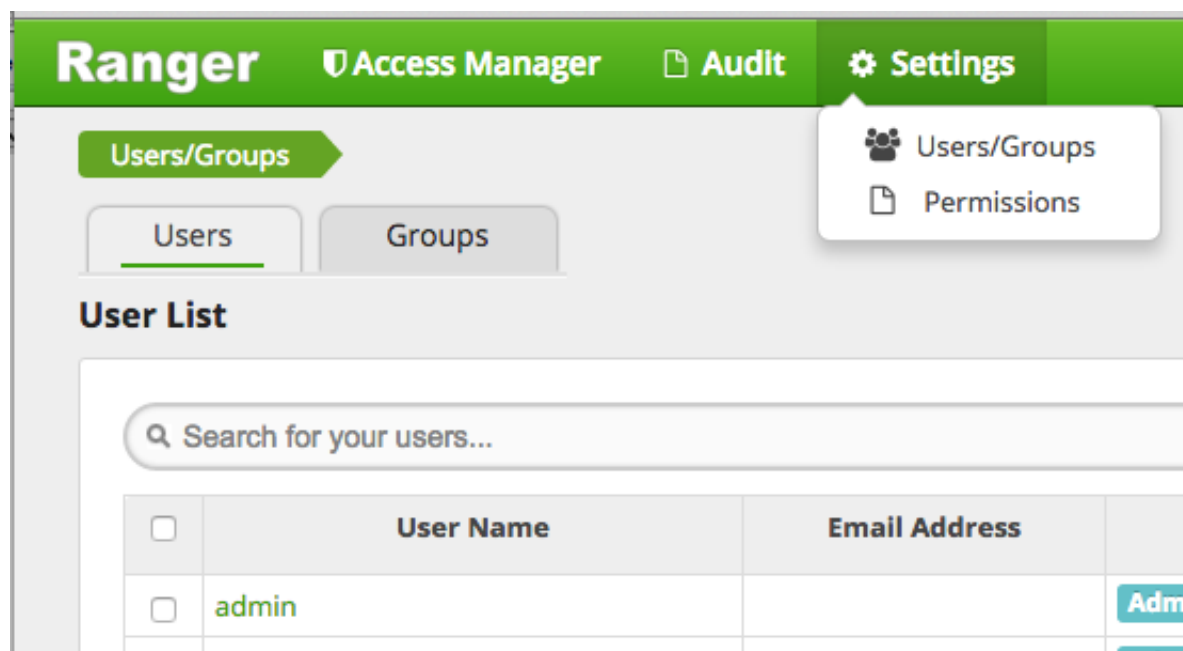
- **Access Manager > Reports** – Opens the Reports page. You can use this page to generate user access reports for resource and tag-based policies based on policy name, resource, group, and user name.



- **Audit** – You can use the Audit page to monitor user activity at the resource level, and also to set up conditional auditing based on users, groups, or time. The Audit page includes the Access, Admin, Login Sessions, and Plugins tabs.



- **Settings** – Enables you to manage and assign policy permissions to users and groups. Clicking or passing the mouse over **Settings** displays a submenu with links to the Users/Groups and Permissions pages.






3.2.3. Configuring Resource-Based Services

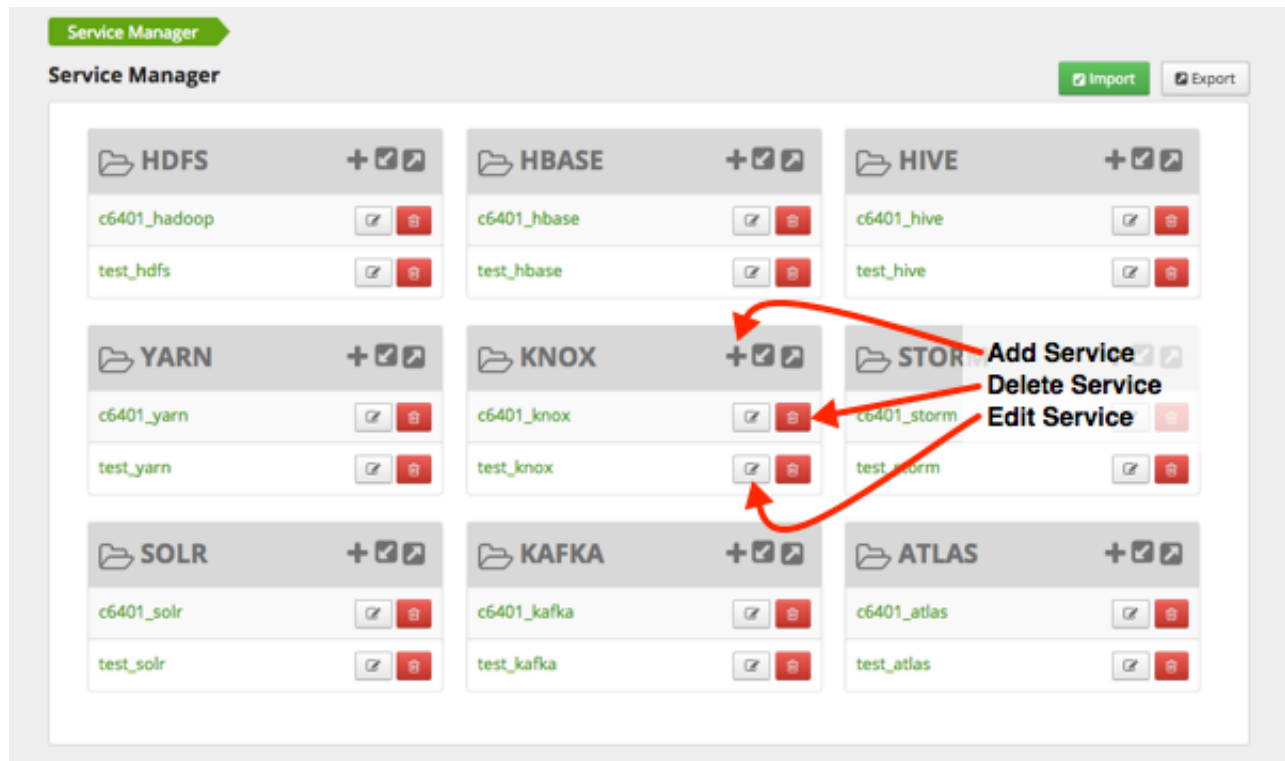
The Service Manager for Resource Based Policies page is displayed when you log in to the Ranger Console. You can also access this page by selecting **Access Manager > Resource Based Policies**. You can use this page to create services for Hadoop resources (HDFS, HBase, Hive, etc.) and add access policies to those resources.



Note

The Ambari Ranger installation procedure automatically configures these services, so there should be no need to add a service manually.

- **To add a new resource-based service**, click the Add icon () in the applicable box on the Service Manager page. Enter the required configuration settings, then click **Add**.
- **To edit a resource-based service**, click the Edit icon () at the right of the service. Edit the service settings, then click **Save** to save your changes.
- **To delete a resource-based service**, click the Delete icon () at the right of the service. Deleting a service also deletes all of the policies for that service.



This section describes how to configure resource-based services for the following Hadoop components:

- [Configure an HBase Service \[285\]](#)
- [Configure an HDFS Service \[287\]](#)
- [Configure a Hive Service \[289\]](#)
- [Configure a Kafka Service \[292\]](#)
- [Configure a Knox Service \[294\]](#)
- [Configure a Solr Service \[295\]](#)
- [Configure a Storm Service \[297\]](#)
- [Configure a YARN Service \[298\]](#)

3.2.3.1. Configure an HBase Service

Use the following steps to add a service to HBase:

1. On the Service Manager page, click the Add icon (⊕) next to HBase.

The Create Service page appears.

2. Enter the following information on the Create Service page:

Table 3.24. Service Details

Field name	Description
Service Name	The name of the service; required when configuring agents.
Description	A description of the service.
Active Status	Enabled or Disabled.
Select Tag Service	Select a tag-based service to apply the service and its tag-based policies to HBase.

Table 3.25. Config Properties

Field name	Description
Username	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 authentication is enabled.)
hbase.security.authentication	As noted in the hadoop configuration file hbase-site.xml.
hbase.zookeeper.property.clientPort	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.
Common Name for Certificate	The name of the certificate. This field is interchangeably named Common Name For Certificate and Ranger Plugin SSL CName in Create Service pages.
Add New Configurations	Add any other new configuration(s).

3. Click **Test Connection**.

4. Click **Add**.

3.2.3.2. Configure an HDFS Service

Use the following steps to add a service to HDFS:

1. On the Service Manager page, click the Add icon () next to HDFS.

The Create Service page appears.

2. Enter the following information on the Create Service page:

Table 3.26. Service Details

Field name	Description
Service Name	The name of the service; required when configuring agents.
Description	A description of the service.
Active Status	Enabled or Disabled.
Select Tag Service	Select a tag-based service to apply the service and its tag-based policies to HDFS.

Table 3.27. Config Properties

Field name	Description
Username	The end system username that can be used for connection.
Password	The password for the username entered above.
NameNode URL	hdfs:// <i>NAMENODE_FQDN</i> :8020 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. This field was formerly named fs.defaultFS.
Authorization Enabled	Authorization involves restricting access to resources. If enabled, user need authorization credentials.
Authentication Type	The type of authorization in use, as noted in the hadoop configuration file core-site.xml; either <code>simple</code> or <code>Kerberos</code> . (Required only if authorization is enabled). This field was formerly named hadoop.security.authorization.
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 service resides, as noted in the hadoop configuration file hdfs-site.xml. (Required only if Kerberos authentication is enabled).
dfs.namenode.kerberos.principal	The principal associated with the NameNode where the service 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 service resides, as noted in the hadoop configuration file hdfs-site.xml. (Required only if Kerberos authentication is enabled).
RPC Protection Type	Only authorised user can view, use, and contribute to a dataset. A list of protection values for secured SASL connections. Values: Authentication, Integrity, Privacy
Common Name For Certificate	The name of the certificate. This field is interchangeably named Common Name For Certificate and Ranger Plugin SSL CName in Create Service pages.
Add New Configurations	Add any other new configuration(s).

3. Click **Test Connection**.

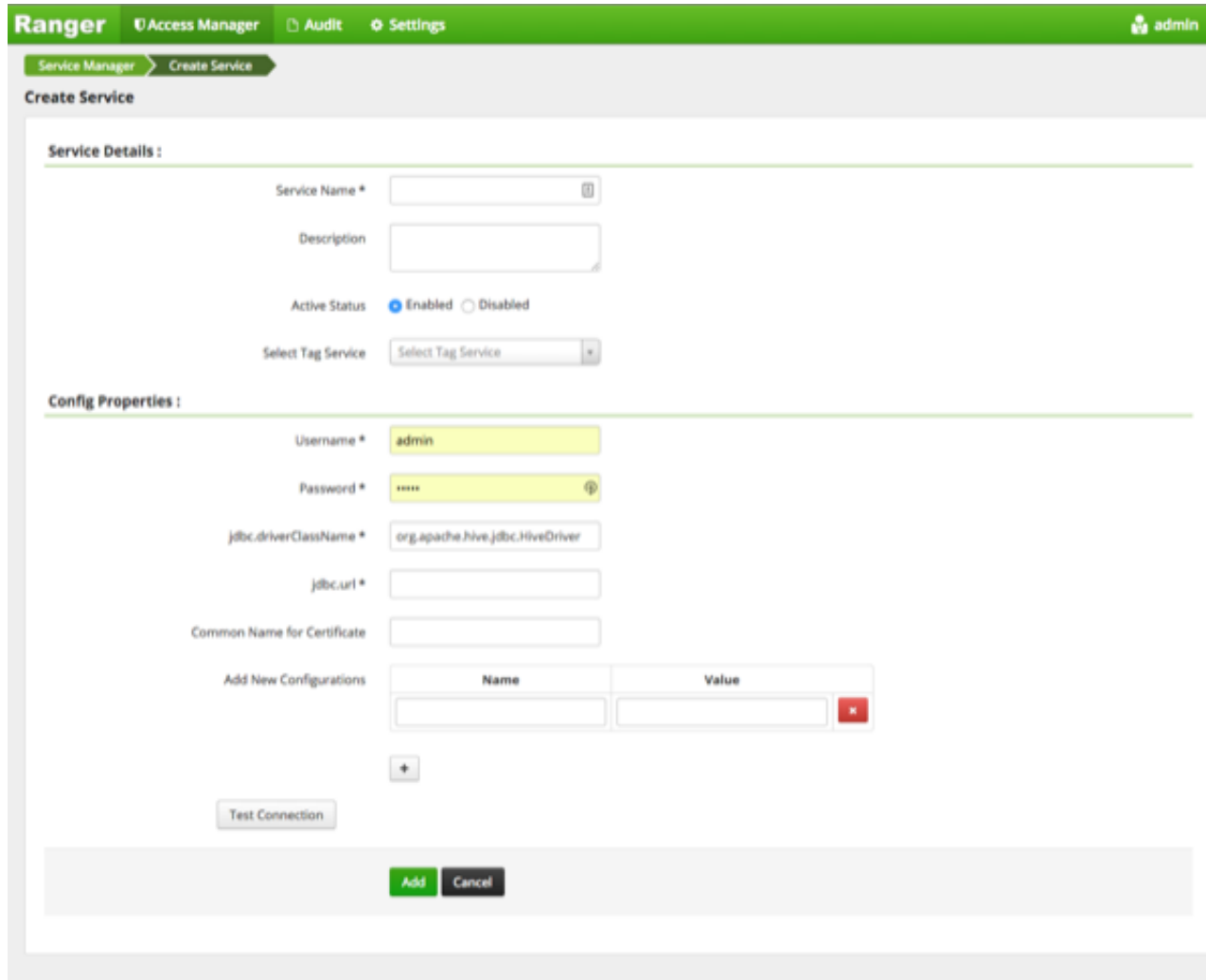
4. Click **Add**.

3.2.3.3. Configure a Hive Service

Use the following steps to add a service to Hive:

1. On the Service Manager page, click the Add icon () next to Hive.

The Create Service page appears.



2. Enter the following information on the Create Service page:

Table 3.28. Service Details

Field name	Description
Service Name	The name of the service; required when configuring agents.
Description	A description of the service.
Active Status	Enabled or Disabled.
Select Tag Service	Select a tag-based service to apply the service and its tag-based policies to Hive.

Table 3.29. Config Properties

Field name	Description
Username	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

Field name	Description
<code>jdbc.url</code>	The complete connection URL, including port and database name. (Default port: 10000.) For example, on the sandbox, <code>jdbc:hive2://sandbox:10000/</code> .
Common Name For Certificate	The name of the certificate. This field is interchangeably named Common Name For Certificate and Ranger Plugin SSL CName in Create Service pages.
Add New Configurations	Add any other new configuration(s).

3. Click **Test Connection**.

4. Click **Add**.

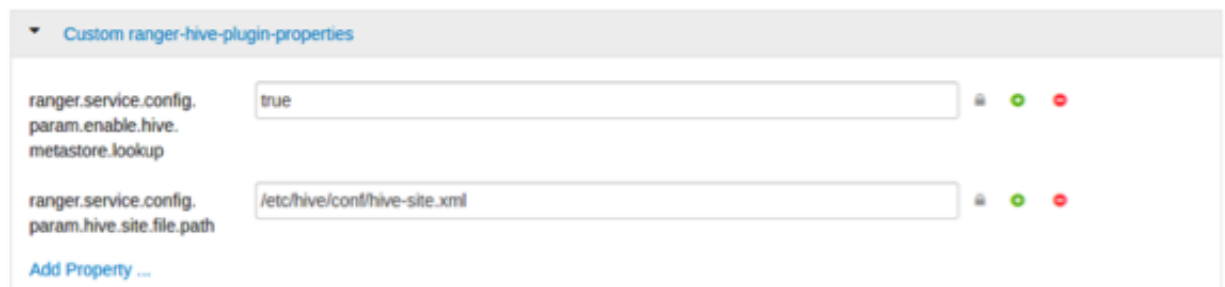
Usually, the Ranger Hive service definition uses the HiveServer2 (HS2) JDBC driver to fetch Hive database/table info for resource lookup and testing the connection. Alternatively, you can configure the service definition to use Hive metastore libraries connecting to the Hive metastore database directly. This is recommended when it is difficult to set up HiveServer2 on your cluster, such as when using HDCloud for AWS.

1. Under Ambari>Hive>Configs>Advanced, edit Hive properties:

2. Add the below properties to `custom ranger-hive-plugin-properties`:

```
ranger.service.config.param.enable.hive.metastore.lookup = true
```

```
ranger.service.config.param.hive.site.file.path = /etc/hive/conf/hive-site.xml
```



3. Save and restart required components.

To test the configuration is successful, create a new Hive service and specify the `jdbc.url` as "none", then run **Test Connection**.

Config Properties :

Username *

Password *

jdbc.driverClassName *

jdbc.url *

Common Name for Certificate

Add New Configurations

Name	Value	
enable.hive.metastore.lookup	<input type="text" value="true"/>	<input type="button" value="x"/>
hive.site.file.path	<input type="text" value="/etc/hive/conf/hive-site.xml"/>	<input type="button" value="x"/>
ambari.service.check.user	<input type="text" value="ambari-qa"/>	<input type="button" value="x"/>

3.2.3.4. Configure a Kafka Service

Use the following steps to add a service to Kafka:

1. On the Service Manager page, click the Add icon () next to Kafka.

The Create Service page appears.

2. Enter the following information on the Create Service page:

Table 3.30. Service Details

Field name	Description
Service Name	The name of the service; required when configuring agents.
Description	A description of the service.
Active Status	Enabled or Disabled.
Select Tag Service	Select a tag-based service to apply the service and its tag-based policies to Kafka.

Table 3.31. Config Properties

Field name	Description
Username	The end system username that can be used for connection.
Password	The password for the username entered above.
ZooKeeper Connect String	Defaults to localhost:2181 (Provide FQDN of zookeeper host : 2181).
Ranger Plugin SSL CName	Provide common.name.for.certificate which is registered with Ranger (in Wire Encryption environment).

Field name	Description
	This field is interchangeably named Common Name For Certificate and Ranger Plugin SSL CName in Create Service pages.
Add New Configurations	Add any other new configuration(s).

3. Click **Test Connection**.
4. Click **Add**.

3.2.3.5. Configure a Knox Service

Use the following steps to add a service to Knox:

1. On the Service Manager page, click the Add icon (+) next to Knox.

The Create Service page appears.

2. Enter the following information on the Create Service page:

Table 3.32. Service Details

Field name	Description
Service Name	The name of the service; required when configuring agents.
Description	A description of the service.
Active Status	Enabled or Disabled.
Select Tag Service	Select a tag-based service to apply the service and its tag-based policies to Knox.

Table 3.33. Config Properties


Field name	Description
Username	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. This field is interchangeably named Common Name For Certificate and Ranger Plugin SSL CName in Create Service pages.
Add New Configurations	Add any other new configuration(s).

3. Click **Test Connection**.

4. Click **Add**.

3.2.3.6. Configure a Solr Service

Use the following steps to add a service to Solr:

1. On the Service Manager page, click the Add icon () next to Solr.

The Create Service page appears.

2. Enter the following information on the Create Service page:

Table 3.34. Service Details

Field name	Description
Service Name	The name of the service; required when configuring agents.
Description	A description of the service.
Active Status	Enabled or Disabled.
Select Tag Service	Select a tag-based service to apply the service and its tag-based policies to Solr.

Table 3.35. Config Properties

Field name	Description
Username	The end system username that can be used for connection.
Password	The password for the username entered above.
Solr URL	For HDP Search's Solr Instance: <code>http://Solr_host:8983</code> For Ambari Infra's Solr Instance: <code>http://Solr_host:8886</code>

Field name	Description
Ranger Plugin SSL CName	Provide common.name.for.certificate which is registered with Ranger (in Wire Encryption environment). This field is interchangeably named Common Name For Certificate and Ranger Plugin SSL CName in Create Service pages.
Add New Configurations	Add any other new configuration(s).

3. Click **Test Connection**.

4. Click **Add**.

3.2.3.7. Configure a Storm Service

Use the following steps to add a service to Storm:

1. On the Service Manager page, click the Add icon () next to Storm.

The Create Service page appears.

2. Enter the following information on the Create Service page:

Table 3.36. Service Details

Field name	Description
Service Name	The name of the service; required when configuring agents.
Description	A description of the service.
Active Status	Enabled or Disabled.
Select Tag Service	Select a tag-based service to apply the service and its tag-based policies to Storm.

Table 3.37. Config Properties

Field name	Description
Username	The end system username that can be used for connection.
Password	The password for the username entered above.
Nimbus URL	Host name of nimbus format, in the form: <code>http://ipaddress:8080</code> . This field was formerly named nimbus.url.
Common Name For Certificate	The name of the certificate. This field is interchangeably named Common Name For Certificate and Ranger Plugin SSL CName in Create Service pages.
Add New Configurations	Add any other new configuration(s).

3. Click **Test Connection**.
4. Click **Add**.

3.2.3.8. Configure a YARN Service

Use the following steps to add a service to YARN:

1. On the Service Manager page, click the Add icon () next to YARN.

The Create Service page appears.

2. Enter the following information on the Create Service page:

Table 3.38. Service Details

Field name	Description
Service Name	The name of the service; required when configuring agents.
Description	A description of the service.
Active Status	Enabled or Disabled.
Select Tag Service	Select a tag-based service to apply the service and its tag-based policies to YARN.

Table 3.39. Config Properties

Field name	Description
Username	The end system username that can be used for connection.
Password	The password for the username entered above.
YARN REST URL	Http or https://RESOURCEMANAGER_FQDN:8088.

Field name	Description
Authentication Type	The type of authorization in use, as noted in the hadoop configuration file core-site.xml; either <code>simple</code> or <code>Kerberos</code> . (Required only if authorization is enabled). This field was formerly named <code>hadoop.security.authorization</code> .
Common Name For Certificate	The name of the certificate. This field is interchangeably named Common Name For Certificate and Ranger Plugin SSL CName in Create Service pages.
Add New Configurations	Add any other new configuration(s).

3. Click **Test Connection**.

4. Click **Add**.

3.2.3.9. Configure an Atlas Service

Use the following steps to add an Atlas service:

1. On the Service Manager page, click the Add icon () next to Storm.

The Create Service page appears.

2. Enter the following information on the Create Service page:

Table 3.40. Service Details

Field name	Description
Service Name	The name of the service; required when configuring agents.
Description	A description of the service.
Active Status	Enabled or Disabled.
Select Tag Service	Select a tag-based service to apply the service and its tag-based policies to Atlas.



Table 3.41. Config Properties

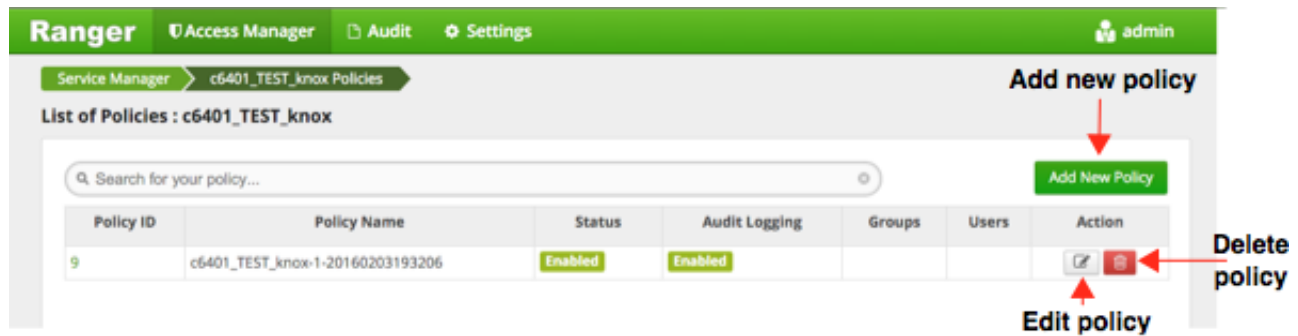
Field name	Description
Username	The end system username that can be used for connection.
Password	The password for the username entered above.
atlas.rest.address	Atlas host and port: : <code>http://atlas_host_FQDN:21000.</code>
Common Name For Certificate	The name of the certificate. This field is interchangeably named Common Name For Certificate and Ranger Plugin SSL CName in Create Service pages.
Add New Configurations	Add any other new configuration(s).

3. Click **Test Connection**.
4. Click **Add**.

3.2.4. Resource-Based Policy Management

To view the policies associated with a service, click the service name on the Resource Based Policies Service Manager page. The policies for that service will be displayed in a list, along with a search box.

- To add a new resource-based policy to the service, click **Add New Policy**.
- To edit a resource-based policy, click the Edit icon () at the right of the entry for that service. Edit the policy settings, then click **Save** to save your changes.
- To delete a resource-based policy, click the Delete icon () at the right of the entry for that service.



This section describes:

- Configuring resource-based policies for the following Hadoop components:
 - [Create an HBase Policy \[302\]](#)
 - [Create an HDFS Policy \[305\]](#)
 - [Create a Hive Policy \[307\]](#)
 - [Create a Kafka Policy \[311\]](#)
 - [Create a Knox Policy \[313\]](#)
 - [Create a Solr Policy \[315\]](#)
 - [Create a Storm Policy \[317\]](#)
 - [Create a YARN Policy \[319\]](#)
 - [Create an Atlas Policy \[322\]](#)
- Importing and Exporting Resource-Based Policies [326]:
 - [Import Resource-Based Policies \[328\]](#)
 - [Export Resource-Based Policies \[332\]](#)

3.2.4.1. Configuring Resource-Based Policies

3.2.4.1.1. Create an HBase Policy

To add a new policy to an existing HBase service:

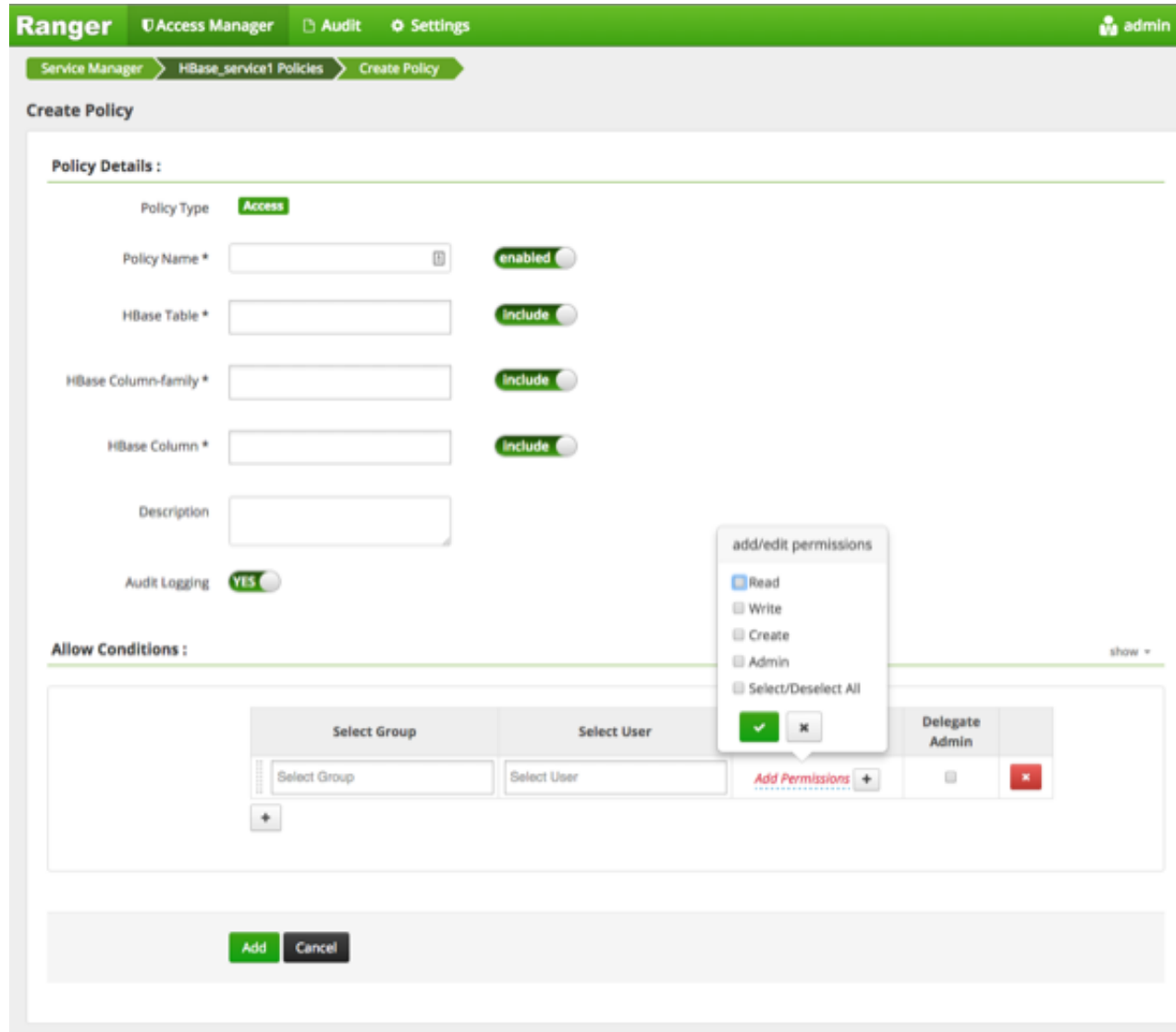
1. On the Service Manager page, select an existing service under HBase.



The List of Policies page appears.

2. Click **Add New Policy**.

The Create Policy page appears.



3. Complete the Create Policy page as follows:

Table 3.42. Policy Details

Label	Description
Policy Name	Enter an appropriate policy name. This name cannot be duplicated across the system. This field is mandatory.
HBase Table	Select the appropriate database. Multiple databases can be selected for a particular policy. This field is mandatory.
HBase Column-family	For the selected table, specify the column families to which the policy applies.

Label	Description
HBase Column	For the selected table and column families, specify the columns to which the policy applies.
Description	(Optional) Describe the purpose of the policy.
Audit Logging	Specify whether this policy is audited. (De-select to disable auditing).

Table 3.43. Allow Conditions

Label	Description
Select Group	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). The public group contains all users, so granting access to the public group grants access to all users.
Select User	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).
Permissions	Add or edit permissions: Read, Write, Create, Admin, Select/Deselect All.
Delegate Admin	When a policy is assigned to a user or a group of users those users become the delegated admin. The delegated admin can update, delete the policies. It can also create child policies based on the original policy (base policy).

For reference information, see: [Wildcard Characters \[324\]](#) and [{USER} Variable \[324\]](#).

- You can use the Plus (+) symbol to add additional conditions. Conditions are evaluated in the order listed in the policy. The condition at the top of the list is applied first, then the second, then the third, and so on.
- Click **Add**.

3.2.4.1.2. Provide User Access to HBase Database Tables from the Command Line

HBase provides the means to manage user access to HBase 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 would be:

```
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 would be:

```
revoke 'user1', 'usertable'
```



Note

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

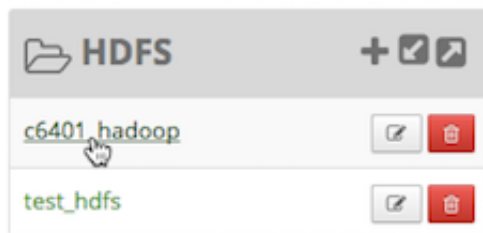
3.2.4.1.3. Create an HDFS Policy

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 Service Manager. If there are no policies, the Ranger plugin checks for permissions set in HDFS.

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

To add a new policy to an existing HDFS service:

1. On the Service Manager page, select an existing service under HDFS.

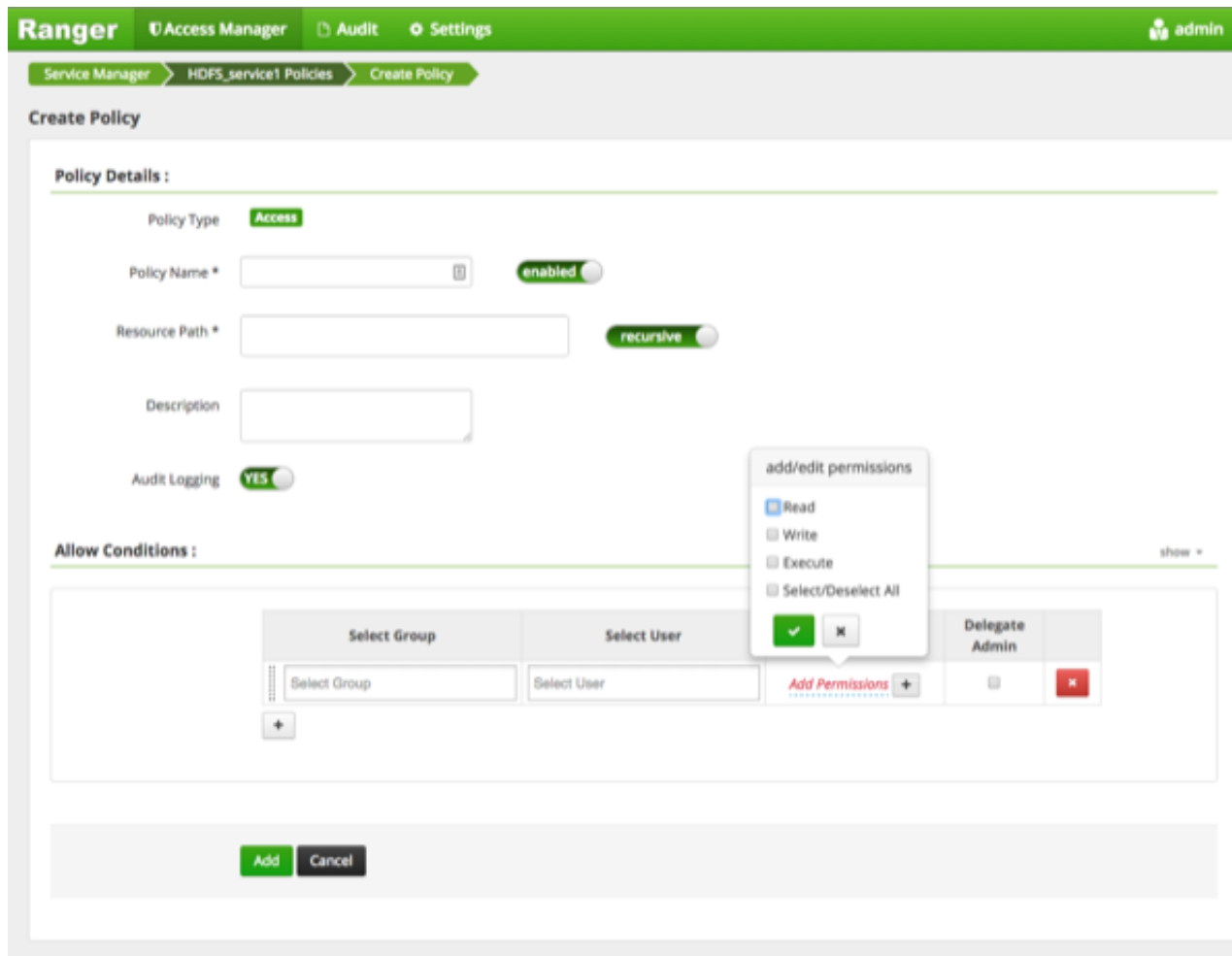


The List of Policies page appears.

Policy ID	Policy Name	Status	Audit Logging	Groups	Users	Action
13	Example-Service-1-20160211205602	Enabled	Enabled		admin	

2. Click **Add New Policy**.

The Create Policy page appears.



3. Complete the Create Policy page as follows:

Table 3.44. Policy Details

Field	Description
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 all 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.
Audit Logging	Specify whether this policy is audited. (De-select to disable auditing).

Table 3.45. Allow Conditions

Label	Description
Select Group	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).

Label	Description
	The public group contains all users, so granting access to the public group grants access to all users.
Select User	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).
Permissions	Add or edit permissions: Read, Write, Create, Admin, Select/Deselect All.
Delegate Admin	When a policy is assigned to a user or a group of users those users become the delegated admin. The delegated admin can update, delete the policies. It can also create child policies based on the original policy (base policy).

For reference information, see: [Wildcard Characters \[324\]](#) and [{USER} Variable \[324\]](#).

4. You can use the Plus (+) symbol to add additional conditions. Conditions are evaluated in the order listed in the policy. The condition at the top of the list is applied first, then the second, then the third, and so on.

5. Click **Add**.

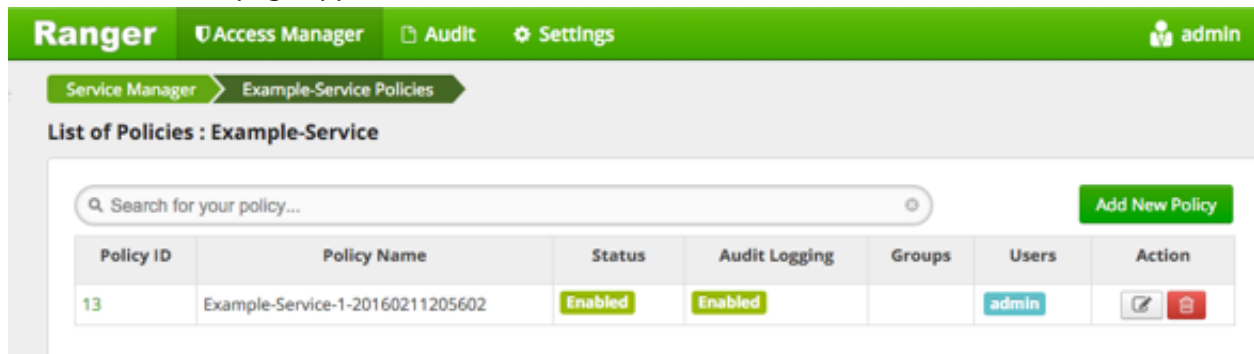
3.2.4.1.4. Create a Hive Policy

To add a new policy to an existing Hive service:

1. On the Service Manager page, select an existing service under Hive.

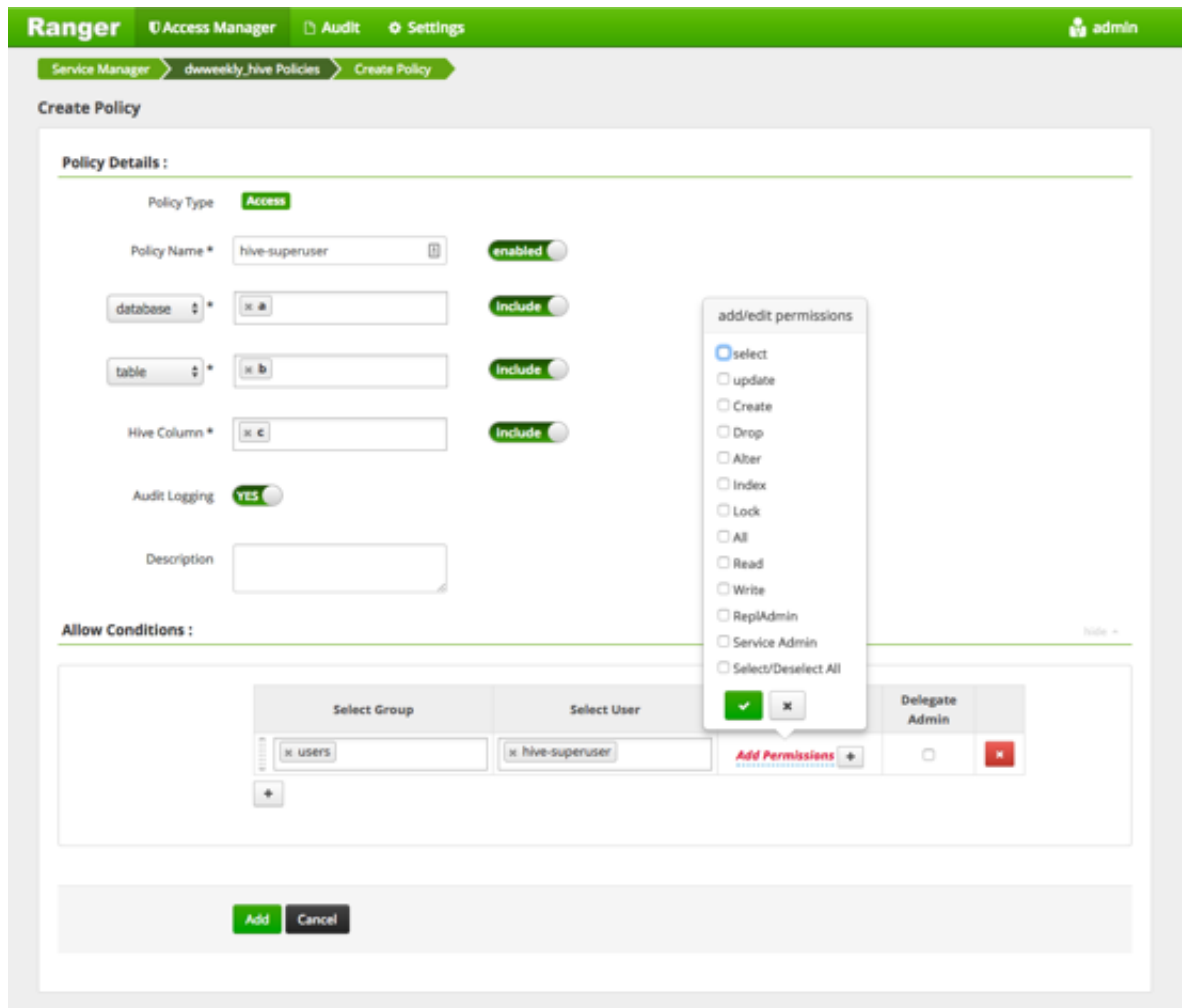


The List of Policies page appears.



2. Click **Add New Policy**.

The Create Policy page appears.



3. Complete the Create Policy page as follows:

Table 3.46. Policy Details

Field	Description
Policy Name	Enter an appropriate policy name. This name cannot be duplicated across the system. This field is mandatory. The policy is enabled by default.
Database	Type in the applicable database name. The autocomplete feature displays available databases based on the entered text. Include is selected by default to allow access. Select Exclude to deny access..
Table	To continue adding a table-based policy, keep Table selected. Type in the applicable table name. The autocomplete feature displays available tables based on the entered text. Include is selected by default to allow access. Select Exclude to deny access.

Field	Description
Column	<p>Type in the applicable Hive column name. The autocomplete feature displays available columns based on the entered text.</p> <p>Include is selected by default to allow access. Select Exclude to deny access.</p> <p>If using the Ranger Hive plugin with HiveServer2 or HiveServer2-LLAP, where column or description permissions include <code>all</code>, you must set a parameter for Hive columns to display as expected: in Ambari>Hive, under <code>ranger-hive-security.xml</code>, enter: <code>xasecure.hive.describetable.showcolumns.authorization.opt</code> <code>all</code>. Failure to set this parameter will result in the error message <code>HiveAccessControlException</code>.</p>
URL	<p>The URL field in the Hive Policy is applicable only for cloud users.</p> <p>Specify the cloud storage path (for example <code>s3a://dev-admin/demo/campaigns.txt</code>) where the end-user permission is needed to read/write the Hive data from/to a cloud storage path.</p> <p>Permissions: READ operation on the URL permits the user to perform HiveServer2 operations which use S3 as data source for Hive tables. WRITE operation on the URL permits the user to perform HiveServer2 operations which write data to the specified S3 location.</p> <p>This feature is a Technical Preview: it is not ready for production deployment.</p>
Description	<p>(Optional) Describe the purpose of the policy.</p> <p>If using the Ranger Hive plugin with HiveServer2 or HiveServer2-LLAP, where column or description permissions include <code>all</code>, you must set a parameter for Hive columns to display as expected: in Ambari>Hive, under <code>ranger-hive-security.xml</code>, enter: <code>xasecure.hive.describetable.showcolumns.authorization.opt</code> <code>all</code>. Failure to set this parameter will result in the error message <code>HiveAccessControlException</code>.</p>
Hive Service Name	<p>hiveservice is used only in conjunction with <code>Permissions=Service Admin</code>. Enables a user who has Service Admin permission in Ranger to run the kill query API: <code>kill query <queryID></code>. Supported value: <code>*</code>. (Required)</p>
Audit Logging	<p>Specify whether this policy is audited. (De-select to disable auditing).</p>

Table 3.47. Allow Conditions

Label	Description
Select Group	<p>Specify a group to which this policy applies. To designate the group as an Administrator for the chosen resource, select the Delegate Admin check box. (Administrators can create child policies based on existing policies).</p> <p>The public group contains all users, so granting access to the public group grants access to all users.</p>
Select User	<p>Specify one or more users to which this policy applies. To designate the group as an Administrator for the chosen resource, select the Delegate Admin check</p>

Label	Description
	box. (Administrators can create child policies based on existing policies).
Permissions	<p>Add or edit permissions: Select, Update, Create, Drop, Alter, Index, Lock, All, ReplAdmin, Service Admin, Select/Deselect All.</p> <p>If using the Ranger Hive plugin with HiveServer2 or HiveServer2-LLAP, where column or description permissions include <code>all</code>, you must set a parameter for Hive columns to display as expected: in Ambari>Hive, under <code>ranger-hive-security.xml</code>, enter: <code>xasecure.hive.describetable.showcolumns.authorization.opt</code> <code>all</code>. Failure to set this parameter will result in the error message <code>HiveAccessControlException</code>.</p> <p>In order to execute <code>repl dump</code>, <code>repl load</code>, or <code>repl status</code> commands, you must set a parameter: in Ambari>Hive, under <code>hive-site.xml</code>, enter: <code>hive.distcp.privileged.doAs=hive</code>.</p> <p>Service Admin is used in conjunction with Hive Service Name and the kill query API: <code>kill query <queryID></code>.</p>
Delegate Admin	When Delegate Admin is selected, administrative privileges are assigned to the applicable users and groups. Delegated administrators can update and delete policies, and can also create child policies based on the original policy.

For reference information, see: [Wildcard Characters \[324\]](#) and [{USER} Variable \[324\]](#).

- You can use the Plus (+) symbol to add additional conditions. Conditions are evaluated in the order listed in the policy. The condition at the top of the list is applied first, then the second, then the third, and so on.
- Click **Add**.



Note

The Ranger Hive plugin only protects HiveServer2; Hive CLI is not supported by Ranger.

3.2.4.1.5. Provide 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 would be:

```
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 would be:

```
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.

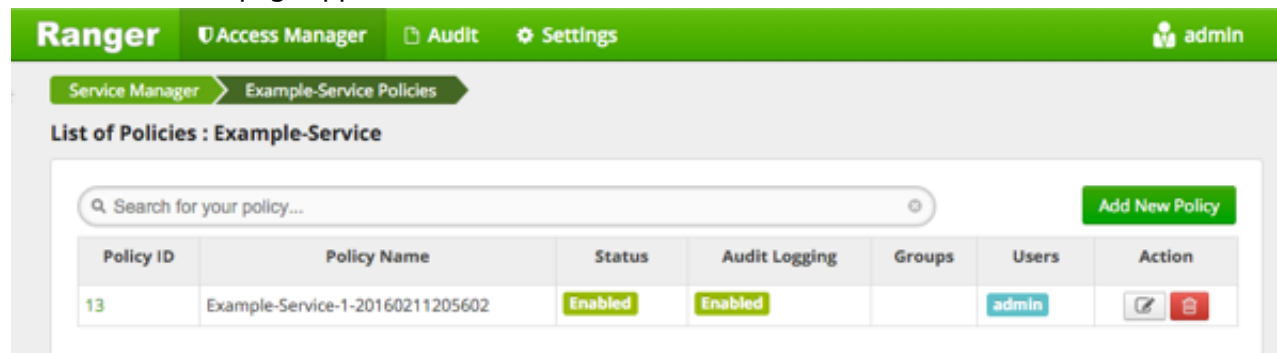
3.2.4.1.6. Create a Kafka Policy

To add a new policy to an existing Kafka service:

1. On the Service Manager page, select an existing service under Kafka.

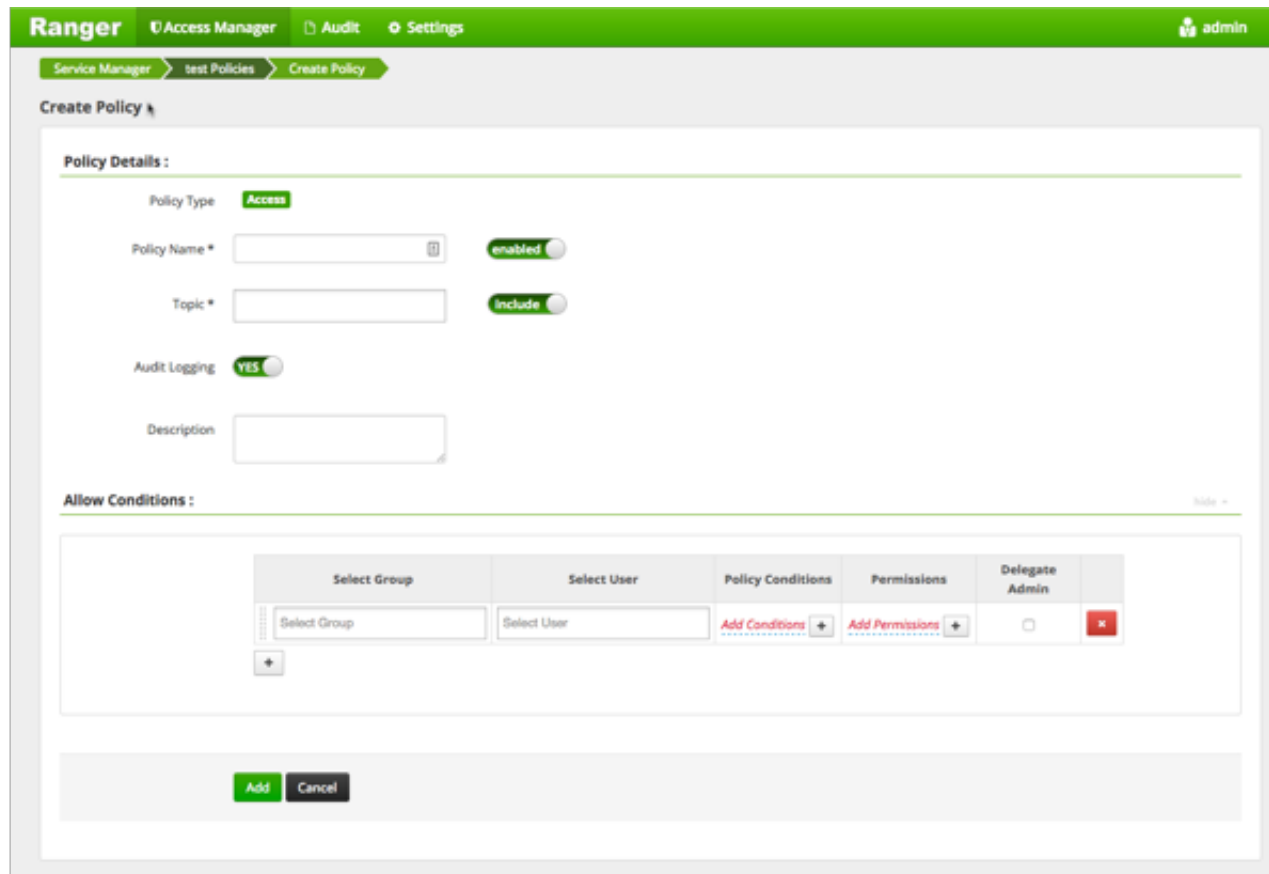


The List of Policies page appears.



2. Click **Add New Policy**.

The Create Policy page appears.



3. Complete the Create Policy page as follows:

Table 3.48. Policy Details

Field	Description
Policy Name	Enter an appropriate policy name. This name cannot be duplicated across the system. This field is mandatory.
Topic	A topic is a category or feed name to which messages are published.
Description	(Optional) Describe the purpose of the policy.
Audit Logging	Specify whether this policy is audited. (De-select to disable auditing).

Table 3.49. Allow Conditions

Label	Description
Select Group	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). The public group contains all users, so granting access to the public group grants access to all users.
Select User	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).

Label	Description
Policy Conditions	Specify IP address range.
Permissions	Add or edit permissions: Read, Write, Create, Admin, Select/Deselect All.
Delegate Admin	When a policy is assigned to a user or a group of users those users become the delegated admin. The delegated admin can update, delete the policies. It can also create child policies based on the original policy (base policy).

For reference information, see: [Wildcard Characters \[324\]](#) and [{USER} Variable \[324\]](#).

- You can use the Plus (+) symbol to add additional conditions. Conditions are evaluated in the order listed in the policy. The condition at the top of the list is applied first, then the second, then the third, and so on.
- Click **Add**.

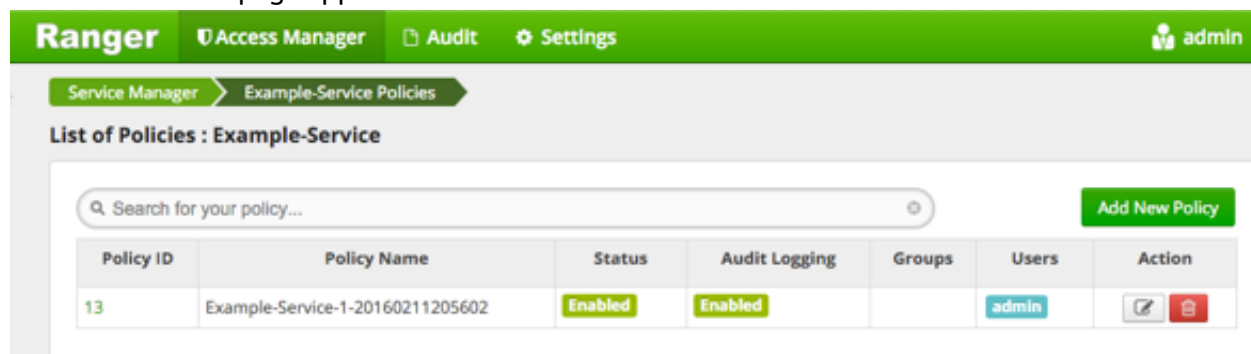
3.2.4.1.7. Create a Knox Policy

To add a new policy to an existing Knox service:

- On the Service Manager page, select an existing service under Knox.

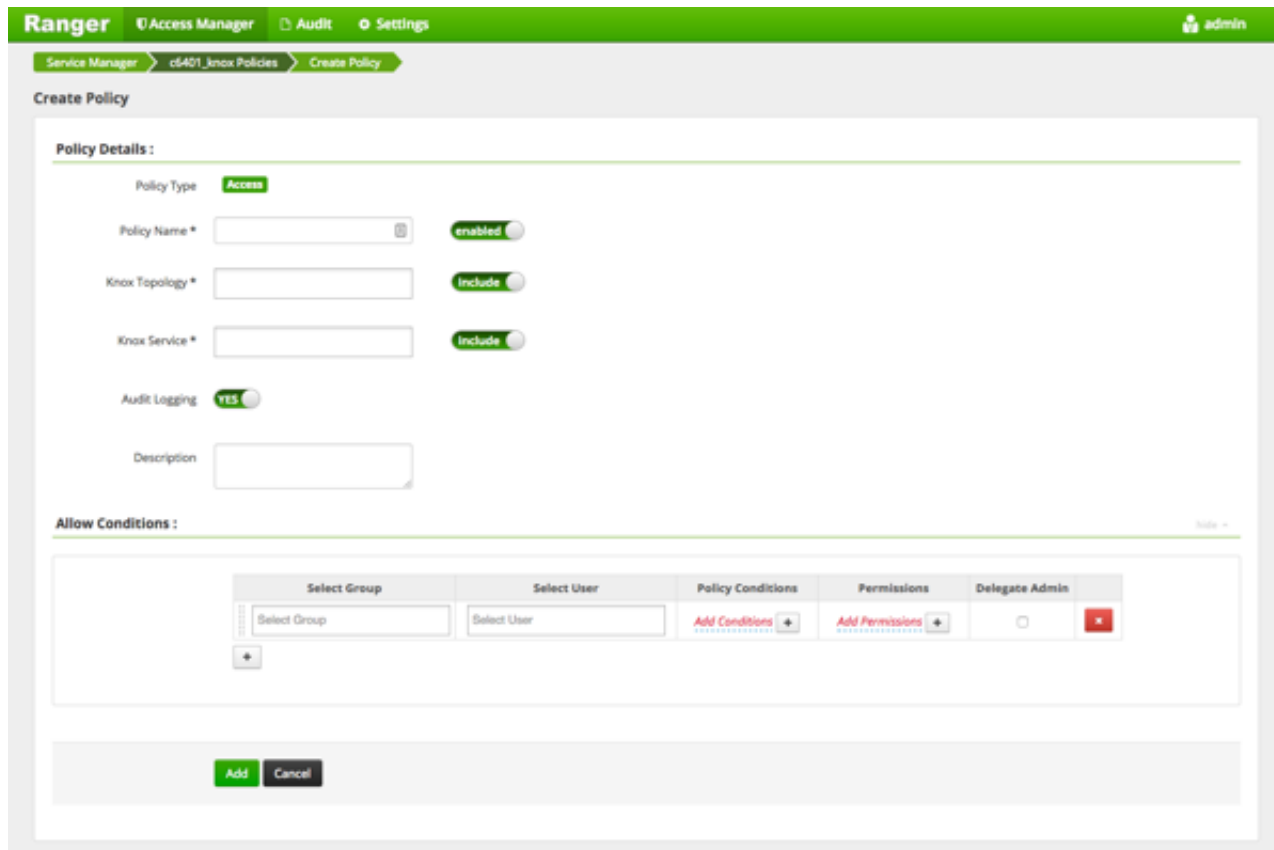


The List of Policies page appears.



- Click **Add New Policy**.

The Create Policy page appears.



3. Complete the Create Policy page as follows:

Table 3.50. Policy Details

Field	Description
Policy Name	Enter an appropriate policy name. This name cannot be duplicated across the system. This field is mandatory.
Knox Topology	Enter an appropriate Topology Name.
Knox Service	Enter an appropriate Service Name.
Description	(Optional) Describe the purpose of the policy.
Audit Logging	Specify whether this policy is audited. (De-select to disable auditing).

Table 3.51. Allow Conditions

Label	Description
Select Group	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). The public group contains all users, so granting access to the public group grants access to all users.
Select User	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).

Label	Description
Policy Conditions	Specify IP address range,
Permissions	Add or edit permissions: Read, Write, Create, Admin, Select/Deselect All.
Delegate Admin	When a policy is assigned to a user or a group of users those users become the delegated admin. The delegated admin can update, delete the policies. It can also create child policies based on the original policy (base policy).

For reference information, see: [Wildcard Characters \[324\]](#) and [{USER} Variable \[324\]](#).

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.

- You can use the Plus (+) symbol to add additional conditions. Conditions are evaluated in the order listed in the policy. The condition at the top of the list is applied first, then the second, then the third, and so on.
- Click **Add**.

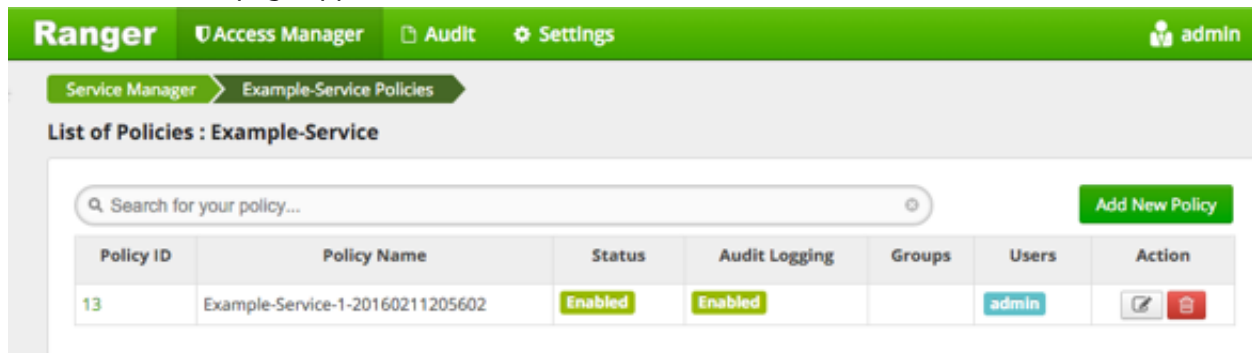
3.2.4.1.8. Create a Solr Policy

To add a new policy to an existing Solr service:

- On the Service Manager page, select an existing service under Solr.

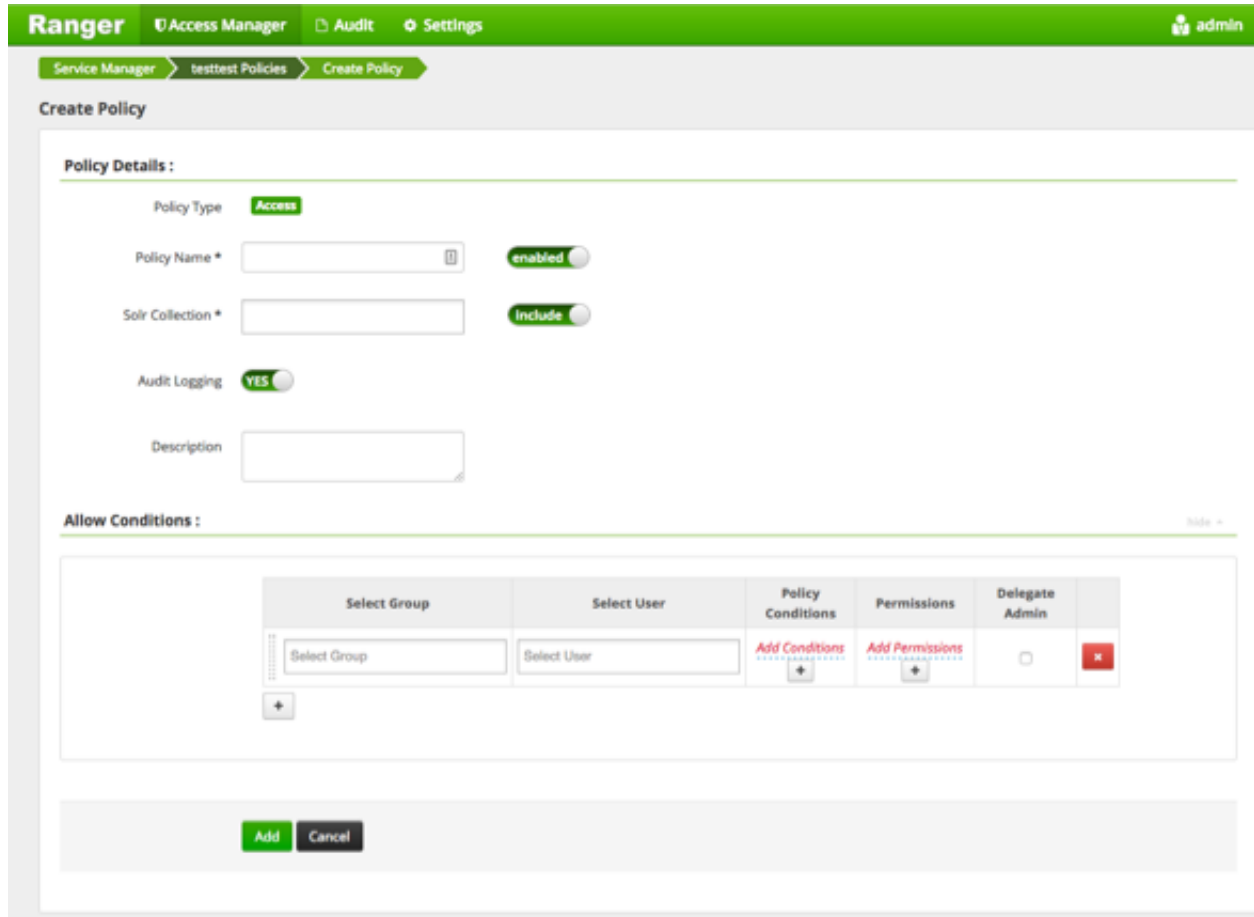


The List of Policies page appears.



- Click **Add New Policy**.

The Create Policy page appears.



3. Complete the Create Policy page as follows:

Table 3.52. Policy Details

Field	Description
Policy Name	Enter an appropriate policy name. This name cannot be duplicated across the system. This field is mandatory.
Solr Collection	For HDP Search's Solr Instance: <code>http:host_ip:8983/solr</code> For Ambari Infra's Solr Instance: <code>http:host_ip:8886/solr</code>
Description	(Optional) Describe the purpose of the policy.
Audit Logging	Specify whether this policy is audited. (De-select to disable auditing).

Table 3.53. Allow Conditions

Label	Description
Select Group	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). The public group contains all users, so granting access to the public group grants access to all users.

Label	Description
Select User	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).
Policy Conditions	Specify IP address range,
Permissions	Add or edit permissions: Read, Write, Create, Admin, Select/Deselect All.
Delegate Admin	When a policy is assigned to a user or a group of users those users become the delegated admin. The delegated admin can update, delete the policies. It can also create child policies based on the original policy (base policy).

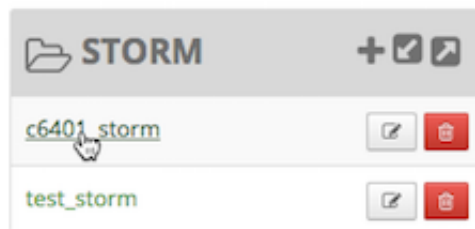
For reference information, see: [Wildcard Characters \[324\]](#) and [{USER} Variable \[324\]](#).

- You can use the Plus (+) symbol to add additional conditions. Conditions are evaluated in the order listed in the policy. The condition at the top of the list is applied first, then the second, then the third, and so on.
- Click **Add**.

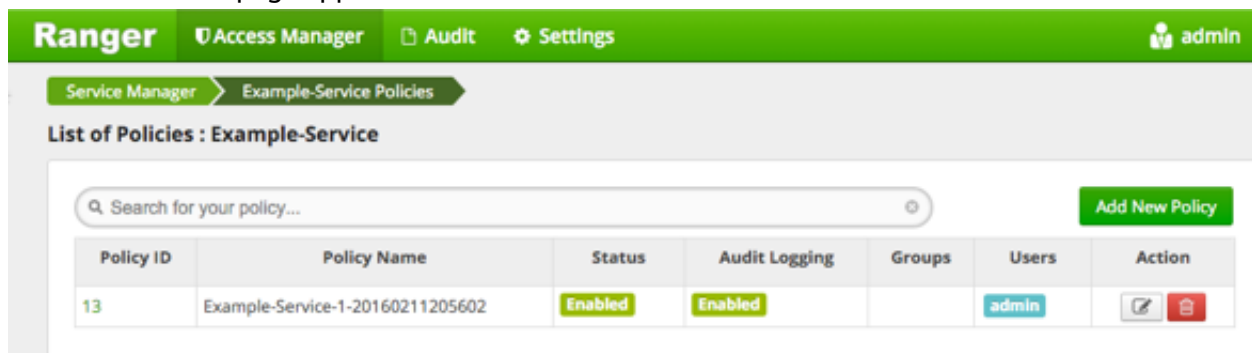
3.2.4.1.9. Create a Storm Policy

To add a new policy to an existing Storm service:

- On the Service Manager page, select an existing service under Storm.



The List of Policies page appears.



- Click **Add New Policy**.

The Create Policy page appears.

3. Complete the Create Policy page as follows:

Table 3.54. Policy Details

Label	Description
Policy Name	Enter an appropriate policy name. This name is cannot be duplicated across the system. This field is mandatory.
Storm Topology	Enter an appropriate Topology Name.
Description	(Optional) Describe the purpose of the policy.
Audit Logging	Specify whether this policy is audited. (De-select to disable auditing).

Table 3.55. Allow Conditions

Label	Description
Select Group	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). The public group contains all users, so granting access to the public group grants access to all users.
Select User	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).
Storm User and Group Permissions [319]	Add or edit permissions: Read, Write, Create, Admin, Select/Deselect All.
Delegate Admin	When a policy is assigned to a user or a group of users those users become the delegated admin. The delegated admin can update, delete the policies. It can also create child policies based on the original policy (base policy).

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.

Table 3.56. Storm 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.

For reference information, see: [Wildcard Characters \[324\]](#) and [{USER} Variable \[324\]](#).

4. You can use the Plus (+) symbol to add additional conditions. Conditions are evaluated in the order listed in the policy. The condition at the top of the list is applied first, then the second, then the third, and so on.
5. Click **Add**.

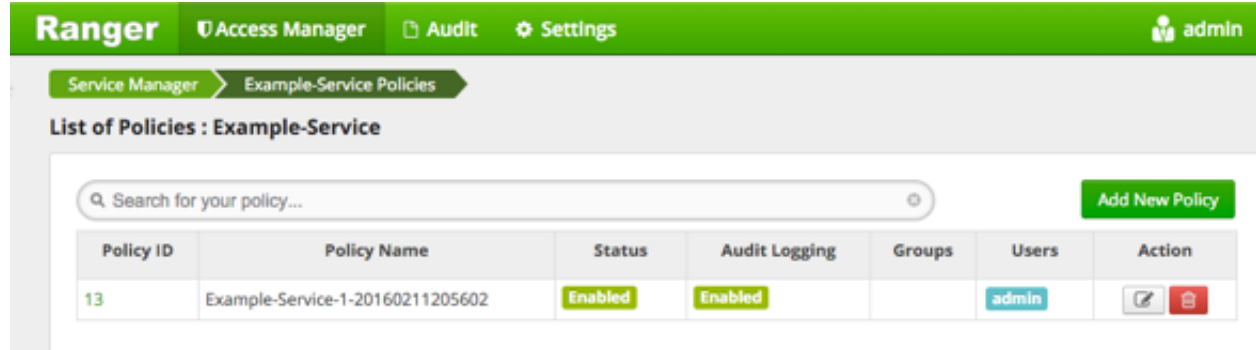
3.2.4.1.10. Create a YARN Policy

To add a new policy to an existing YARN service:

1. On the Service Manager page, select an existing service under YARN.



The List of Policies page appears.



2. Click **Add New Policy**.

The Create Policy page appears.

3. Complete the Create Policy page as follows:

Table 3.57. Policy Details

Field	Description
Policy Name	Enter an appropriate policy name. This name cannot be duplicated across the system. This field is mandatory.
Queue	The fundamental unit of scheduling in yarn.
Recursive	You can indicate whether all files or folders within the existing folder comes under the policy. Can be used instead of wildcard characters.
Description	(Optional) Describe the purpose of the policy.
Audit Logging	Specify whether this policy is audited. (De-select to disable auditing).

Table 3.58. Allow Conditions

Label	Description
Select Group	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). The public group contains all users, so granting access to the public group grants access to all users.

Label	Description
Select User	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).
Permissions	Add or edit permissions: Read, Write, Create, Admin, Select/Deselect All.
Delegate Admin	When a policy is assigned to a user or a group of users those users become the delegated admin. The delegated admin can update, delete the policies. It can also create child policies based on the original policy (base policy).

For reference information, see: [Wildcard Characters \[324\]](#) and [{USER} Variable \[324\]](#).

4. Click **Add**.

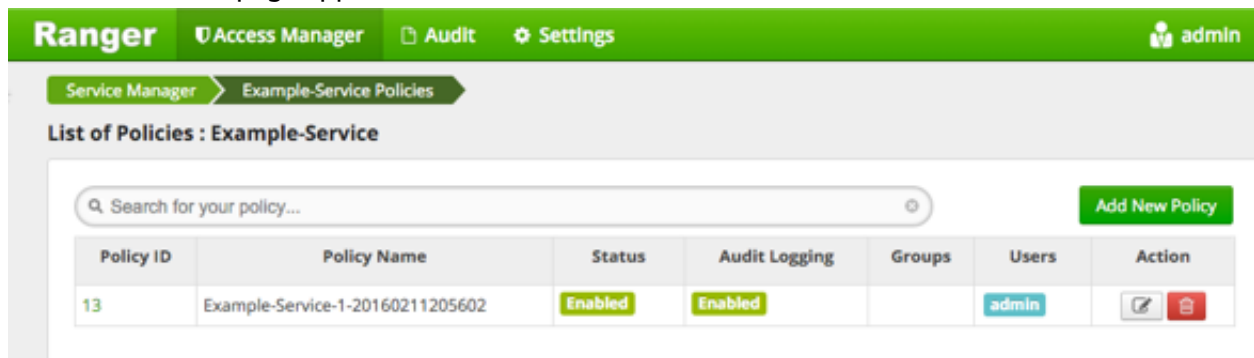
3.2.4.1.11. Create an Atlas Policy

To add a new policy to an existing Atlas service:

1. On the Service Manager page, select an existing service under Atlas.

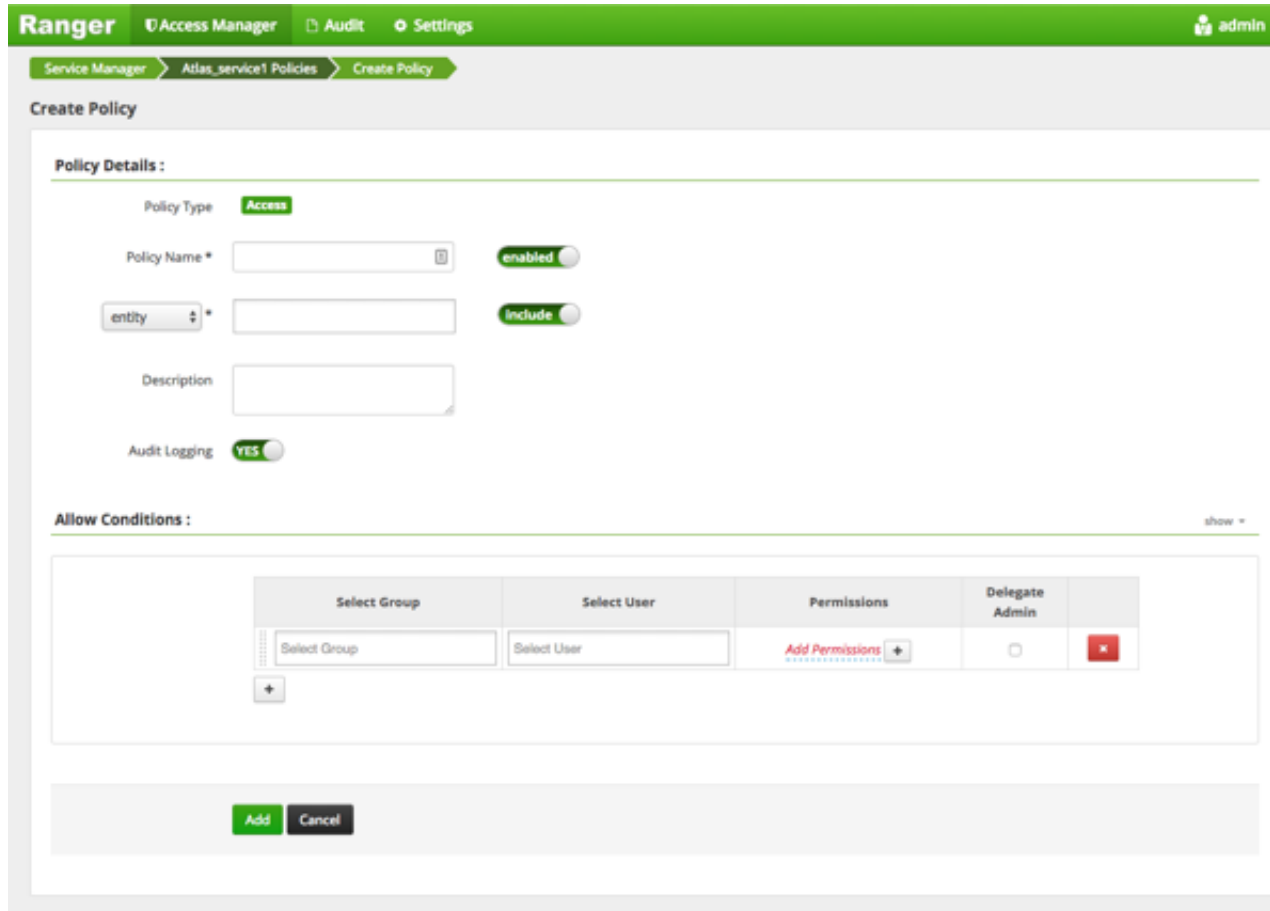


The List of Policies page appears.



2. Click **Add New Policy**.

The Create Policy page appears.



3. Complete the Create Policy page as follows:

Table 3.59. Policy Details

Field	Description
Policy Name	Enter an appropriate policy name. This name cannot be duplicated across the system. This field is mandatory.
entity	Select entity, type, operation, taxonomy, or term.
Description	(Optional) Describe the purpose of the policy.
Audit Logging	Specify whether this policy is audited. (De-select to disable auditing).

Table 3.60. Allow Conditions

Label	Description
Select Group	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). The public group contains all users, so granting access to the public group grants access to all users.
Select User	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).
Permissions	Add or edit permissions: Read, Write, Create, Admin, Select/Deselect All.
Delegate Admin	When a policy is assigned to a user or a group of users those users become the delegated admin. The delegated admin can update, delete the policies. It can also create child policies based on the original policy (base policy).

For reference information, see: [Wildcard Characters \[324\]](#) and [{USER} Variable \[324\]](#).

- You can use the Plus (+) symbol to add additional conditions. Conditions are evaluated in the order listed in the policy. The condition at the top of the list is applied first, then the second, then the third, and so on.
- Click **Add**.

3.2.4.1.12. Wildcard and Variable Reference Information

- [Wildcard Characters \[324\]](#)
- [{USER} Variable \[324\]](#)
 - [{USER} Variable Recommended Practices and Customizability \[324\]](#)

3.2.4.1.12.1. Wildcard Characters

Wildcard characters can be included in the resource path, the database name, the table name, or the column name:

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

3.2.4.1.12.2. {USER} Variable

The variable `{USER}` can be used to autofill the accessing user, for example:

In **Select User**, choose `{USER}`.

In **Resource Path**, enter `data_{USER}`.

3.2.4.1.12.2.1. {USER} Variable Recommended Practices and Customizability

Ranger requires that string `{USER}` is used to represent accessing user as the user in the policy-item in a Ranger policy. However, Ranger provides flexible way of customizing the string that is used as shorthand to represent the accessing user's name in the policy resource specification. By default, Ranger policy resource specification expects characters `{` and `}` as delimiters for string `USER`, however, ranger supports customizable way of specifying delimiter characters, escaping those delimiters, and the string `USER` itself by prefixing it with another, user-specified string on a per resource-level basis in the service definition of each component supported by Ranger.

For example, if for a certain HDFS installation, if the path names may contain '{' or '}' as valid characters, but not '%' character, then the service-definition for HDFS can be specified as:

```
"resources": [
{
  "itemId": 1,
  "name": "path",
  "type": "path",
  "level": 10,
  "parent": "",
  "mandatory": true,
  "lookupSupported": true,
  "recursiveSupported": true,
  "excludesSupported": false,
  "matcher": "org.apache.ranger.plugin.resourcematcher.
RangerPathResourceMatcher",
  "matcherOptions": {"wildcard": true, "ignoreCase": false},
  "replaceTokens":true, "tokenDelimiterStart": "%", "tokenDelimiterEnd": "%",
  "tokenDelimiterPrefix": "rangerToken:"}
  "validationRegex": "",
  "validationMessage": "",
  "uiHint": "",
  "label": "Resource Path",
  "description": "HDFS file or directory
path"
}
]
```

Corresponding ranger policy for the use case for HDFS will be written as follow:

```
resource: path=/home/%rangerToken:USER%
user: {USER}
permissions: all, delegateAdmin=true
```

The following customizable *matcherOptions* are available for this feature:

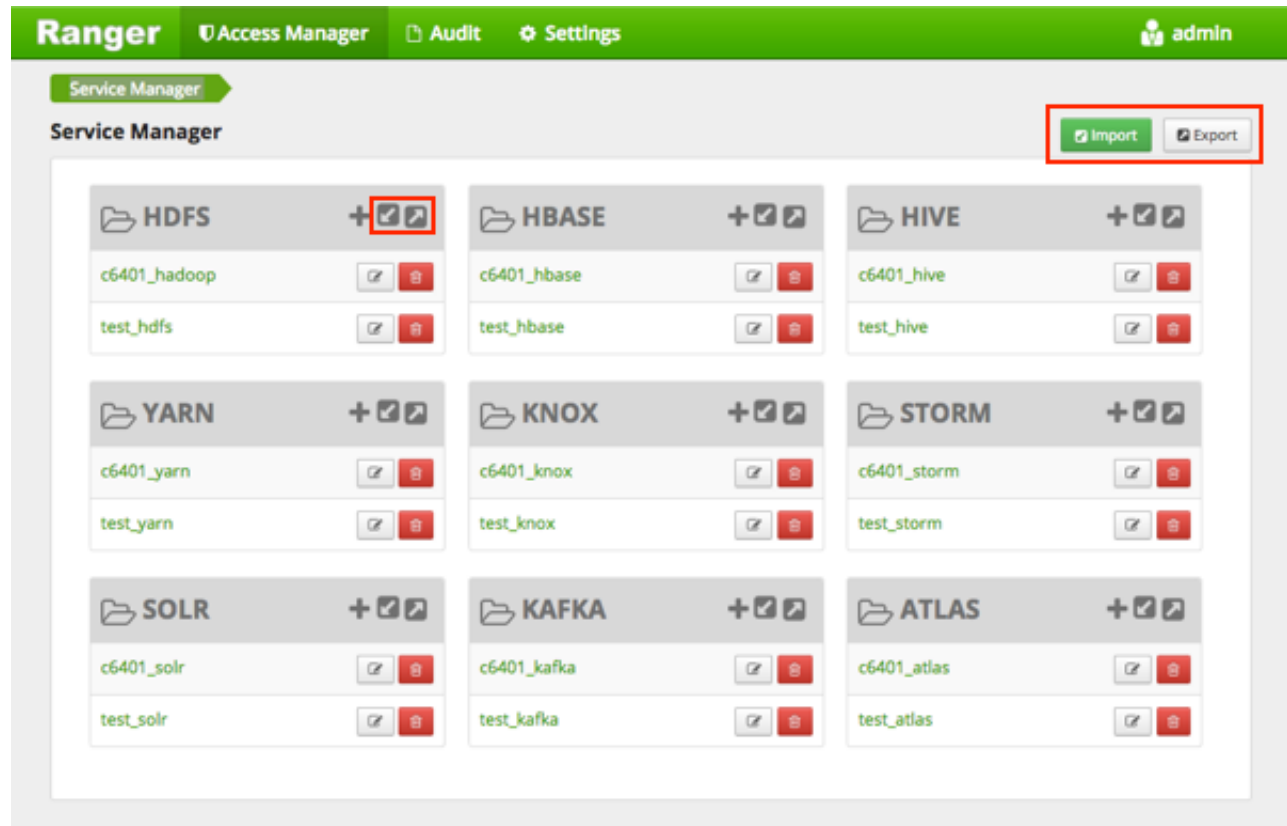
- *replaceTokens*: `true` if short-hand for user in resource-spec needs to be replaced at run-time with current-user's name; `false` if the resource-spec needs to be interpreted as it is. Default value: `true`.
- *tokenDelimiterStart*: Identifies start character of short-hand for current-user in resource specification. Default value: `{`.
- *tokenDelimiterEnd*: Identifies end character of short-hand for current-user in resource specification. Default value: `}`.
- *tokenDelimiterEscape*: Identifies escape character for escaping *tokenDelimiterStart* or *tokenDelimiterEnd* values in resource specification. Default value: `\`.
- *tokenDelimiterPrefix*: Identifies special prefix which together with string 'USER' makes up short-hand for current-user's name in the resource specification. Default value: `.`

3.2.4.2. Importing and Exporting Resource-Based Policies

You can export and import policies from the Ranger Admin UI for cluster resiliency (backups), during recovery operations, or when moving policies from test clusters to production clusters. You can export/import a specific subset of policies (such as those that pertain to specific resources or user/groups) or clone the entire repository (or multiple repositories) via Ranger Admin UI.

Interfaces

You can import and export policies from the Access Manager page:



The screenshot displays the Ranger Admin UI's Service Manager interface. At the top, a green navigation bar contains the 'Ranger' logo and menu items for 'Access Manager', 'Audit', and 'Settings'. The user 'admin' is logged in. Below the navigation bar, the 'Service Manager' section is active. In the top right corner of this section, there are 'Import' and 'Export' buttons, both highlighted with a red box. The main area is a grid of service categories, each with a folder icon, a name, and a '+ [edit] [delete]' icon. The categories and their policies are:

Service	Policy 1	Policy 2
HDFS	c6401_hadoop	test_hdfs
HBASE	c6401_hbase	test_hbase
HIVE	c6401_hive	test_hive
YARN	c6401_yarn	test_yarn
KNOX	c6401_knox	test_knox
STORM	c6401_storm	test_storm
SOLR	c6401_solr	test_solr
KAFKA	c6401_kafka	test_kafka
ATLAS	c6401_atlas	test_atlas

You can also export policies from the Reports page:

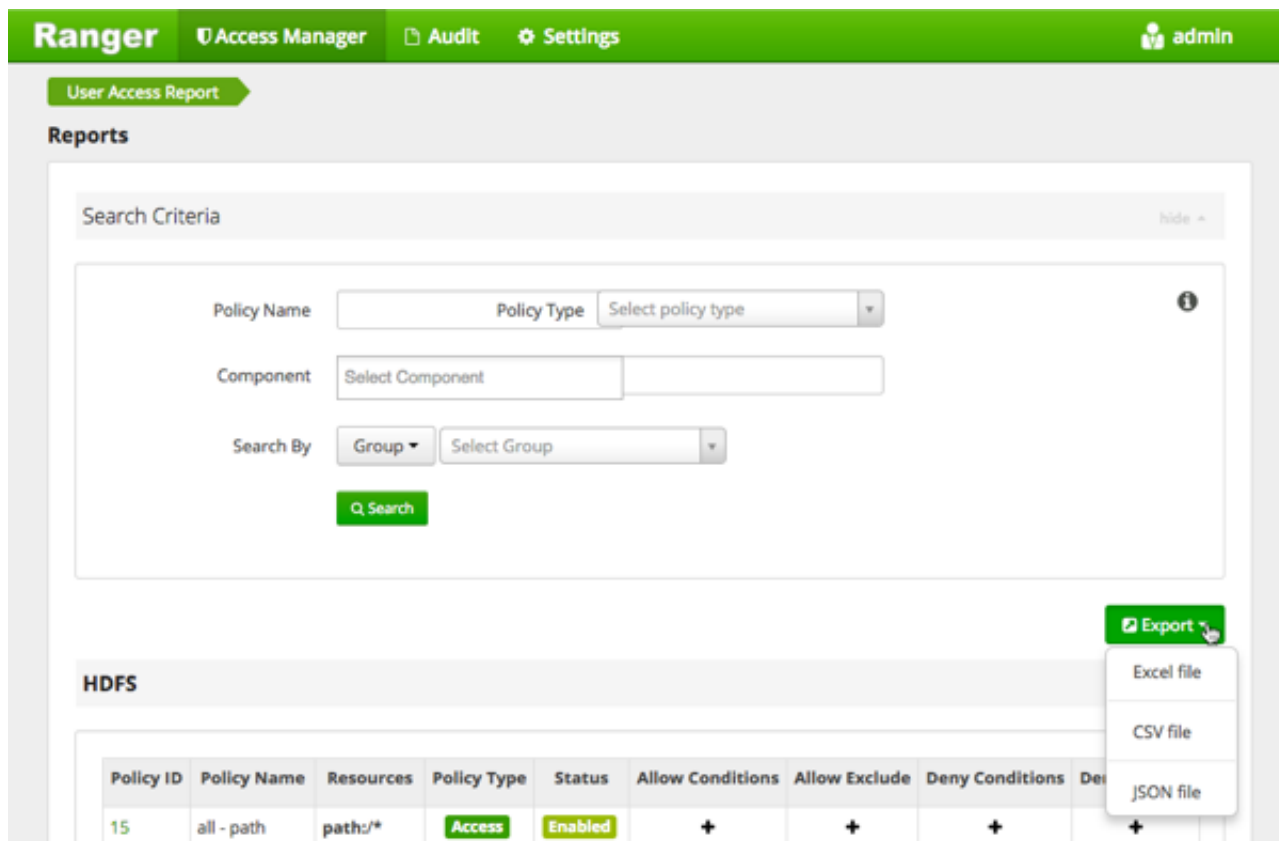


Table 3.61. Export Policy Options

	Access Manager Page	Reports Page
Formats	JSON	JSON Excel CSV
Filtering Supported	No	Yes
Specific Service Export	Yes	Via filtering

Filtering

When exporting from the Reports page, you can apply filters before saving the file.

Export Formats

You can export policies in the following formats:

- Excel
- JSON
- CSV

Note: CSV format is not supported for importing policies.

When you export policies from the Access Manager page, the policies are automatically downloaded in JSON format. If you wish to export in Excel or CSV format, export the policies from the Reports page dropdown menu.

Required User Roles

The Ranger admin user can import and export only Resource & Tag based policies. The credentials for this user are set in Ranger **Configs > Advanced ranger-env** in the fields labeled **admin_username** (default: *admin/admin*).

The Ranger KMS keyadmin user can import and export only KMS policies. The default credentials for this user are *keyadmin/keyadmin*.

Limitations

To successfully import policies, use the following database versions:

- MariaDB: 10.1.16+
- MySQL: 5.6.x+
- Oracle: 11gR2+
- PostgreSQL: 8.4+
- MS SQL: 2008 R2+

Partial import is not supported.

See also: [Importing and Exporting Tag-Based Policies \[362\]](#)

3.2.4.2.1. Import Resource-Based Policies

- [Import Resource-Based Policies for a Specific Service \[328\]](#)
- [Import Resource-Based Policies for All Services \[330\]](#)

3.2.4.2.1.1. Import Resource-Based Policies for a Specific Service

To import the policies for a specific service (HBase, YARN, etc):


1. From the Access Manager page, click the Import icon beside the service:



The Import Policy page opens.

Import Policy ×

Select File :

Select file  Override Policy :

No file chosen

Specify Service Mapping :

Source	To	Destination
<input type="text" value="Enter service name"/>		<input type="text" value="Select service name"/> ▼ ×

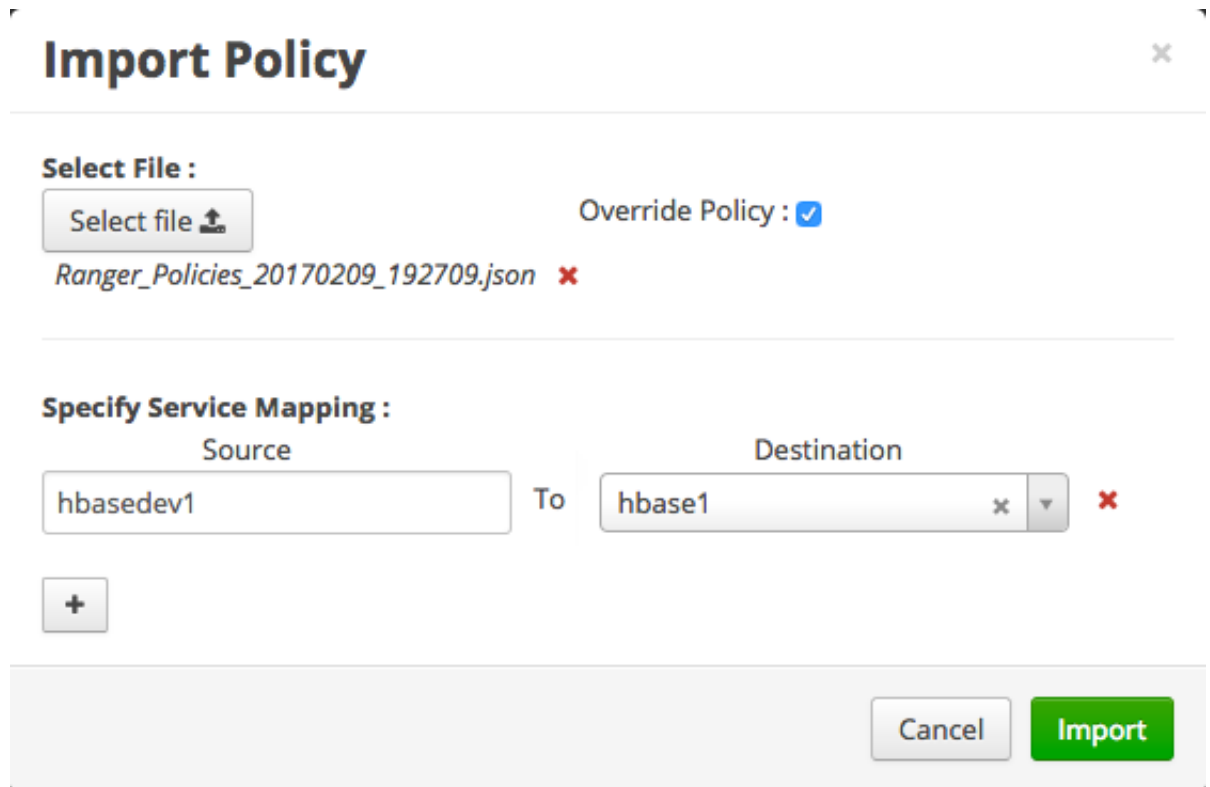
2. Select the file to import.

You can only import policies in JSON format.

3. (Optional) Configure the import operation:

- The Override Policy option deletes all policies of the destination repositories.
- Service Mapping maps the downloaded file repository, i.e. source repository to destination repository.

For example:



The screenshot shows the 'Import Policy' dialog box. At the top, there is a title 'Import Policy' with a close button (X) on the right. Below the title, there is a section 'Select File :'. It contains a 'Select file' button with an upload icon, and a checked checkbox for 'Override Policy :'. Below this, the filename 'Ranger_Policies_20170209_192709.json' is displayed with a red 'X' icon to its right. The next section is 'Specify Service Mapping :'. It has two input fields: 'Source' containing 'hbasedev1' and 'Destination' containing 'hbase1'. A 'To' label is between the fields. A red 'X' icon is to the right of the destination field. Below the fields is a '+' button. At the bottom right of the dialog are 'Cancel' and 'Import' buttons.

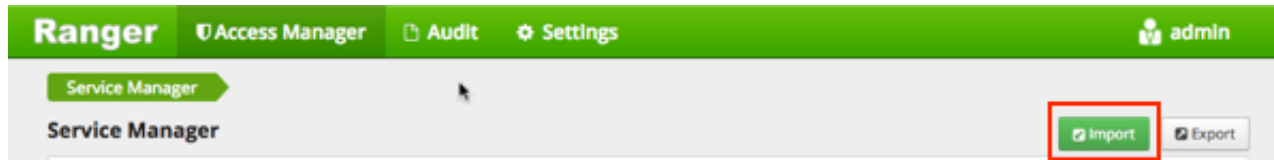
4. Click **Import**.

A confirmation message appears: "Success: File import successfully."

3.2.4.2.1.2. Import Resource-Based Policies for All Services

To import the policies for all services:

1. From the Access Manager page, click the Import button:




The Import Policy page opens.

Import Policy ✕

Service Type :

✕ hdfs ✕ hbase ✕ hive ✕ yarn ✕ Knox ✕ storm ✕ solr ✕ kafka
✕ atlas

Select File :

Select file  Override Policy :

No file chosen

Specify Service Mapping :

Source	To	Destination
<input type="text" value="Enter service name"/>	To	<input type="text" value="Select service name"/> ✕

+

Cancel Import

2. Select the file to import.

You can only import policies in JSON format.

3. (Optional) Configure the import operation:

- Service Types enables you to remove specific services from the import.
- The Override Policy option deletes all policies of the destination repositories.
- Service Mapping maps the downloaded file repository, i.e. source repository to destination repository.

For example:

Import Policy

Service Type :

hdfs
 hbase
 hive
 yarn
 Knox
 storm
 solr
 kafka

atlas

Select file :

Select file

Override Policy :

File Name : all_in_one.xls ✘

Specify service Mapping :

hadoopdev1 To hdfs1 ✘

hivedev1 To hive1 ✘

hbasedev1 To hbase1 ✘

4. Click **Import**.

A confirmation message appears: "Success: File import successfully."

3.2.4.2.2. Export Resource-Based Policies

- [Export Resource-Based Policies for a Specific Service \[332\]](#)
- [Export All Resource-Based Policies for All Services \[333\]](#)

3.2.4.2.2.1. Export Resource-Based Policies for a Specific Service

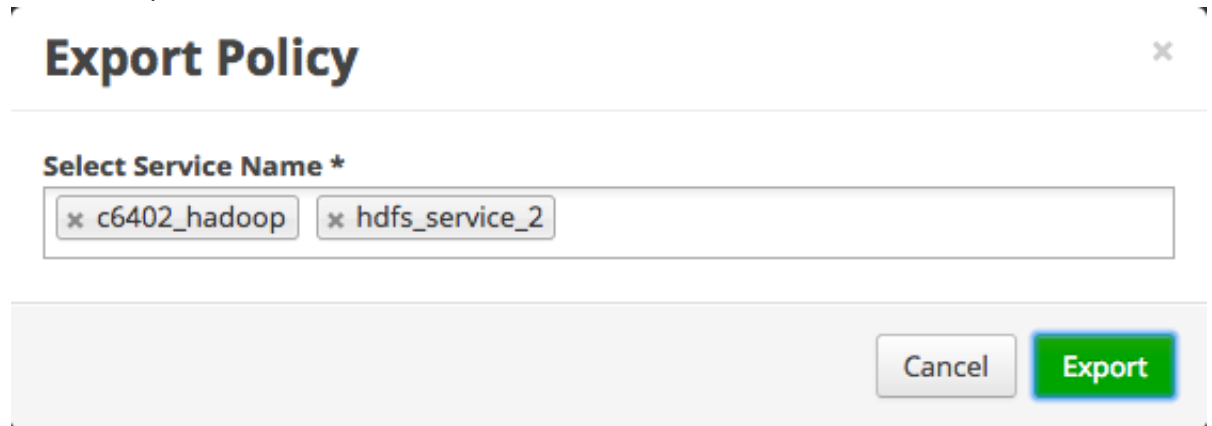
To export the policies for a specific service (HBase, YARN, etc):

1. From the Access Manager page, click the Export icon beside the service:



The Export Policy page opens.

2. Click the Export button.



Export Policy [X]

Select Service Name *

x c6402_hadoop x hdfs_service_2

Cancel Export

3. The file downloads in your browser as a JSON file.

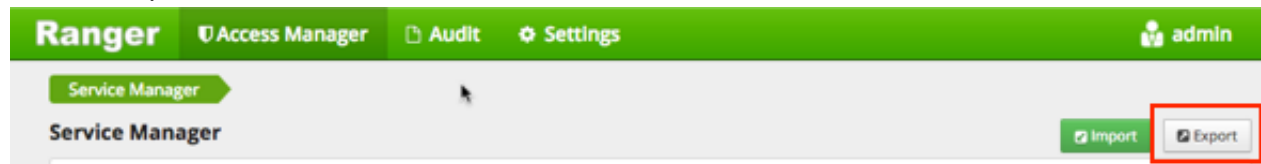
If you wish to export in Excel or CSV format, export the policies from the Reports page dropdown menu.

3.2.4.2.2.2. Export All Resource-Based Policies for All Services

To export all policies for all services (HBase, YARN, etc):

- From the Access Manager page:

1. Click the Export button:



The Export Policy page opens.

2. Remove components or specific services and click Export.

Export Policy ✕

Service Type :

✕ hdfs
✕ hbase
✕ hive
✕ yarn
✕ Knox
✕ storm
✕ solr
✕ kafka

✕ atlas

Select Service Name *

✕ c6402_hadoop
✕ hdfs_service_2
✕ c6402_hbase
✕ c6402_hive

✕ c6402_yarn
✕ c6402_knox
✕ c6402_atlas

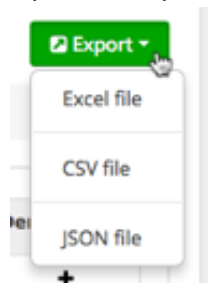
Cancel
Export

3. The file downloads in your browser as a JSON file.

If you wish to export in Excel or CSV format, export the policies from the Reports page dropdown menu.

- From the Reports page:

1. Apply filters before exporting file.
2. Open the Export drop-down menu:



3. Select the file format.

The file downloads in your browser.

3.2.5. Row-level Filtering and Column Masking in Hive

You can use Apache Ranger row-level filters to set access policies for rows in Hive tables. You can also use Ranger column masking to set policies that mask data in Hive columns, for example to show only the first or last four characters of column data.

In this section:

- [Row-level Filtering in Hive with Ranger Policies \[335\]](#)
- [Dynamic Resource-Based Column Masking in Hive with Ranger Policies \[339\]](#)
- [Dynamic Tag-Based Column Masking in Hive with Ranger Policies \[344\]](#)

3.2.5.1. Row-level Filtering in Hive with Ranger Policies

Row-level filtering helps simplify Hive queries. By moving the access restriction logic down into the Hive layer, Hive applies the access restrictions every time data access is attempted. This helps simplify authoring of the Hive query, and provides seamless behind-the-scenes enforcement of row-level segmentation without having to add this logic to the predicate of the query.

Row-level filtering also improves the reliability and robustness of Hadoop. By providing row-level security to Hive tables and reducing the security surface area, Hive data access can be restricted to specific rows based on user characteristics (such as group membership) and the runtime context in which this request is issued.

Typical use cases where row-level filtering can be beneficial include:

- A hospital can create a security policy that allows doctors to view data rows only for their own patients, and that allows insurance claims administrators to view only specific rows for their specific site.
- A bank can create a policy to restrict access to rows of financial data based on the employee's business division, locale, or based on the employee's role (for example: only employees in the finance department are allowed to see customer invoices, payments, and accrual data; only European HR employees can see European employee data).
- A multi-tenant application can create logical separation of each tenant's data so that each tenant can see only their own data rows.

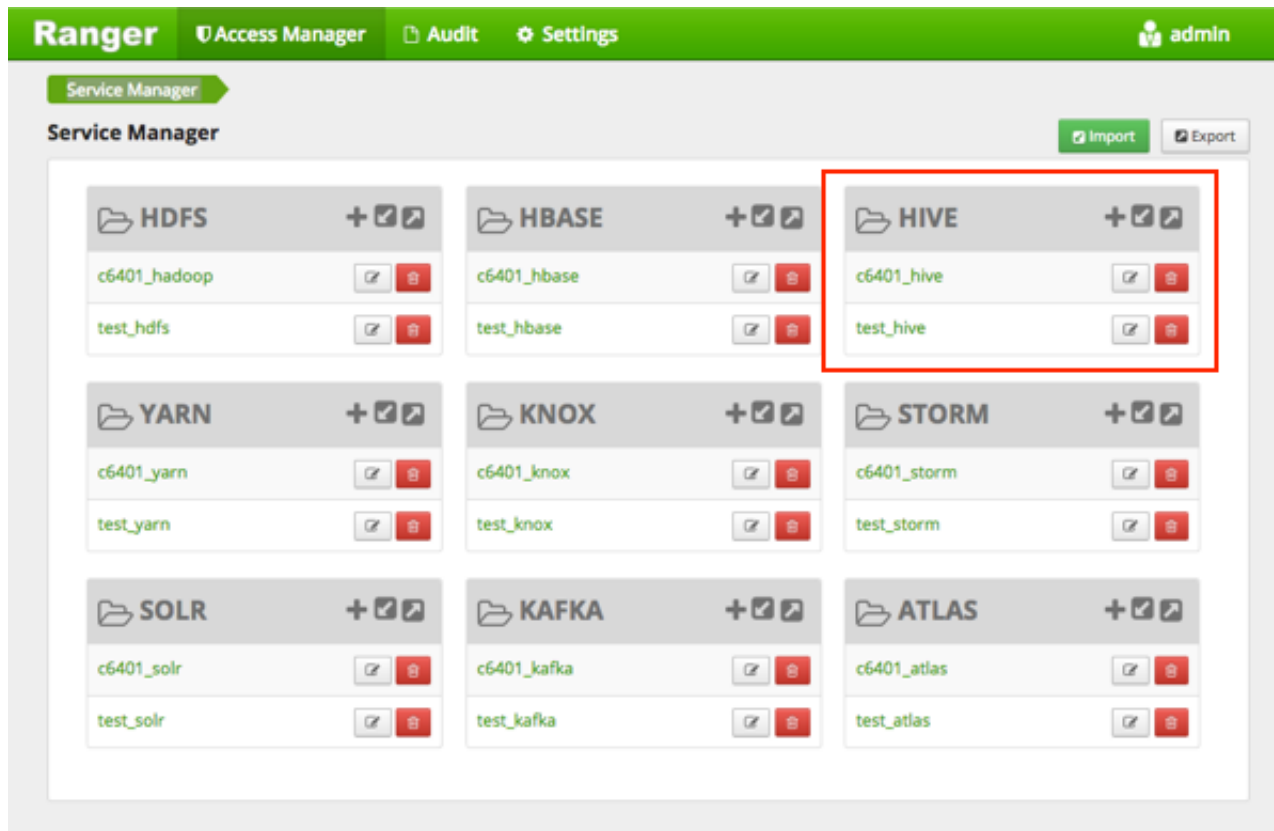
You can use Apache Ranger row-level filters to set access policies for rows in Hive tables. Row-level filter policies are similar to other Ranger access policies. You can set filters for specific users, groups, and conditions.

The following conditions apply when using row-level filters:

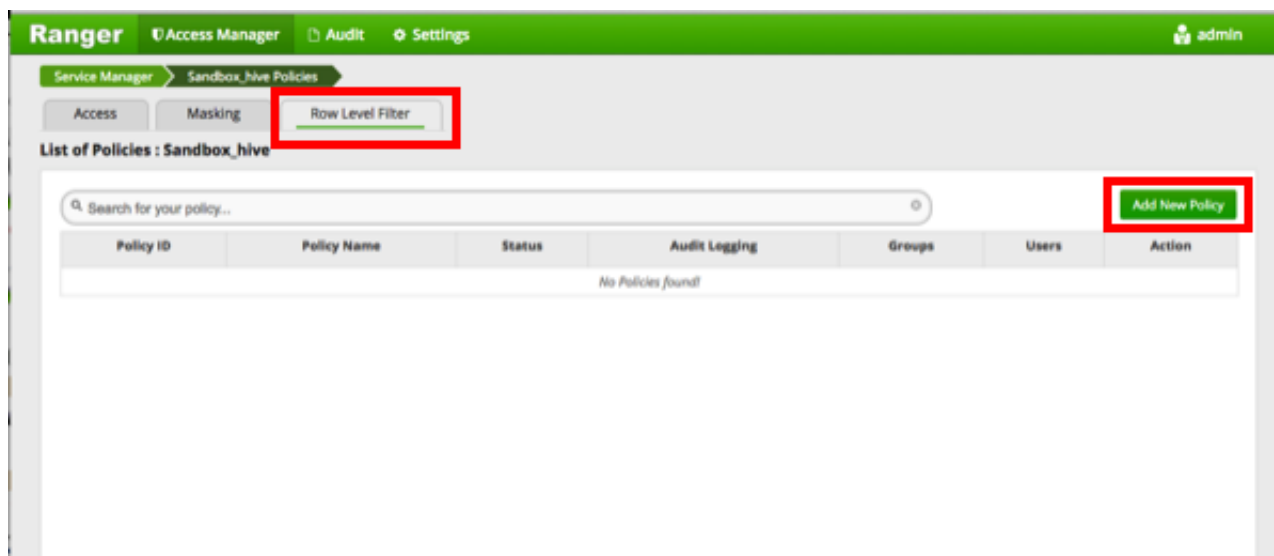
- The filter expression must be a valid WHERE clause for the table or view.
- Each table or view should have its own row-level filter policy.
- Wildcard matching is not supported on database or table names.
- Filters are evaluated in the order listed in the policy.
- An audit log entry is generated each time a row-level filter is applied to a table or view.

Use the following steps to create a row-level filtering policy:

1. On the Service Manager page, select an existing Hive Service.



2. Select the Row Level Filter tab, then click **Add New Policy**.



3. On the Create Policy page, add the following information for the row-level filter:

Table 3.62. Policy Details

Field	Description
Policy Name (required)	Enter an appropriate policy name. This name cannot be duplicated across the system. The policy is enabled by default.
Hive Database (required)	Type in the applicable database name. The auto-complete feature displays available databases based on the entered text.
Hive Table (required)	Type in the applicable table name. The auto-complete feature displays available tables based on the entered text.
Audit Logging	Audit Logging is set to Yes by default. Select No to turn off audit logging.
Description	Enter an optional description for the policy.

Table 3.63. Row Filter Conditions

Label	Description
Select Group	Specify the groups to which this policy applies. The public group contains all users, so granting access to the public group grants access to all users.
Select User	Specify one or more users to which this policy applies.
Access Types	Currently select is the only available access type. This will be used in conjunction with the WHERE clause specified in the Row Level Filter field.
Add Row Filter	<ul style="list-style-type: none"> To create a row filter for the specified users and groups, Click Add Row Filter, then type a valid WHERE clause in the Enter filter expression box. To allow Select access for the specified users and groups without row-level restrictions, do not add a row filter (leave the setting as "Add Row Filter"). Filters are evaluated in the order listed in the policy. The filter at the top of the Row Filter Conditions list is applied first, then the second, then the third, and so on.

Ranger Access Manager Audit Settings admin

Service Manager Sandbox_hive Policies Create Policy

Create Policy

Policy Details :

Policy Type **Row Level Filter**

Policy Name * row-filter.hr.employee **enabled**

Hive Database * hr

Hive Table * employee

Audit Logging **YES**

Description Row-level filter policy for hr.employee table

Row Filter Conditions :

Select Group	Select User	Access Types		
Select Group	admin	select	Add Row Filter +	✖
Select Group	ambari-qa	select	loc_state = 'CA'	✖
public	Select User	select	loc_state In 'CA'	✖

Enter filter expression
enter expression
✔ ✖

Add **Cancel**

- To move a condition in the Row Filter Conditions list (and therefore change the order in which it is evaluated), click the dotted rows icon at the left of the condition row, then drag the condition to a new position in the list.

Ranger Access Manager Audit Settings admin

Service Manager Sandbox_Hive Policies Edit Policy

Edit Policy

Policy Details :

Policy Type **Row Level Filter**

Policy ID **22**

Policy Name * row-filter:hr:employee **enabled**

Hive Database * **hr**

Hive Table * **employee**

Audit Logging **YES**

Description Row-level filter policy for hr.employee table

Row Filter Conditions :

Select Group	Select User	Access Types	Row Level Filter	
public	Select User	select	loc_state in 'CA'	x
Select Group	ambari-qa	select	loc_state = 'CA'	x
+	Select Group	admin	select	Add Row Filter +

Save Cancel Delete

5. Click **Add** to add the new row-level filter policy.

3.2.5.2. Dynamic Resource-Based Column Masking in Hive with Ranger Policies

You can use Apache Ranger dynamic resource-based column masking capabilities to protect sensitive data in Hive in near real-time. You can set policies that mask or anonymize sensitive data columns (such as PII, PCI, and PHI) dynamically from Hive query output. For example, you can mask sensitive data within a column to show only the first or last four characters.

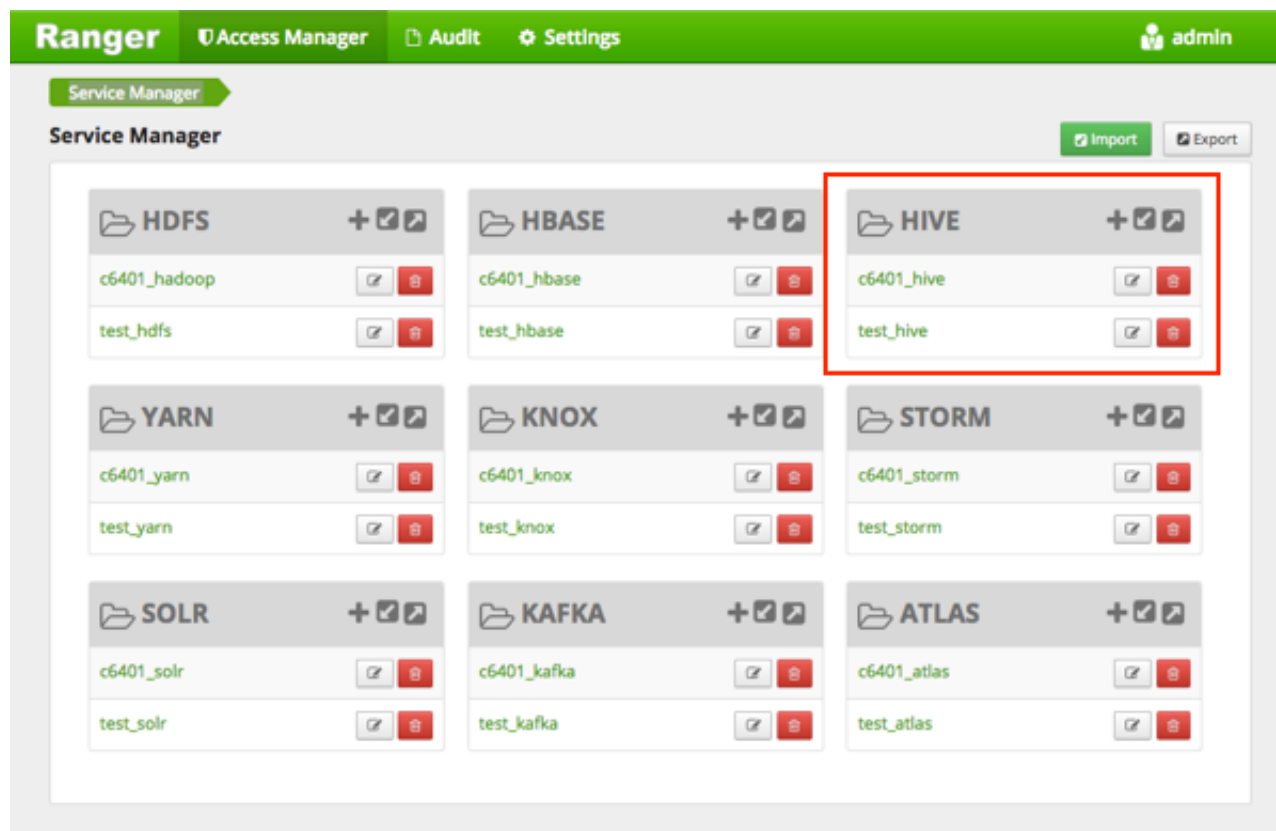
Dynamic column masking policies are similar to other Ranger access policies for Hive. You can set filters for specific users, groups, and conditions. With dynamic column-level masking, sensitive information never leaves Hive, and no changes are required at the consuming application or the Hive layer. There is also no need to produce additional protected duplicate versions of datasets.

The following conditions apply when using Ranger column masking policies to mask data returned in Hive query results:

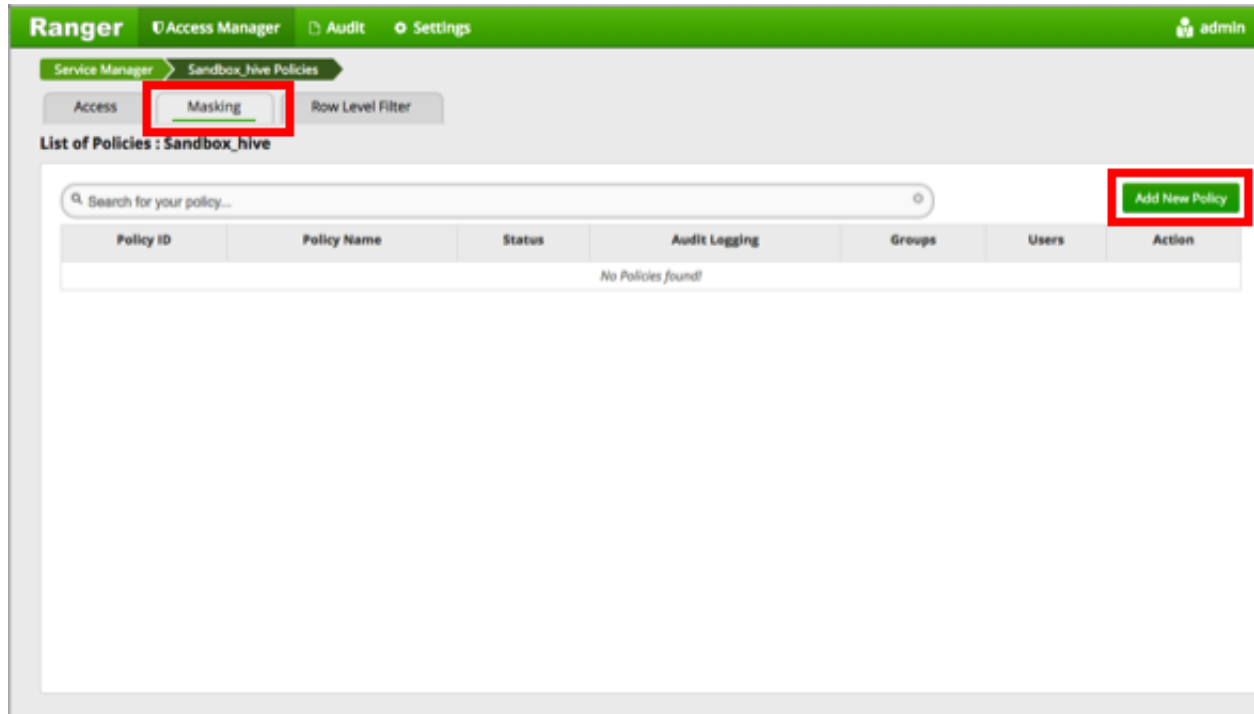
- A variety of masking types are available, such as show last 4 characters, show first 4 characters, Hash, Nullify, and date masks (show only year).
- You can specify a masking type for specific users, groups, and conditions.
- Wildcard matching is not supported.
- Each column should have its own masking policy.
- Masks are evaluated in the order listed in the policy.
- An audit log entry is generated each time a masking policy is applied to a column .

Use the following steps to create a masking policy:

1. On the Service Manager page, select an existing Hive Service.



2. Select the Masking tab, then click **Add New Policy**.



3. On the Create Policy page, add the following information for the column-masking filter:

Table 3.64. Policy Details

Field	Description
Policy Name (required)	Enter an appropriate policy name. This name cannot be duplicated across the system. The policy is enabled by default.
Hive Database (required)	Type in the applicable database name. The auto-complete feature displays available databases based on the entered text.
Hive Table (required)	Type in the applicable table name. The auto-complete feature displays available tables based on the entered text.
Hive Column (required)	Type in the applicable column name. The auto-complete feature displays available columns based on the entered text.
Audit Logging	Audit Logging is set to Yes by default. Select No to turn off audit logging.
Description	Enter an optional description for the policy.

Table 3.65. Mask Conditions

Label	Description
Select Group	Specify the groups to which this policy applies. The public group contains all users, so granting access to the public group grants access to all users.
Select User	Specify one or more users to which this policy applies.
Access Types	Currently select is the only available access type.

Label	Description
Select Masking Type	<p>To create a row filter for the specified users and groups, click Select Masking Option, then select a masking type:</p> <ul style="list-style-type: none"> • Redact – mask all alphabetic characters with "x" and all numeric characters with "n". • Partial mask: show last 4 – Show only the last four characters. • Partial mask: show first 4 – Show only the first four characters. • Hash – Replace all characters with a hash of entire cell value. • Nullify – Replace all characters with a NULL value. • Unmasked (retain original value) – No masking is applied. • Date: show only year – Show only the year portion of a date string and default the month and day to 01/01 • Custom – Specify a custom masked value or expression. Custom masking can use any valid Hive UDF (Hive that returns the same data type as the data type in the column being masked). <p>Masking conditions are evaluated in the order listed in the policy. The condition at the top of the Masking Conditions list is applied first, then the second, then the third, and so on.</p>

Policy Details :

Policy Type: **Masking**

Policy Name: maskchr.employee.ssn **enabled**

Hive Database: hr

Hive Table: employee

Hive Column: ssn

Audit Logging: **YES**

Description: Masking for ssn column in hr.employee table

Mask Conditions :

Select Group	Select User	Access Type	Masking Option
Select Group	admin	select	Unmasked (retain original value)
Select Group	ambari-qa	select	Partial mask: show last 4
public	Select User	select	Nullify

Add **Cancel**

- To move a condition in the Mask Conditions list (and therefore change the order in which it is evaluated), click the dotted rows icon at the left of the condition row, then drag the condition to a new position in the list.

Ranger Access Manager Audit Settings admin

Service Manager > Hive_service_1 Policies > Create Policy

Create Policy

Policy Details :

Policy Type: **Masking**

Policy Name: maskhr.employee.ssn **enabled**

Hive Database: hr

Hive Table: employee

Hive Column: ssn

Audit Logging: **YES**

Description: Masking for ssn column in hr.employee table

Mask Conditions :

Select Group	Select User	Access Types	Select Masking Option
Select Group	admin	select	Unmasked (retain original value)
Select Group	ambari-ga	select	Partial mask: show last 4
public	Select User	select	Nullify

Add **Cancel**

5. Click **Add** to add the new column masking filter policy.

3.2.5.3. Dynamic Tag-Based Column Masking in Hive with Ranger Policies

Where Ranger **resource-based** masking policy for Hive anonymizes data from a Hive column identified by the database, table, and column, **tag-based** masking policy anonymizes Hive column data based on tags and tag attribute values associated with Hive column (usually specified as metadata classification in Atlas).

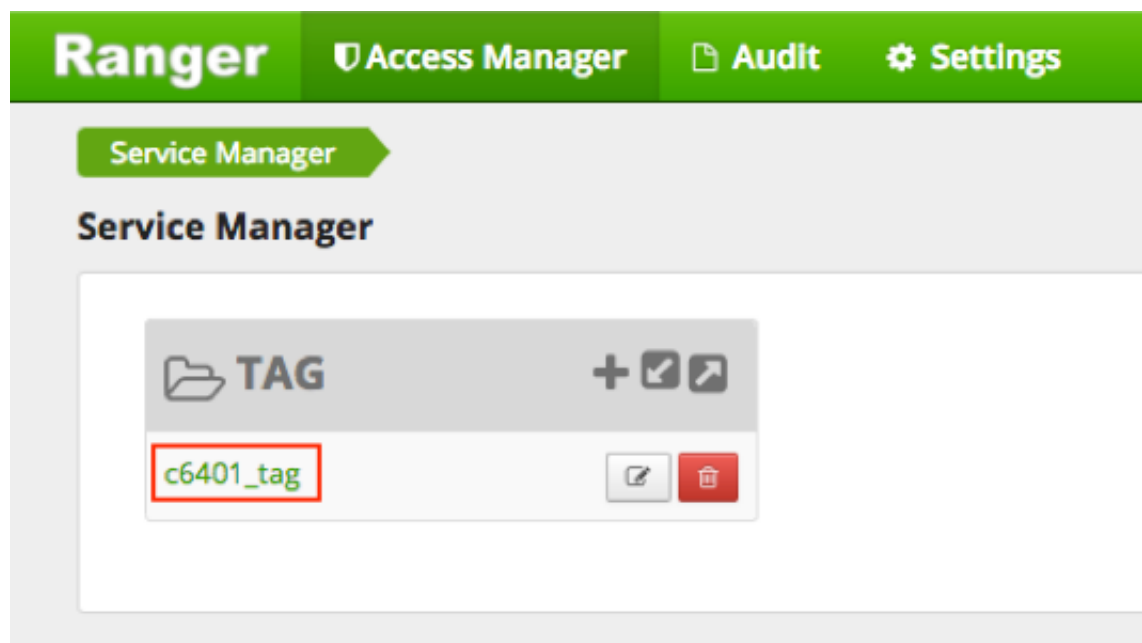
The following conditions apply when using Ranger column masking policies to mask data returned in Hive query results:

- A variety of masking types are available, such as show last 4 characters, show first 4 characters, Hash, Nullify, and date masks (show only year).
- You can specify a masking type for specific users, groups, and conditions.

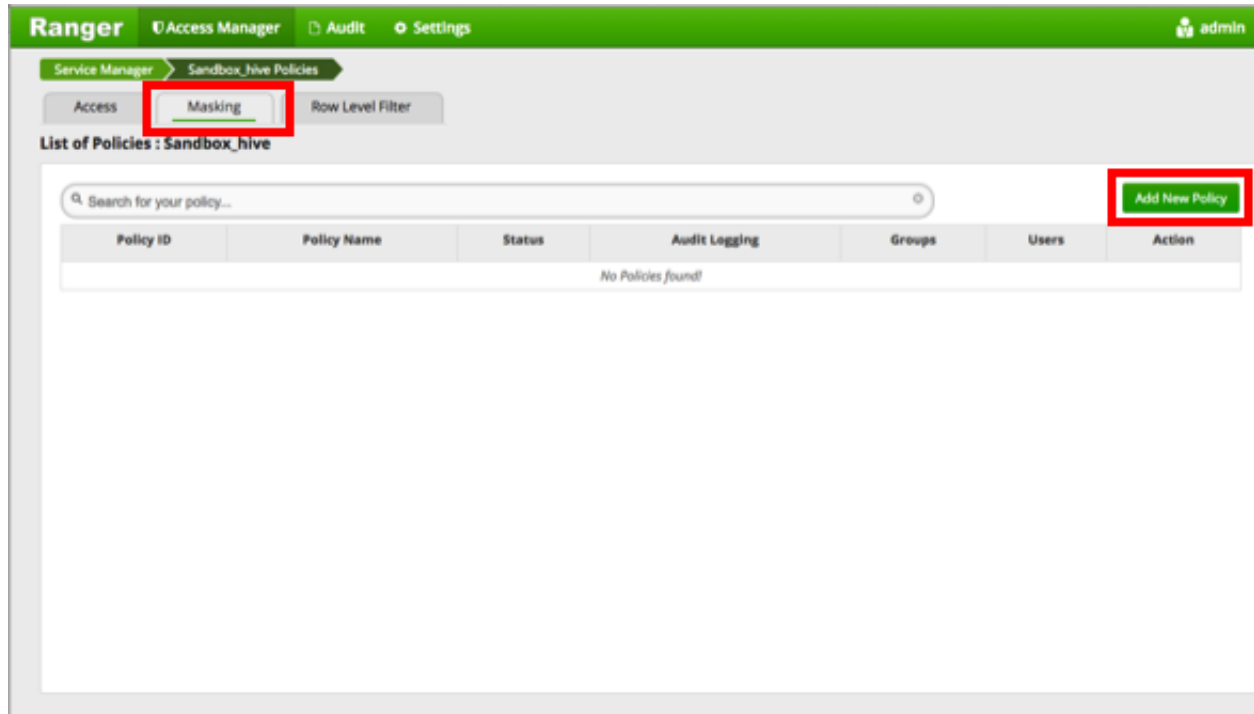
- Wildcard matching is not supported.
- If there are multiple tag masking policies applied to the same Hive column, the masking policy with the lexicographically smallest policy-name is chosen for enforcement, E.G., policy "a" is enforced before policy "aa".
- Masks are evaluated in the order listed in the policy.
- An audit log entry is generated each time a masking policy is applied to a column.

To add a new tag-based Hive column masking policy:

1. Select **Access Manager > Tag Based Policies**, then select a tag-based service.



2. Select the **Masking** tab, then click **Add New Policy**.



3. On the **Create Policy** page, add the following information for the column-masking filter:

Table 3.66. Policy Details

Field	Description
Policy Type (required)	Set to Hive by default.
Policy Name (required)	Enter an appropriate policy name. This name cannot be duplicated across the system. The policy is enabled by default.
TAG (required)	Enter the applicable tag name, E.G., MASK .
Audit Logging	Audit Logging is set to Yes by default. Select No to turn off audit logging.
Description	Enter an optional description for the policy.

Table 3.67. Mask Conditions

Label	Description
Select Group	Specify the groups to which this policy applies. The public group contains all users, so granting access to the public group grants access to all users.
Select User	Specify one or more users to which this policy applies.
Policy Conditions	Click Add Conditions to add or edit policy conditions. Currently "Accessed after expiry_date? (yes/no)" is the only available policy condition. To set this condition, type yes in the text box, then select the green check mark button to add the condition.

Label	Description
Access Types	Currently hive and select are the only available access types.
Select Masking Option	<p>To create a row filter for the specified users and groups, click Select Masking Option, then select a masking type:</p> <ul style="list-style-type: none"> • Redact – mask all alphabetic characters with "x" and all numeric characters with "n". • Partial mask: show last 4 – Show only the last four characters. • Partial mask: show first 4 – Show only the first four characters. • Hash – Replace all characters with a hash of entire cell value. • Nullify – Replace all characters with a NULL value. • Unmasked (retain original value) – No masking is applied. • Date: show only year – Show only the year portion of a date string and default the month and day to 01/01 • Custom – Specify a custom masked value or expression. Custom masking can use any valid Hive UDF (Hive that returns the same data type as the data type in the column being masked). <p>Masking conditions are evaluated in the order listed in the policy. The condition at the top of the Masking Conditions list is applied first, then the second, then the third, and so on.</p>

Ranger Access Manager Audit Settings admin

Service Manager Hive-tags Policies Edit Policy

Edit Policy

Please ensure that users/groups listed in this policy have access to the tag via an Access Policy. This policy does not implicitly grant access to the tag.

Policy Details :

Policy Type: **Masking**

Policy ID: **21**

Policy Name *: masking policy **enabled**

TAG *: MASK

Audit Logging: **YES**

Description: Mask our Ranger resources marked with tag "MASK" for user

Mask Conditions :

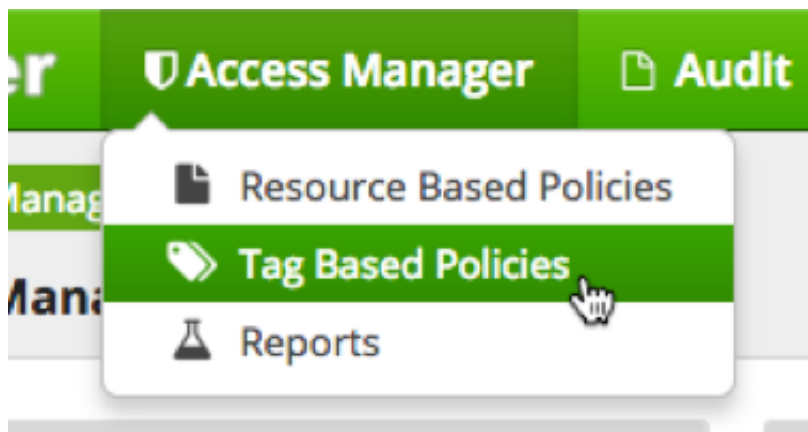
Select Group	Select User	Policy Conditions	Access Types
Select Group	hive	Add Conditions	HIVE
			HIVE : Unmasked (retain original value)

Save Cancel Delete

- You can use the Plus (+) symbols to add additional conditions. Conditions are evaluated in the order listed in the policy. The condition at the top of the list is applied first, then the second, then the third, and so on.
- Click **Add** to add the new policy.

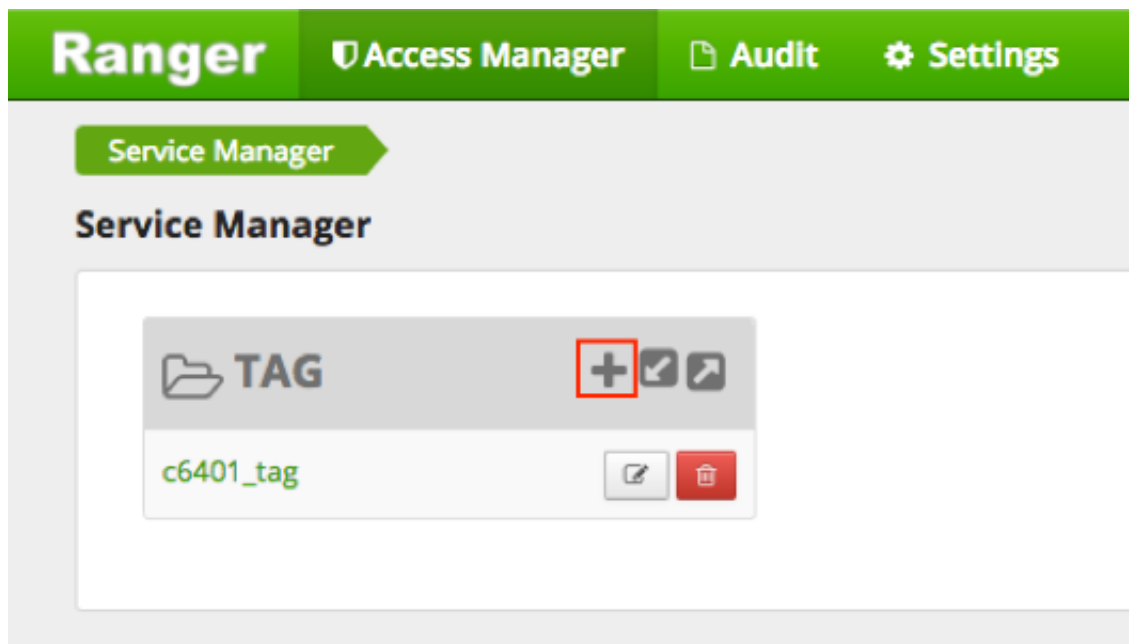
3.2.6. Adding Tag-based Service

You can access the Service Manager for Tag-Based Policies page by selecting **Access Manager > Tag Based Policies**. You can use this page to create tag-based services and add tag-based access policies that can be applied to Hadoop resources. Using tag-based policies enables you to control access to resources across multiple Hadoop components without creating separate services and policies in each component. You can also use Ranger TagSync to synchronize the Ranger tag store with an external metadata service such as Apache Atlas.

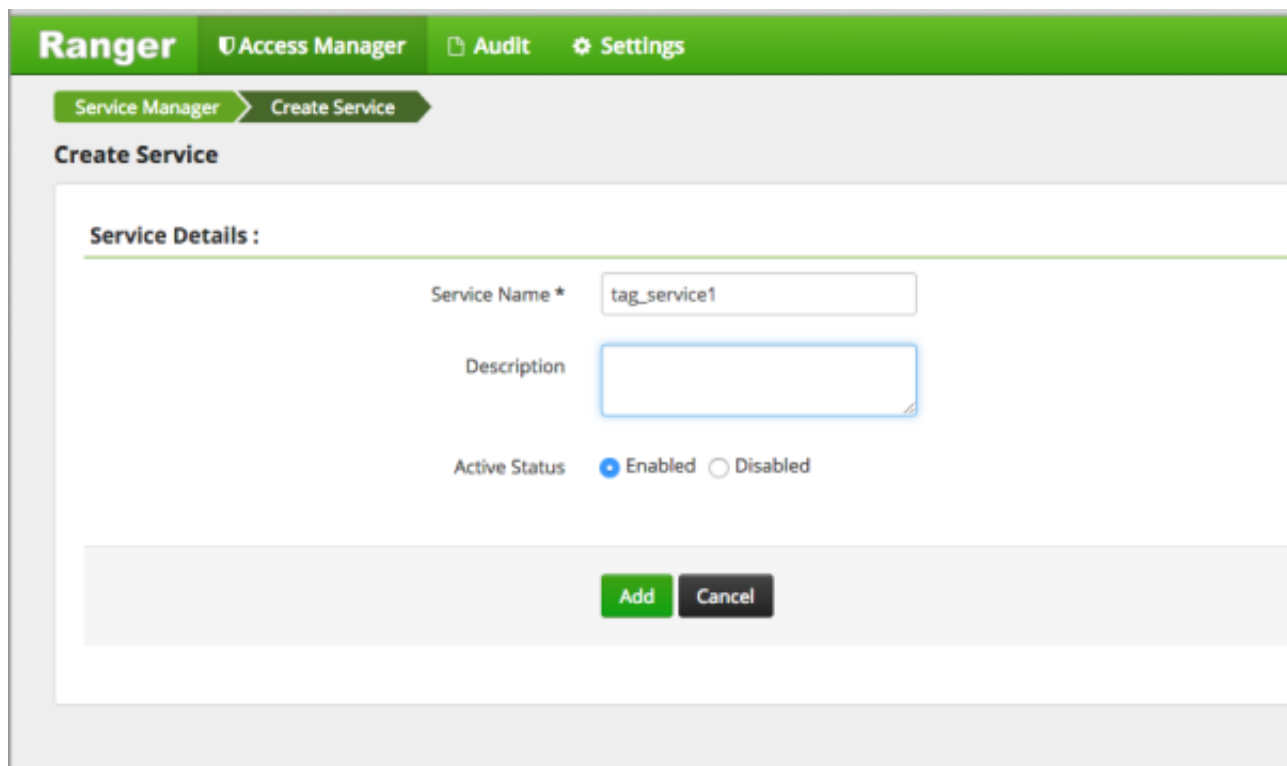


To add a new tag-based service:

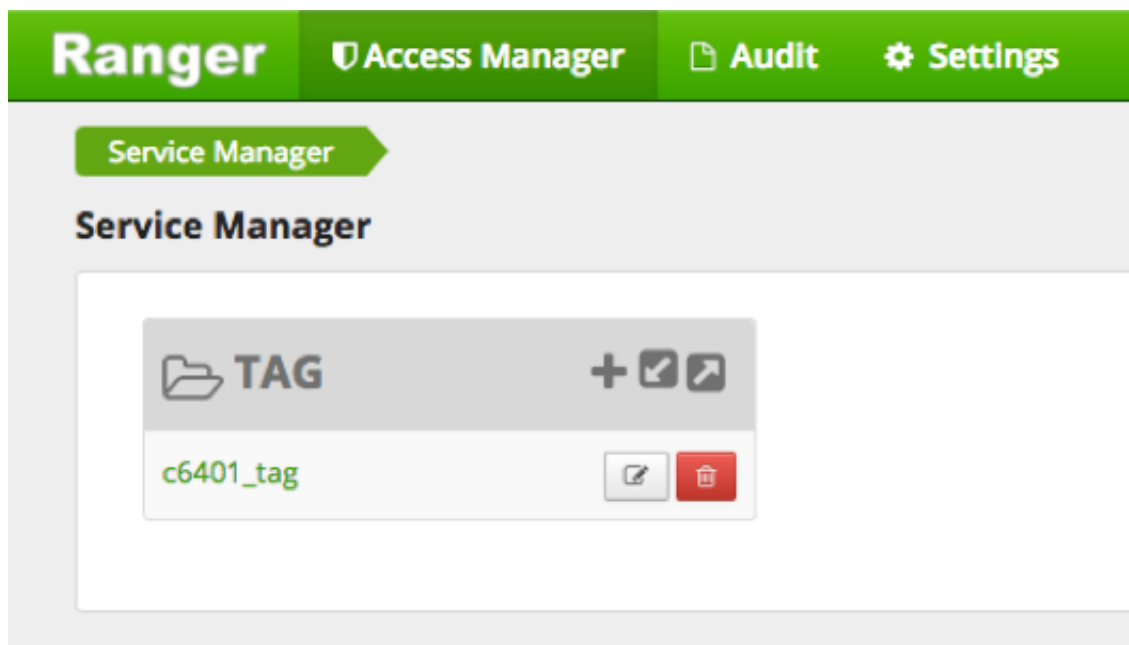
1. Click the Add icon (+) in the TAG box on the Service Manager page.



2. On the Service Details page, type in a service name and an optional description. The service is enabled by default, but you can disable it by selecting **Disabled**. To add the service, click **Add**.



3. The new tag service appears on the Service Manager page.



3.2.7. Tag-Based Policy Management

This section describes how to configure tag-based policies:

- [Adding Tag-Based Policies \[351\]](#)

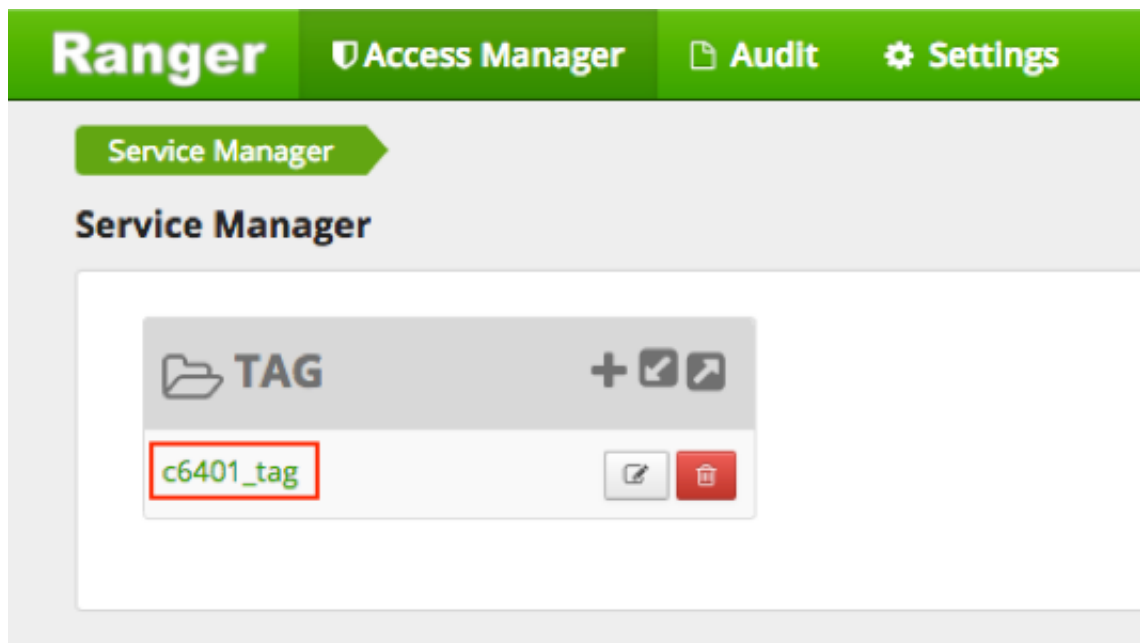
- [Adding a Tag-based PII Policy \[356\]](#)
- [Default EXPIRES_ON Policy \[360\]](#)
- [Importing and Exporting Tag-Based Policies \[362\]](#)
- [Import Tag-Based Policies \[364\]](#)
- [Export Tag-Based Policies \[368\]](#)

3.2.7.1. Adding Tag-Based Policies

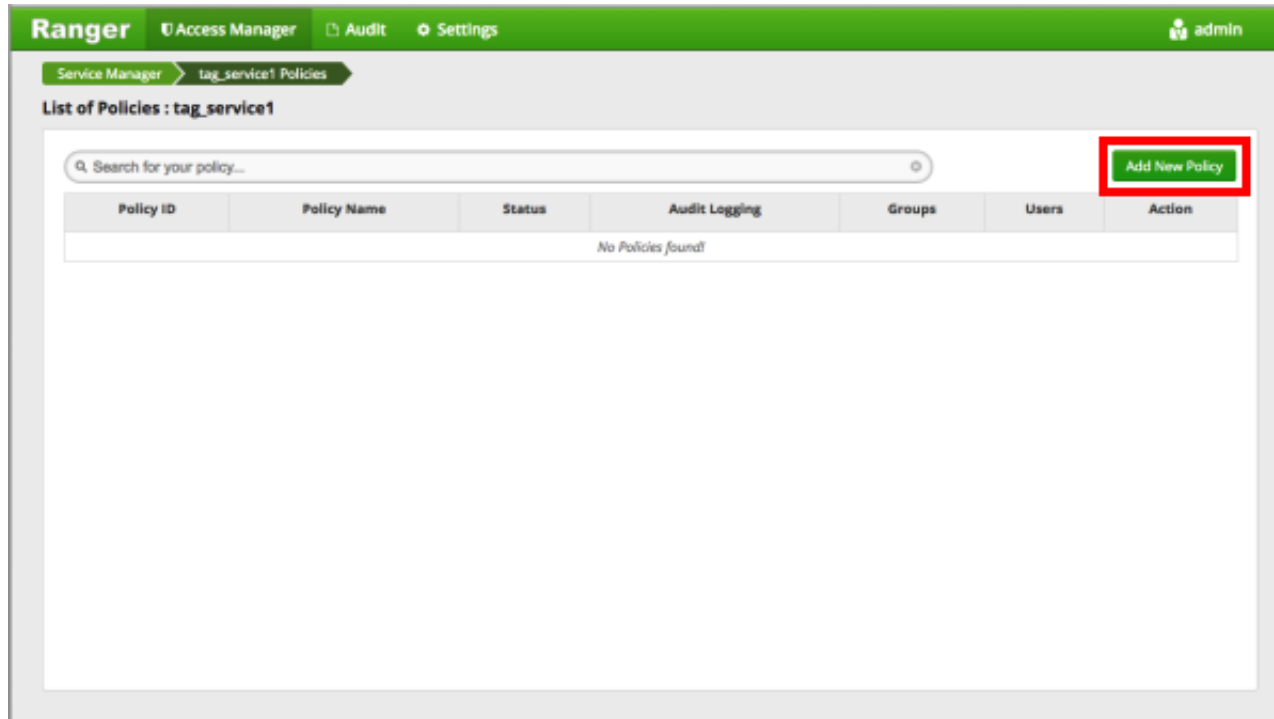
Tag-based policies enable you to control access to resources across multiple Hadoop components without creating separate services and policies in each component. You can also use Ranger TagSync to synchronize the Ranger tag store with an external metadata service such as Apache Atlas.

To add a new tag-based policy:

1. Select **Access Manager > Tag Based Policies**, then select a tag-based service.



2. On the List of Policies page, click **Add New Policy**.



The Create Policy page appears:

3. Enter information on the Create Policy page as follows:

Table 3.68. Policy Details

Field	Description
Policy Type	Set to Access by default.
Policy Name	Enter an appropriate policy name. This name cannot be duplicated across the system. This field is mandatory.
TAG	Enter the applicable tag name.
Description	(Optional) Describe the purpose of the policy.

Field	Description
Audit Logging	Specify whether this policy is audited. (De-select to disable auditing).

Table 3.69. Allow, Exclude from Allow, Deny, and Exclude from Deny Conditions

Label	Description
Select Group	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). The public group contains all users, so setting a condition for the public group applies to all users.
Select User	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).
Policy Conditions	Click Add Conditions to add or edit policy conditions. " Accessed after expiry_date (yes/no)? ": To set this condition, type yes in the text box, then select the green check mark button to add the condition. Enter boolean expression: Available for allow or deny conditions on tag-based policies. For examples and details, see Using Tag Attributes and Values in Ranger Tag-based Policy Conditions [354] .
Component Permissions	Click Add Permissions to add or edit component conditions. To add component permissions, enter the component name in the text box, then use the check boxes to specify component permissions. Select the green check mark button to add the chosen component conditions to the policy.



Important

Currently there is no Atlas hook for HBase, HDFS, or Kafka. For these components, you must [manually create entities in Atlas](#). You can then associate tags with these entities and control access using Ranger tag-based policies.

If **Deny Conditions** does not appear on your **Policy Details** page, you must first [Enable Deny Conditions for Policies](#).

4. You can use the Plus (+) symbols to add additional conditions. Conditions are evaluated in the order listed in the policy. The condition at the top of the list is applied first, then the second, then the third, and so on.
5. Click **Add** to add the new policy.

3.2.7.1.1. Using Tag Attributes and Values in Ranger Tag-based Policy Conditions

Enter boolean expression allows Ranger to use tag attributes and values when configuring tag-based policy Allow or Deny conditions. It allows admins to provide boolean expression(s) using tag attributes.

The policy condition is introduced in the tag service definition:

```
{
  "itemId":2,
  "name":"expression",
  "evaluator": "org.apache.ranger.plugin.conditionevaluator.
RangerScriptConditionEvaluator",
  "evaluatorOptions" : {"engineName":"JavaScript", "ui.
isMultiline":"true"},
  "label":"Enter boolean expression",
  "description": "Boolean expression"
}
```

The following variables can be referenced in the boolean expression:

- `ctx`: Context handler containing APIs to access metadata information from the request.
- `tag`: Information about the current tag.
- `tagAttr`: Map containing all the current tag attributes and corresponding values.

The following APIs available from the request:

- `getUser()`: Returns a string.
- `getUserGroups()`: Returns a set of strings containing groups.
- `getClientIPAddress()`: Returns a string containing client IP address.
- `getAction()`: Returns a string containing information about the action being requested.

Example

For two scenarios:

- User “sam” needs to be denied a policy based on the IP address of the machine from where the resources are accessed.

Set the deny condition for user `sam` with the following boolean expression:

```
if ( tagAttr.get('ipAddr').equals(ctx.getClientIPAddress()) ) {
  ctx.result = true;
}
```

- Deny one particular user, “bob” from a group, “users”, only when this user is accessing resources from a particular IP defined as an tag attribute in Atlas.

Set the deny condition for group `users` with the following boolean expression:

```
if (tagAttr.get('ipAddr').equals(ctx.getClientIPAddress()) && ctx.getUser().
equals("bob")) {
  ctx.result=true;
}
```

Deny Conditions:

Select Group	Select User	Policy Conditions	Component Permissions
Select Group	x sam	expression: JavaScript Condition	deny
x users x bob	Select User	expression: JavaScript Condition	deny

JavaScript Expression 1: `(tagAttr.get('ipAddr').equals(ctx.getClientIPAddress())) { ctx.result = true;}`

JavaScript Expression 2: `(tagAttr.get('ipAddr').equals(ctx.getClientIPAddress()) && ctx.getUser().equals('bob')) { ctx.result=true;}`

3.2.7.1.2. Adding a Tag-based PII Policy

In this example we create a tag-based policy for objects tagged "PII" in Atlas. Access to objects tagged "PII" is allowed for members of the "audit" group. All other users (the "public" group) are denied access.

To add a PII tag-based policy:

1. Select **Access Manager** > **Tag Based Policies**, then select a tag-based service.

Ranger Access Manager Audit Settings

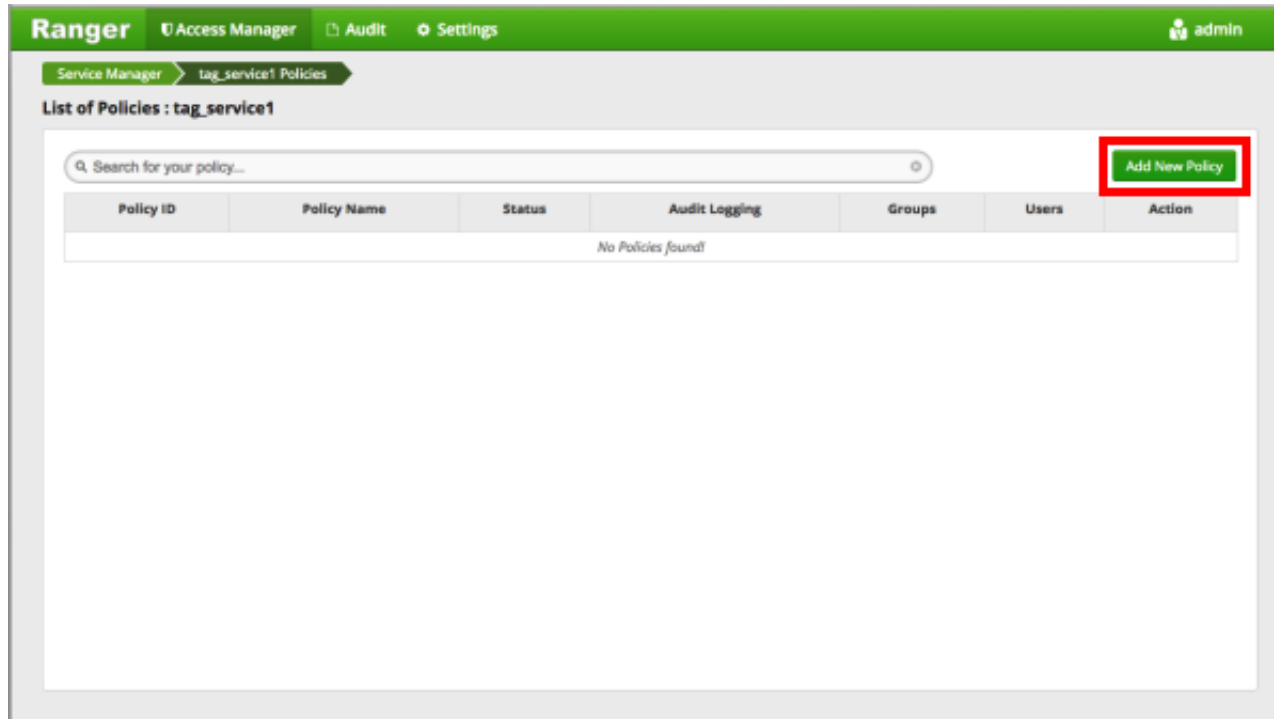
Service Manager

Service Manager

TAG

c6401_tag

2. On the List of Policies page, click **Add New Policy**.



The Create Policy page appears:

Ranger | Access Manager | Audit | Settings | admin

Service Manager > tag_service1 Policies > Create Policy

Create Policy

Policy Details :

Policy Type: **Access**

Policy Name * **enabled**

TAG *

Description

Audit Logging: **YES**

Allow Conditions : show ▾

Select Group	Select User	Policy Conditions	Component Permissions
<input type="text" value="Select Group"/>	<input type="text" value="Select User"/>	Add Conditions +	Add Permissions +

Exclude from Allow Conditions : show ▾

Deny Conditions : hide ▾

Select Group	Select User	Policy Conditions	Component Permissions
<input type="text" value="Select Group"/>	<input type="text" value="Select User"/>	Add Conditions +	Add Permissions +

Exclude from Deny Conditions : show ▾

Add **Cancel**

3. Enter the following information on the Create Policy page:

Table 3.70. Policy Details

Field	Description
Policy Type	Set to Access by default.
Policy Name	PII
TAG	PII
Audit Logging	YES
Description	Restrict access to resources with the PII tag.

Table 3.71. Allow Conditions

Label	Description
Select Group	audit
Select User	<none>
Policy Conditions	<none>
Component Permissions	hive (select all permissions)

Table 3.72. Deny Conditions

Label	Description
Select Group	public
Select User	<none>
Policy Conditions	<none>
Component Permissions	hive (select all permissions)

If **Deny Conditions** does not appear on your **Policy Details** page, you must first [Enable Deny Conditions for Policies](#).

Table 3.73. Exclude from Allow Conditions

Label	Description
Select Group	audit
Select User	<none>
Policy Conditions	<none>
Component Permissions	hive (select all permissions)

The screenshot displays the Ranger Access Manager configuration page for a policy named 'PII'. The policy is of type 'Access', is enabled, and has a TAG of 'PII'. Audit logging is set to 'YES'. The description is 'Restrict access to resources with the PII tag.' Below the description, there are sections for 'Allow Conditions' and 'Deny Conditions'. The 'Allow Conditions' section contains one condition for the 'audit' group. The 'Deny Conditions' section contains one condition for the 'public' group. An 'Exclude from Deny Conditions' section also lists the 'audit' group. At the bottom, there are 'Add' and 'Cancel' buttons.

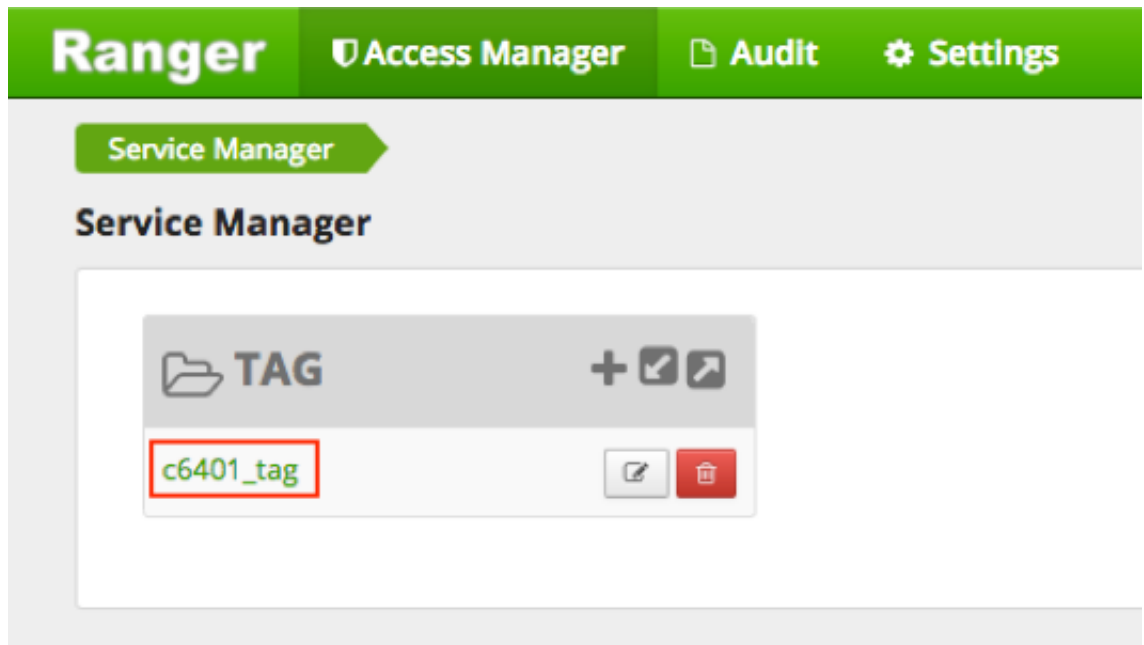
In this example we used Allow Conditions to grant access to the "audit" group, and then used Deny Conditions to deny access to the "public" group. Because the "public" group includes all users, we then used Exclude from Deny Conditions to exclude the "audit" group, in effect reinstating the "audit" group's original Allow access condition.

4. Click **Add** to add the new policy.

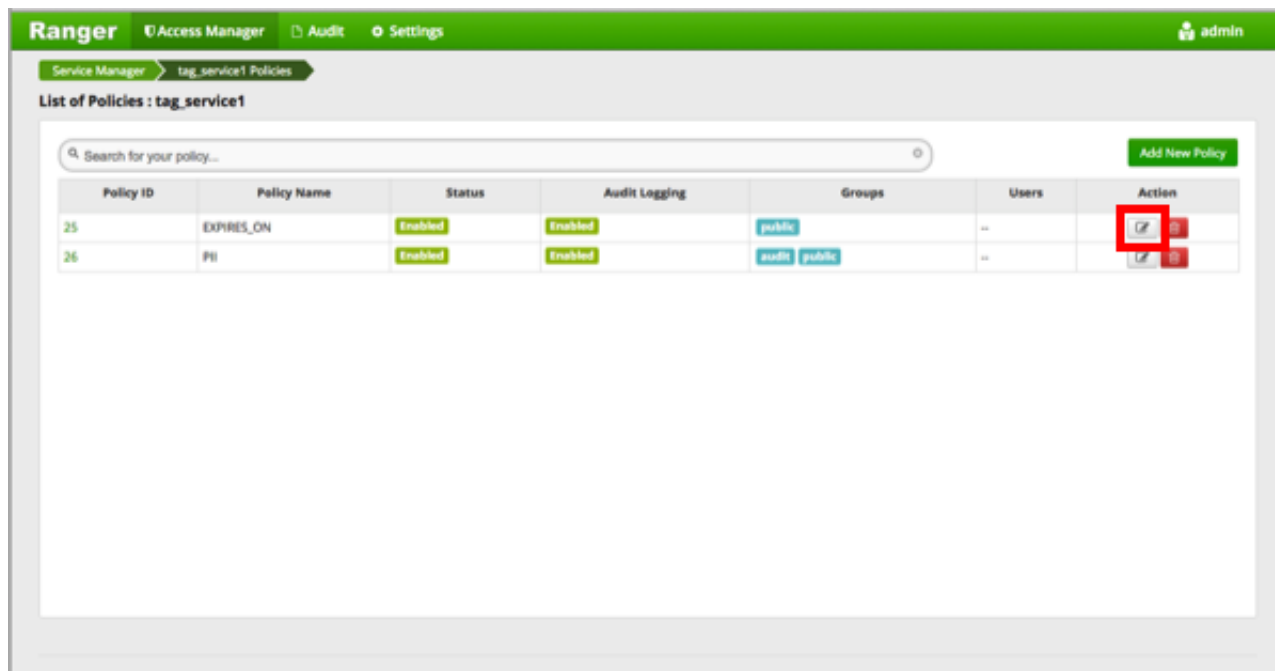
3.2.7.1.3. Default EXPIRES_ON Policy

An EXPIRES_ON tag-based policy is created automatically when a tag service instance created. This default policy denies access to objects tagged with EXPIRES_ON after the expiry date specified in the Atlas tag attribute. You can use the following steps to review the default EXPIRES_ON policy.

1. Select **Access Manager > Tag Based Policies**, then select a tag-based service.



2. On the List of Policies page, click the Edit icon for the default EXIRES_ON policy.



The Edit Policy page appears:

Ranger Access Manager Audit Settings admin

Edit Policy

Policy Details :

Policy Type: **Access**

Policy ID: **25**

Policy Name: **enabled**

TAG:

Audit Logging: **YES**

Description:

Allow Conditions :

Select Group	Select User	Policy Conditions	Component Permissions
<input type="text" value="Select Group"/>	<input type="text" value="Select User"/>	Add Conditions +	Add Permissions +

Exclude from Allow Conditions: show +

Deny Conditions :

Select Group	Select User	Policy Conditions	Component Permissions
<input type="text" value="x public"/>	<input type="text" value="Select User"/>	accessed-after-expiry: yes	<input type="checkbox"/> HDFS <input type="checkbox"/> HBASE <input type="checkbox"/> HIVE <input type="checkbox"/> HMS <input type="checkbox"/> KNOX <input type="checkbox"/> STORM <input type="checkbox"/> YARN <input type="checkbox"/> KAFKA <input type="checkbox"/> SOLR <input type="checkbox"/> ATLAS

Exclude from Deny Conditions: show +

Save **Cancel** **Delete**

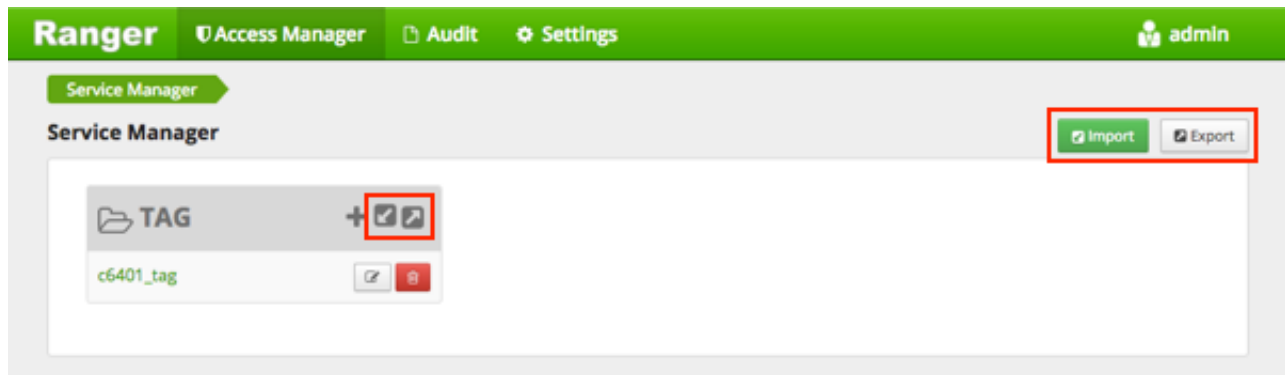
- We can see that the default EXPIRES_ON policy denies access to all users, and for all components, after the expiry date specified in the Atlas tag attribute.

3.2.7.2. Importing and Exporting Tag-Based Policies

You can export and import policies from the Ranger Admin UI for cluster resiliency (backups), during recovery operations, or when moving policies from test clusters to production clusters. You can export/import a specific subset of policies (such as those that pertain to specific resources or user/groups) or clone the entire repository (or multiple repositories) via Ranger Admin UI.

Interfaces

You can import and export policies from the Access Manager>Tag Based Policies page:



You can also export policies from the Reports page:

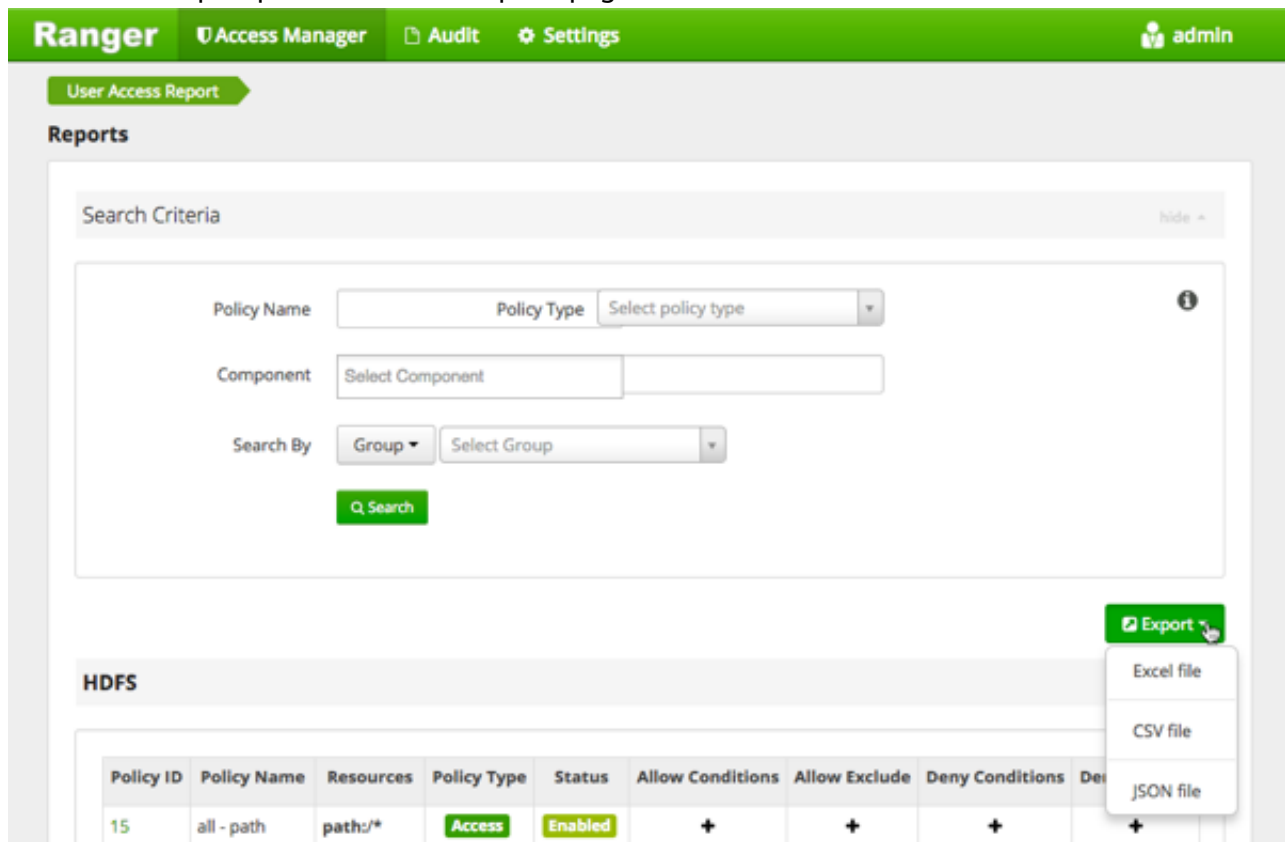


Table 3.74. Export Policy Options

	Access Manager Page	Reports Page
Formats	JSON	JSON Excel CSV
Filtering Supported	No	Yes
Specific Service Export	Yes	Via filtering

Filtering

When exporting from the Reports page, you can apply filters before saving the file.

Export Formats

You can export policies in the following formats:

- Excel
- JSON
- CSV

Note: CSV format is not supported for importing policies.

When you export policies from the Access Manager page, the policies are automatically downloaded in JSON format. If you wish to export in Excel or CSV format, export the policies from the Reports page dropdown menu.

Required User Roles

The Ranger admin user can import and export only Resource & Tag based policies. The credentials for this user are set in Ranger **Configs > Advanced ranger-env** in the fields labeled **admin_username** (default: *admin/admin*).

The Ranger KMS keyadmin user can import and export only KMS policies. The default credentials for this user are *keyadmin/keyadmin*.

Limitations

To successfully import policies, use the following database versions:

- MariaDB: 10.1.16+
- MySQL: 5.6.x+
- Oracle: 11gR2+
- PostgreSQL: 8.4+
- MS SQL: 2008 R2+

Partial policy import is not supported.

See also: [Importing and Exporting Resource-Based Policies \[326\]](#)

3.2.7.2.1. Import Tag-Based Policies

To import tag-based policies:


- Via the Import icon:
 1. From the Access Manager>Tag Based Policies page, click the Import icon beside the service:



The Import Policy page opens.

Import Policy [Close]



Select File :


Select file  Override Policy :

No file chosen

Specify Service Mapping :

Source Destination

Enter service name To Select service name  



Cancel **Import**

2. Select the file to import.

You can only import policies in JSON format.


3. (Optional) Configure the import operation:

- The Override Policy option deletes all policies of the destination repositories.
- Service Mapping maps the downloaded file repository, i.e. source repository to destination repository.

For example:

Import Policy

Select File :

Select file  Override Policy :

Ranger_Policies_20170209_192920.json **x**

Specify Service Mapping :

Source Destination

tagdev1 To tag1 **x**

+

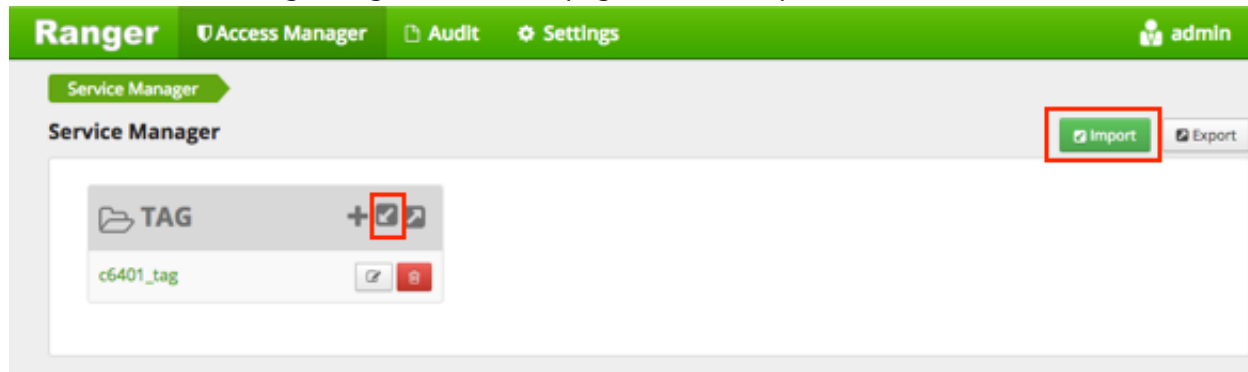
Cancel **Import**

4. Click **Import**.

A confirmation message appears: "Success: File import successfully."

• Via the Import button:

1. From the Access Manager>Tag Based Policies page, click the Import button:




The Import Policy page opens.

Import Policy ✕

Service Type :

✕ tag

Select File :

Select file  Override Policy :

No file chosen

Specify Service Mapping :

Source	To	Destination
<input type="text" value="Enter service name"/>	To	<input style="border: 1px solid gray; background-color: #f0f0f0; border-radius: 4px; padding: 2px 10px;" type="text" value="Select service name"/> ✕

+

Cancel Import

2. Select the file to import.

You can only import policies in JSON format.


3. (Optional) Configure the import operation:


- Service Types enables you to remove specific services from the import.
- The Override Policy option deletes all policies of the destination repositories.
- Service Mapping maps the downloaded file repository, i.e. source repository to destination repository.

For example:

Import Policy



Select File :


Select file  Override Policy :

Ranger_Policies_20170209_192920.json 

Specify Service Mapping :

Source Destination

tagdev1 To tag1  



Cancel **Import**

4. Click **Import**.

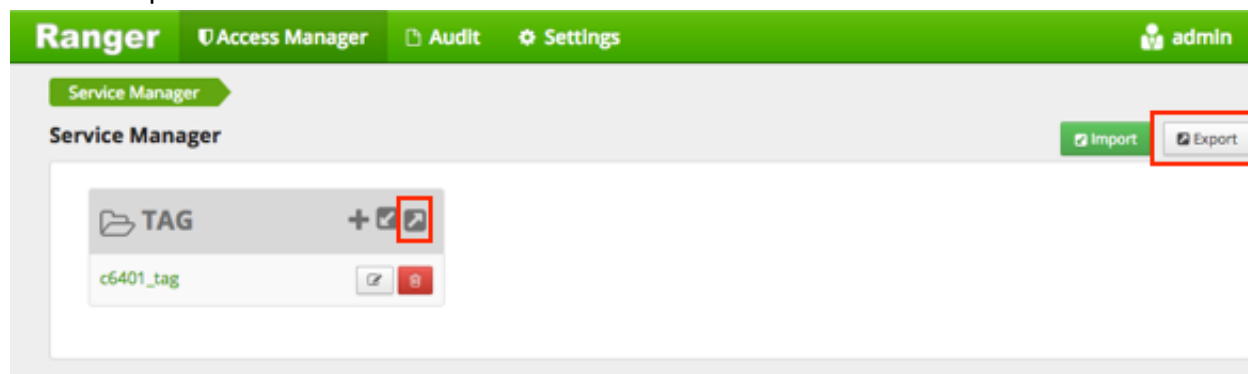
A confirmation message appears: "Success: File import successfully."

3.2.7.2.2. Export Tag-Based Policies

To export all tag-based policies:

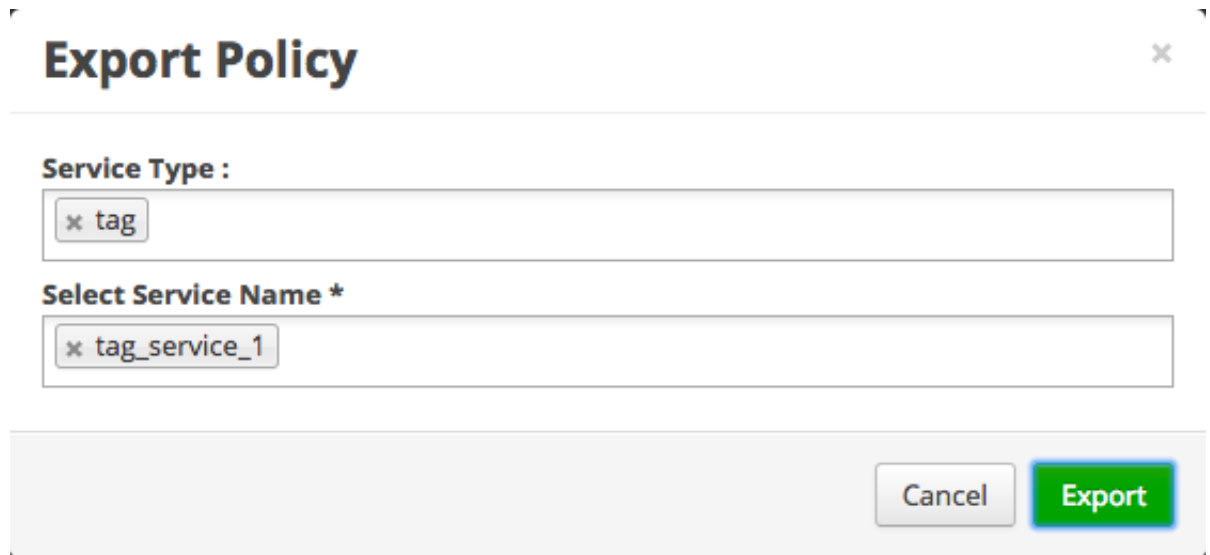
- From the Access Manager>Tag Based Policies page:

1. Click the Export button or icon:



The Export Policy page opens.

2. Remove components or specific services and click Export.



Export Policy

Service Type :
tag

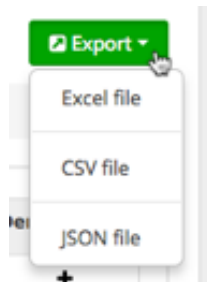
Select Service Name *:
tag_service_1

Cancel Export

3. The file downloads in your browser as a JSON file.

If you wish to export in Excel or CSV format, export the policies from the Reports page dropdown menu.

- From the Reports page:
 1. Filter **Component** to **tag** and click **Search**.
 2. (Optional) Apply filters before exporting file.
 3. Open the Export drop-down menu:



4. Select the file format.

The file downloads in your browser.

3.2.8. Users/Groups and Permissions Administration

To view the list of users and groups who can access the Ranger portal or its services, select **Settings > Users/Groups** in the top menu.

The Users/Groups page lists:

- Internal users who can log in to the Ranger portal; created by the Ranger console Service Manager.

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

The screenshot shows the Ranger interface with the 'Users/Groups' section selected. The 'Groups' tab is active, displaying a 'Group List' table. The table has columns for Group Name, Group Source, and Visibility. There are buttons for 'Add New Group', 'Set Visibility', and a trash icon.

Group Name	Group Source	Visibility
public	Internal	Visible
slider	External	Visible
hadoop	External	Visible
splash	External	Visible
root	External	Visible
...	External	Visible

3.2.8.1. Add a User

To add a new user to the user list:

1. Select **Settings > Users/Groups**.

The Users/Groups page appears.

The screenshot shows the Ranger interface with the 'Settings' menu open and 'Users/Groups' selected. The 'Users/Groups' page is displayed, showing a 'User List' table. The table has columns for User Name, Email Address, Role, User Source, Groups, and Visibility. There are buttons for 'Add New User', 'Set Visibility', and a trash icon.

User Name	Email Address	Role	User Source	Groups	Visibility
admin		Admin	Internal	admin, hadoop, hdfs	Visible
rangerusersync		Admin	Internal	--	Visible
rangertagsync		Admin	Internal	--	Visible
amb_ranger_admin		Admin	Internal	--	Visible
hadoop		User	External	--	Visible
ambari-qa		User	External	hadoop, users	Visible
slider		User	External	slider	Visible
hive		User	External	hadoop	Visible
zeppelin		User	External	hadoop	Visible
splash		User	External	splash, root	Visible

2. Click **Add New User**.

The User Detail page appears.

The screenshot shows the Ranger User Detail page. The header is green and contains the 'Ranger' logo, navigation links for 'Access Manager', 'Audit', and 'Settings', and a user profile icon labeled 'admin'. Below the header, there are breadcrumb links for 'Users/Groups' and 'User Create'. The main content area is titled 'User Detail' and contains a form with the following fields: 'User Name *', 'New Password *', 'Password Confirm *', 'First Name *', 'Last Name', 'Email Address', 'Select Role *' (with 'Admin' selected), and 'Group' (with 'Please select' and a '+' button). At the bottom of the form are 'Save' and 'Cancel' buttons.

3. Add the required user details, then click **Save**.

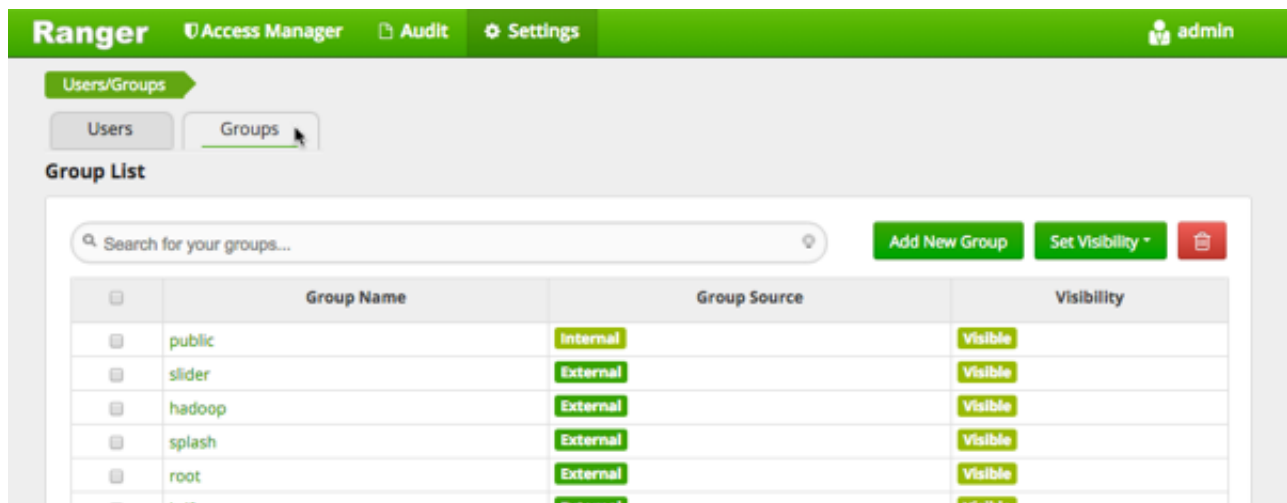
The user is immediately added to the list.

3.2.8.2. Edit a User

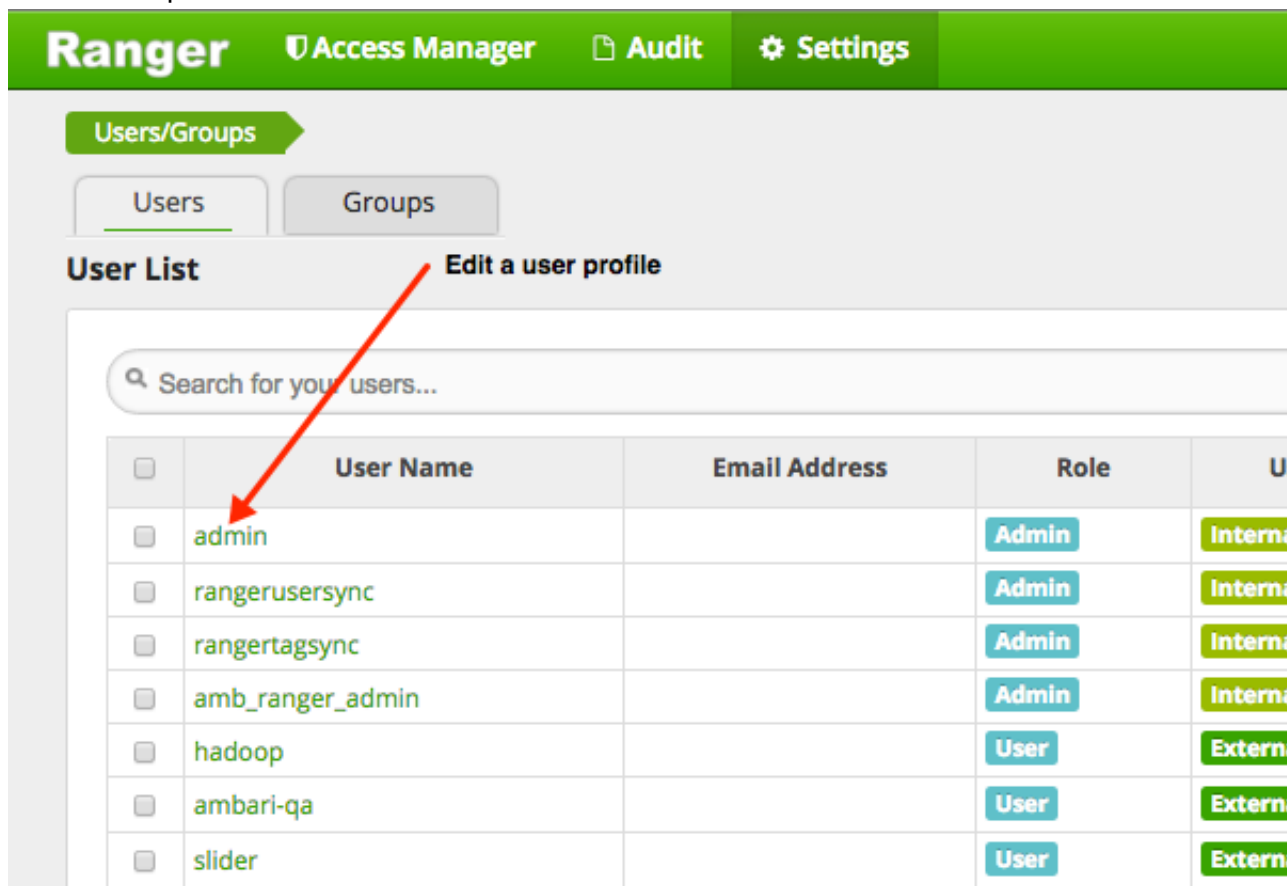
To edit a user:

1. Select **Settings > Users/Groups**.

The Users/Groups page opens to the Users tab.



2. Select a user profile to edit.



- The User Detail page appears.



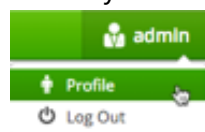
Note

You can only fully edit internal users. For external users, you can only edit the user role.

You can edit all properties except the User Name when editing an internal user.

You can only change the user role when editing an external user.

- To edit your own user profile, click *username*>Profile.



The User Profile page appears.

The screenshot shows the 'User Profile' page with the 'Basic Info' tab selected. The form contains the following fields:

- First Name *: Admin
- Last Name *: Last Name
- Email Address: Email Address
- Select Role *: Admin

At the bottom of the form, there are two buttons: 'Save' (highlighted in green) and 'Cancel'.

3. Edit the appropriate details, then click **Save**.

3.2.8.3. Delete a User


Only users with role "admin" may delete a user. To permanently delete a user:

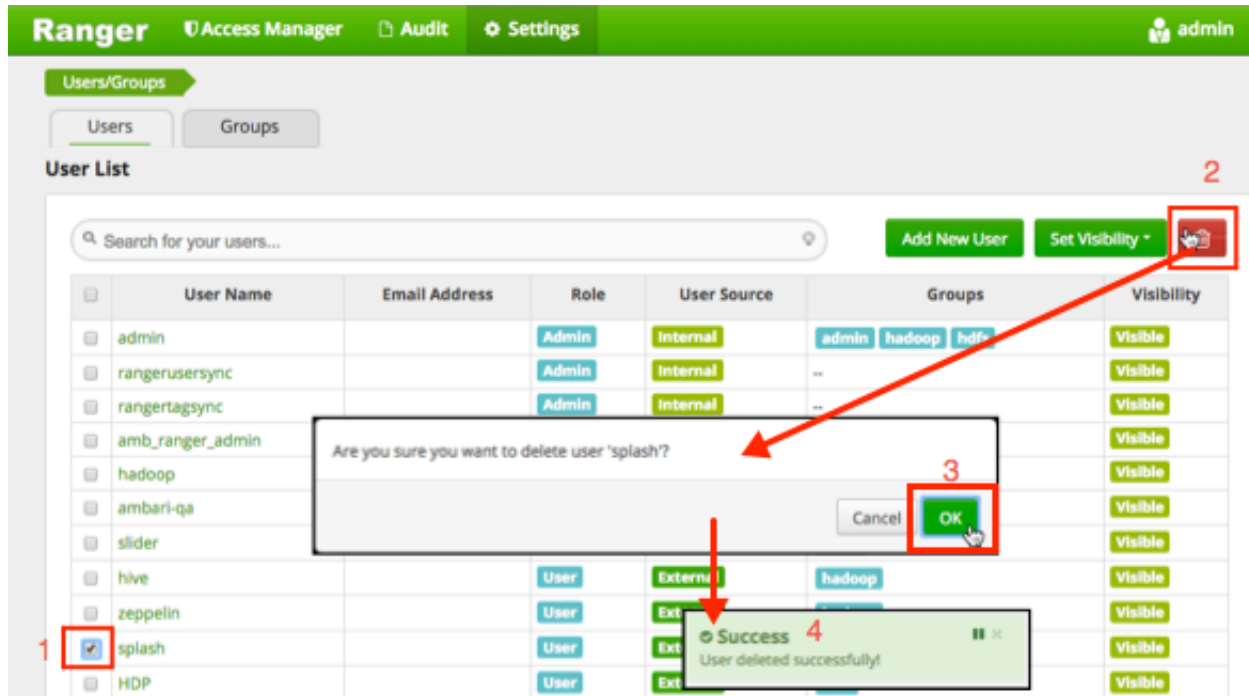
1. Select **Settings > Users/Groups**.

The Users/Groups page appears.

The screenshot shows the Ranger 'Users/Groups' page. The 'Settings' menu is open, and 'Users/Groups' is selected. Below the navigation, there are tabs for 'Users' and 'Groups'. The 'User List' section features a search bar and two buttons: 'Add New User' and 'Set Visibility'. The main content is a table with the following data:

	User Name	Email Address	Role	User Source	Groups	Visibility
<input type="checkbox"/>	admin		Admin	Internal	admin, hadoop, hdfs	Visible
<input type="checkbox"/>	rangerusersync		Admin	Internal	--	Visible
<input type="checkbox"/>	rangertagsync		Admin	Internal	--	Visible
<input type="checkbox"/>	amb_ranger_admin		Admin	Internal	--	Visible
<input type="checkbox"/>	hadoop		User	External	--	Visible
<input type="checkbox"/>	ambari-qa		User	External	hadoop, users	Visible
<input type="checkbox"/>	slider		User	External	slider	Visible
<input type="checkbox"/>	hive		User	External	hadoop	Visible
<input type="checkbox"/>	zeppelin		User	External	hadoop	Visible
<input type="checkbox"/>	splash		User	External	splash, root	Visible

2. Select the check box of the user you want to delete and click the Delete icon () at the right of the **User List** menu bar.



The screenshot shows the Ranger interface for managing users. The 'User List' table contains the following data:

	User Name	Email Address	Role	User Source	Groups	Visibility
<input type="checkbox"/>	admin		Admin	Internal	admin, hadoop, hdp	Visible
<input type="checkbox"/>	rangerusersync		Admin	Internal	--	Visible
<input type="checkbox"/>	rangertagsync		Admin	Internal	--	Visible
<input type="checkbox"/>	amb_ranger_admin					Visible
<input type="checkbox"/>	hadoop					Visible
<input type="checkbox"/>	ambari-qa					Visible
<input type="checkbox"/>	slider					Visible
<input type="checkbox"/>	hive		User	External	hadoop	Visible
<input type="checkbox"/>	zeppelin		User	Ext		Visible
<input checked="" type="checkbox"/>	splash		User	Ext		Visible
<input type="checkbox"/>	HDP		User	Ext		Visible

3. You are prompted to confirm the user deletion; select OK.

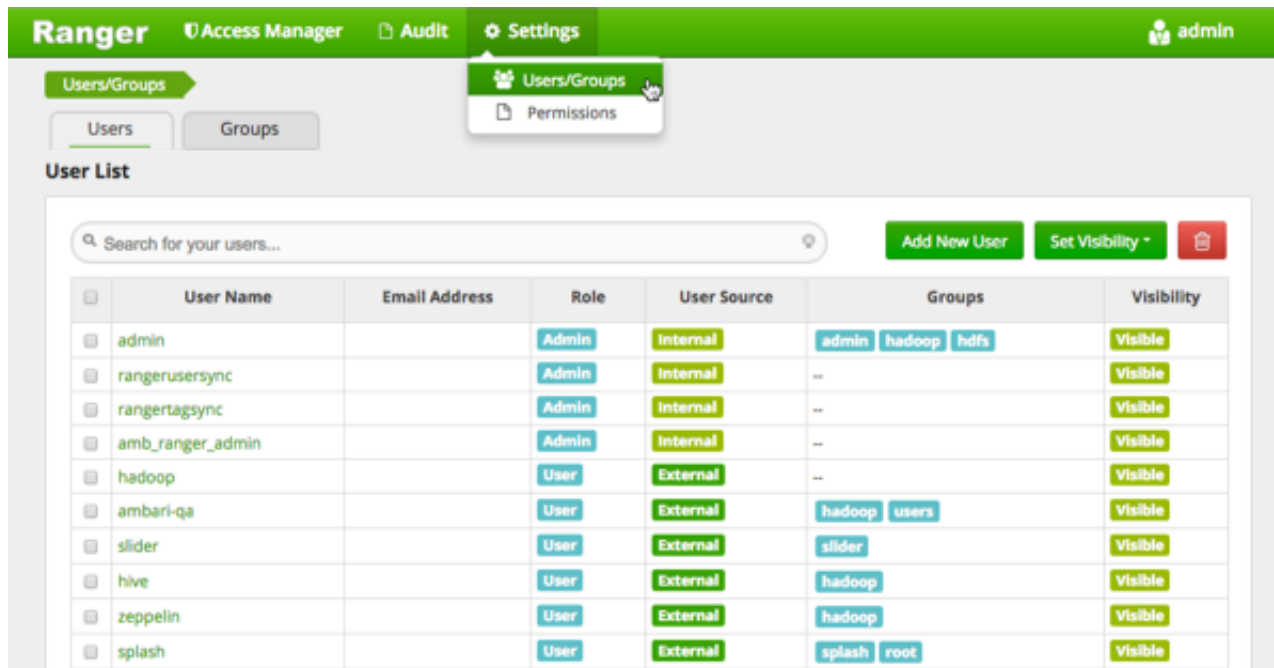
You receive confirmation that the operation has succeeded.

3.2.8.4. Add a Group

To add a group:

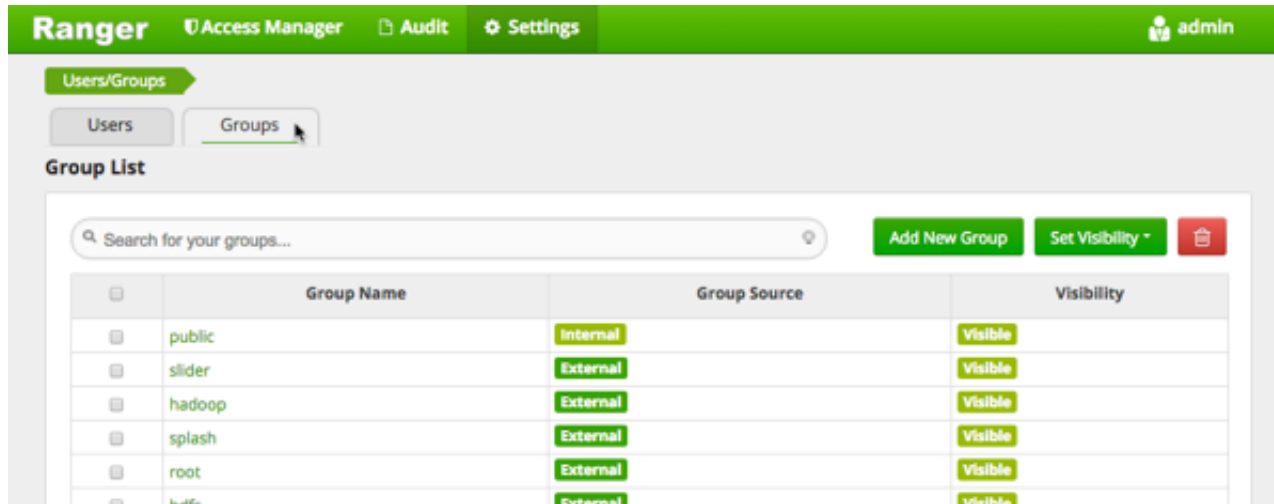
1. Select **Settings > Users/Groups**.

The Users/Groups page opens to the Users tab.



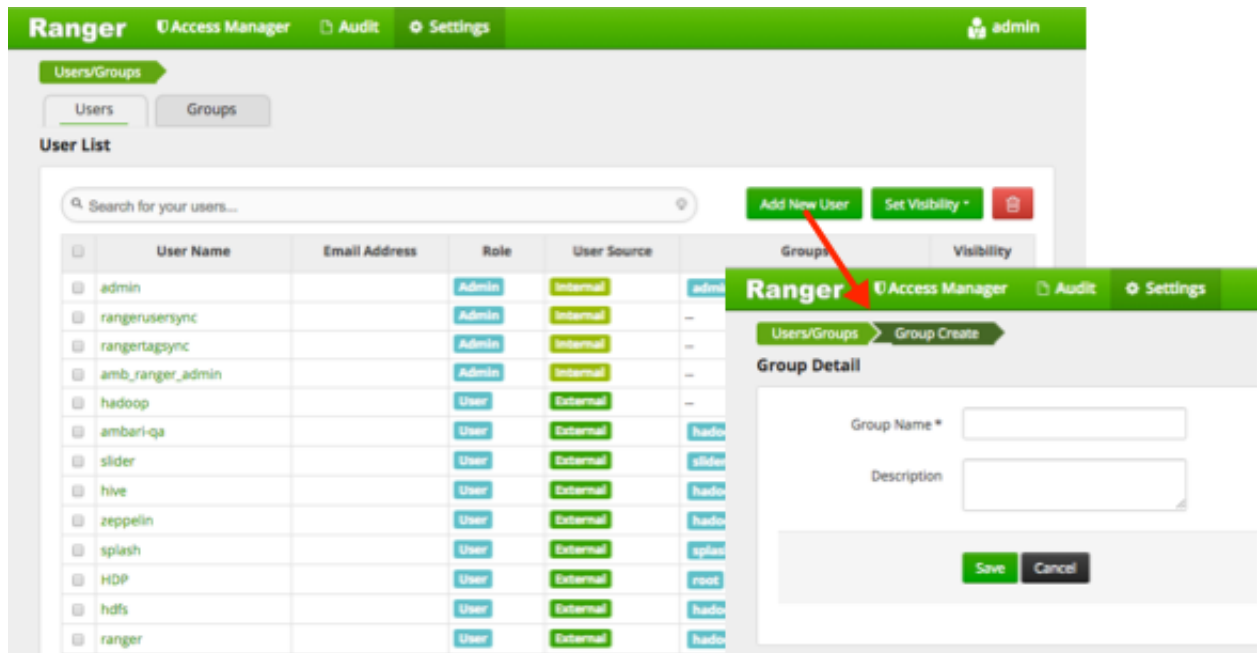
2. Click the **Groups** tab.

The Groups page appears.



3. Click **Add New Group**

The Group Create page appears.



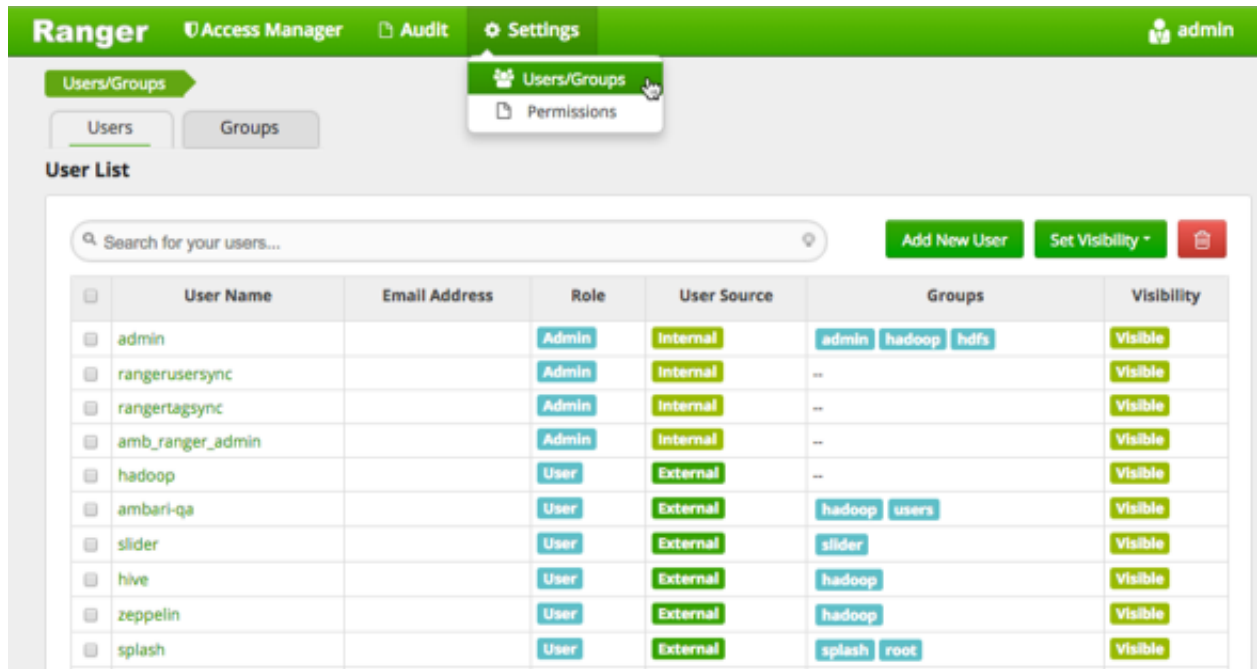
4. Enter a unique name for the group, and an optional description, then click **Save**.

3.2.8.5. Edit a Group

To edit a group:

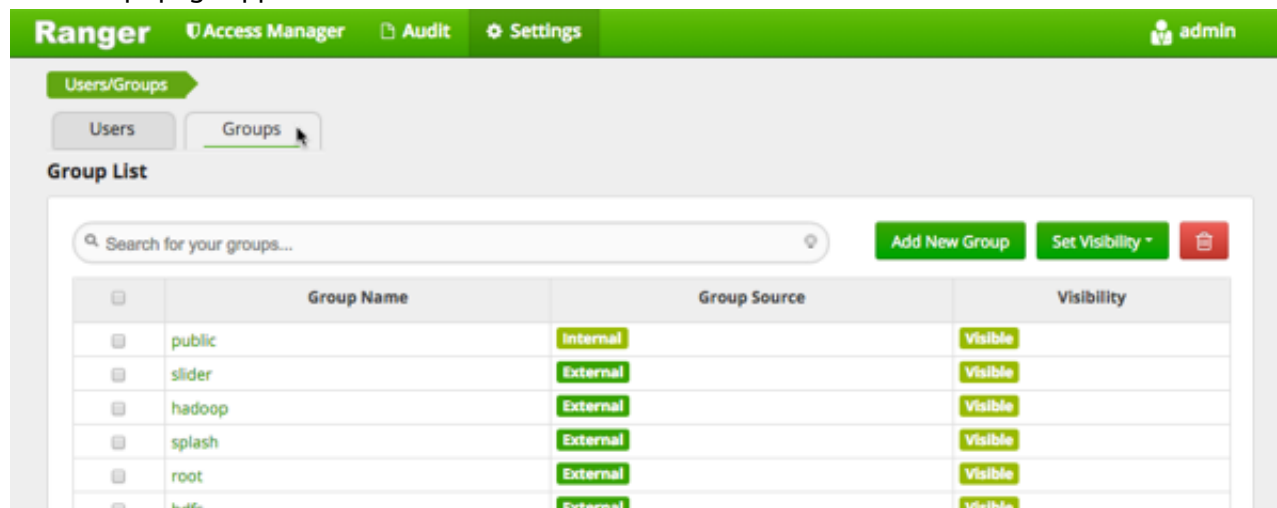
1. Select **Settings > Users/Groups**.

The Users/Groups page opens to the Users tab.

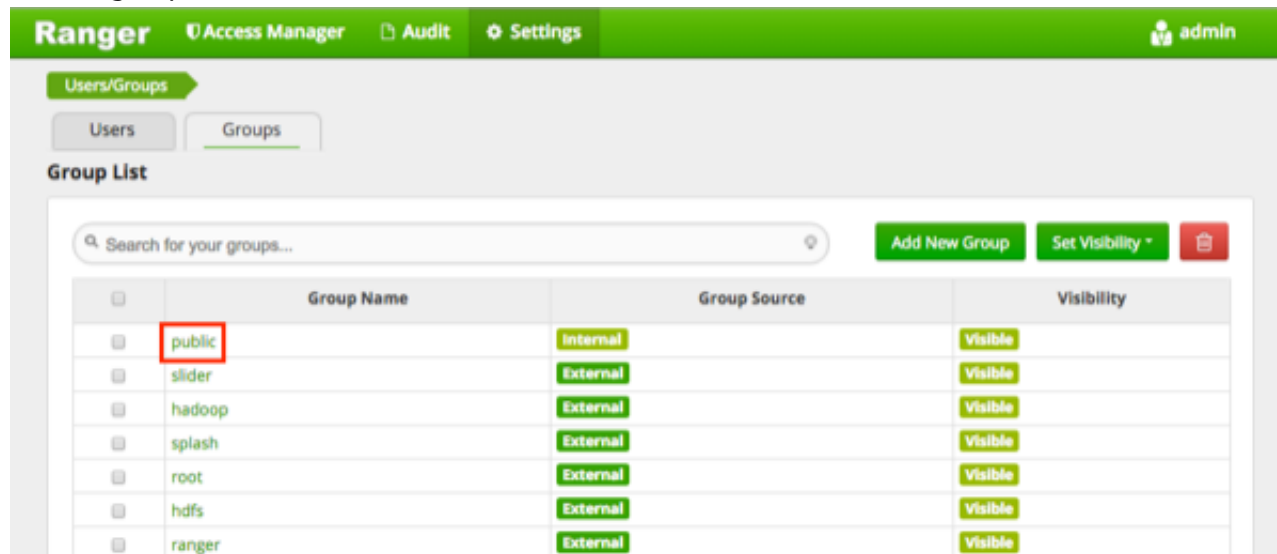


2. Click the **Groups** tab.

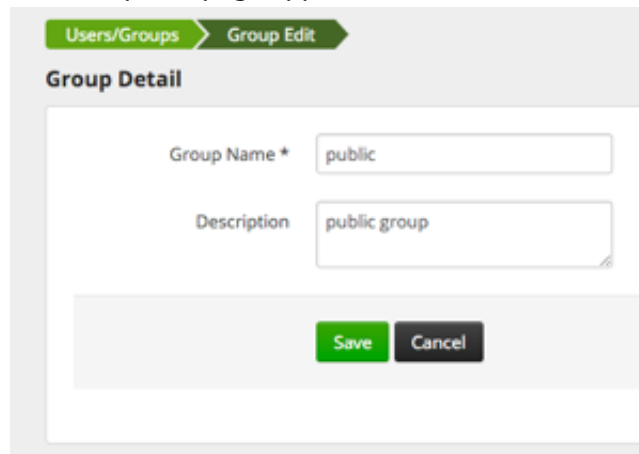
The Groups page appears.



3. Select a group name to edit.



4. The Group Edit page appears.



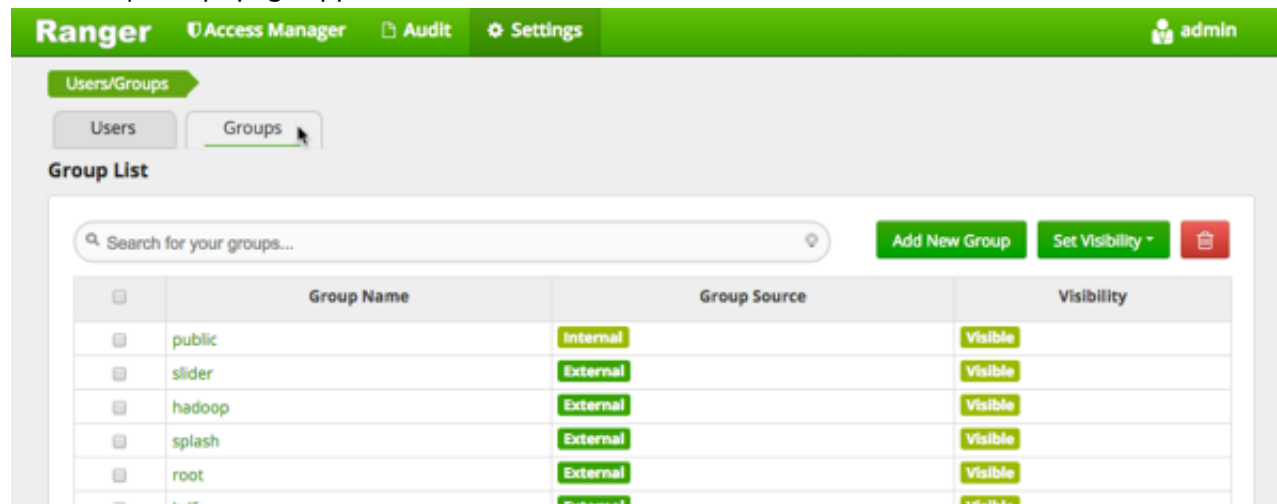
5. Edit the group details, then click **Save**.

3.2.8.6. Delete a Group

Only users with role "admin" may delete a group. To permanently delete a group:

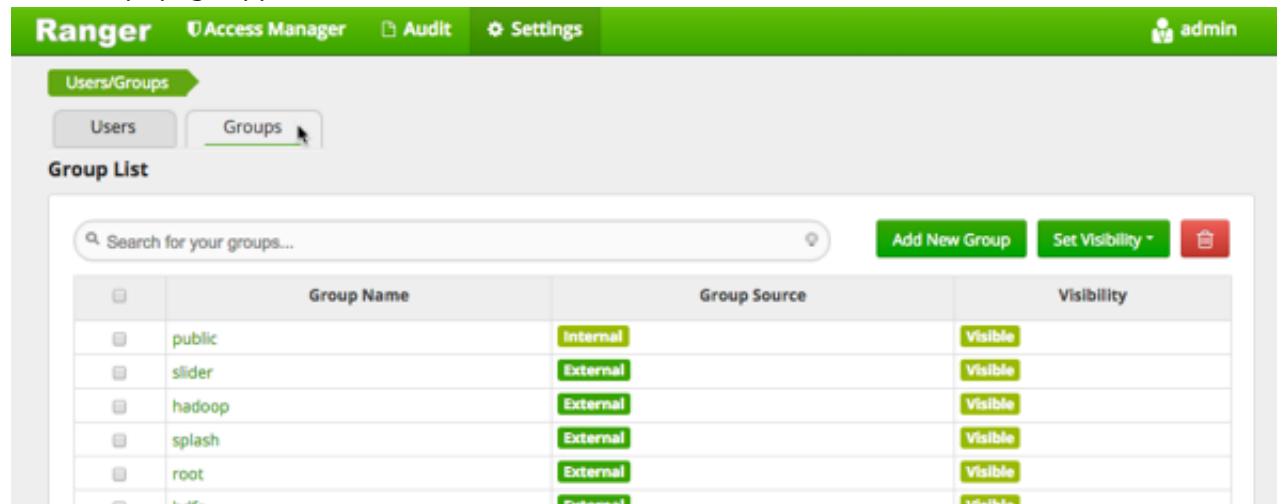
1. Select **Settings > Users/Groups**.


The Users/Groups page appears.



2. Click the **Groups** tab.

The Groups page appears.



3. Select the check box of the group you want to delete and click the Delete icon () at the right of the **Group List** menu bar.

The screenshot shows the Ranger interface with the 'Users/Groups' section active. A table lists various groups, including 'splash'. A red box highlights the checkbox for 'splash'. A confirmation dialog asks 'Are you sure you want to delete group 'splash?'' with 'Cancel' and 'OK' buttons. A red box highlights the 'OK' button. Below the dialog, a green success message states 'Success Group deleted successfully!'. A red arrow points from the 'OK' button to the success message.

4. You are prompted to confirm the group deletion; select OK.

You receive confirmation that the operation has succeeded.

Users in a deleted group will be reassigned to no group. You can [edit the user](#) to reassign it to groups.

3.2.8.7. Add or Edit Permissions

To add or edit user or group:

1. Select **Settings > Permissions**.

The Users/Groups page opens to the Permissions page.

The screenshot shows the Ranger 'Permissions' page. It features a search bar and a table with columns for 'Permissions', 'Groups', 'Users', and 'Action'. The table lists several permissions and the users associated with them.

Permissions	Groups	Users	Action
Resource Based Policies		admin rangeruserync koyadmin hue + More...	
Users/Groups		admin rangeruserync amb ranger_admin	
Reports		admin rangeruserync koyadmin hue + More...	
Audit		admin rangeruserync amb ranger_admin	
Key Manager		koyadmin	

2. Click the Edit icon () next to the permission you would like to edit.

The Edit Permission page appears.

Edit Permission

Policy Details :

Module Name * Users/Groups

User and Group Permissions :

Permissions	Select Group	Select User	Allow Access
	Select Group	x-admin x-rangerusersync x-amb_ranger_admin keyadmin hive kms ambari-qa hdfs knox ranger .	at

Save Cancel

3. Edit the permission settings, then click **Save**.

You can select multiple users and groups from the drop-down menus.

3.2.9. Reports Administration

You can use the Reports page to help manage policies more efficiently as the number of policies increases. The page lists all HDFS, HBase, Hive, YARN, Knox, Storm, Solr, Kafka, Atlas, and tag-based policies.

Ranger Access Manager Audit Settings admin

User Access Report

Reports

Search Criteria

Policy Name Policy Type Select policy type

Component Select Component Resource

Search By Group Select Group

Search

Export

HDFS


Policy ID	Policy Name	Resources	Policy Type	Status	Allow Conditions
2	all - path	path/*	Access	Enabled	+
16	all - path	path/*	Access	Enabled	+

3.2.9.1. View Reports

To view reports on one or more policies, select **Access Manager > Reports**.

The screenshot shows the Ranger Reports interface. At the top, there is a navigation menu with 'Access Manager', 'Audit', and 'Settings'. Below this is a 'User Access Report' section with a 'Reports' tab. The main area contains a 'Search Criteria' form with fields for 'Policy Name', 'Policy Type', 'Component', 'Resource', and 'Search By'. A green 'Search' button is at the bottom of the form. To the right of the form is an 'Export' button. Below the search form is a table titled 'HDFS' with the following data:

Policy ID	Policy Name	Resources	Policy Type	Status	Allow Conditions
2	all - path	path/*	Access	Enabled	+
16	all - path	path/*	Access	Enabled	+

More policy information is available when you click  below **Allow Conditions**.

This screenshot shows the expanded view of the 'Allow Conditions' for the policy with ID 1. The table has columns for 'Policy ID', 'Policy Name', 'Resources', 'Policy Type', 'Status', and 'Allow Conditions'. The 'Allow Conditions' cell is expanded to show a detailed view with sub-sections for 'Groups' and 'Users'. The 'Users' section lists 'hadoop' and 'ambari-qa' with their respective access levels: 'read', 'write', and 'execute'. Below this, there is a row for 'HDFS Global Allow' with a policy ID of 11.

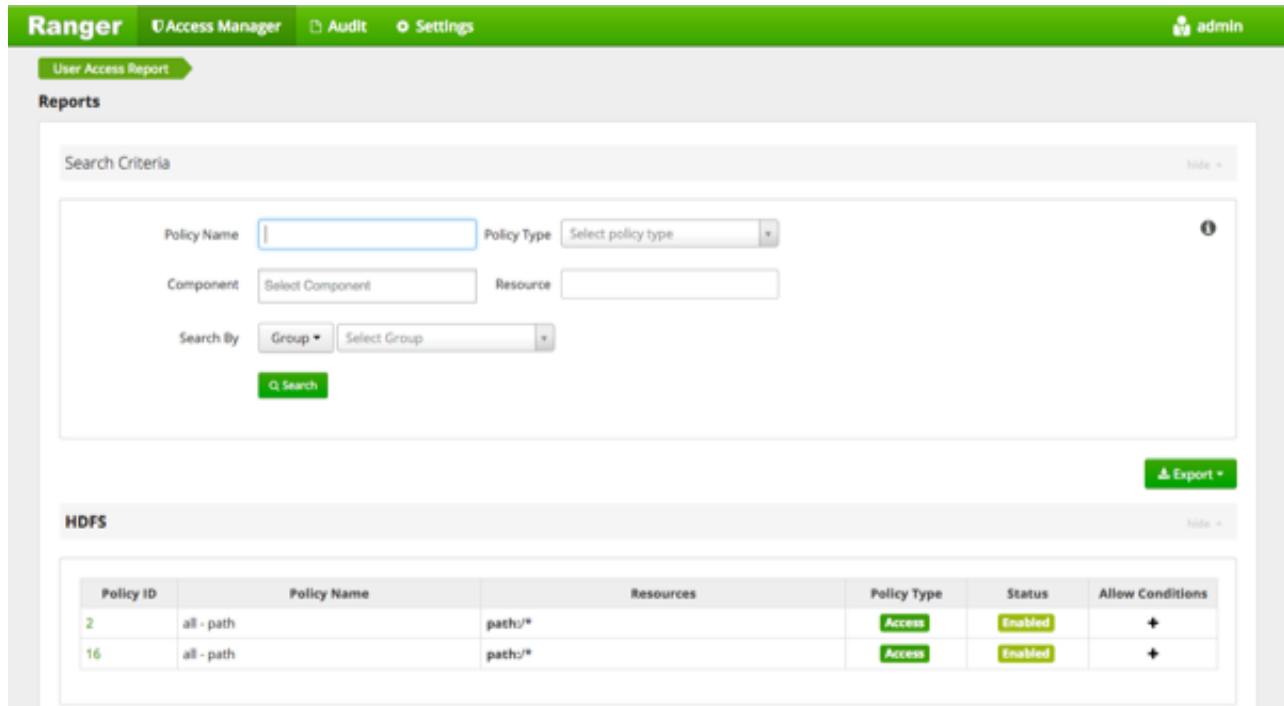
Policy ID	Policy Name	Resources	Policy Type	Status	Allow Conditions													
1	all - path	path/*	Access	Enabled	+													
<table border="1"> <thead> <tr> <th colspan="2">Groups</th> <th colspan="2">Users</th> <th colspan="2">Accesses</th> </tr> </thead> <tbody> <tr> <td colspan="2">--</td> <td>hadoop</td> <td>ambari-qa</td> <td>read</td> <td>write</td> <td>execute</td> </tr> </tbody> </table>						Groups		Users		Accesses		--		hadoop	ambari-qa	read	write	execute
Groups		Users		Accesses														
--		hadoop	ambari-qa	read	write	execute												
11	HDFS Global Allow	path/*	Access	Enabled	+													

3.2.9.2. Search Reports

You can search based on:

- **Policy Name** – The policy name assigned to the policy.
- **Policy Type** – The policy type assigned to the policy (Access, Masking, or Row Level Filter).
- **Component** – The component assigned to the policy (HDFS, HBase, Hive, YARN, Knox, Storm, Solr, Kafka, Atlas, and tag).
- **Resource** – The resource path used when creating the policy.

- **Group, Username** – The group and the users to which the policy is assigned.



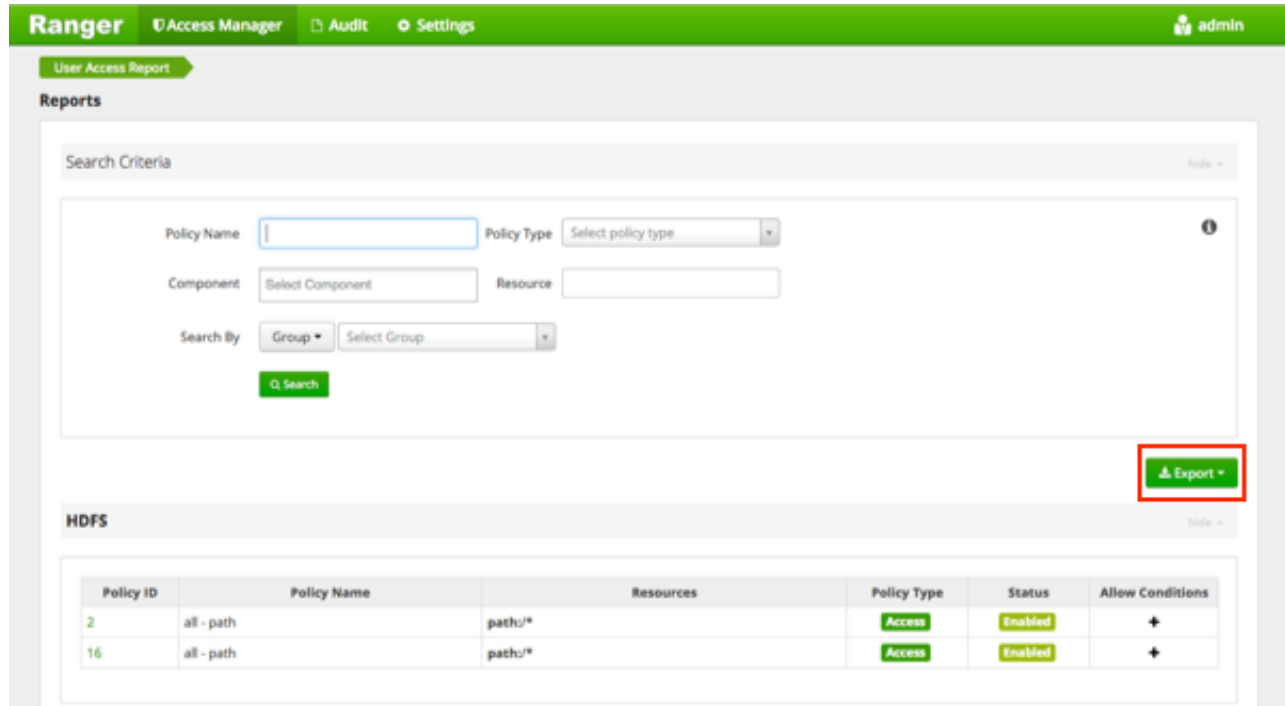
The screenshot displays the Ranger User Access Report interface. At the top, there is a green navigation bar with the Ranger logo and links for Access Manager, Audit, and Settings. The user 'admin' is logged in. Below the navigation bar, the 'User Access Report' section is active, showing a 'Reports' tab. The 'Search Criteria' section includes fields for Policy Name, Policy Type (a dropdown menu), Component, Resource, and Search By (a dropdown menu). A green 'Search' button is located below these fields. To the right of the search criteria is an 'Export' button. Below the search criteria is the 'HDFS' section, which contains a table of policies.

Policy ID	Policy Name	Resources	Policy Type	Status	Allow Conditions
2	all - path	path:/*	Access	Enabled	+
16	all - path	path:/*	Access	Enabled	+

3.2.9.3. Export Reports

You can export a list of reports in three file formats:

- CSV file
- Excel file
- JSON



The screenshot shows the Ranger web interface. At the top, there is a green navigation bar with 'Ranger' and links for 'Access Manager', 'Audit', and 'Settings'. The user 'admin' is logged in. Below the navigation bar, there is a 'User Access Report' breadcrumb and a 'Reports' section. The 'Search Criteria' section contains several input fields: 'Policy Name' (text input), 'Policy Type' (dropdown menu), 'Component' (text input), 'Resource' (text input), and 'Search By' (dropdown menu). A green 'Search' button is located below these fields. To the right of the search criteria, there is a green 'Export' button with a download icon, which is highlighted with a red box. Below the search criteria, there is an 'HDFS' section containing a table of policies.

Policy ID	Policy Name	Resources	Policy Type	Status	Allow Conditions
2	all - path	path:/*	Access	Enabled	+
16	all - path	path:/*	Access	Enabled	+

For more information on exporting policies from the reports page, see: [Export Tag-Based Policies](#) and [Export Resource-Based Policies](#).

3.2.9.4. Edit Policies from the Reports Page

You can edit policies from the Reports page by selecting the Policy ID.

The screenshot displays the Ranger web interface. The top navigation bar includes 'Ranger', 'Access Manager', 'Audit', and 'Settings', with a user profile 'admin'. The main content area is titled 'Edit Policy' and shows the following details:

- Policy Details:**
 - Policy Type: Access
 - Policy ID: 1
 - Policy Name: all - path (with an 'enabled' toggle)
 - Resource Path: (empty field)
 - Recursive: ON
 - Audit Logging: YES
 - Description: Policy for all - path
- Allow Conditions:**
 - Select Group: (dropdown menu)
 - Select User: hadoop, ambari-qa
 - Permissions: Read, Write, Execute
 - Delegate Admin: (checkbox)

At the bottom, there are 'Save', 'Cancel', and 'Delete' buttons. A sidebar on the left shows a list of policies under 'HDFS' and 'HBASE' categories, with a red arrow pointing to policy ID 11.

3.2.10. Special Requirements for High Availability Environments

In a High Availability (HA) environment, the 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 service set up in the Security Manager. Any policies created within that HDFS service 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 service, you must include the `fs.default.name` property, and it must be set to the full host name 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 service details to enable directory lookup for policy creation.

If, while the primary NameNode 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 failover 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.

For **Test Connection** to be successful for HBase and HDFS in a Ranger HA environment, complete the following: In `/etc/ranger/admin`, create a symbolic link between `hbase-site.xml` and `hdfs-site.xml`:

```
cd /etc/ranger/admin
ln -s /etc/hadoop/conf/hdfs-site.xml hdfs-site.xml
ln -s /etc/hbase/conf/hbase-site.xml hbase-site.xml
```

3.2.11. Adding a New Component to Apache Ranger

This section describes 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.

In order to support new component authorization using Apache Ranger, the component details need 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 a (public) REST API. In order to support a new component in both the UI and the Server, the Admin Tool must be modified.

Required UI changes to support the new component:

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

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

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

2. Add new configuration information to the Service Form:

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

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 Service Type

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

2. Add new required parameters in existing 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 Service Creation/Update of any component: Apache Ranger uses only one common object `VXService`. As Apache Ranger has three components, it will have all the required parameters of all of those three components in `VXPolicy/VXService`. 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/VXService` 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/VXService, 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 Service, then on the basis of serviceType, Apache Ranger will create one default policy for the given service. Everything else will work fine, as it is common for all components.

Required database changes for the new component:

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

- x_asset (for service)
- x_resource (for service)

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 service for our new component, and also for previous components.

3.2.12. 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 Service Manager for a given service-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()

3.2.13. Apache Ranger Public REST API

- [Service Definition APIs \[390\]](#)

- [Get Service Definition by ID \[391\]](#)
- [Get Service Definition by Name \[394\]](#)
- [Create Service Definition \[397\]](#)
- [Update Service Definition by ID \[400\]](#)
- [Update Service Definition by Name \[404\]](#)
- [Delete Service Definition by ID \[404\]](#)
- [Delete Service Definition by Name \[404\]](#)
- [Search Service Definitions \[404\]](#)
- [Service APIs \[406\]](#)
 - [Get Service by ID \[406\]](#)
 - [Get Service by Name \[406\]](#)
 - [Create Service \[407\]](#)
 - [Update Service by ID \[407\]](#)
 - [Update Service by Name \[408\]](#)
 - [Delete Service by ID \[408\]](#)
 - [Delete Service by Name \[408\]](#)
 - [Search Services \[409\]](#)
- [Policy APIs \[411\]](#)
 - [Get Policy by ID \[411\]](#)
 - [Get Policy by Service Name and Policy Name \[412\]](#)
 - [Create Policy \[413\]](#)
 - [Update Policy by ID \[415\]](#)
 - [Update Policy by Service Name and Policy Name \[417\]](#)
 - [Delete Policy by ID \[420\]](#)
 - [Delete Policy by Service Name and Policy Name \[420\]](#)
 - [Search Policies in a Service \[420\]](#)

3.2.13.1. Service Definition APIs

- [Get Service Definition by ID \[391\]](#)

- [Get Service Definition by Name \[394\]](#)
- [Create Service Definition \[397\]](#)
- [Update Service Definition by ID \[400\]](#)
- [Update Service Definition by Name \[404\]](#)
- [Delete Service Definition by ID \[404\]](#)
- [Delete Service Definition by Name \[404\]](#)
- [Search Service Definitions \[404\]](#)

3.2.13.1.1. Get Service Definition by ID

API Name	Get Service Definition
Request Type	GET
Request URL	service/public/v2/api/servicedef/{id}
Request Params	
Response	<pre>{ "accessTypes": [{ "impliedGrants": [], "itemId": 1, "label": "Read", "name": "read" }, { "impliedGrants": [], "itemId": 2, "label": "Write", "name": "write" }, { "impliedGrants": [], "itemId": 3, "label": "Execute", "name": "execute" }], "configs": [{ "itemId": 1, "label": "Username", "mandatory": true, "name": "username", "subType": "", "type": "string", "uiHint": "", "validationMessage": "", "validationRegex": "" }, { "itemId": 2, "label": "Password", "mandatory": true, "name": "password", "subType": "", "type": "password", "uiHint": "", "validationMessage": "", "validationRegex": "" }, { "itemId": 3, "label": "Namenode URL", "mandatory": true, "name": "fs.default.name", "subType": "", "type": "string", </pre>

API Name	Get Service Definition
	<pre> "uiHint": "", "validationMessage": "", "validationRegex": "" }, { "defaultValue": "false", "itemId": 4, "label": "Authorization Enabled", "mandatory": true, "name": "hadoop.security. authorization", "subType": "YesTrue:NoFalse", "type": "bool", "uiHint": "", "validationMessage": "", "validationRegex": "" }, { "defaultValue": "simple", "itemId": 5, "label": "Authentication Type", "mandatory": true, "name": "hadoop.security. authentication", "subType": "authnType", "type": "enum", "uiHint": "", "validationMessage": "", "validationRegex": "" }, { "itemId": 6, "mandatory": false, "name": "hadoop.security. auth_to_local", "subType": "", "type": "string", "uiHint": "", "validationMessage": "", "validationRegex": "" }, { "itemId": 7, "mandatory": false, "name": "dfs.datanode.kerberos. principal", "subType": "", "type": "string", "uiHint": "", "validationMessage": "", "validationRegex": "" }, { "itemId": 8, "mandatory": false, "name": "dfs.namenode.kerberos. principal", "subType": "", "type": "string", "uiHint": "", "validationMessage": "", "validationRegex": "" }, { "itemId": 9, "mandatory": false, "name": "dfs.secondary.namenode. kerberos.principal", "subType": "", "type": "string", "uiHint": "", "validationMessage": "", "validationRegex": "" }, { "defaultValue": "authentication", "itemId": 10, </pre>

API Name	Get Service Definition
	<pre> "label": "RPC Protection Type", "mandatory": false, "name": "hadoop.rpc.protection", "subType": "rpcProtection", "type": "enum", "uiHint": "", "validationMessage": "", "validationRegex": "" }, { "itemId": 11, "label": "Common Name for Certificate", "mandatory": false, "name": "commonNameForCertificate", "subType": "", "type": "string", "uiHint": "", "validationMessage": "", "validationRegex": "" }], "contextEnrichers": [], "createTime": 1450756476000, "description": "HDFS Repository", "enums": [{ "defaultIndex": 0, "elements": [{ "itemId": 1, "label": "Simple", "name": "simple" }, { "itemId": 2, "label": "Kerberos", "name": "kerberos" }] }, { "itemId": 1, "name": "authnType" }, { "defaultIndex": 0, "elements": [{ "itemId": 1, "label": "Authentication", "name": "authentication" }, { "itemId": 2, "label": "Integrity", "name": "integrity" }, { "itemId": 3, "label": "Privacy", "name": "privacy" }] }, { "itemId": 2, "name": "rpcProtection" }] }, "guid": "0d047247-bafe-4cf8-8e9b- d5d377284b2d", "id": 1, "implClass": "org.apache.ranger.services.hdfs. RangerServiceHdfs", "isEnabled": true, "label": "HDFS Repository", "name": "hdfs", "options": {}, "policyConditions": [], "resources": [</pre>

API Name	Get Service Definition
	<pre> { "description": "HDFS file or directory path", "excludesSupported": false, "itemId": 1, "label": "Resource Path", "level": 10, "lookupSupported": true, "mandatory": true, "matcher": "org.apache.ranger.plugin. resourcematcher.RangerPathResourceMatcher", "matcherOptions": { "ignoreCase": "false", "wildCard": "true" }, "name": "path", "recursiveSupported": true, "type": "path", "uiHint": "", "validationMessage": "", "validationRegex": "" }], "updateTime": 1450756477000, "version": 1 } </pre>

3.2.13.1.2. Get Service Definition by Name

API Name	Get Service Definition
Request Type	GET
Request URL	service/public/v2/api/servicedef/name/{name}
Request Params	
Response	<pre> { "accessTypes": [{ "impliedGrants": [], "itemId": 1, "label": "select", "name": "select" }, { "impliedGrants": [], "itemId": 2, "label": "update", "name": "update" }, { "impliedGrants": [], "itemId": 3, "label": "Create", "name": "create" }, { "impliedGrants": [], "itemId": 4, "label": "Drop", "name": "drop" }, { "impliedGrants": [], "itemId": 5, "label": "Alter", "name": "alter" }, { "impliedGrants": [], "itemId": 6, "label": "Index", "name": "index" }, { "impliedGrants": [], </pre>

API Name	Get Service Definition
	<pre> "itemId": 7, "label": "Lock", "name": "lock" }, { "impliedGrants": ["select", "update", "create", "drop", "alter", "index", "lock"], "itemId": 8, "label": "All", "name": "all" }], "configs": [{ "itemId": 1, "label": "Username", "mandatory": true, "name": "username", "type": "string", "uiHint": "", "validationMessage": "", "validationRegex": "" }, { "itemId": 2, "label": "Password", "mandatory": true, "name": "password", "type": "password", "uiHint": "", "validationMessage": "", "validationRegex": "" }, { "defaultValue": "org.apache.hive.jdbc. HiveDriver", "itemId": 3, "mandatory": true, "name": "jdbc.driverClassName", "type": "string", "uiHint": "", "validationMessage": "", "validationRegex": "" }, { "defaultValue": "", "itemId": 4, "mandatory": true, "name": "jdbc.url", "type": "string", "uiHint": "", "validationMessage": "", "validationRegex": "" }, { "itemId": 5, "label": "Common Name for Certificate", "mandatory": false, "name": "commonNameForCertificate", "type": "string", "uiHint": "", "validationMessage": "", "validationRegex": "" } }], "contextEnrichers": [], "createTime": 1450756479000, "description": "Hive Server2", "enums": [], </pre>

API Name	Get Service Definition
	<pre> "guid": "3e1afb5a-184a-4e82-9d9c-87a5cacc243c", "id": 3, "implClass": "org.apache.ranger.services.hive. RangerServiceHive", "isEnabled": true, "label": "Hive Server2", "name": "hive", "options": {}, "policyConditions": [{ "description": "List of Hive resources", "evaluator": "org.apache. ranger.plugin.conditionevaluator. RangerHiveResourcesAccessedTogetherCondition", "evaluatorOptions": {}, "itemId": 1, "label": "Hive Resources Accessed Together?", "name": "resources-accessed-together" }], "resources": [{ "description": "Hive Database", "excludesSupported": true, "itemId": 1, "label": "Hive Database", "level": 10, "lookupSupported": true, "mandatory": true, "matcher": "org.apache.ranger.plugin. resourcematcher.RangerDefaultResourceMatcher", "matcherOptions": { "ignoreCase": "true", "wildCard": "true" }, "name": "database", "recursiveSupported": false, "type": "string", "uiHint": "", "validationMessage": "", "validationRegex": "" }, { "description": "Hive Table", "excludesSupported": true, "itemId": 2, "label": "Hive Table", "level": 20, "lookupSupported": true, "mandatory": true, "matcher": "org.apache.ranger.plugin. resourcematcher.RangerDefaultResourceMatcher", "matcherOptions": { "ignoreCase": "true", "wildCard": "true" }, "name": "table", "parent": "database", "recursiveSupported": false, "type": "string", "uiHint": "", "validationMessage": "", "validationRegex": "" }, { "description": "Hive UDF", "excludesSupported": true, "itemId": 3, "label": "Hive UDF", "level": 20, "lookupSupported": true, "mandatory": true, "matcher": "org.apache.ranger.plugin. resourcematcher.RangerDefaultResourceMatcher", </pre>

API Name	Get Service Definition
	<pre> "matcherOptions": { "ignoreCase": "true", "wildCard": "true" }, "name": "udf", "parent": "database", "recursiveSupported": false, "type": "string", "uiHint": "", "validationMessage": "", "validationRegex": "" }, { "description": "Hive Column", "excludesSupported": true, "itemId": 4, "label": "Hive Column", "level": 30, "lookupSupported": true, "mandatory": true, "matcher": "org.apache.ranger.plugin. resourcematcher.RangerDefaultResourceMatcher", "matcherOptions": { "ignoreCase": "true", "wildCard": "true" }, "name": "column", "parent": "table", "recursiveSupported": false, "type": "string", "uiHint": "", "validationMessage": "", "validationRegex": "" }], "updateTime": 1450756479000, "version": 1 } </pre>

3.2.13.1.3. Create Service Definition

API Name	Create Service Definition
Request Type	Post
Request URL	service/public/v2/api/servicedef
Request Params	<pre> { "accessTypes": [{ "impliedGrants": [], "itemId": 1, "label": "select", "name": "select" }, { "impliedGrants": [], "itemId": 2, "label": "update", "name": "update" }, { "impliedGrants": [], "itemId": 3, "label": "Create", "name": "create" }, { "impliedGrants": [], "itemId": 4, "label": "Drop", "name": "drop" }], { "impliedGrants": ["select", "update",] } } </pre>

API Name	Create Service Definition
	<pre> "create", "drop"], "itemId": 5, "label": "All", "name": "all" }], "configs": [{ "itemId": 1, "label": "Username", "mandatory": true, "name": "username", "type": "string", "uiHint": "", "validationMessage": "", "validationRegex": "" }, { "itemId": 2, "label": "Password", "mandatory": true, "name": "password", "type": "password", "uiHint": "", "validationMessage": "", "validationRegex": "" }], "contextEnrichers": [], "description": "Test Component", "enums": [], "implClass": "org.apache.ranger.services.test.RangerServiceTest", "isEnabled": true, "label": "Test Component", "name": "test", "options": {}, "policyConditions": [], "resources": [{ "description": "Root Of Resource Hierarchy for Test Component", "excludesSupported": true, "itemId": 1, "label": "Test Root Resource", "level": 10, "lookupSupported": true, "mandatory": true, "matcher": "org.apache.ranger.plugin.resourcematcher.RangerDefaultResourceMatcher", "matcherOptions": { "ignoreCase": "true", "wildCard": "true" }, "name": "root", "recursiveSupported": false, "type": "string", "uiHint": "", "validationMessage": "", "validationRegex": "" }, { "description": "Sub Resource for Test Component", "excludesSupported": true, "itemId": 2, "label": "Test sub resource", "level": 20, "lookupSupported": true, "mandatory": true, "matcher": "org.apache.ranger.plugin.resourcematcher.RangerDefaultResourceMatcher", "matcherOptions": { "ignoreCase": "true", "wildCard": "true" } }] </pre>

API Name	Create Service Definition
	<pre> }, "name": "sub", "parent": "root", "recursiveSupported": false, "type": "string", "uiHint": "", "validationMessage": "", "validationRegex": "" }], "version": 1 } </pre>
Response	<pre> { "accessTypes": [{ "impliedGrants": [], "itemId": 1, "label": "select", "name": "select" }, { "impliedGrants": [], "itemId": 2, "label": "update", "name": "update" }, { "impliedGrants": [], "itemId": 3, "label": "Create", "name": "create" }, { "impliedGrants": [], "itemId": 4, "label": "Drop", "name": "drop" }, { "impliedGrants": ["select", "update", "create", "drop"], "itemId": 5, "label": "All", "name": "all" }], "configs": [{ "itemId": 1, "label": "Username", "mandatory": true, "name": "username", "type": "string", "uiHint": "", "validationMessage": "", "validationRegex": "" }, { "itemId": 2, "label": "Password", "mandatory": true, "name": "password", "type": "password", "uiHint": "", "validationMessage": "", "validationRegex": "" }], "contextEnrichers": [], "createTime": 1451347300617, "createdBy": "Admin", "description": "Test Component", "enums": [], </pre>

API Name	Create Service Definition
	<pre> "guid": "f889f2d3-920a-4504-9905-809bbc417902", "id": 101, "implClass": "org.apache.ranger.services.test. RangerServiceTest", "isEnabled": true, "label": "Test Component", "name": "test", "options": {}, "policyConditions": [], "resources": [{ "description": "Root Of Resource Hierarchy for Test Component", "excludesSupported": true, "itemId": 1, "label": "Test Root Resource", "level": 10, "lookupSupported": true, "mandatory": true, "matcher": "org.apache.ranger.plugin. resourcematcher.RangerDefaultResourceMatcher", "matcherOptions": { "ignoreCase": "true", "wildCard": "true" }, "name": "root", "recursiveSupported": false, "type": "string", "uiHint": "", "validationMessage": "", "validationRegEx": "" }, { "description": "Sub Resource for Test Component", "excludesSupported": true, "itemId": 2, "label": "Test sub resource", "level": 20, "lookupSupported": true, "mandatory": true, "matcher": "org.apache.ranger.plugin. resourcematcher.RangerDefaultResourceMatcher", "matcherOptions": { "ignoreCase": "true", "wildCard": "true" }, "name": "sub", "parent": "root", "recursiveSupported": false, "type": "string", "uiHint": "", "validationMessage": "", "validationRegEx": "" }], "updateTime": 1451347300618, "updatedBy": "Admin", "version": 1 } </pre>

3.2.13.1.4. Update Service Definition by ID

API Name	Update Service Definition
Request Type	PUT
Request URL	service/public/v2/api/servicedef/{id}
Request Params	<pre> { "accessTypes": [{ "impliedGrants": [], "itemId": 1, "label": "select", "name": "select" }] } </pre>

API Name	Update Service Definition
	<pre> }, { "impliedGrants": [], "itemId": 2, "label": "update", "name": "update" }, { "impliedGrants": [], "itemId": 3, "label": "Create", "name": "create" }, { "impliedGrants": [], "itemId": 4, "label": "Drop", "name": "drop" }, { "impliedGrants": [], "itemId": 5, "label": "Index", "name": "index" }, { "impliedGrants": ["select", "update", "create", "drop", "index"], "itemId": 6, "label": "All", "name": "all" }], "configs": [{ "itemId": 1, "label": "Username", "mandatory": true, "name": "username", "type": "string", "uiHint": "", "validationMessage": "", "validationRegex": "" }, { "itemId": 2, "label": "Password", "mandatory": true, "name": "password", "type": "password", "uiHint": "", "validationMessage": "", "validationRegex": "" }], "contextEnrichers": [], "description": "Test Component", "enums": [], "implClass": "org.apache.ranger.services.test.RangerServiceTest", "isEnabled": true, "label": "Test Component", "name": "test", "options": {}, "policyConditions": [], "resources": [{ "description": "Root Of Resource Hierarchy for Test Component", "excludesSupported": true, "itemId": 1, "label": "Test Root Resource", </pre>

API Name	Update Service Definition
	<pre> "level": 10, "lookupSupported": true, "mandatory": true, "matcher": "org.apache.ranger.plugin. resourcematcher.RangerDefaultResourceMatcher", "matcherOptions": { "ignoreCase": "true", "wildCard": "true" }, "name": "root", "recursiveSupported": false, "type": "string", "uiHint": "", "validationMessage": "", "validationRegex": "" }, { "description": "Sub Resource for Test Component", "excludesSupported": true, "itemId": 2, "label": "Test sub resource", "level": 20, "lookupSupported": true, "mandatory": true, "matcher": "org.apache.ranger.plugin. resourcematcher.RangerDefaultResourceMatcher", "matcherOptions": { "ignoreCase": "true", "wildCard": "true" }, "name": "sub", "parent": "root", "recursiveSupported": false, "type": "string", "uiHint": "", "validationMessage": "", "validationRegex": "" }] } </pre>
<p>Response</p>	<pre> { "accessTypes": [{ "impliedGrants": [], "itemId": 1, "label": "select", "name": "select" }, { "impliedGrants": [], "itemId": 2, "label": "update", "name": "update" }, { "impliedGrants": [], "itemId": 3, "label": "Create", "name": "create" }, { "impliedGrants": [], "itemId": 4, "label": "Drop", "name": "drop" }, { "impliedGrants": [], "itemId": 5, "label": "Index", "name": "index" }], { "impliedGrants": ["select", "update", </pre>

API Name	Update Service Definition
	<pre> "create", "drop", "index"], "itemId": 6, "label": "All", "name": "all" }], "configs": [{ "itemId": 1, "label": "Username", "mandatory": true, "name": "username", "type": "string", "uiHint": "", "validationMessage": "", "validationRegex": "" }, { "itemId": 2, "label": "Password", "mandatory": true, "name": "password", "type": "password", "uiHint": "", "validationMessage": "", "validationRegex": "" }], "contextEnrichers": [], "createTime": 1451347301000, "createdBy": "Admin", "description": "Test Component", "enums": [], "guid": "f889f2d3-920a-4504-9905-809bbc417902", "id": 101, "implClass": "org.apache.ranger.services.test. RangerServiceTest", "isEnabled": true, "label": "Test Component", "name": "test", "options": {}, "policyConditions": [], "resources": [{ "description": "Root Of Resource Hierarchy for Test Component", "excludesSupported": true, "itemId": 1, "label": "Test Root Resource", "level": 10, "lookupSupported": true, "mandatory": true, "matcher": "org.apache.ranger.plugin. resourcematcher.RangerDefaultResourceMatcher", "matcherOptions": { "ignoreCase": "true", "wildCard": "true" }, "name": "root", "recursiveSupported": false, "type": "string", "uiHint": "", "validationMessage": "", "validationRegex": "" }, { "description": "Sub Resource for Test Component", "excludesSupported": true, "itemId": 2, "label": "Test sub resource", "level": 20, "lookupSupported": true, </pre>

API Name	Update Service Definition
	<pre> "mandatory": true, "matcher": "org.apache.ranger.plugin. resourcematcher.RangerDefaultResourceMatcher", "matcherOptions": { "ignoreCase": "true", "wildCard": "true" }, "name": "sub", "parent": "root", "recursiveSupported": false, "type": "string", "uiHint": "", "validationMessage": "", "validationRegex": "" }], "updateTime": 1451351474321, "updatedBy": "Admin", "version": 2 } </pre>

3.2.13.1.5. Update Service Definition by Name

API Name	Update Service Definition
Request Type	PUT
Request URL	service/public/v2/api/servicedef/{name}
Request Params	Application/json • Example:
Response	200-Application/json

3.2.13.1.6. Delete Service Definition by ID

API Name	Delete Service Definition
Request Type	DELETE
Request URL	service/public/v2/api/servicedef/{id}
Request Param	
Response	204-No Content

3.2.13.1.7. Delete Service Definition by Name

API Name	Delete Service Definition
Request Type	DELETE
Request URL	service/public/v2/api/servicedef/name/{name}
Request Param	
Response	204-No Content

3.2.13.1.8. Search Service Definitions

API Name	Search Service Definitions
Request Type	GET
Request URL	service/public/v2/api/servicedef
Request Params	Query Params pageSize int <i>The page size required</i> startIndex int <i>The startrecord index</i>

API Name	Search Service Definitions
	<p>serviceType string <i>The service definition names("hdfs","hive","hbase","knox","storm","solr","kafka","yarn")</i></p> <p>isEnabled boolean <i>The enabled status : true if enabled; false otherwise</i></p> <p>Example :</p> <p>pageSize=25&startIndex=0</p>
Response	<pre>[{ "accessTypes": [{ "impliedGrants": [], "itemId": 1, "label": "Read", "name": "read" }, { "impliedGrants": [], "itemId": 2, "label": "Write", "name": "write" }, { "impliedGrants": [], "itemId": 3, "label": "Execute", "name": "execute" }], "configs": [{ "itemId": 1, "label": "Username", "mandatory": true, "name": "username", "subType": "", "type": "string", "uiHint": "", "validationMessage": "", "validationRegex": "" }, ... { "description": "Sub Resource for Test Component", "excludesSupported": true, "itemId": 2, "label": "Test sub resource", "level": 20, "lookupSupported": true, "mandatory": true, "matcher": "org. apache.ranger.plugin.resourcematcher. RangerDefaultResourceMatcher", "matcherOptions": { "ignoreCase": "true", "wildCard": "true" }, "name": "sub", "parent": "root", "recursiveSupported": false, "type": "string", "uiHint": "", "validationMessage": "", "validationRegex": "" }], "updateTime": 1451351474000, "updatedBy": "Admin", "version": 2 }]</pre>

3.2.13.2. Service APIs

- [Get Service by ID \[406\]](#)
- [Get Service by Name \[406\]](#)
- [Create Service \[407\]](#)
- [Update Service by ID \[407\]](#)
- [Update Service by Name \[408\]](#)
- [Delete Service by ID \[408\]](#)
- [Delete Service by Name \[408\]](#)
- [Search Services \[409\]](#)

3.2.13.2.1. Get Service by ID

API Name	Get Service
Request Type	GET
Request URL	service/public/v2/api/service/{id}
Request Params	
Response	<pre>{ "configs": { "fs.default.name": "hdfs://akulkarni-etp-real-final-1.novalocal:8020", "hadoop.security.auth_to_local": "DEFAULT", "hadoop.security.authentication": "simple", "hadoop.security.authorization": "false", "password": "*****", "username": "hadoop" }, "createTime": 1450757397000, "createdBy": "amb_ranger_admin", "description": "hdfs repo", "guid": "ec082eea-0c22-43b8-84e0-129422f689b9", "id": 1, "isEnabled": true, "name": "c11_hadoop", "policyUpdateTime": 1450757398000, "policyVersion": 2, "tagVersion": 1, "type": "hdfs", "updateTime": 1450757398000, "updatedBy": "amb_ranger_admin", "version": 3 }</pre>

3.2.13.2.2. Get Service by Name

API Name	Get Service
Request Type	GET
Request URL	service/public/v2/api/service/name/{name}
Request Params	
Response	{

API Name	Get Service
	<pre> "configs": { "jdbc.driverClassName": "org.apache.hive. jdbc.HiveDriver", "jdbc.url": "jdbc:hive2://akulkarni-etp- real-final-1.novalocal:10000", "password": "*****", "username": "hive" }, "createTime": 1450757992000, "createdBy": "amb_ranger_admin", "description": "hive repo", "guid": "2bca8f98-4859-43c3-a8f4- d31a15f28793", "id": 3, "isEnabled": true, "name": "cl1_hive", "policyUpdateTime": 1450757995000, "policyVersion": 4, "tagUpdateTime": 1450916660000, "tagVersion": 74, "type": "hive", "updateTime": 1450757995000, "updatedBy": "amb_ranger_admin", "version": 78 } </pre>

3.2.13.2.3. Create Service

API Name	Create Service
Request Type	Post
Request URL	service/public/v2/api/service
Request Params	<pre> { "configs": { "password": "*****", "username": "hadoop" }, "description": "test service", "isEnabled": true, "name": "cl1_test", "type": "test", "version": 1 } </pre>
Response	<pre> { "configs": { "password": "*****", "username": "hadoop" }, "createTime": 1451348710255, "createdBy": "Admin", "description": "test service", "guid": "e72cb64d-66d7-4632-b5ae- c6966cb52105", "id": 6, "isEnabled": true, "name": "cl1_test", "tagVersion": 1, "type": "test", "updateTime": 1451348710256, "updatedBy": "Admin", "version": 1 } </pre>

3.2.13.2.4. Update Service by ID

API Name	Update Service
Request Type	PUT

API Name	Update Service
Request URL	service/public/v2/api/service/{id}
Request Params	Application/json • Example:
Response	200-Application/json

3.2.13.2.5. Update Service by Name

API Name	Update Service
Request Type	PUT
Request URL	service/public/v2/api/service/name/{name}
Request Params	<pre>{ "configs": { "password": "*****", "username": "admin" }, "description": "test service", "isEnabled": true, "name": "cl1_test", "type": "test" }</pre>
Response	<pre>{ "configs": { "password": "*****", "username": "admin" }, "createTime": 1451348710000, "createdBy": "Admin", "description": "test service", "guid": "e72cb64d-66d7-4632-b5ae-c6966cb52105", "id": 6, "isEnabled": true, "name": "cl1_test", "policyUpdateTime": 1451351474000, "policyVersion": 3, "tagVersion": 1, "type": "test", "updateTime": 1451352016713, "updatedBy": "Admin", "version": 5 }</pre>

3.2.13.2.6. Delete Service by ID

API Name	Delete Service
Request Type	DELETE
Request URL	service/public/v2/api/service/{id}
Request Param	
Response	204-No Content

3.2.13.2.7. Delete Service by Name

API Name	Delete Service
Request Type	DELETE
Request URL	service/public/v2/api/service/name/{name}
Request Param	
Response	204-No Content

3.2.13.2.8. Search Services

API Name	Search Services
Request Type	GET
Request URL	service/public/v2/api/service
Request Params	<p>Query Parameters:</p> <p>pageSize int <i>The page size required</i></p> <p>startIndex int <i>The startrecord index</i></p> <p>serviceName string <i>The service name</i></p> <p>serviceNamePartial string <i>Partial service name</i></p> <p>serviceType string <i>The service types(such as "hdfs","hive","hbase","knox","storm")</i></p> <p>isEnabled boolean <i>The enabled status (true/false): true is enabled, false otherwise</i></p> <p>Example :</p> <p>pageSize=25&startIndex=0</p>
Response	<pre>[{ "configs": { "fs.default.name": "hdfs://akulkarni- etp-real-final-1.novalocal:8020", "hadoop.security.auth_to_local": "DEFAULT", "hadoop.security.authentication": "simple", "hadoop.security.authorization": "false", "password": "*****", "username": "hadoop" }, "createTime": 1450757397000, "createdBy": "amb_ranger_admin", "description": "hdfs repo", "guid": "ec082eea-0c22-43b8-84e0-129422f689b9", "id": 1, "isEnabled": true, "name": "c1l_hadoop", "policyUpdateTime": 1450757398000, "policyVersion": 2, "tagVersion": 1, "type": "hdfs", "updateTime": 1450757398000, "updatedBy": "amb_ranger_admin", "version": 3 }, { "configs": { "password": "*****", "username": "yarn", "yarn.url": "http://akulkarni-etp- real-final-1.novalocal:8088" }, "createTime": 1450757747000, "createdBy": "amb_ranger_admin", "description": "yarn repo", "guid": "080970a9-2216-4660-962e-2b48046bf87e", "id": 2, "isEnabled": true, "name": "c1l_yarn", "policyUpdateTime": 1450757747000, "policyVersion": 1, "tagVersion": 1, "type": "yarn", "updateTime": 1450757747000,</pre>

API Name	Search Services
	<pre> "updatedBy": "amb_ranger_admin", "version": 2 }, { "configs": { "jdbc.driverClassName": "org.apache. hive.jdbc.HiveDriver", "jdbc.url": "jdbc:hive2://akulkarni- etp-real-final-1.novalocal:10000", "password": "*****", "username": "hive" }, "createTime": 1450757992000, "createdBy": "amb_ranger_admin", "description": "hive repo", "guid": "2bca8f98-4859-43c3-a8f4- d31a15f28793", "id": 3, "isEnabled": true, "name": "c11_hive", "policyUpdateTime": 1450757995000, "policyVersion": 4, "tagUpdateTime": 1450916660000, "tagVersion": 74, "type": "hive", "updateTime": 1450757995000, "updatedBy": "amb_ranger_admin", "version": 78 }, { "configs": { "hadoop.security.authentication": "simple", "hbase.security.authentication": "simple", "hbase.zookeeper.property.clientPort": "2181", "hbase.zookeeper.quorum": "akulkarni- etp-real-final-1.novalocal", "password": "*****", "username": "hbase", "zookeeper.znode.parent": "/hbase- unsecure" }, "createTime": 1450758200000, "createdBy": "amb_ranger_admin", "description": "hbase repo", "guid": "6495d4c9-cd1b-4bdf-a023- bdc82806186f", "id": 4, "isEnabled": true, "name": "c11_hbase", "policyUpdateTime": 1450758202000, "policyVersion": 2, "tagVersion": 1, "type": "hbase", "updateTime": 1450758202000, "updatedBy": "amb_ranger_admin", "version": 3 }, { "configs": { "password": "*****", "username": "kafka", "zookeeper.connect": "akulkarni-etp- real-final-1.novalocal:2181" }, "createTime": 1450758481000, "createdBy": "amb_ranger_admin", "description": "kafka repo", "guid": "bd25a697-7c45-4c75-b23d- bb02071c98c2", "id": 5, "isEnabled": true, "name": "c11_kafka", "policyUpdateTime": 1450805416000, "policyVersion": 2, </pre>

API Name	Search Services
	<pre> "tagVersion": 1, "type": "kafka", "updateTime": 1450805416000, "updatedBy": "amb_ranger_admin", "version": 3 }, { "configs": { "password": "*****", "username": "admin" }, "createTime": 1451348710000, "createdBy": "Admin", "description": "test service", "guid": "e72cb64d-66d7-4632-b5ae- c6966cb52105", "id": 6, "isEnabled": true, "name": "c11_test", "policyUpdateTime": 1451352708000, "policyVersion": 4, "tagVersion": 1, "type": "test", "updateTime": 1451352708000, "updatedBy": "Admin", "version": 6 }] </pre>

3.2.13.3. Policy APIs

- [Get Policy by ID \[411\]](#)
- [Get Policy by Service Name and Policy Name \[412\]](#)
- [Create Policy \[413\]](#)
- [Update Policy by ID \[415\]](#)
- [Update Policy by Service Name and Policy Name \[417\]](#)
- [Delete Policy by ID \[420\]](#)
- [Delete Policy by Service Name and Policy Name \[420\]](#)
- [Search Policies in a Service \[420\]](#)

3.2.13.3.1. Get Policy by ID

API Name	Get Policy
Request Type	Get
Request URL	<code>service/public/v2/api/policy/{id}</code>
Request Params	
Response	<pre> { "allowExceptions": [], "createTime": 1450757397000, "createdBy": "amb_ranger_admin", "denyExceptions": [], "denyPolicyItems": [], "description": "Default Policy for Service: c11_hadoop", "guid": "4c2f7afb-23fa-45e9-9b41-29bdc7423b65", "id": 1, "isAuditEnabled": true, "isEnabled": true, "name": "c11_hadoop-1-20151222040957", "policyItems": [</pre>

API Name	Get Policy
	<pre> { "accesses": [{ "isAllowed": true, "type": "read" }, { "isAllowed": true, "type": "write" }, { "isAllowed": true, "type": "execute" }], "conditions": [], "delegateAdmin": true, "groups": [], "users": ["ambari-qa"] }, "resourceSignature": "6f95606340leda656fleae8870clafac", "resources": { "path": { "isExcludes": false, "isRecursive": true, "values": ["/*"] } }, "service": "cli_hadoop", "updateTime": 1450757398000, "updatedBy": "amb_ranger_admin", "version": 2 } </pre>

3.2.13.3.2. Get Policy by Service Name and Policy Name

API Name	Get Policy
Request Type	Get
Request URL	service/public/v2/api/service/{service-name}/policy/{policy-name}
Request Params	
Response	<pre> { "allowExceptions": [], "createTime": 1450757992000, "createdBy": "amb_ranger_admin", "denyExceptions": [], "denyPolicyItems": [], "description": "Default Policy for Service: cli_hive", "guid": "d6218120-1b66-43e6-9fef-9c917a8e9e25", "id": 4, "isAuditEnabled": true, "isEnabled": true, "name": "cli_hive-2-20151222041952", "policyItems": [{ "accesses": [{ "isAllowed": true, "type": "select" }, { "isAllowed": true, "type": "update" }] }] } </pre>

API Name	Get Policy
	<pre> { "isAllowed": true, "type": "create" }, { "isAllowed": true, "type": "drop" }, { "isAllowed": true, "type": "alter" }, { "isAllowed": true, "type": "index" }, { "isAllowed": true, "type": "lock" }, { "isAllowed": true, "type": "all" }], "conditions": [], "delegateAdmin": true, "groups": [], "users": ["ambari-qa"] }], "resourceSignature": "c834ed2b8c7462d2aa8bbffdb05226c8", "resources": { "database": { "isExcludes": false, "isRecursive": false, "values": ["*"] }, "udf": { "isExcludes": false, "isRecursive": false, "values": ["*"] } }, "service": "cli_hive", "updateTime": 1450757995000, "updatedBy": "amb_ranger_admin", "version": 2 }] } </pre>

3.2.13.3.3. Create Policy

API name	Create Policy
Request Type	POST
Request URL	service/public/v2/api/policy
Request Params	<pre> { "allowExceptions": [], "denyExceptions": [], "denyPolicyItems": [{ "accesses": [{ "isAllowed": true, "type": "drop" }] }], } </pre>

API name	Create Policy
	<pre> "conditions": [], "delegateAdmin": true, "groups": [], "users": ["hadoop"] }, "description": "Policy for Service: cll_test", "isAuditEnabled": true, "isEnabled": true, "name": "c11_test-1", "policyItems": [{ "accesses": [{ "isAllowed": true, "type": "select" }, { "isAllowed": true, "type": "update" }, { "isAllowed": true, "type": "create" }, { "isAllowed": true, "type": "drop" }], "conditions": [], "delegateAdmin": true, "groups": ["public"], "users": [] }], "resources": { "root": { "isExcludes": false, "isRecursive": false, "values": ["abc"] }, "sub": { "isExcludes": false, "isRecursive": false, "values": ["*"] } }, "service": "c11_test", "version": 1 } </pre>
<p>Response</p>	<pre> { "allowExceptions": [], "createTime": 1451350456093, "createdBy": "Admin", "denyExceptions": [], "denyPolicyItems": [{ "accesses": [{ "isAllowed": true, "type": "drop" }], "conditions": [], "delegateAdmin": true, "groups": [], "users": ["hadoop"] }] } </pre>

API name	Create Policy
	<pre> }], "description": "Policy for Service: cll_test", "guid": "ff0b3c4a-6aa0-4803-9314-17f3b8950482", "id": 8, "isAuditEnabled": true, "isEnabled": true, "name": "c11_test-1", "policyItems": [{ "accesses": [{ "isAllowed": true, "type": "select" }, { "isAllowed": true, "type": "update" }, { "isAllowed": true, "type": "create" }, { "isAllowed": true, "type": "drop" }], "conditions": [], "delegateAdmin": true, "groups": ["public"], "users": [] }], "resourceSignature": "8a2fac99ba72c687defacff39d6354fb", "resources": { "root": { "isExcludes": false, "isRecursive": false, "values": ["abc"] }, "sub": { "isExcludes": false, "isRecursive": false, "values": ["*"] } }, "service": "c11_test", "updateTime": 1451350456094, "updatedBy": "Admin", "version": 1 } </pre>

3.2.13.3.4. Update Policy by ID

API Name	update policy
Request Type	PUT
Request URL	service/public/v2/api/policy/{id}
Request Params	<pre> { "id": 8, "allowExceptions": [], "denyExceptions": [], "denyPolicyItems": [{ "accesses": [</pre>

API Name	update policy
	<pre> { "isAllowed": true, "type": "drop" }], "conditions": [], "delegateAdmin": true, "groups": [], "users": ["admin"] }], "description": "Policy for Service: cl1_test", "isAuditEnabled": true, "isEnabled": true, "name": "cl1_test-1", "policyItems": [{ "accesses": [{ "isAllowed": true, "type": "select" }, { "isAllowed": true, "type": "update" }, { "isAllowed": true, "type": "create" }, { "isAllowed": true, "type": "drop" }], "conditions": [], "delegateAdmin": true, "groups": ["public"], "users": [] }], "resources": { "root": { "isExcludes": false, "isRecursive": false, "values": ["abc"] }, "sub": { "isExcludes": false, "isRecursive": false, "values": ["*"] } }, "service": "cl1_test", "version": 1 } </pre>
Response	<pre> { "allowExceptions": [], "createTime": 1451350456000, "createdBy": "Admin", "denyExceptions": [], "denyPolicyItems": [{ "accesses": [{ "isAllowed": true, "type": "drop" }], "conditions": [], </pre>

API Name	update policy
	<pre> "delegateAdmin": true, "groups": [], "users": ["admin"] }], "description": "Policy for Service: cll_test", "guid": "ff0b3c4a-6aa0-4803-9314-17f3b8950482", "id": 8, "isAuditEnabled": true, "isEnabled": true, "name": "c1l_test-1", "policyItems": [{ "accesses": [{ "isAllowed": true, "type": "select" }, { "isAllowed": true, "type": "update" }, { "isAllowed": true, "type": "create" }, { "isAllowed": true, "type": "drop" }], "conditions": [], "delegateAdmin": true, "groups": ["public"], "users": [] }], "resourceSignature": "8a2fac99ba72c687defacff39d6354fb", "resources": { "root": { "isExcludes": false, "isRecursive": false, "values": ["abc"] }, "sub": { "isExcludes": false, "isRecursive": false, "values": ["*"] } }, "service": "c1l_test", "updateTime": 1451955041580, "updatedBy": "Admin", "version": 3 } </pre>

3.2.13.3.5. Update Policy by Service Name and Policy Name

API Name	update policy
Request Type	PUT
Request URL	service/public/v2/api/service/{service-name}/policy/{policy-name}
Request Params	{

API Name	<p>update policy</p> <pre> "allowExceptions": [], "denyExceptions": [], "denyPolicyItems": [{ "accesses": [{ "isAllowed": true, "type": "drop" }], "conditions": [], "delegateAdmin": true, "groups": [], "users": ["hadoop", "admin"] }], "description": "Policy for Service: cll_test", "isAuditEnabled": true, "isEnabled": true, "name": "c11_test-1", "policyItems": [{ "accesses": [{ "isAllowed": true, "type": "select" }, { "isAllowed": true, "type": "update" }, { "isAllowed": true, "type": "create" }, { "isAllowed": true, "type": "drop" }], "conditions": [], "delegateAdmin": true, "groups": ["public"], "users": [] }], "resources": { "root": { "isExcludes": false, "isRecursive": false, "values": ["abc"] }, "sub": { "isExcludes": false, "isRecursive": false, "values": ["*"] } }, "service": "c11_test", "version": 1 } </pre>
	200 - Application/json
Response	<pre> { "allowExceptions": [], "createTime": 1451350456000, "createdBy": "Admin", "denyExceptions": [], "denyPolicyItems": [</pre>

API Name	update policy
	<pre> { "accesses": [{ "isAllowed": true, "type": "drop" }], "conditions": [], "delegateAdmin": true, "groups": [], "users": ["hadoop", "admin"] }], "description": "Policy for Service: cll_test", "guid": "ff0b3c4a-6aa0-4803-9314-17f3b8950482", "id": 8, "isAuditEnabled": true, "isEnabled": true, "name": "c11_test-1", "policyItems": [{ "accesses": [{ "isAllowed": true, "type": "select" }, { "isAllowed": true, "type": "update" }, { "isAllowed": true, "type": "create" }, { "isAllowed": true, "type": "drop" }], "conditions": [], "delegateAdmin": true, "groups": ["public"], "users": [] }], "resourceSignature": "8a2fac99ba72c687defacff39d6354fb", "resources": { "root": { "isExcludes": false, "isRecursive": false, "values": ["abc"] }, "sub": { "isExcludes": false, "isRecursive": false, "values": ["*"] } }, "service": "c11_test", "updateTime": 1451352707567, "updatedBy": "Admin", "version": 2 } </pre>

3.2.13.3.6. Delete Policy by ID

API Name	Delete Policy
Request Type	DELETE
Request URL	service/public/v2/api/policy/{id}
Request Params	
Response	204 - No Content

3.2.13.3.7. Delete Policy by Service Name and Policy Name

API Name	Delete Policy
Request Type	DELETE
Request URL	service/public/v2/api/policy
Request Params	<p><u>Query Parameters:</u></p> <p>servicename string The name of service</p> <p>polycyname string The name of policy</p> <p>Example:</p> <p>servicename=service-name&polycyname=policy-name</p>
Response	204 - No Content

3.2.13.3.8. Search Policies in a Service

API Name	Search Policies in a Service
API Name	Search Policies in a Service
Request Type	GET
Request URL	service/public/v2/api/service/{service-name}/policy
Request Params	<p><u>Query Parameters:</u></p> <p>pageSize int <i>The page size required</i></p> <p>startIndex int <i>The start record index</i></p> <p>policyName string <i>The Exact Name of the policy</i></p> <p>policyNamePartial string <i>The Partial Name of the policy</i></p> <p>policyId string <i>The policy ID</i></p> <p>polResource string <i>The policy resource value</i></p> <p>resource:resource-type string <i>The policy resource value for given resource-type</i></p> <p>user string <i>The user name</i></p> <p>group string <i>The group name</i></p> <p>isRecursive boolean <i>The <u>isRecursive</u> property ("true" or "false")</i></p> <p>isEnabled boolean <i>The enable/disabled property ("true" or "false")</i></p> <p>Example =</p> <p>pageSize=25&startIndex=0&resource:database=finance</p>
Response	[

API Name	Search Policies in a Service
API Name	Search Policies in a Service
	<pre> { "allowExceptions": [], "createTime": 1450757992000, "createdBy": "amb_ranger_admin", "denyExceptions": [], "denyPolicyItems": [], "description": "Default Policy for Service: cll_hive", "guid": "4a322a05-cl7f-4d6c- b291-94cae3e6c353", "id": 3, "isAuditEnabled": true, "isEnabled": true, "name": "c1l_hive-1-20151222041951", "policyItems": [{ "accesses": [{ "isAllowed": true, "type": "select" }, { "isAllowed": true, "type": "update" }, { "isAllowed": true, "type": "create" }, { "isAllowed": true, "type": "drop" }, { "isAllowed": true, "type": "alter" }, { "isAllowed": true, "type": "index" }, { "isAllowed": true, "type": "lock" }, { "isAllowed": true, "type": "all" }], "conditions": [], "delegateAdmin": true, "groups": [], "users": ["ambari-qa"] }], "resourceSignature": "6e79c1c989c79b7e53af663d3bdc2de6", "resources": { "column": { "isExcludes": false, "isRecursive": false, "values": ["*"] }, "database": { "isExcludes": false, "isRecursive": false, "values": ["*"] } } } </pre>

API Name	Search Policies in a Service
API Name	Search Policies in a Service
	<pre> "table": { "isExcludes": false, "isRecursive": false, "values": ["*"] }, "service": "c1l_hive", "updateTime": 1450757994000, "updatedBy": "amb_ranger_admin", "version": 2 }, { "allowExceptions": [], "createTime": 1450757992000, "createdBy": "amb_ranger_admin", "denyExceptions": [], "denyPolicyItems": [], "description": "Default Policy for Service: c1l_hive", "guid": "d6218120-1b66-43e6-9fef-9c917a8e9e25", "id": 4, "isAuditEnabled": true, "isEnabled": true, "name": "c1l_hive-2-20151222041952", "policyItems": [{ "accesses": [{ "isAllowed": true, "type": "select" }, { "isAllowed": true, "type": "update" }, { "isAllowed": true, "type": "create" }, { "isAllowed": true, "type": "drop" }, { "isAllowed": true, "type": "alter" }, { "isAllowed": true, "type": "index" }, { "isAllowed": true, "type": "lock" }, { "isAllowed": true, "type": "all" }], "conditions": [], "delegateAdmin": true, "groups": [], "users": ["ambari-ga"] }], "resourceSignature": "c834ed2b8c7462d2aa8bbffdb05226c8", "resources": { "database": { </pre>

API Name	Search Policies in a Service
	<pre> "isExcludes": false, "isRecursive": false, "values": ["*"] }, "udf": { "isExcludes": false, "isRecursive": false, "values": ["*"] } }, "service": "cli_hive", "updateTime": 1450757995000, "updatedBy": "amb_ranger_admin", "version": 2 }] </pre>

4. Data Protection: Wire Encryption

Encryption is applied to electronic information to ensure its privacy and confidentiality. Wire encryption protects data as it moves into, through, and out of an Hadoop cluster over RPC, HTTP, Data Transfer Protocol (DTP), and JDBC:

- *Clients* typically communicate directly with the Hadoop cluster. Data can be protected using RPC encryption or Data Transfer Protocol:
 - **RPC encryption:** Clients interacting directly with the Hadoop cluster through RPC. A client uses RPC to connect to the NameNode (NN) to initiate file read and write operations. RPC connections in Hadoop use Java's Simple Authentication & Security Layer (SASL), which supports encryption.
 - **Data Transfer Protocol:** The NN gives the client the address of the first DataNode (DN) to read or write the block. The actual data transfer between the client and a DN uses Data Transfer Protocol.
- *Users* typically communicate with the Hadoop cluster using a Browser or a command line tools, data can be protected as follows:
 - **HTTPS encryption:** Users typically interact with Hadoop using a browser or component CLI, while applications use REST APIs or Thrift. Encryption over the HTTP protocol is implemented with the support for SSL across a Hadoop cluster and for the individual components such as Ambari.
 - **JDBC:** HiveServer2 implements encryption with Java SASL protocol's quality of protection (QOP) setting. With this the data moving between a HiveServer2 over JDBC and a JDBC client can be encrypted.
- Additionally, within-cluster communication between processes can be protected using HTTPS encryption during MapReduce shuffle:
 - **HTTPS encryption during shuffle:** When data moves between the Mappers and the Reducers over the HTTP protocol, this step is called shuffle. Reducer initiates the connection to the Mapper to ask for data; it acts as an SSL client.

This chapter provides information about configuring and connecting to wire-encrypted components.

For information about configuring HDFS data-at-rest encryption, see [HDFS "Data at Rest" Encryption](#).

4.1. Enabling RPC Encryption

The most common way for a client to interact with a Hadoop cluster is through RPC. A client connects to a NameNode over RPC protocol to read or write a file. RPC connections in Hadoop use the Java Simple Authentication and Security Layer (SASL) which supports encryption. When the `hadoop.rpc.protection` property is set to `privacy`, the data over RPC is encrypted with symmetric keys.

**Note**

RPC encryption covers not only the channel between a client and a Hadoop cluster but also the inter-cluster communication among Hadoop services.

Enable Encrypted RPC by setting the following properties in `core-site.xml`.

```
hadoop.rpc.protection=privacy
```

(Also supported are the 'authentication' and 'integrity' settings.)

4.2. Enabling Data Transfer Protocol

The NameNode gives the client the address of the first DataNode to read or write the block. The actual data transfer between the client and the DataNode is over Hadoop's Data Transfer Protocol. To encrypt this protocol you must set `dfs.encrypt.data.transfer=true` on the NameNode and all DataNodes. The actual algorithm used for encryption can be customized with `dfs.encrypt.data.transfer.algorithm` set to either "3des" or "rc4". If nothing is set, then the default on the system is used (usually 3DES.) While 3DES is more cryptographically secure, RC4 is substantially faster.

Enable Encrypted DTP by setting the following properties in `hdfs-site.xml`:

```
dfs.encrypt.data.transfer=true
dfs.encrypt.data.transfer.algorithm=3des
```

rc4 is also supported.

**Note**

Secondary NameNode is not supported with the HTTPS port. It can only be accessed via `http://<SNN>:50090`.

4.3. Enabling SSL: Understanding the Hadoop SSL Keystore Factory

The Hadoop SSL Keystore Factory manages SSL for core services that communicate with other cluster services over HTTP, such as MapReduce, YARN, and HDFS. Other components that have services that are typically not distributed, or only receive HTTP connections directly from clients, use built-in Java JDK SSL tools. Examples include HBase and Oozie.

The following table shows HDP cluster services that use HTTP and support SSL for wire encryption.

Table 4.1. Components that Support SSL

Component	Service	SSL Management
Accumulo		JDK: User and default
Ambari	//TODO	//TODO
Atlas	//TODO	//TODO

Component	Service	SSL Management
Falcon	REST API	JDK: User and default
HBase	REST API	Configured in hbase-site.xml
HDFS	WebHDFS	Hadoop SSL Keystore Factory
HDP Security Administration	Server/Agent	JDK: User and default
Hive	HiveServer2	Configured in hive-site.xml
JobHistory	Hadoop SSL Keystore Factory	
Kafka		JDK: User and default
Knox	Hadoop cluster (REST client)	JDK: default only
Knox Gateway server	JDK: User and default	
MapReduce	Shuffle	Hadoop SSL Keystore Factory
Oozie		Configured in oozie-site.xml
Ranger	//TODO	//TODO
Solr		JDK: User and default
TaskTracker	Hadoop SSL Keystore Factory	
Yarn	Resource Manager	Hadoop SSL Keystore Factory

When enabling support for SSL, it is important to know which SSL Management method is being used by the Hadoop service. Services that are co-located on a host must configure the server certificate and keys, and in some cases the client truststore, in the Hadoop SSL Keystore Factory and JDK locations. When using CA signed certificates, configure the Hadoop SSL Keystore Factory to use the Java keystore and truststore locations.

The following list describes major differences between certificates managed by the Hadoop SSL Keystore Management Factory and certificates managed by JDK:

- Hadoop SSL Keystore Management Factory:
 - Supports only JKS formatted keys.
 - Supports toggling the shuffle between HTTP and HTTPS.
 - Supports two way certificate and name validation.
 - Uses a common location for both the keystore and truststore that is available to other Hadoop core services.
 - Allows you to manage SSL in a central location and propagate changes to all cluster nodes.
 - Automatically reloads the keystore and truststore without restarting services.
- SSL Management with JDK:
 - Allows either HTTP or HTTPS.
 - Uses hard-coded locations for truststores and keystores that may vary between hosts. Typically, this requires you to generate key pairs and import certificates on each host.
 - Requires the service to be restarted to reload the keystores and truststores.
 - Requires certificates to be installed in the client CA truststore.



Note

For more information on JDK SSL Management, see "Using SSL" in [Monitoring and Managing Using JMX Technology](#).

4.4. Creating and Managing SSL Certificates

This section contains the following topics:

- Obtaining a certificate from a third-party Certificate Authority (CA)
- Creating an internal CA (OpenSSL)
- Installing Certificates in the Hadoop SSL Keystore Factory (HDFS, MapReduce, and YARN)
- Using an internal CA (OpenSSL)



Note

For more information about the `keytool` utility, see the Oracle `keytool` reference: [keytool - Key and Certificate Management Tool](#).

For more information about OpenSSL, see [OpenSSL Documentation](#).



Note

Java-based Hadoop components such as HDFS, MapReduce, and YARN support JKS format, while Python based services such as Hue use PEM format.

4.4.1. Obtain a Certificate from a Trusted Third-Party Certification Authority (CA)

A third-party Certification Authority (CA) accepts certificate requests from entities, authenticates applications, issues certificates, and maintains status information about certificates. Associated cryptography guarantees that a signed certificate is computationally difficult to forge. Thus, as long as the CA is a genuine and trusted authority, clients have high assurance that they are connecting to the machines that they are attempting to connect with.

To obtain a certificate signed by a third-party CA, generate and submit a Certificate Signing Request (CSR) for each cluster node:

1. From the service user account associated with the component (such as `hive`, `hbase`, `oozie`, or `hdfs`, shown below as `<service_user>`), generate the host key:

```
su -l <service_user> -C "keytool -keystore <client-keystore> -genkey -alias <host>"
```

2. At the prompts, enter the information required by the CSR.



Note

Request generation information and requirements vary depending on the certificate authority. Check with your CA for details.

Example using default keystore `keystore.jks`:

```
su -l hdfs -c "keytool -keystore keystore.jks -genkey -alias n3"

Enter keystore password: *****
What is your first and last name?
[Unknown]: hortonworks.com
What is the name of your organizational unit?
[Unknown]: Development
What is the name of your organization?
[Unknown]: Hortonworks
What is the name of your City or Locality?
[Unknown]: SantaClara
What is the name of your State or Province?
[Unknown]: CA
What is the two-letter country code for this unit?
[Unknown]: US
Is <CN=hortonworks.com, OU=Development, O=Hortonworks, L=SantaClara, ST=CA, C=US correct?
[no]: yes

Enter key password for <host>
(RETURN if same as keystore password):
```

By default, `keystore` uses JKS format for the keystore and truststore. The keystore file is created in the user's home directory. Access to the keystore requires the password and alias.

3. Verify that the key was generated; for example:

```
su -l hdfs -c "keytool -list -v -keystore keystore.jks"
```

4. Create the CSR file:

```
su -l hdfs -c "keytool -keystore <keystorename> -certreq -alias <host> -keyalg rsa -file <host>.csr"
```

This command generates a certificate signing request that can be sent to a CA. The file `<host>.csr` contains the CSR.

The CSR is created in the user's home directory.

5. Confirm that the `keystore.jks` and `<host>.csr` files exist by running the following command and making sure that the files are listed in the output:

```
su -l hdfs -c "ls ~/"
```

6. Submit the CSR to your Certificate Authority.

7. To import and install keys and certificates, follow the instructions sent to you by the CA.

4.4.2. Create and Set Up an Internal CA (OpenSSL)

OpenSSL provides tools to allow you to create your own private certificate authority.

Considerations:

- The encryption algorithms may be less secure than a well-known, trusted third-party.
- Unknown CAs require that the certificate be installed in corresponding client truststores.



Note

When accessing the service from a client application such as HiveCLI or cURL, the CA must resolve on the client side or the connection attempt may fail. Users accessing the service through a browser will be able to add an exception if the certificate cannot be verified in their local truststore.

Prerequisite: Install `openssl`. For example, on CentOS run `yum install openssl`.

To create and set up a CA:

1. Generate the key and certificate for a component process.

The first step in deploying HTTPS for a component process (for example, Kafka broker) is to generate the key and certificate for each node in the cluster. You can use the Java `keytool` utility to accomplish this task. Start with a temporary keystore, so that you can export and sign it later with the CA.

Use the following `keytool` command to create the key and certificate:

```
$ keytool -keystore <keystore-file> -alias localhost -validity <validity> -genkey
```

where:

`<keystore-file>` is the keystore file that stores the certificate. The keystore file contains the private key of the certificate; therefore, it needs to be kept safely.

`<validity>` is the length of time (in days) that the certificate will be valid.

Make sure that the common name (CN) matches the fully qualified domain name (FQDN) of the server. The client compares the CN with the DNS domain name to ensure that it is indeed connecting to the desired server, not a malicious server.

2. Create the Certificate Authority (CA)

After step 1, each machine in the cluster has a public-private key pair and a certificate that identifies the machine. The certificate is unsigned, however, which means that an attacker can create such a certificate to pretend to be any machine.

To prevent forged certificates, it is very important to sign the certificates for each machine in the cluster.

A CA is responsible for signing certificates, and associated cryptography guarantees that a signed certificate is computationally difficult to forge. Thus, as long as the CA is a

genuine and trusted authority, the clients have high assurance that they are connecting to the machines that they are attempting to connect with.

Here is a sample `openssl` command to generate a CA:

```
openssl req -new -x509 -keyout ca-key -out ca-cert -days 365
```

The generated CA is simply a public-private key pair and certificate, intended to sign other certificates.

3. Add the generated CA to the *server's* truststore:

```
keytool -keystore server.truststore.jks -alias CARoot -import -file ca-cert
```

4. Add the generated CA to the *client's* truststore, so that clients know that they can trust this CA:

```
keytool -keystore client.truststore.jks -alias CARoot -import -file ca-cert
```

In contrast to the keystore in step 1 that stores each machine's own identity, the truststore of a client stores all of the certificates that the client should trust. Importing a certificate into one's truststore also means trusting all certificates that are signed by that certificate.

Trusting the CA means trusting all certificates that it has issued. This attribute is called a "chain of trust," and is particularly useful when deploying SSL on a large cluster. You can sign all certificates in the cluster with a single CA, and have all machines share the same truststore that trusts the CA. That way all machines can authenticate all other machines.

5. Sign all certificates generated in Step 1 with the CA generated in Step 2:

- a. Export the certificate from the keystore:

```
keytool -keystore server.keystore.jks -alias localhost -certreq -file cert-file
```

- b. Sign the certificate with the CA:

```
openssl x509 -req -CA ca-cert -CAkey ca-key -in cert-file -out cert-signed -days <validity> -CAcreateserial -passin pass:<ca-password>
```

6. Import the CA certificate and the signed certificate into the keystore. For example:

```
$ keytool -keystore server.keystore.jks -alias CARoot -import -file ca-cert
$ keytool -keystore server.keystore.jks -alias localhost -import -file cert-signed
```

The parameters are defined as follows:

Parameter	Description
keystore	The location of the keystore
ca-cert	The certificate of the CA
ca-key	The private key of the CA
ca-password	The passphrase of the CA
cert-file	The exported, unsigned certificate of the server
cert-signed	The signed certificate of the server

All of the preceding steps can be placed into a bash script.

In the following example, note that one of the commands assumes a password of test1234. Specify your own password before running the script.

```
#!/bin/bash

#Step 1
keytool -keystore server.keystore.jks -alias localhost -validity 365 -genkey

#Step 2
openssl req -new -x509 -keyout ca-key -out ca-cert -days 365
keytool -keystore server.truststore.jks -alias CARoot -import -file ca-cert
keytool -keystore client.truststore.jks -alias CARoot -import -file ca-cert

#Step 3
keytool -keystore server.keystore.jks -alias localhost -certreq -file cert-
file
openssl x509 -req -CA ca-cert -CAkey ca-key -in cert-file -out cert-signed -
days 365 -CAcreateserial -passin pass:test1234
keytool -keystore server.keystore.jks -alias CARoot -import -file ca-cert
keytool -keystore server.keystore.jks -alias localhost -import -file cert-
signed
```

To finish the setup process:

1. Set up the CA directory structure:

```
mkdir -m 0700 /root/CA /root/CA/certs /root/CA/crl /root/CA/newcerts /root/
CA/private
```

2. Move the CA key to /root/CA/private and the CA certificate to /root/CA/certs.

```
mv ca.key /root/CA/private;mv ca.crt /root/CA/certs
```

3. Add required files:

```
touch /root/CA/index.txt; echo 1000 >> /root/CA/serial
```

4. Set permissions on the ca.key:

```
chmod 0400 /root/ca/private/ca.key
```

5. Open the OpenSSL configuration file:

```
vi /etc/pki/tls/openssl.cnf
```

6. Change the directory paths to match your environment:

```
[ CA_default ]

dir                = /root/CA                # Where everything is kept
certs              = /root/CA/certs          # Where the issued certs are
kept
crl_dir            = /root/CA/crl            # Where the issued crl are kept
database           = /root/CA/index.txt      # database index file.
```

```

#unique_subject = no                # Set to 'no' to allow creation
of                                  # several certificates with same
subject.
new_certs_dir    = /root/CA/newcerts # default place for new certs.

certificate      = /root/CA/cacert.pem # The CA certificate
serial          = /root/CA/serial     # The current serial number
crlnumber       = /root/CA/crlnumber  # the current crl number
# must be commented out to leave
a V1 CRL
crl             = $dir/crl.pem        # The current CRL
private_key     = /root/CA/private/cakey.pem # The private key
RANDFILE       = /root/CA/private/.rand # private random number file

x509_extensions = usr_cert          # The extensions to add to the cert

```

7. Save the changes and restart OpenSSL.

Example of setting up an OpenSSL internal CA:

```

openssl genrsa -out ca.key 8192; openssl req -new -x509 -extensions v3_ca -key
ca.key -out ca.crt -days 365

Generating RSA private key, 8192 bit long modulus
.....++
.....++
e is 65537 (0x10001)
You are about to be asked to enter information that will be incorporated
into your certificate request.
What you are about to enter is what is called a Distinguished Name or a DN.
There are quite a few fields but you can leave some blank
For some fields there will be a default value,
If you enter '.', the field will be left blank.

Country Name (2 letter code) [XX]:US
State or Province Name (full name) []:California
Locality Name (eg, city) [Default City]:SantaClara
Organization Name (eg, company) [Default Company Ltd]:Hortonworks
Organizational Unit Name (eg, section) []:
Common Name (eg, your name or your server's hostname) []:nn
Email Address []:it@hortonworks.com

mkdir -m 0700 /root/CA /root/CA/certs /root/CA/crl /root/CA/newcerts /root/CA/
private
ls /root/CA
certs crl newcerts private

```

4.4.3. Installing Certificates in the Hadoop SSL Keystore Factory (HDFS, MapReduce, and YARN)

HDFS, MapReduce, and YARN use the Hadoop SSL Keystore Factory to manage SSL Certificates. This factory uses a common directory for server keystore and client truststore. The Hadoop SSL Keystore Factory allows you to use CA certificates managed in their own stores.

1. Create a directory for the server and client stores.

```
mkdir -p <SERVER_KEY_LOCATION> ; mkdir -p <CLIENT_KEY_LOCATION>
```

2. Import the server certificate from each node into the HTTP Factory truststore.

```
cd <SERVER_KEY_LOCATION> ; keytool -import -noprompt -alias <remote-  
hostname> -file <remote-hostname>.jks -keystore <TRUSTSTORE_FILE> -storepass  
<SERVER_TRUSTSTORE_PASSWORD>
```

3. Create a single truststore file containing the public key from all certificates, by importing the public key for each CA or from each self-signed certificate pair:

```
keytool -import -noprompt -alias <host> -file $CERTIFICATE_NAME -keystore  
<ALL_JKS> -storepass <CLIENT_TRUSTSTORE_PASSWORD>
```

4. Copy the keystore and truststores to every node in the cluster.

5. Validate the common truststore file on all hosts.

```
keytool -list -v -keystore <ALL_JKS> -storepass <CLIENT_TRUSTSTORE_PASSWORD>
```

6. Set permissions and ownership on the keys:

```
chgrp -R <YARN_USER>:hadoop <SERVER_KEY_LOCATION>  
chgrp -R <YARN_USER>:hadoop <CLIENT_KEY_LOCATION>  
chmod 755 <SERVER_KEY_LOCATION>  
chmod 755 <CLIENT_KEY_LOCATION>  
chmod 440 <KEYSTORE_FILE>  
chmod 440 <TRUSTSTORE_FILE>  
chmod 440 <CERTIFICATE_NAME>  
chmod 444 <ALL_JKS>
```



Note

The complete path of the <SERVER_KEY_LOCATION> and the <CLIENT_KEY_LOCATION> from the root directory /etc must be owned by the yarn user and the hadoop group.

4.4.4. Using a CA-Signed Certificate

To use a CA-signed certificate:

1. Run the following command to create a self-signing rootCA and import the rootCA into the client truststore. This is a private key; it should be kept private. The following command creates a 2048-bit key:

```
openssl genrsa -out <clusterCA>.key 2048
```

2. Self-sign the rootCA. The following command signs for 300 days. It will start an interactive script that requests name and location information.

```
openssl req -x509 -new -key <clusterCA>.key -days 300 -out <clusterCA>
```

3. Import the rootCA into the client truststore:

```
keytool -importcert -alias <clusterCA> -file $clusterCA -keystore  
<clustertruststore> -storepass <clustertruststorekey>
```



Note

Make sure that the `ssl-client.xml` file on every host is configured to use this `$clustertrust store`.

When configuring with Hive point to this file; when configuring other services install the certificate in the Java truststore.

4. For each host, sign the `certreq` file with the `rootCA`:

```
openssl x509 -req -CA $clusterCA.pem -CAkey <clusterCA>.key -in <host>.cert
-out $host.signed -days 300 -CAcreateserial
```

5. On each host, import the `rootCA` and the signed cert back in:

```
keytool -keystore <hostkeystore> -storepass <hoststorekey> -alias
<clusterCA> -import -file cluster1CA.pem
keytool -keystore <hostkeystore> -storepass <hoststorekey> -alias `hostname
-s` -import -file <host>.signed -keypass <hostkey>
```

4.5. Enabling SSL for HDP Components

The following table contains links to instructions for enabling SSL on specific HDP components.



Note

These instructions assume that you have already created keys and signed certificates for each component of interest, across your cluster. (See [Section 4.3, "Enabling SSL: Understanding the Hadoop SSL Keystore Factory" \[425\]](#) for more information.)

Table 4.2. Configure SSL Data Protection for HDP Components

HDP Component	Notes/Link
Hadoop, MapReduce, YARN	Section 4.1, "Enabling RPC Encryption" [424] ; Section 4.6, "Enable SSL for WebHDFS, MapReduce Shuffle, Tez, and YARN" [435]
Oozie	Section 4.8, "Enable SSL on Oozie" [439]
HBase	Section 4.9, "Enable SSL on the HBase REST Server" [440]
Hive (HiveServer2)	Section 4.11, "Enable SSL on HiveServer2" [443]
Kafka	Section 4.12, "Enable SSL for Kafka Clients" [445]
Ambari Server	Set Up SSL for Ambari
Falcon	Enabled by default (see Installing the Falcon Package)
Sqoop	Clients of Hive and HBase, see Hortonworks Data Platform Data Movement and Integration, Apache Sqoop
Knox Gateway	Configure SSL for Knox [455]
Flume	Apache Flume Component Guide
Accumulo	Apache Foundation Blog, Apache Accumulo: Generating Keystores for configuring Accumulo with SSL

HDP Component	Notes/Link
Phoenix	Command Line Installation , Installing Apache Phoenix: Configuring Phoenix for Security and Apache Phoenix, Flume Plug-in
HUE	Command Line Installation , Installing Hue , Configure Hue

4.6. Enable SSL for WebHDFS, MapReduce Shuffle, Tez, and YARN

This section explains how to set up SSL for WebHDFS, YARN and MapReduce. Before you begin, make sure that the SSL certificate is properly configured, including the keystore and truststore that will be used by WebHDFS, MapReduce, and YARN.

HDP supports the following SSL modes:

- One-way SSL: SSL client validates the server identity only.
- Mutual authentication (2WAY SSL): The server and clients validate each others' identities. 2WAY SSL can cause performance delays and is difficult to set up and maintain.



Note

In order to access SSL enabled HDP Services through the Knox Gateway, additional configuration on the Knox Gateway is required, see [Configure SSL for Knox](#).

To enable one-way SSL set the following properties and restart all services:

1. Set the following property values (or add the properties if required) in `core-site.xml`:

```
hadoop.ssl.require.client.cert=false
```

```
hadoop.ssl.hostname.verifier=DEFAULT
```

```
hadoop.ssl.keystores.factory.class=org.apache.hadoop.security.ssl.FileBasedK
```

```
hadoop.ssl.server.conf=ssl-server.xml
```

```
hadoop.ssl.client.conf=ssl-client.xml
```



Note

Specify the `hadoop.ssl.server.conf` and `hadoop.ssl.client.conf` values as the relative or absolute path to Hadoop SSL Keystore Factory configuration files. If you specify only the file name, put the files in the same directory as the `core-site.xml`.

2. Set the following properties (or add the properties if required) in `hdfs-site.xml`:

- `dfs.http.policy=<Policy>`

- `dfs.client.https.need-auth=true` (optional for mutual client/server certificate validation)
- `dfs.datanode.https.address=<hostname>:50475`
- `dfs.namenode.https-address=<hostname>:50470`

where `<Policy>` is either:

- `HTTP_ONLY`: service is provided only on HTTP
- `HTTPS_ONLY`: service is provided only on HTTPS
- `HTTP_AND_HTTPS`: service is provided both on HTTP and HTTPS

3. Set the following properties in `mapred-site.xml`:

```
mapreduce.jobhistory.http.policy=HTTPS_ONLY
mapreduce.jobhistory.webapp.https.address=<JHS>:<JHS_HTTPS_PORT>
mapreduce.ssl.enabled=true
mapreduce.shuffle.ssl.enabled=true
```

4. Set the following properties in `yarn-site.xml`:

```
yarn.http.policy=HTTPS_ONLY
yarn.log.server.url=https://<JHS>:<JHS_HTTPS_PORT>/jobhistory/logs
yarn.resourcemanager.webapp.https.address=<RM>:<RM_HTTPS_PORT>
yarn.nodemanager.webapp.https.address=0.0.0.0:<NM_HTTPS_PORT>
```

5. Create an `ssl-server.xml` file for the Hadoop SSL Keystore Factory:

- a. Copy the example SSL Server configuration file and modify the settings for your environment:

```
cp /etc/hadoop/conf/ssl-server.xml.example /etc/hadoop/conf/ssl-server.xml
```

- b. Configure the server SSL properties:

Table 4.3. Configuration Properties in `ssl-server.xml`

Property	Default Value	Description
<code>ssl.server.keystore.type</code>	JKS	The type of the keystore, JKS = Java Keystore, the de-facto standard in Java
<code>ssl.server.keystore.location</code>	None	The location of the keystore file
<code>ssl.server.keystore.password</code>	None	The password to open the keystore file
<code>ssl.server.truststore.type</code>	JKS	The type of the trust store
<code>ssl.server.truststore.location</code>	None	The location of the truststore file
<code>ssl.server.truststore.password</code>	None	The password to open the truststore

For example:

```
<property>
```



```
<name>ssl.server.truststore.location</name>
<value>/etc/security/serverKeys/truststore.jks</value>
<description>Truststore to be used by NN and DN. Must be specified.</
description>
</property>

<property>
  <name>ssl.server.truststore.password</name>
  <value>changeit</value>
  <description>Optional. Default value is "".
  </description>
</property>

<property>
  <name>ssl.server.truststore.type</name>
  <value>jks</value>
  <description>Optional. The keystore file format, default value is
  "jks".</description>
</property>

<property>
  <name>ssl.server.truststore.reload.interval</name>
  <value>10000</value>
  <description>Truststore reload check interval, in milliseconds.
  Default value is 10000 (10 seconds).</description>
</property>

<property>
  <name>ssl.server.keystore.location</name>
  <value>/etc/security/serverKeys/keystore.jks</value>
  <description>Keystore to be used by NN and DN. Must be specified.</
description>
</property>

<property>
  <name>ssl.server.keystore.password</name>
  <value>changeit</value>
  <description>Must be specified.</description>
</property>

<property>
  <name>ssl.server.keystore.keypassword</name>
  <value>changeit</value>
  <description>Must be specified.</description>
</property>

<property>
  <name>ssl.server.keystore.type</name>
  <value>jks</value>
  <description>Optional. The keystore file format, default value is
  "jks".</description>
</property>
```

6. Create an `ssl-client.xml` file for the Hadoop SSL Keystore Factory:

a. Copy the client truststore example file:

```
cp /etc/hadoop/conf/ssl-server.xml.example /etc/hadoop/conf/ssl-server.xml
```

- b. Configure the client trust store values:

```
ssl.client.truststore.location=/etc/security/clientKeys/all.jks
ssl.client.truststore.password=clientTrustStorePassword
ssl.client.truststore.type=jks
```

7. Set the following properties in the `tez-site.xml` file:

```
tez.runtime.shuffle.ssl.enable=true
tez.runtime.shuffle.keep-alive.enabled=true
```

8. Copy the configuration files (`core-site.xml`, `hdfs-site.xml`, `mapred-site.xml`, `yarn-site.xml`, `ssl-server.xml`, `tez-site.xml` and `ssl-client.xml`), including the `ssl-server` and `ssl-client` store files if the Hadoop SSL Keystore Factory uses its own keystore and truststore files, to all nodes in the cluster.
9. Restart services on all nodes in the cluster.

4.7. Enable SSL for HttpFS

Use the following steps to configure [HttpFS](#) to work over SSL.

1. Edit the `httpfs-env.sh` script in the configuration directory and set `HTTPFS_SSL_ENABLED` to `true`.

In addition, the following 2 properties can be defined (shown here with default values):

- `HTTPFS_SSL_KEYSTORE_FILE=$HOME/.keystore`
- `HTTPFS_SSL_KEYSTORE_PASS=password`

2. In the HttpFS `tomcat/conf` directory, replace the `server.xml` file with the `ssl-server.xml` file.
3. Create an SSL certificate for the HttpFS server. As the `httpfs` Unix user, use the Java `keytool` command to create the SSL certificate:

```
$ keytool -genkey -alias tomcat -keyalg RSA
```

You will be asked a series of questions in an interactive prompt. It will create the keystore file, which will be named `.keystore` and located in the `httpfs` user home directory.

The password you enter for “keystore password” must match the value of the `HTTPFS_SSL_KEYSTORE_PASS` environment variable set in the `httpfs-env.sh` script in the configuration directory.

The answer to “What is your first and last name?” (i.e. “CN”) must be the host name of the machine where the HttpFS Server will be running.

4. Start HttpFS. It should work over HTTPS.
5. Utilizing the Hadoop FileSystem API or the Hadoop FS shell, use the `swebhdfs://` scheme. Make sure the JVM is picking up the truststore containing the public key of the SSL certificate if you are using a self-signed certificate.

4.8. Enable SSL on Oozie

The default SSL configuration makes all Oozie URLs use HTTPS except for the JobTracker callback URLs. This simplifies the configuration because no changes are required outside of Oozie. Oozie inherently does not trust the callbacks, they are used as hints.



Note

Before you begin ensure that the SSL certificate has been generated and properly configured. By default Oozie uses the user default keystore. In order to access SSL enabled HDP Services through the Knox Gateway, additional configuration on the Knox Gateway is required, see [Configure SSL for Knox](#).

1. If Oozie server is running, stop Oozie.
2. Change the Oozie environment variables for HTTPS if required:
 - OOZIE_HTTPS_PORT set to Oozie HTTPS port. The default value is 11443.
 - OOZIE_HTTPS_KEYSTORE_FILE set to the keystore file that contains the certificate information. Default value `$(HOME)/.keystore`, that is the home directory of the Oozie user.
 - OOZIE_HTTPS_KEYSTORE_PASS set to the password of the keystore file. Default value password.



Note

See [Oozie Environment Setup](#) for more details.

3. Run the following command to enable SSL on Oozie:

```
su -l oozie -c "/usr/hdp/current/oozie-server/bin/oozie-setup.sh prepare-war -secure"
```

4. Start the Oozie server.



Note

To revert back to unsecured HTTP, run the following command:

```
su -l oozie -c "/usr/hdp/current/oozie-server/bin/oozie-setup.sh prepare-war"
```

4.8.1. Configure the Oozie Client to Connect Using SSL

Use the following procedure to configure the Oozie client to connect using SSL. The first two steps are only necessary if you are using a self-signed Certificate. Also, these steps must be performed on every machine on which you intend to use the Oozie Client.

1. Copy or download the `.cert` file onto the client machine.

2. Run the following command (as root) to import the certificate into the JRE keystore. This will allow any Java program, including the Oozie client, to connect to the Oozie Server using the self-signed certificate.

```
sudo keytool -import -alias tomcat -file path/to/certificate.cert -keystore  
${JRE_cacerts}
```

Where `${JRE_cacerts}` is the path to the JRE `.certs` file. Its location may differ depending on the operating system, but its typically named `cacerts` and is located at `${JAVA_HOME}/lib/security/cacerts`, but it may be in a different directory under `${JAVA_HOME}` (you may want to create a backup copy of this file first). The default password is `changeit`.

3. When using the Oozie Client, you must use `https://`
`oozie.server.hostname:11443/oozie` rather than `http://`
`oozie.server.hostname:11000/oozie` – Java will not automatically redirect from the HTTP address to the HTTPS address.

4.8.2. Connect to the Oozie Web UI Using SSL

Use `https://oozie.server.hostname:11443/oozie` to connect to the Oozie web UI using SSL, but most browsers should redirect if you use `http://`
`oozie.server.hostname:11000/oozie`.



Important

If you are using a self-signed certificate, the browser will warn you that it cannot verify the certificate. You will probably need to add the certificate as an exception.

4.8.3. Configure Oozie HCatalogJob Properties

Integrate Oozie HCatalog by adding following property to `oozie-hcatalog` `job.properties`. For example if you are using Ambari, set the properties as:

```
hadoop.rpc.protection=privacy
```



Note

This property is in addition to any properties you must set for secure clusters.

4.9. Enable SSL on the HBase REST Server

Perform the following task to enable SSL on an HBase REST API.



Note

In order to access SSL-enabled HDP Services through the Knox Gateway, additional configuration on the Knox Gateway is required, see [Configure SSL for Knox](#).

1. Create and install an SSL certificate for HBase, for example to use a self-signed certificate:

a. Create an HBase keystore:

```
su -l hbase -c "keytool -genkey -alias hbase -keyalg RSA -keysize 1024 -keystore hbase.jks"
```

At the keytool command prompt:

- Enter the key password
- Enter the keystore password

**Note**

Add these two specified values to the corresponding properties in `hbase-site.xml` in step 2.

b. Export the certificate:

```
su -l hbase -c "keytool -exportcert -alias hbase -file certificate.cert -keystore hbase.jks"
```

c. (Optional) Add certificate to the Java keystore:

- If you are not root run:

```
sudo keytool -import -alias hbase -file certificate.cert -keystore /usr/jdk64/jdk1.7.0_45/jre/lib/security/cacerts
```

- If you are root:

```
keytool -import -alias hbase -file certificate.cert -keystore /usr/jdk64/jdk1.7.0_45/jre/lib/security/cacerts
```

2. Add the following properties to the `hbase-site.xml` configuration file on each node in your HBase cluster:

```
<property>
  <name>hbase.rest.ssl.enabled</name>
  <value>true</value>
</property>

<property>
  <name>hbase.rest.ssl.keystore.store</name>
  <value>/path/to/keystore</value>
</property>

<property>
  <name>hbase.rest.ssl.keystore.password</name>
  <value>keystore-password</value>
</property>

<property>
  <name>hbase.rest.ssl.keystore.keypassword</name>
  <value>key-password</value>
</property>
```

3. Restart all HBase nodes in the cluster.



Note

For clusters using self-signed certificates: Define the truststore as a custom property on the JVM. If the self-signed certificate is not added to the system truststore (cacerts), specify the Java KeyStore (.jks) file containing the certificate in applications by invoking the `javax.net.ssl.trustStore` system property. Run the following command argument in the application client container to use a self-signed certificate in a .jks file:

```
-Djavax.net.ssl.trustStore=/path/to/keystore
```

4.10. Enable SSL on the HBase Web UI

Perform the following task to enable SSL and TLS on an HBase Web UI.



Note

In order to access SSL-enabled HDP Services through the Knox Gateway, additional configuration on the Knox Gateway is required, see [Configure SSL for Knox](#).

1. Create and install an SSL certificate for HBase, for example to use a self-signed certificate:

- a. Create an HBase keystore:

```
su -l hbase -c "keytool -genkey -alias hbase -keyalg RSA -keysize 1024 -keystore hbase.jks"
```

At the keytool command prompt:

- Enter the key password
- Enter the keystore password



Note

Add these two specified values to the corresponding properties in `hbase-site.xml` in step 2.

- b. Export the certificate:

```
su -l hbase -c "keytool -exportcert -alias hbase -file certificate.cert -keystore hbase.jks"
```

- c. **(Optional)** Add certificate to the Java keystore:

- If you are not root run:

```
sudo keytool -import -alias hbase -file certificate.cert -keystore /usr/jdk64/jdk1.7.0_45/jre/lib/security/cacerts
```

- If you are root:

```
keytool -import -alias hbase -file certificate.cert -keystore /usr/
jdk64/jdk1.7.0_45/jre/lib/security/cacerts
```

2. Add the following properties to the `hbase-site.xml` configuration file on each node in your HBase cluster:

```
<property>
  <name>hbase.ssl.enabled</name>
  <value>true</value>
</property>

<property>
  <name>hadoop.ssl.enabled</name>
  <value>true</value>
</property>

<property>
  <name>ssl.server.keystore.keypassword</name>
  <value>key-password</value>
</property>

<property>
  <name><ssl.server.keystore.password</name>
  <value>keystore-password</value>
</property>

<property>
  <name>ssl.server.keystore.location</name>
  <value>/tmp/server-keystore.jks</value>
</property>
```

3. Restart all HBase nodes in the cluster.



Note

For clusters using self-signed certificates: Define the truststore as a custom property on the JVM. If the self-signed certificate is not added to the system truststore (cacerts), specify the Java KeyStore (.jks) file containing the certificate in applications by invoking the `javax.net.ssl.trustStore` system property. Run the following command argument in the application client container to use a self-signed certificate in a .jks file:

```
-Djavax.net.ssl.trustStore=/path/to/keystore
```

4.11. Enable SSL on HiveServer2

When using HiveServer2 without Kerberos authentication, you can enable SSL.



Note

In order to access SSL enabled HDP Services through the Knox Gateway, additional configuration on the Knox Gateway is required, see [Apache Knox Gateway Administrator Guide, Gateway Security, Configure Wire Encryption](#).

Perform the following steps on the HiveServer2:

1. Log into the cluster as the `hive` user. Having `hive` user permissions when creating the Java keystore file sets up the proper `user: :group` ownership, which allows `HiveServer` to access the file and prevents `HiveServer` startup failure.
2. Run the following command to create a keystore for `hiveserver2`:

```
keytool -genkey -alias hbase -keyalg RSA -keysize 1024 -keystore hive.jks
```

3. Edit the `hive-site.xml`, set the following properties to enable SSL:

```
<property>
  <name>hive.server2.use.SSL</name>
  <value>true</value>
  <description></description>
</property>

<property>
  <name>hive.server2.keystore.path</name>
  <value>keystore-file-path</value>
  <description></description>
</property>

<property>
  <name>hive.server2.keystore.password</name>
  <value>keystore-file-password</value>
  <description></description>
</property>
```

4.



Note

When `hive.server2.transport.mode` is binary and `hive.server2.authentication` is `KERBEROS`, SSL encryption does not currently work. Set `hive.server2.thrift.sasl.qop` to `auth-conf` to enable encryption

On the client-side, specify SSL settings for Beeline or JDBC client as follows:

```
jdbc:hive2://<host>:<port>/<database>;ssl=true;sslTrustStore=<path-to-truststore>;trustStorePassword=<password>
```

4.11.1. Setting up SSL with self-signed certificates



Note

In product systems, use a CA-signed SSL certificated rather than a self-signed certificated. A self-signed certificated is a good way to test before deploying in production.

Use the following steps to create and verify self-signed SSL certificates for use with `HiveServer2`:

1. List the keystore entries to verify that the certificate was added. Note that a keystore can contain multiple such certificates: `keytool -list -keystore keystore.jks`
2. Export this certificate from `keystore.jks` to a certificate file: `keytool -export -alias example.com -file example.com.crt -keystore keystore.jks`

3. Add this certificate to the client's truststore to establish trust: `keytool -import -trustcacerts -alias example.com -file example.com.crt -keystore truststore.jks`
4. Verify that the certificate exists in truststore.jks: `keytool -list -keystore truststore.jks`
5. Then start HiveServer2, and try to connect with beeline using:
`jdbc:hive2://<host>:<port>/<database>;ssl=true;sslTrustStore=<path-to-truststore>;trustStorePassword=<truststore-password>`

4.11.2. Selectively disabling SSL protocol versions

To disable specific SSL protocol versions, use the following steps:

1. Run `openssl ciphers -v` (or the corresponding command if not using openssl) to view all protocol versions.
2. In addition to 1, an additional step of going over the HiveServer2 logs may be required to see all the protocols that the node running HiveServer2 is supporting. For that, search for "SSL Server Socket Enabled Protocols:" in the HiveServer2 log file.
3. Add all the SSL protocols that need to be disabled to `hive.ssl.protocol.blacklist`. Ensure that the property in `hiveserver2-site.xml` does not override that in `hive-site.xml`.

4.12. Enable SSL for Kafka Clients

Kafka allows clients to connect over SSL. By default SSL is disabled, but it can be enabled as needed.

Before you begin, be sure to generate the key, SSL certificate, keystore, and truststore that will be used by Kafka.

4.12.1. Configuring the Kafka Broker

The Kafka Broker supports listening on multiple ports and IP addresses. To enable this feature, specify one or more comma-separated values in the `listeners` property in `server.properties`.

Both PLAINTEXT and SSL ports are required if SSL is not enabled for inter-broker communication (see the following subsection for information about enabling inter-broker communication):

```
listeners=PLAINTEXT://host.name:port,SSL://host.name:port
```

The following SSL configuration settings are needed on the broker side:

```
ssl.keystore.location = /var/private/ssl/kafka.server.keystore.jks
ssl.keystore.password = test1234
ssl.key.password = test1234
ssl.truststore.location = /var/private/ssl/kafka.server.truststore.jks
ssl.truststore.password = test1234
```

The following optional settings are available:

Property	Description	Value(s)
<code>ssl.client.auth</code>	Specify whether client authentication is required, requested, or not required. none: no client authentication. required: client authentication is required. requested: client authentication is requested, but a client without certs can still connect. Note: If you set <code>ssl.client.auth</code> to <code>requested</code> or <code>required</code> , then you must provide a truststore for the Kafka broker. The truststore should contain all CA certificates that are used to sign clients' keys.	none
<code>ssl.cipher.suites</code>	Specify one or more cipher suites: named combinations of authentication, encryption, MAC and key exchange algorithms used to negotiate the security settings for a network connection using the TLS or SSL network protocol.	
<code>ssl.enabled.protocols</code>	Specify the SSL protocols that you will accept from clients. Note: SSL is deprecated; its use in production is not recommended.	TLsv1.2, TLsv1.1, TLsv1
<code>ssl.keystore.type</code>	Specify the SSL keystore type.	JKS
<code>ssl.truststore.type</code>	Specify the SSL truststore type.	JKS

Enabling SSL for Inter-Broker Communication

To enable SSL for inter-broker communication, add the following setting to the broker properties file (default is PLAINTEXT):

```
security.inter.broker.protocol = SSL
```

Enabling Additional Cipher Suites

To enable any cipher suites other than the defaults that come with JVM (see [Java Cryptography documentation](#)), you will need to install JCE Unlimited Strength Policy files ([download link](#)).

Validating the Configuration

After you start the broker, you should see the following information in the `server.log` file:

```
with addresses: PLAINTEXT -> EndPoint(192.168.64.1,9092,PLAINTEXT), SSL -> EndPoint(192.168.64.1,9093,SSL)
```

To make sure that the server keystore and truststore are set up properly, run the following command:

```
openssl s_client -debug -connect localhost:9093 -tls1
```

(Note: TLSv1, TLSv1.1, and TLSv1.2 should be listed under `ssl.enabled.protocols`)

In the `openssl` output you should see the server certificate; for example:

```
Server certificate
-----BEGIN CERTIFICATE-----
MIID+DCCAUACCQCx2Rz1tXx3NTANBgkqhkiG9w0BAQsFADB6MQswCQYDVQQGEwJV
UzELMAkGA1UECAwCQ0ExFDASBgNVBACMC1NhbnRhIENsYXJhMQwwCgYDVQQKDANv
cmcxDDAKBgNVBAsMA29yZzEOMAwGA1UEAwwFa2FmYWxHDAABgkqhkiG9w0BCQEW
DXRlc3RAdGVzdC5jb20wHhcNMTUwNzMwMDQyOTMwWhcNMTYwNzI5MDQyOTMwWjBt
MQswCQYDVQQGEwJVUzELMAkGA1UECBMCQ0ExFDASBgNVBACTC1NhbnRhIENsYXJh
MQwwCgYDVQQKEwNvcmcxDDAKBgNVBAsTA29yZzEOMAwGA1UEAxMwU3JpaGFyc2hh
IENoaW50YWxhcGFuaTCCABCwggEsBgqhkiG9w0BAQsFADQ9f1OBHXUSKVLf
Spwu7OTn9hG3UjzvRADDHj+AtlEmaUVdQCJR+1k9jVj6v8X1ujD2y5tVbNeBO4Ad
NG/yZmC3a5lQpaSfn+gEexAiwk+7qdf+t8Yb+DtX58aophUPBPuD9tPFHsMCNVQT
WhaRMvZ1864rYdcq7/IiAxmd0UgBxwIVAjdGUi8VIwvMspK5gqLrhAvwWBz1AoGB
APfhoIXWmz3ey7yrXDa4V715lK+7+jrqqv1XTAs9B4JnUV1XjrrUWU/mcQcQgYC0
SRZxI+hMKBYTt88JMoZIpue8FngLVHyNKOCjrh4rs6Z1kW6jfwv6ITVi8ftiegEk
O8yk8b6oUZCJqIPf4VrlnwaSi2ZegHtVJWQBTDv+z0kqA4GEAAKBgB+Pdz0306bq
TpUAb2FERMPLFsx06H0x+TULivcp7HbS5yrkV9bXZmv/FD98x76QxXrOq1WpQhY
YDeGDjH+XQkJ6ZxBVBZNJDIPcnfQpfzXAvryQ+cm8oXUsKidtHf4pLMYViXX6BWX
Oc2hX4rG+lC8/NXW+1zVvCr9To9fngzjMA0GCSqGSIb3DQEBCwUAA4IBAQBfyVse
RJ+uginlWg5trZscqH0tlocbnek4UuV/xis2eAu9l4EFOM5kRt5GmkGZRcM/zHF8
BRJwXbf0fytmQKSPFk8R4/NGDolzoK+F7uXeJ0S2u/T29xk0u2i4tjvleq6OCphE
i9vdjM0E0Whf9SHRHOXirOYFX3cL775XwKdzKKRkk+AszFR+mRu90rdoaePQtgGh
9Kfwr4+6AU/dPtdGuomtBQqMxCzlrLd8EYhVVQ97wHIZ3sPv1M5PIhOJ/YHSBJIC
75eo/4acDxZ+j3sR5kcFulzYwFLGDYBaKH/w3mYcGTALeBlzUkX53NVizIvhUd69
XJO4lDSDtG0lfort
-----END CERTIFICATE-----
subject=/C=US/ST=CA/L=Santa Clara/O=org/OU=org/CN=JBrown
issuer=/C=US/ST=CA/L=Santa Clara/O=org/OU=org/CN=kafak/emailAddress=test@test.
com
```

If the certificate does not display, or if there are any other error messages, then your keystore is not set up properly.

4.12.2. Configuring Kafka Producer and Kafka Consumer

SSL is supported for new Kafka Producers and Consumer processes; the older API is not supported. Configuration settings for SSL are the same for producers and consumers.

If client authentication is not needed in the broker, then the following is a minimal configuration example:

```
security.protocol = SSL
ssl.truststore.location = /var/private/ssl/kafka.client.truststore.jks
ssl.truststore.password = test1234
```

If client authentication is required, first create a keystore (described earlier in this chapter). Next, specify the following settings:

```
ssl.keystore.location = /var/private/ssl/kafka.client.keystore.jks
ssl.keystore.password = test1234
ssl.key.password = test1234
```

One or more of the following optional settings might also be needed, depending on your requirements and the broker configuration:

Property	Description	Value(s)
<code>ssl.provider</code>	The name of the security provider used for SSL connections. Default value is the default security provider of the JVM.	
<code>ssl.cipher.suites</code>	Specify one or more cipher suites: named combinations of authentication, encryption, MAC and key exchange algorithms used to negotiate the security settings for a network connection using the TLS or SSL network protocol.	
<code>ssl.enabled.protocols</code>	List at least one of the protocols configured on the broker side. Note: SSL is deprecated; its use in production is not recommended.	TLsv1.2, TLsv1.1, TLsv1
<code>ssl.keystore.type</code>	Specify the SSL keystore type.	JKS
<code>ssl.truststore.type</code>	Specify the SSL truststore type.	JKS

The following two examples launch console-producer and console-consumer processes:

```
kafka-console-producer.sh --broker-list localhost:9093 --topic test --
producer.config client-ssl.properties

kafka-console-consumer.sh --bootstrap-server localhost:9093 --topic test --
new-consumer --consumer.config client-ssl.properties
```

4.13. Enable SSL for Accumulo

One of the major features added in Accumulo 1.6.0 was the ability to configure Accumulo so that the Thrift communications will run over SSL. [Apache Thrift](#) is the remote procedure call library that is leveraged for both intra-server and client communication with Accumulo. Issuing these calls over a secure socket ensures that unwanted actors cannot inspect the traffic sent across the wire. Given the sometimes sensitive nature of data stored in Accumulo and the authentication details for users, secure communications are critical.

Due to the complex and deployment-specific nature of the security model for some systems, Accumulo expects users to provide their own certificates, guaranteeing that they are, in fact, secure. However, for those who require security but do not already operate within the confines of an established security infrastructure, OpenSSL and the Java keytool command can be used to generate the necessary components to enable wire encryption.

To enable SSL with Accumulo, it is necessary to generate a certificate authority and certificates that are signed by that authority. Typically, each client and server has its own certificate, which provides the finest level of control over a secure cluster when the certificates are properly secured.

4.13.1. Generate a Certificate Authority

The certificate authority (CA) controls what certificates can be used to authenticate with each other. To create a secure connection with two certificates, each certificate must be signed by a certificate authority in the "truststore" (A Java KeyStore which contains at least one Certificate Authority's public key). When creating your own certificate authority, a single CA is typically sufficient (and would result in a single public key in the truststore).

Alternatively, a third party can also act as a certificate authority (to add an additional layer of security); however, these are typically not a free service.

The following is an example of creating a certificate authority and adding its public key to a Java KeyStore to provide to Accumulo.

```
# Create a private key
openssl genrsa -des3 -out root.key 4096

# Create a certificate request using the private key
openssl req -x509 -new -key root.key -days 365 -out root.pem

# Generate a Base64-encoded version of the PEM just created
openssl x509 -outform der -in root.pem -out root.der

# Import the key into a Java KeyStore
keytool -import -alias root-key -keystore truststore.jks -file root.der

# Remove the DER formatted key file (as we don't need it anymore)
rm root.der
```

Remember to protect `root.key` and never distribute it, as the private key is the basis for your circle of trust. The `keytool` command will prompt you about whether or not the certificate should be trusted: enter "yes". The `truststore.jks` file, a "truststore", is meant to be shared with all parties communicating with one another. The password provided to the truststore verifies that the contents of the truststore have not been tampered with.

4.13.2. Generate a Certificate/Keystore Per Host

It is desirable to generate a certificate for each host in the system. Additionally, each client connecting to the Accumulo instance running with SSL should be issued its own certificate. Issuing individual certificates to each entity provides proper control to revoke/reissue certificates to clients as necessary, without widespread interruption.

The following commands create a private key for the server, generate a certificate signing request created from that private key, use the certificate authority to generate the certificate using the signing request. and then create a Java KeyStore with the certificate and the private key for our server.

```
# Create the private key for our server
openssl genrsa -out server.key 4096

# Generate a certificate signing request (CSR) with our private key
openssl req -new -key server.key -out server.csr

# Use the CSR and the CA to create a certificate for the server (a reply to
the CSR)
openssl x509 -req -in server.csr -CA root.pem -CAkey root.key -CAcreateserial
-out server.crt -days 365

# Use the certificate and the private key for our server to create PKCS12 file
openssl pkcs12 -export -in server.crt -inkey server.key -certfile server.crt -
name 'server-key' -out server.p12

# Create a Java KeyStore for the server using the PKCS12 file (private key)
keytool -importkeystore -srckeystore server.p12 -srcstoretype pkcs12 -
destkeystore server.jks -deststoretype JKS
```

```
# Remove the PKCS12 file as we don't need it
rm server.p12

# Import the CA-signed certificate to the keystore
keytool -import -trustcacerts -alias server-crt -file server.crt -keystore
server.jks
```

This, combined with the truststore, provides what is needed to configure Accumulo servers to run over SSL. The private key (`server.key`), the certificate signed by the CA (`server.pem`), and the keystore (`server.jks`) should be restricted to only be accessed by the user running Accumulo on the host it was generated for. Use `chown` and `chmod` to protect the files, and do not distribute them over non-secure networks.

4.13.3. Configure Accumulo Servers

Now that the Java KeyStores have been created with the necessary information, the Accumulo configuration must be updated so that Accumulo creates the Thrift server over SSL instead of a normal socket. Configure the following properties in `accumulo-site.xml`:

```
<property>
  <name>rpc.javax.net.ssl.keyStore</name>
  <value>/path/to/server.jks</value>
</property>
<property>
  <name>rpc.javax.net.ssl.keyStorePassword</name>
  <value>server_password</value>
</property>
<property>
  <name>rpc.javax.net.ssl.trustStore</name>
  <value>/path/to/truststore.jks</value>
</property>
<property>
  <name>rpc.javax.net.ssl.trustStorePassword</name>
  <value>truststore_password</value>
</property>
<property>
  <name>instance.rpc.ssl.enabled</name>
  <value>>true</value>
</property>
```

The keystore and truststore paths are both absolute paths on the local file system (not HDFS). Remember that the server keystore should only be readable by the user running Accumulo and, if you place plain-text passwords in `accumulo-site.xml`, make sure that `accumulo-site.xml` is also not globally readable. To keep these passwords out of `accumulo-site.xml`, consider configuring your system with the new Hadoop `CredentialProvider` class. See [ACCUMULO-2464](#) for more information on what will be available in Accumulo-1.6.1.

Also, be aware that if unique passwords are used for each server when generating the certificate, this will result in different `accumulo-site.xml` files for each host. Unique configuration files for each host will add complexity to the configuration management of your instance. The use of a `CredentialProvider` (a feature from Hadoop which allows for acquisitions of passwords from alternate systems) can help alleviate the issues with unique `accumulo-site.xml` files on each host. A Java KeyStore can be created using the `CredentialProvider` tools, which eliminates the need for passwords to be stored in

accumulo-site.xml, and can instead point to the CredentialProvider URI which is consistent across hosts.

4.13.4. Configure Accumulo Clients

To configure Accumulo clients, use `$HOME/.accumulo/config`. This is a simple [Java properties file](#): each line is a configuration, key, and value separated by a space, and lines beginning with a `#` symbol are ignored. For example, if we generated a certificate and placed it in a keystore (as described above), we would generate the following file for the Accumulo client.

```
instance.rpc.ssl.enabled true
rpc.javax.net.ssl.keyStore /path/to/client-keystore.jks
rpc.javax.net.ssl.keyStorePassword client-password
rpc.javax.net.ssl.trustStore /path/to/truststore.jks
rpc.javax.net.ssl.trustStorePassword truststore-password
```

When creating a ZooKeeperInstance, the implementation will automatically look for this configuration file and set up a connection with the methods defined in this file. The ClientConfiguration class also contains methods that can be used instead of a configuration file on the file system. Again, the paths to the keystore and truststore are on the local file system, not HDFS.

4.14. Enable SSL for Apache Atlas

This section describes how to enable SSL for Apache Atlas on an Ambari cluster.

4.14.1. Configuring Apache Atlas SSL

Both one-way (server authentication) and two-way (server and client authentication) SSL are supported. Use the following steps to enable Apache Atlas SSL:

1. Create a keystore file:

```
cd /usr/jdk64/jdk1.8.0_112/bin/
keytool -genkey -alias serverkey -keypass <keypass> -keyalg RSA -sigalg
  SHA1withRSA -keystore atlas.keystore -storepass <keypass> -validity 3650 -
  dnname "CN=Nicola Marangoni, OU=PS, O=Hortonworks, L=Munich, ST=BY, C=DE"
```

2. Create a .jceks file:

```
cd /usr/hdp/current/atlas-server/bin
./cputil.py
Please enter the full path to the credential provider:jceks://file/home/
atlas/test.jceks
0 [main] WARN org.apache.hadoop.util.NativeCodeLoader - Unable to load
  native-hadoop library for your platform... using builtin-java classes where
  applicable
Please enter the password value for keystore.password:<keypass>
Please enter the password value for keystore.password again:<keypass>
Please enter the password value for truststore.password:<keypass>
Please enter the password value for truststore.password again:<keypass>
Please enter the password value for password:<keypass>
Please enter the password value for password again:<keypass>
cp /root/atlas.keystore /home/atlas/
cd /home/atlas
```

- Assign 440 rights to both of these files, and make atlas:hadoop owners for these files (so that Atlas can read these files):

```
chmod 440 atlas.keystore test.jceks
```

- Select **Atlas > Configs > Advanced**, then select **Advanced application-properties** and set the following properties:

Table 4.4. Atlas Advanced application-properties

Property	Value	Description
atlas.enableTLS	true	Enable or disable the SSL listener. Set this value to <code>true</code> to enable SSL (default value is <code>false</code>).

Add the following properties by selecting **Custom application-properties > Add Property**.

Table 4.5. Atlas Custom application-properties

Property	Value	Description
keystore.file	/home/atlas/atlas.keystore	The path to the keystore file leveraged by the server. This file contains the server certificate.
truststore.file	/home/atlas/atlas.keystore	The path to the truststore file. This file contains the certificates of other trusted entities (e.g. the certificates for client processes if two-way SSL is enabled). In most instances this can be set to the same value as the keystore.file property (especially if one-way SSL is enabled).
client.auth.enabled	true	Enable/disable client authentication (disabled by default). If enabled, the client must authenticate to the server during the transport session key creation process (i.e. two-way SSL is in effect).
cert.stores.credential.provider.path	jceks://file//home/atlas/test.jceks	The path to the Credential Provider store file. The passwords for the keystore, truststore, and server certificate are maintained in this secure file. Utilize the <code>cputil</code> script in the 'bin' directory (see below) to populate this file with the passwords required.
atlas.ssl.exclude.cipher.suites	*NULL.*, *RC4.*, *MD5.*, *DES.*, *DSS.*	This excluded Cipher Suites list - NULL, RC4, MD5, DES, DSS are weak and unsafe Cipher Suites that are excluded by default. If additional Ciphers need to be excluded, set this property with the default Cipher Suites such as *NULL.*, *RC4.*, *MD5.*, *DES.*, *DSS.* and add the additional Cipher Suites to the list with a comma separator. They can be added with their full name or a regular expression. The Cipher Suites listed in the atlas.ssl.exclude.cipher.suites property take precedence over the

Property	Value	Description
		default Cipher Suites. You should retain the default Cipher Suites, and add additional ones to increase security.



Important

Enabling or disabling HTTPS will **not** automatically reconfigure the `atlas.rest.address` property. To update this property, select **Atlas > Configs > Advanced**, then select **Advanced application-properties**. Change the URL strings in the `atlas.rest.address` property to "https" if SSL is enabled (if the `atlas.enableTLS` property is set to `true`) "https". If SSL is not enabled, use "http". For example:

- `http:<server_one>:21000,http:<server_two>:21000,http:<server_three>`
- `https:<server_one>:21443,https:<server_two>:21443,https:<server_th`

The default HTTP port is 21000 and the default HTTPS port is 21443. These values can be overridden using the `atlas.server.http.port` and `atlas.server.https.port` properties, respectively.

5.



Important

After manually editing these settings, select **Actions > Stop All** on the Ambari dashboard to stop all services, then select **Actions > Start All** to restart all services.



Note

If you disable Atlas SSL, you must clear your browser cookies in order to log in to the Atlas UI using HTTP request headers.

4.14.2. Credential Provider Utility Script

In order to prevent the use of clear-text passwords, the Atlas platform uses the Credential Provider facility for secure password storage (see the [Hadoop Credential Command Reference](#) for more information about this facility). The `cputil` script can be used to create the required password store.

To create the credential provider for Atlas:

1. Use the following command to switch to the Atlas bin directory:

```
cd /usr/hdp/current/atlas-server/bin
```

2. Run the following command:

```
./cputil.py
```

3. When prompted, enter the path for the generated credential provider. The format for the path is:

```
/local/file/path/file.jceks
```

- Only one absolute path is allowed. The credential provider files generally use the .jceks extension.
4. When prompted, enter the passwords for the keystore, truststore, and server key (these passwords must match the passwords used when actually creating the associated certificate store files).
 5. The credential provider is generated and saved to the specified path.

4.15. SPNEGO setup for WebHCat

To set up secure WebHCat, set the following properties in the `/etc/hcatalog/conf/webhcat-site.xml` file:

```
</property>
  <name>templeton.kerberos.principal</name>
  <value>HTTP/host1234.example.com@EXAMPLE.COM</value>
  <description/>
</property>
```

The `templeton.kerberos.principal` property must use the host name of the WebHCat Server.

```
<property>
  <name>templeton.kerberos.keytab</name>
  <value>/etc/security/keytabs/spnego.service.keytab</value>
  <description/>
</property>
```

```
<property>
  <name>templeton.kerberos.secret</name>
  <value>secret</value>
  <description/>
</property>
```

```
<property>
  <name>templeton.hive.properties</name>
  <value>hive.metastore.local=false,hive.metastore.uris=thrift://host1234.
example.com:9083,
      hive.metastore.sasl.enabled=true,hive.metastore.execute.
setugi=true,
      hive.exec.mode.local.auto=false,
      hive.metastore.kerberos.principal=hive/_HOST@EXAMPLE.COM</
value>
  <description>Properties to set when running hive.</description>
</property>
```

Be sure to set the `templeton.hive.properties` property with the host name for your Thrift server.

4.16. Configure SSL for Hue

HTTPS is a simple HTTP in conjunction with SSL (Secure Sockets Layer) and used for establishing an encrypted link between the web browser and the web server. Using HTTPS enables you to prevent collection of sensitive information between your web browser and a web server.

4.16.1. Enabling SSL on Hue by Using a Private Key

If you have a private key, follow these steps to enable SSL on Hue:

1. Configure Hue to use your private key by adding the following syntax to the `/etc/hue/conf/hue.ini` file:

```
ssl_certificate=$PATH_TO_CERTIFICATE
ssl_private_key=$PATH_TO_KEY
ssl_cipher_list="DEFAULT:!aNULL:!eNULL:!LOW:!EXPORT:!SSLv2" (default)
```

2. Restart Hue:

```
/etc/init.d/hue restart
```

4.16.2. Enabling SSL on Hue Without Using a Private Key

If you do not have a private key and want to run tests, you can enable SSL on Hue by creating a self-signed certificate:

1. Create a key:

```
openssl genrsa 1024 > host.key
```

2. Create a self-signed certificate:

```
openssl req -new -x509 -nodes -sha1 -key host.key > host.cert
```

3. Move the `host.key` and `host.cert` files to the `ssl` directory:

```
mv host.key /etc/ssl
mv host.cert /etc/ssl
```

4. Configure Hue to use your private key by adding the following syntax to the `/etc/hue/conf/hue.ini` file:

```
ssl_certificate=$PATH_TO_CERTIFICATE
ssl_private_key=$PATH_TO_KEY
ssl_cipher_list="DEFAULT:!aNULL:!eNULL:!LOW:!EXPORT:!SSLv2" (default)
```

5. Restart Hue:

```
/etc/init.d/hue restart
```

4.17. Configure SSL for Knox

For the simplest of evaluation deployments, the initial startup of the Knox Gateway will generate a self-signed cert for use on the same machine as the gateway instance. These certificates are issued for "localhost" and will require specifically disabling hostname verification on client machines other than where the gateway is running.

4.17.1. Self-Signed Certificate with Specific Hostname for Evaluations

In order to continue to use self-signed certificates for larger evaluation deployments, a certificate can be generated for a specific hostname. This will allow clients to properly verify

the hostname presented in the certificate as the host that they requested in the request URL.

To create a self-signed certificate:

1. Create a certificate: where `$gateway-hostname` is the FQDN of the Knox Gateway.

```
cd $gateway bin/knoxcli.cmd create-cert --hostname $gateway-hostname
```

2. Export the certificate in PEM format:

```
keytool -export -alias gateway-identity -rfc -file $certificate_path -
keystore $gateway /data/security/keystores/gateway.jks
```



Note

cURL option accepts certificates in PEM format only.

3. Restart the gateway:

```
cd $gateway bin/gateway.sh stop bin/gateway.sh start
```

4. After copying the certificate to a client, use the following command to verify:

```
curl --cacert $certificate_path -u $username : $password https://
$gateway-hostname : $gateway_port /gateway/ $cluster_name /webhdfs/v1?op=
GETHOMEDIRECTORY
```

4.17.2. CA-Signed Certificates for Production

For production deployments or any deployment in which a certificate authority issued certificate is needed, the following steps are required.

1. Import the desired certificate/key pair into a java keystore using keytool and ensure the following:

- The certificate alias is `gateway-identity`.
- The store password matches the master secret created earlier.
- Note the key password used - as we need to create an alias for this password.

2. Add a password alias for the key password:

```
cd $gateway bin/knoxcli.cmd create-cert create-alias gateway-identity-
passphrase --value $actualpassphrase
```



Note

The password alias must be `gateway-identity-passphrase`.

4.17.3. Setting Up Trust for the Knox Gateway Clients

In order for clients to trust the certificates presented to them by the gateway, they will need to be present in the client's truststore as follows:

1. Export the gateway-identity cert from the `$gateway/data/security/keystores/gateway.jks` using java keytool or another key management tool.
2. Add the exported certificate to the cacerts or other client specific truststore or the `gateway.jks` file can be copied to the clients to be used as the truststore.



Note

If taking this approach be sure to change the password of the copy so that it no longer matches the master secret used to protect server side artifacts.

4.18. Securing Phoenix

To configure Phoenix to run in a secure Hadoop cluster, use the instructions on [this page](#).

4.19. Set Up SSL for Ambari

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

Ambari Server should not be running when you do this. Either make these changes before you start Ambari the first time, or bring the server down before running the setup command.

1. Log into the Ambari Server host.
2. Locate your certificate. If you want to create a temporary self-signed certificate, use this as an example:

```
openssl genrsa -out $wserver.key 2048

openssl req -new -key $wserver.key -out $wserver.csr

openssl x509 -req -days 365 -in $wserver.csr -signkey
$wserver.key -out $wserver.crt
```

Where `$wserver` is the Ambari Server host name.

The certificate you use must be PEM-encoded, not DER-encoded. If you attempt to use a DER-encoded certificate, you see the following error:

```
unable to load certificate 140109766494024:error:0906D06C:PEM
routines:PEM_read_bio:no start line:pem_lib.c :698:Expecting:
TRUSTED CERTIFICATE
```

You can convert a DER-encoded certificate to a PEM-encoded certificate using the following command:

```
openssl x509 -in cert.crt -inform der -outform pem -out cert.pem
```

where `cert.crt` is the DER-encoded certificate and `cert.pem` is the resulting PEM-encoded certificate.

3. Run the special setup command and answer the prompts.

```
ambari-server setup-security
```

- Select `1` for Enable HTTPS for Ambari server.
- Respond `y` to Do you want to configure HTTPS ?
- Select the port you want to use for SSL. The default port number is `8443`.
- Provide the complete path to your certificate file (`$wserver.crt` from above) and private key file (`$wserver.key` from above).
- Provide the password for the private key.
- Start or restart the Server

```
ambari-server restart
```

4. Trust Store Setup - If you plan to use Ambari Views with your Ambari Server, after enabling SSL for Ambari using the instructions below, you must also [set up a truststore for the Ambari server](#).

4.19.1. Set Up Truststore for Ambari Server

If you plan to [Set Up SSL for Ambari \[457\]](#) or to enable wire encryption for HDP, you must configure the Truststore for Ambari and add certificates.

Ambari Server should not be running when you do this. Either make these changes before you start Ambari the first time, or bring the server down before running the setup command.

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

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

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

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

```
ambari-server setup-security
Using python /usr/bin/python2.6
Security setup options...
=====
Choose one of the following options:
 [1] Enable HTTPS for Ambari server.
 [2] Encrypt passwords stored in ambari.properties file.
 [3] Setup Ambari kerberos JAAS configuration.
 [4] Setup truststore.
```

```
[5] Import certificate to truststore.
=====
Enter choice, (1-5): *4*
Do you want to configure a truststore [y/n] (y)? *y*
TrustStore type [jks/jceks/pkcs12] (jks): *jks*
Path to TrustStore file : *<path to the ambari-server-truststore keystore>*
Password for TrustStore:
Re-enter password:
Ambari Server 'setup-security' completed successfully.
```

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

```
ambari-server restart
```

4.20. Configure Ambari Ranger SSL

4.20.1. Configuring Ambari Ranger SSL Using Public CA Certificates

If you have access to Public CA issued certificates, use the following steps to configure Ambari Ranger SSL.

4.20.1.1. Prerequisites

- Copy the keystore/truststore files into a different location (e.g. `/etc/security/serverKeys`) than the `/etc/<component>/conf` folder.
- Make sure that the JKS file names are unique.
- Make sure that the correct permissions are applied.
- Make sure that passwords are secured.

4.20.1.2. Configuring Ranger Admin

1. Stop Ranger by selecting **Service Actions > Stop**.

The screenshot shows the Ambari interface for the 'test_cluster'. The top navigation bar includes 'Dashboard', 'Services', 'Hosts', 'Alerts', and 'Admin'. The 'Services' tab is selected, and the 'Ranger' service is highlighted in the left sidebar. The main content area shows the 'Summary' tab for Ranger, with a table of service components and their statuses:

Component	Status
Ranger Admin	Started
Ranger Usersync	Started
Ranger HDFS plugin	Disabled
Ranger YARN plugin	Disabled
Ranger Hive plugin	Disabled
Ranger HBase plugin	Disabled
Ranger Storm plugin	Disabled
Ranger Knox plugin	Disabled
Ranger Kafka plugin	Disabled

The 'Service Actions' dropdown menu is open, showing options: Start, Stop (highlighted in red), Restart All, Enable Ranger Admin HA, Run Service Check, and Turn On Maintenance Mode.

2. Use the following steps to disable the HTTP port and enable the HTTPS port with the required keystore information.
 - a. Select **Configs > Advanced**. Under Ranger Settings, clear the **HTTP enabled** check box (this blocks all agent calls to the HTTP port even if the port is up and working).

The screenshot shows the Ambari Ranger configuration page. The left sidebar lists services: HDFS, MapReduce2, YARN, Tez, Hive, HBase, Pig, Sqoop, Oozie, ZooKeeper, Falcon, Storm, Flume, Accumulo, Ambari Metrics, Atlas, Kafka, Knox, Mahout, Ranger, Slider, SmartSense, and Spark. The Ranger service is highlighted. The main content area shows the Ranger configuration page with tabs for Ranger Admin, Ranger User Info, Ranger Plugin, Ranger Audit, and Advanced. The Advanced tab is selected, showing Admin Settings and Ranger Settings. The Ranger Settings section includes fields for External URL, Authentication method, and HTTP enabled. The External URL field is highlighted with a red box, containing the value 'https://c6403.ambari.apache.org:6182'. The HTTP enabled checkbox is also highlighted with a red box.

- b. Under Ranger Settings, provide the value in the **External URL** box in the format `https://<hostname of policy manager>:<https port>`.

The screenshot shows a close-up of the Ranger Settings section. The External URL field is highlighted with a red box, containing the value 'https://c6403.ambari.apache.org:6182'. The Authentication method is set to LDAP. The HTTP enabled checkbox is also visible.

- c. Under Advanced ranger-admin-site, set the following properties:

- `ranger.https.attrib.keystore.file` – Provide the location of the Public CA issued keystore file.
- `ranger.service.https.attrib.keystore.pass` – Enter the password for the keystore.
- `ranger.service.https.attrib.keystore.keyalias` – Enter the alias name for the keystore private key.
- `ranger.service.https.attrib.clientAuth` – Enter `want` as the value. This validates the client cert from all agents, but not the requests from web applications. Setting this value to `want` requires the client to have a certificate to use to sign traffic. If you do not want to put certificates on the client machines to do two-way SSL, this parameter can be set to `false` to enable one-way SSL.
- `ranger.service.https.attrib.ssl.enabled` – set this property to `true`.
- `ranger.service.https.port` – Make sure that this port is available, or change the value to an available port number.

Advanced ranger-admin-site

ranger.audit.source.type	<input type="text" value="solr"/>	🔒	🟢	ⓘ
ranger.credential.provider.path	<input type="text" value="/etc/ranger/admin/rangeradmin.jceks"/>	🔒	🟢	ⓘ
ranger.https.attrib.keystore.file	<input type="text" value="/etc/ranger/admin/conf/ranger-admin-keystore.jks"/>	🔒	🟢	ⓘ
ranger.jpa.audit.jdbc.credential.alias	<input type="text" value="rangeraudit"/>	🔒	🟢	ⓘ
ranger.jpa.audit.jdbc.dialect	<input type="text" value="{{jdbc_dialect}}"/>	🔒	🟢	ⓘ
ranger.jpa.audit.jdbc.driver	<input type="text" value="{{ranger_jdbc_driver}}"/>	🔒	🟢	ⓘ
ranger.jpa.audit.jdbc.password	<input type="password" value="....."/>	🔒		
ranger.jpa.audit.jdbc.url	<input type="text" value="{{audit_jdbc_url}}"/>	🔒	🟢	ⓘ
ranger.jpa.audit.jdbc.user	<input type="text" value="{{ranger_audit_db_user}}"/>	🔒	🟢	ⓘ
ranger.jpa.jdbc.credential.alias	<input type="text" value="rangeradmin"/>	🔒	🟢	ⓘ
ranger.jpa.jdbc.dialect	<input type="text" value="{{jdbc_dialect}}"/>	🔒	🟢	ⓘ
ranger.jpa.jdbc.password	<input type="password" value="....."/>	🔒		
ranger.jpa.jdbc.user	<input type="text" value="{{ranger_db_user}}"/>	🔒	🟢	ⓘ
Group Search Base	<input type="text" value="{{ranger_ug_ldap_group_searchbase}}"/>	🔒	🟢	ⓘ
Group Search Filter	<input type="text" value="{{ranger_ug_ldap_group_searchfilter}}"/>	🔒	🟢	ⓘ
ranger.service.host	<input type="text" value="{{ranger_host}}"/>	🔒	🟢	ⓘ
ranger.service.http.port	<input type="text" value="6080"/>	🔒	🟢	ⓘ
ranger.service.https.attrib.clientAuth	<input type="text" value="want"/>	🔒	🟢	ⓘ
ranger.service.https.attrib.keystore.keyalias	<input type="text" value="rangeradmin"/>	🔒	🟢	ⓘ
ranger.service.https.attrib.keystore.pass	<input type="password" value="....."/>	🔒		
ranger.service.https.attrib.ssl.enabled	<input type="text" value="true"/>	🔒	🟢	ⓘ
ranger.service.https.port	<input type="text" value="6182"/>	🔒	🟢	ⓘ

3. Under Custom ranger-admin-site, add the following properties:

- `ranger.service.https.attrib.keystore.file` – Specify the same value provided for the `ranger.https.attrib.keystore.file` property.

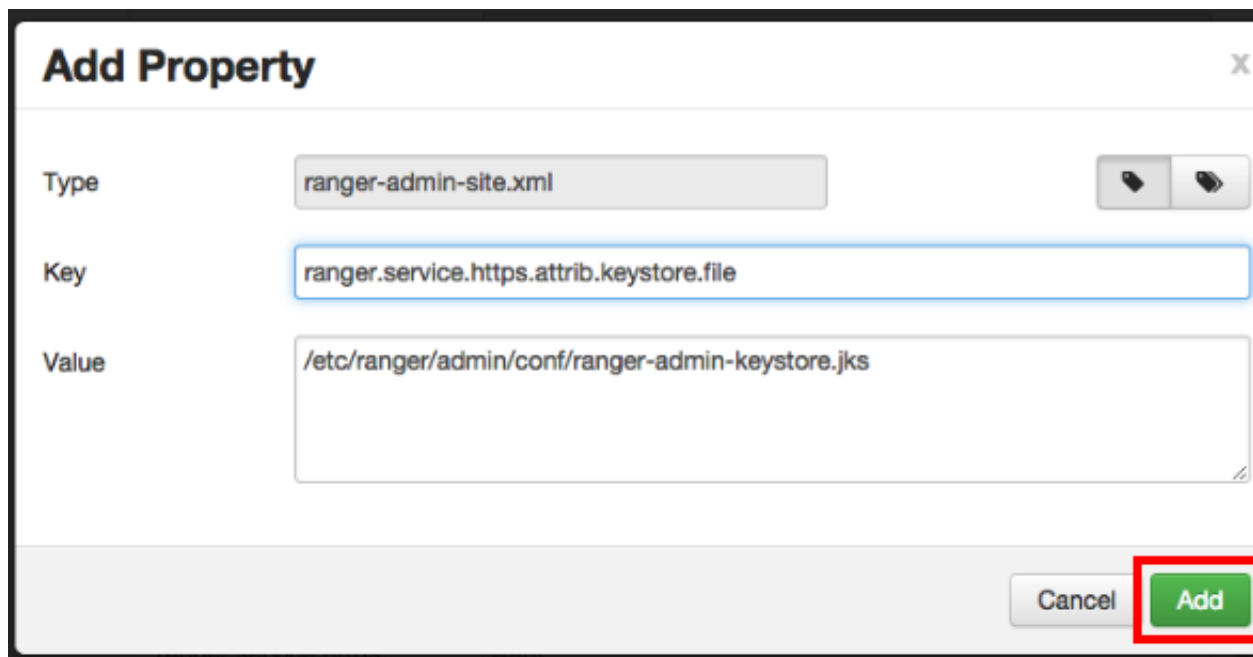
- `ranger.service.https.attrib.client.auth` – Specify the same value provided for the `ranger.service.https.attrib.clientAuth` property.

To add a Custom `ranger-admin-site` property:

- a. Select **Custom `ranger-admin-site`**, then click **Add Property**.

The screenshot displays the Ranger configuration interface. At the top, the 'AD Settings' section is expanded, showing two properties: 'ranger.ldap.ad.domain' with the value 'localhost' and 'ranger.ldap.ad.url' with the value 'ldap://ad.xasecure.net:389'. Below this, several other configuration sections are listed, including 'LDAP Settings', 'Advanced ranger-admin-site', 'Advanced ranger-env', 'Advanced ranger-ugsync-site', 'Custom admin-properties', 'Custom ranger-admin-site', 'Custom ranger-site', 'Custom ranger-ugsync-site', and 'Custom usersync-properties'. The 'Custom ranger-admin-site' section is expanded, and the 'Add Property ...' button is highlighted with a red rectangle.

- b. On the Add Property pop-up, type the property name in the **Key** box, type the property value in the **Value** box, then click **Add**.



Add Property

Type: ranger-admin-site.xml

Key: ranger.service.https.attrib.keystore.file

Value: /etc/ranger/admin/conf/ranger-admin-keystore.jks

Buttons: Cancel, Add

4. Save your changes and start Ranger Admin.

When you attempt to access the Ranger Admin UI with the HTTPS protocol on the port specified by the `ranger.service.https.port` property, the browser should report that it does not trust the site. Click Proceed anyway and you should be able to access the Ranger Admin UI over HTTPS.

4.20.1.3. Configuring Ranger Usersync

1. Stop Ranger Usersync by selecting the **Ranger Usersync** link, then select **Started > Stop** next to Ranger Usersync.

The screenshot shows the Ambari web interface for a cluster named 'test_cluster'. The top navigation bar includes 'Dashboard', 'Services', 'Hosts 1', 'Alerts 16', and 'Admin'. The main header displays the host ID 'c6403.ambari.apache.org' and a 'Back' link. Below the header, there are tabs for 'Summary', 'Configs', 'Alerts 16', and 'Versions'. The 'Summary' tab is active, showing a list of components and their status. A dropdown menu is open for the 'ResourceManager / YARN' component, with the 'Stop' option highlighted in red. To the right, the 'Host Metrics' section shows various metrics like CPU Usage, Disk Usage, Load, Memory Usage, Network Usage, and Processes, all with 'No Data Available' placeholders. The 'Host Actions' dropdown is also visible in the top right.

Component	Status
Accumulo GC / Accumulo	Stopped
Accumulo Master / Accumulo	Stopped
Accumulo Monitor / Accumulo	Stopped
Accumulo Tracer / Accumulo	Stopped
App Timeline Server / YARN	Started
Atlas Metadata Server / Atlas	Started
DRPC Server / Storm	Started
Falcon Server / Falcon	Stopped
HBase Master / HBase	Stopped
History Server / MapReduce2	Started
Hive Metastore / Hive	Started
HiveServer2 / Hive	Started
SmartSense HST Server / SmartSense	Stopped
Kafka Broker / Kafka	Started
Knox Gateway / Knox	Started
Metrics Collector / Ambari Metrics	Stopped
MySQL Server / Hive	Started
NameNode / HDFS	Started
Nimbus / Storm	Started
Oozie Server / Oozie	Started
Ranger Admin / Ranger	Started
Ranger Usersync / Ranger	Started
ResourceManager / YARN	Started
SNameNode / HDFS	Started
Spark History Server / Spark	Started
Storm UI Server / Storm	Started

2. Navigate back to Ranger and select **Configs > Advanced**, then click **Advanced ranger-ugsync-site**. Set the following properties:

- `ranger.usersync.truststore.file` – Enter the path to the truststore file.
- `ranger.usersync.truststore.password` – Enter the truststore password.



The screenshot shows a configuration interface for Ranger. It has two rows of configuration properties. The first row is for 'ranger.usersync.truststore.file' and its value is '/usr/hdp/current/ranger-usersync/conf/mytruststore.jks'. To the right of the input field are three icons: a lock, a green plus sign, and a blue 'C' icon. The second row is for 'ranger.usersync.truststore.password' and its value is masked with dots. To the right of the input field is a lock icon.

3. Start Ranger Usersync by selecting the **Ranger Usersync** link on the Summary tab, then select **Stopped > Start** next to Ranger Usersync.

4.20.1.4. Configuring Ranger Plugins for SSL

The following section shows how to configure the Ranger HDFS plugin for SSL. You can use the same procedure for other Ranger components. Additional steps required to configure the Ranger KMS plugin and server are provided in subsequent sections.

4.20.1.4.1. Configuring the Ranger HDFS Plugin for SSL

The following steps show how to configure the Ranger HDFS plugin for SSL. You can use the same procedure for other Ranger components.

1. Stop HDFS by selecting **Service Actions > Stop**.

The screenshot shows the Ambari interface for a cluster named 'test_cluster'. The 'Services' tab is active, displaying the HDFS service. The 'Service Actions' dropdown menu is open, with the 'Stop' option highlighted in a red box. The main content area shows the HDFS Summary, including NameNode and SNameNode status (both Started), DataNodes (1/1 Started), and various metrics like NameNode Uptime (21.82 days) and Disk Usage (0.17% DFS used, 7.84% Non DFS used). The Metrics section shows several 'No Data Available' warnings and one 'Under Replicated Blocks' warning with a value of 481.

2. Under Ranger Settings, provide the value in the **External URL** box in the format `https://<hostname of policy manager>:<https port>`.

3. Select **Advanced ranger-hdfs-policymgr-ssl** and set the following properties:

- `xasecure.policymgr.clientssl.keystore` – Enter the public CA signed keystore for the machine that is running the HDFS agent.
- `xasecure.policymgr.clientssl.keystore.password` – Enter the keystore password.

▼ **Advanced ranger-hdfs-policymgr-ssl**

xasecure.policymgr. clientsssl.keystore	<input type="text" value="/usr/hdp/current/hadoop-client/conf/ranger-plugin-keystore.jks"/>	+	C
xasecure.policymgr. clientsssl.keystore. credential.file	<input type="text" value="jceks://file{{credential_file}}"/>	+	C
xasecure.policymgr. clientsssl.keystore. password	<input type="password" value="....."/>		
xasecure.policymgr. clientsssl.truststore	<input type="text" value="/usr/hdp/current/hadoop-client/conf/ranger-plugin-truststore.jk"/>	+	C
xasecure.policymgr. clientsssl.truststore. credential.file	<input type="text" value="jceks://file{{credential_file}}"/>	+	C
xasecure.policymgr. clientsssl.truststore. password	<input type="password" value="....."/>		

- 4. Select **Advanced ranger-hdfs-plugin-properties**, then select the **Enable Ranger for HDFS** check box.

▼ **Advanced ranger-hdfs-plugin-properties**

Enable Ranger for HDFS	<input checked="" type="checkbox"/>	🔒	↻	C
Ranger repository config password	<input type="password" value="....."/>	<input type="password" value="....."/>	🔒	
Ranger repository config user	<input type="text" value="hadoop"/>	🔒	+	C
common.name.for. certificate	<input type="text"/>	🔒	+	
hadoop.rpc.protection	<input type="text"/>	🔒	+	
Policy user for HDFS	<input type="text" value="ambari-qa"/>	🔒	+	C

5. Click **Save** to save your changes.
6. Start HDFS by selecting **Service Actions > Start**.
7. Restart Ranger Admin.
8. Log into the Ranger Policy Manager UI as the admin user. Click the **Edit** button of the HDFS repository and provide the CN name of the keystore as the value for **Common Name For Certificate**, then save your changes.
9. Start the HDFS service.
10. Select **Audit > Agents**. You should see an entry for your repo name with HTTP Response Code 200.



Note

This procedure assumes that the keystores provided for the Admin and agent are signed by a public CA.

Provide an identifiable string as the value for Common Name when generating certificates. Important note: In the case where multiple servers talking to ranger admin for downloading policies for the same service/repository (e.g. HBase Master and Region servers, multiple NameNodes or Hive servers in an HA environment, be sure to use the repo name or a common string across all of the nodes (such as HbasePlugin, HdfsPlugin etc). (Note this and enter the same value in Common Name For Certificate field in the edit repository page in the Policy Manager UI).



Note

Ranger Admin will use the JAVA truststore, so you need to add your plugin certificate inside the Java truststore. Alternatively, you can specify a custom truststore by editing `/usr/hdp/2.3.2.0-2950/ranger-admin/ews/ranger-admin-services.sh`. You will need to add the following after the `JAVA_OPTS` line:

```
-Djavax.net.ssl.trustStore=/etc/security/ssl/truststore.jks  
-Djavax.net.ssl.trustStorePassword=hadoop
```

For example:

```
JAVA_OPTS=" ${JAVA_OPTS} -XX:MaxPermSize=256m -Xmx1024m -Xms1024m  
-Djavax.net.ssl.trustStore=/etc/security/ssl/truststore.jks  
-Djavax.net.ssl.trustStorePassword=hadoop"
```

4.20.1.4.2. Configuring the Ranger KMS Plugin for SSL

To configure the Ranger KMS (Key Management Service) plugin for SSL, use the procedure described in the previous section for HDFS, then perform the following additional step.

Log into the Policy Manager UI (as the keyadmin user) and click the **Edit** button of your KMS repository. Provide the CN name of the keystore as the value for Common Name For Certificate and save your changes. This property is not provided by default, so it must be added.

The screenshot shows the Ranger web interface for creating a service. The page has a green header with 'Ranger', 'Access Manager', and 'Encryption' tabs. Below the header, there are navigation buttons for 'Service Manager' and 'Edit Service'. The main content area is titled 'Create Service' and is divided into two sections: 'Service Details' and 'Config Properties'.

Service Details:

- Service Name *:
- Description:
- Active Status: Enabled Disabled

Config Properties:

- KMS URL *:
- Username *:
- Password *:

Below the configuration fields, there is a section for 'Add New Configurations' with a table:

Name	Value
<input type="text" value="commonNameForCertificate"/>	<input type="text" value="ip-172-31-26-219.ec2.internal"/> <input type="button" value="x"/>

There is a '+' button below the table to add more configurations. At the bottom of the form, there is a 'Test Connection' button and three action buttons: 'Save' (green), 'Cancel' (black), and 'Delete' (red).

4.20.1.4.3. Configuring the Ranger KMS Server for SSL

Use the following steps to configure Ranger KMS (Key Management Service) Server for SSL.

1. Stop Ranger KMS by selecting **Service Actions > Stop**.
2. Select **Custom ranger-kms-site**, then add the following properties as shown below:
 - ranger.https.attrib.keystore.file
 - ranger.service.https.attrib.keystore.file (duplicate of above – workaround for now)
 - ranger.service.https.attrib.clientAuth
 - ranger.service.https.attrib.client.auth (duplicate of above – workaround for now)
 - ranger.service.https.attrib.keystore.keyalias
 - ranger.service.https.attrib.keystore.pass

ranger.service.https.attrib.ssl.enabled

ranger.service.https.port

▼ Custom ranger-kms-site

ranger.https.attrib.keystore.file	<input type="text" value="/etc/ranger/kms/conf/ranger-kms-keystore.jks"/>	● ●
ranger.service.https.attrib.client.auth	<input type="text" value="want"/>	● ●
ranger.service.https.attrib.clientAuth	<input type="text" value="false"/>	● ●
ranger.service.https.attrib.keystore.file	<input type="text" value="/etc/ranger/kms/conf/ranger-kms-keystore.jks"/>	● ●
ranger.service.https.attrib.keystore.keyalias	<input type="text" value="rangerkms"/>	● ●
ranger.service.https.attrib.keystore.pass	<input type="text" value="rangerkms"/>	● ●
ranger.service.https.attrib.ssl.enabled	<input type="text" value="true"/>	● ●
ranger.service.https.port	<input type="text" value="9393"/>	● ●

[Add Property ...](#)

3. Under Advanced kms_env, update the value of **kms_port** to match the value of **ranger.service.https.port**.
4. Save your changes and restart Ranger KMS.

When you attempt to access the Ranger KMS UI with the HTTPS protocol on the port specified by the `ranger.service.https.port` property, the browser should report that it does not trust the site. Click Proceed anyway and you should be able to access the Ranger Admin UI over HTTPS.

4.20.1.4.4. Configure Ranger KMS Database for SSL-enabled MySQL

When an SSL-enabled database is configured for use with Ranger KMS, you must add certain configurations to Ranger:

1. In Ambari>Ranger KMS>Configs>Advanced>Custom kms-properties, add the following parameters:
 - `db_ssl_enabled=True`
 - `db_ssl_required=True`
 - `db_ssl_verifyServerCertificate=True`

- `javax_net_ssl_keyStore=/etc/ranger/admin/keystore`
- `javax_net_ssl_keyStorePassword=ranger`
- `javax_net_ssl_trustStore=/etc/ranger/admin/truststore`
- `javax_net_ssl_trustStorePassword=ranger`

Change keystore and truststore file paths according to your environment.

If certificate verification is not required, you can set value `false` in property `db_ssl_verifyServerCertificate`. In this case, keystore and truststore file location need not to be valid and/or mandatory.

2. In Ambari>Ranger KMS>Configs>Advanced>Custom dbks-site, add the following parameters:

- `ranger.ks.db.ssl.enabled=true`
- `ranger.ks.db.ssl.required=true`
- `ranger.ks.db.ssl.verifyServerCertificate=true`
- `ranger.ks.keystore.file=/etc/ranger/admin/keystore`
- `ranger.ks.keystore.password=ranger`
- `ranger.ks.truststore.file=/etc/ranger/admin/truststore`
- `ranger.ks.truststore.password=password`

Change keystore file path according to your environment.

If certificate verification is not required, then you can set value `false` in property `ranger.db.ssl.verifyServerCertificate`. In this case, keystore and truststore file location need not to be valid and/or mandatory.

3. Install/restart Ranger KMS.

4.20.1.5. Configure Ranger HBase Plugin for SSL

About this task

Use the following steps to configure the Ranger HBase plugin for SSL.

Before you begin

Follow the steps listed for [Configuring the Ranger HDFS Plugin for SSL \[467\]](#), modified for HBase, and perform the following additional step.

Steps

Copy the truststore and keystore from the HBase master to all worker nodes running the RegionServers:

1. From Ambari>HDFS>Configs>Advanced>Advanced ranger-hdfs-policymgr-ssl, copy the /path/keystore.file.name from xasecure.policymgr.clientssl.keystore and distribute it to all nodes.
2. From Ambari>HDFS>Configs>Advanced>Advanced ranger-hdfs-policymgr-ssl, copy the /path/truststore.file.name from xasecure.policymgr.clientssl.truststore and distribute it to all nodes.
3. Restart HBase.

4.20.2. Configuring Ambari Ranger SSL Using a Self-Signed Certificate

If you do not have access to Public CA issued certificates, use the following steps to create a self-signed certificate and configure Ambari Ranger SSL.

4.20.2.1. Prerequisites

- Copy the keystore/truststore files into a different location (e.g. /etc/security/serverKeys) than the /etc/<component>/conf folder.
- Make sure that the JKS file names are unique.
- Make sure that the correct permissions are applied.
- Make sure that passwords are secured.

4.20.2.2. Configuring Ranger Admin

1. Stop Ranger by selecting **Service Actions > Stop**.

The screenshot shows the Ambari interface for the 'test_cluster'. The 'Ranger' service is selected in the left sidebar. The main area displays the 'Summary' tab for the Ranger service, showing the following components and their status:

Component	Status
Ranger Admin	Started
Ranger Usersync	Started
Ranger HDFS plugin	Disabled
Ranger YARN plugin	Disabled
Ranger Hive plugin	Disabled
Ranger HBase plugin	Disabled
Ranger Storm plugin	Disabled
Ranger Knox plugin	Disabled
Ranger Kafka plugin	Disabled

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

- Start
- Stop** (highlighted with a red box)
- Restart All
- Enable Ranger Admin HA
- Run Service Check
- Turn On Maintenance Mode

- Use the following CLI commands to change to the Ranger Admin directory and create a self-signed certificate.

```
cd /etc/ranger/admin/conf
keytool -genkey -keyalg RSA -alias rangeradmin -keystore ranger-admin-keystore.jks -storepass xasecure -validity 360 -keysize 2048
chown ranger:ranger ranger-admin-keystore.jks
chmod 400 ranger-admin-keystore.jks
```

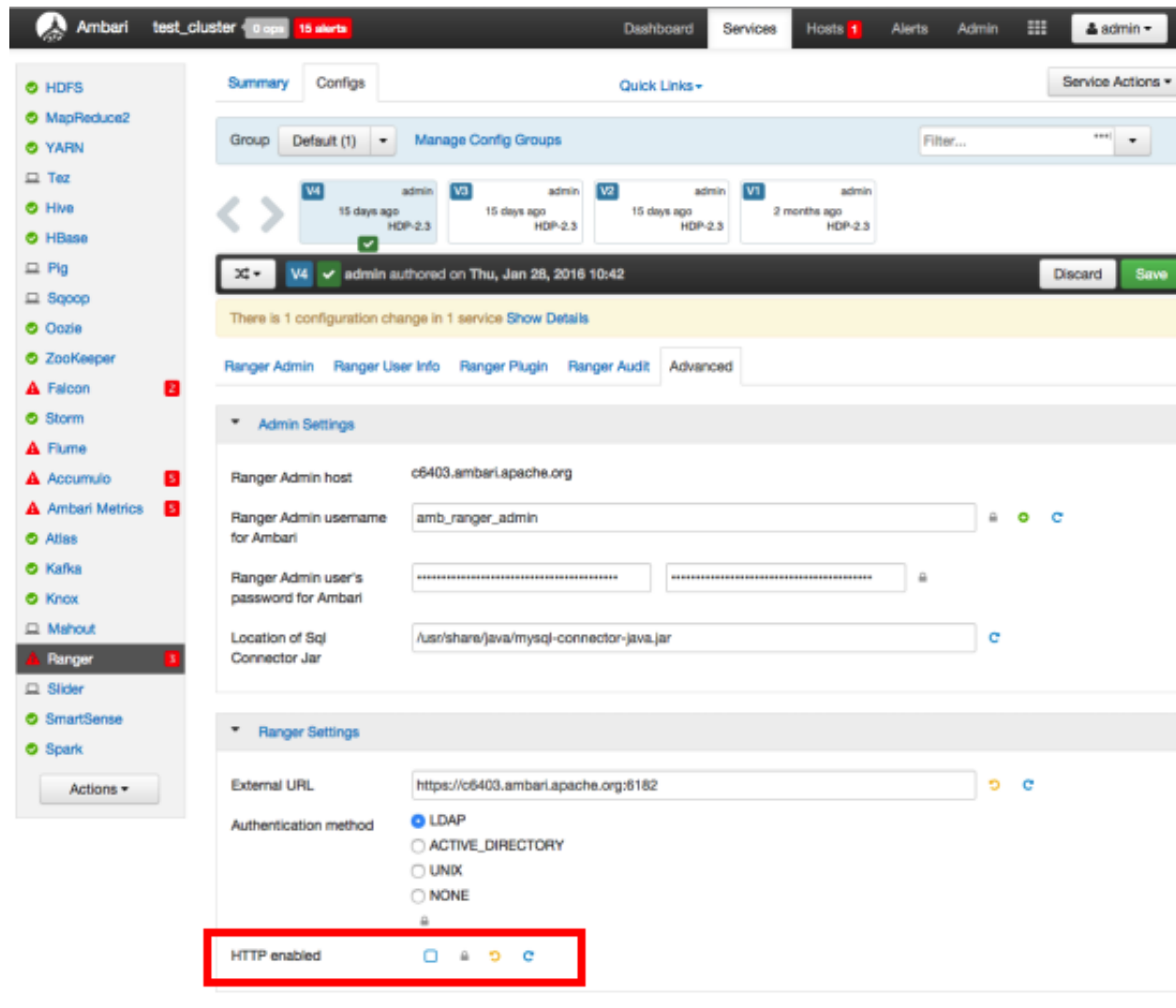
When prompted, provide the host name as the value for the "What is your first and last name?" question. then provide answers to the subsequent questions to create the keystore.



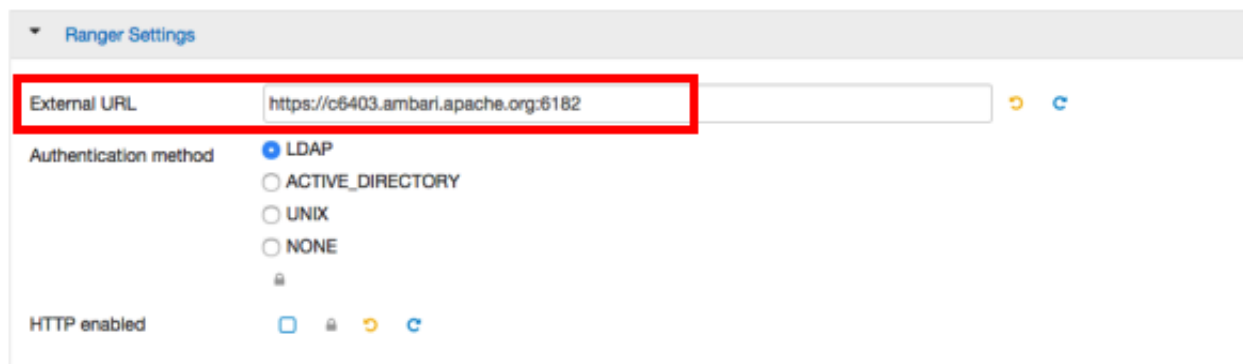
Note

When prompted for your password, press the Enter key. This will not work for Java keytool version 1.5.

- Use the following steps to disable the HTTP port and enable the HTTPS port with the required keystore information.
 - Select **Configs > Advanced**. Under Ranger Settings, clear the **HTTP enabled** check box (this blocks all agent calls to the HTTP port even if the port is up and working).



b. Under Ranger Settings, provide the value in the **External URL** box in the format `https://<hostname of policy manager>:<https port>`.



c. Under Advanced ranger-admin-site, set the following properties:

- `ranger.https.attrib.keystore.file` – Provide the location of the keystore file created previously: `/etc/ranger/admin/conf/ranger-admin-keystore.jks`.
- `ranger.service.https.attrib.keystore.pass` – Enter the password for the keystore (in this case, `xasecure`).
- `ranger.service.https.attrib.keystore.keyalias` – Enter the alias name for the keystore private key (in this case, `rangeradmin`).
- `ranger.service.https.attrib.clientAuth` – Enter `want` as the value. This validates the client cert from all agents, but not the requests from web applications. Setting this value to `want` requires the client to have a certificate to use to sign traffic. If you do not want to put certificates on the client machines to do two-way SSL, this parameter can be set to `false` to enable one-way SSL.
- `ranger.service.https.attrib.ssl.enabled` – set this property to `true`.
- `ranger.service.https.port` – Make sure that this port is available, or change the value to an available port number.

Advanced ranger-admin-site

ranger.audit.source.type	<input type="text" value="solr"/>	🔒	🟢	ⓘ
ranger.credential.provider.path	<input type="text" value="/etc/ranger/admin/rangeradmin.jceks"/>	🔒	🟢	ⓘ
ranger.https.attrib.keystore.file	<input type="text" value="/etc/ranger/admin/conf/ranger-admin-keystore.jks"/>	🔒	🟢	ⓘ
ranger.jpa.audit.jdbc.credential.alias	<input type="text" value="rangeraudit"/>	🔒	🟢	ⓘ
ranger.jpa.audit.jdbc.dialect	<input type="text" value="{{jdbc_dialect}}"/>	🔒	🟢	ⓘ
ranger.jpa.audit.jdbc.driver	<input type="text" value="{{ranger_jdbc_driver}}"/>	🔒	🟢	ⓘ
ranger.jpa.audit.jdbc.password	<input type="password" value="....."/>	🔒		
ranger.jpa.audit.jdbc.url	<input type="text" value="{{audit_jdbc_url}}"/>	🔒	🟢	ⓘ
ranger.jpa.audit.jdbc.user	<input type="text" value="{{ranger_audit_db_user}}"/>	🔒	🟢	ⓘ
ranger.jpa.jdbc.credential.alias	<input type="text" value="rangeradmin"/>	🔒	🟢	ⓘ
ranger.jpa.jdbc.dialect	<input type="text" value="{{jdbc_dialect}}"/>	🔒	🟢	ⓘ
ranger.jpa.jdbc.password	<input type="password" value="....."/>	🔒		
ranger.jpa.jdbc.user	<input type="text" value="{{ranger_db_user}}"/>	🔒	🟢	ⓘ
Group Search Base	<input type="text" value="{{ranger_ug_ldap_group_searchbase}}"/>	🔒	🟢	ⓘ
Group Search Filter	<input type="text" value="{{ranger_ug_ldap_group_searchfilter}}"/>	🔒	🟢	ⓘ
ranger.service.host	<input type="text" value="{{ranger_host}}"/>	🔒	🟢	ⓘ
ranger.service.http.port	<input type="text" value="6080"/>	🔒	🟢	ⓘ
ranger.service.https.attrib.clientAuth	<input type="text" value="want"/>	🔒	🟢	ⓘ
ranger.service.https.attrib.keystore.keyalias	<input type="text" value="rangeradmin"/>	🔒	🟢	ⓘ
ranger.service.https.attrib.keystore.pass	<input type="password" value="....."/>	🔒		
ranger.service.https.attrib.ssl.enabled	<input type="text" value="true"/>	🔒	🟢	ⓘ
ranger.service.https.port	<input type="text" value="6182"/>	🔒	🟢	ⓘ

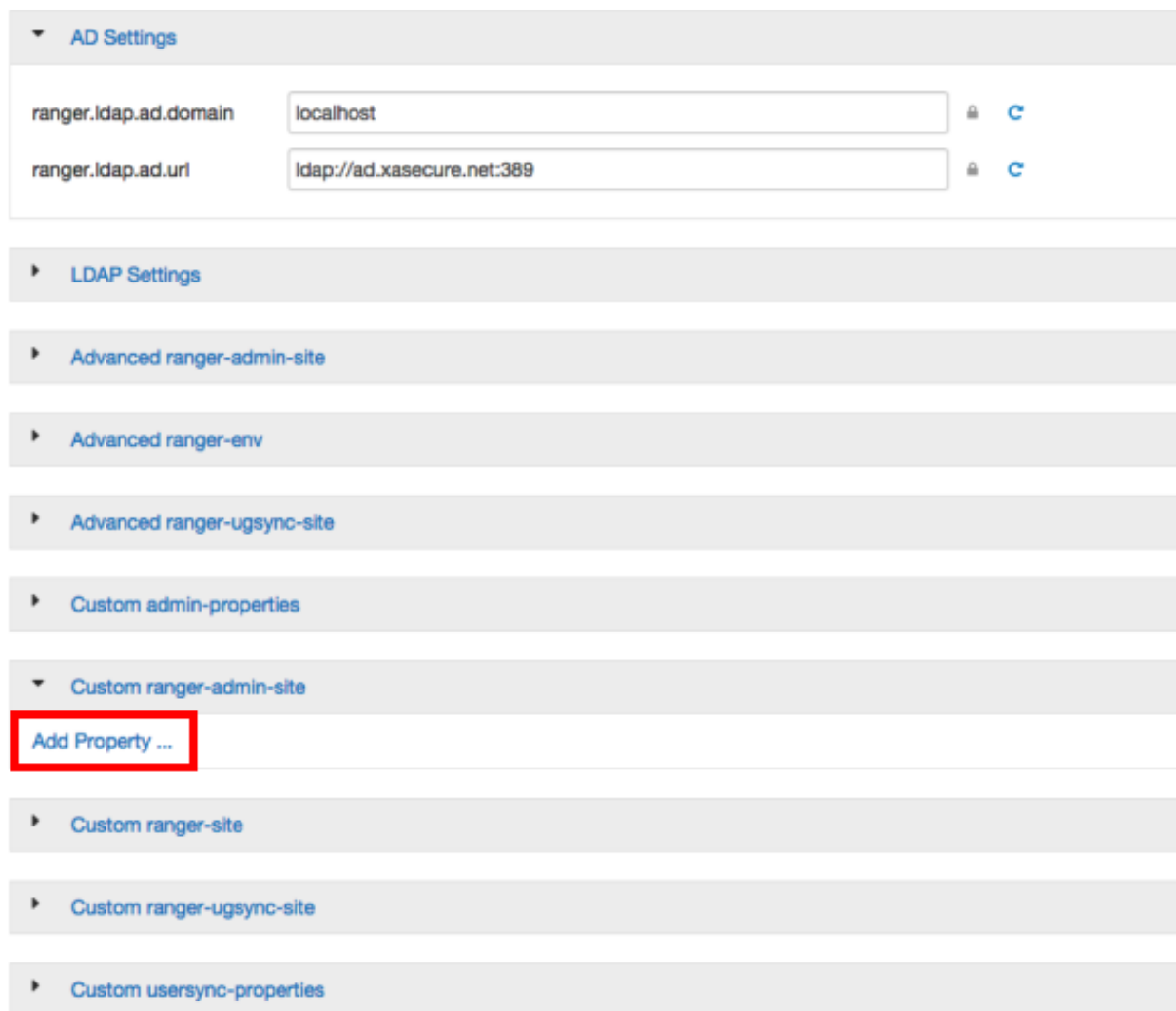
4. Under Custom ranger-admin-site, add the following properties:

- `ranger.service.https.attrib.keystore.file` – Specify the same value provided for the `ranger.https.attrib.keystore.file` property.

- `ranger.service.https.attrib.client.auth` – Specify the same value provided for the `ranger.service.https.attrib.clientAuth` property.

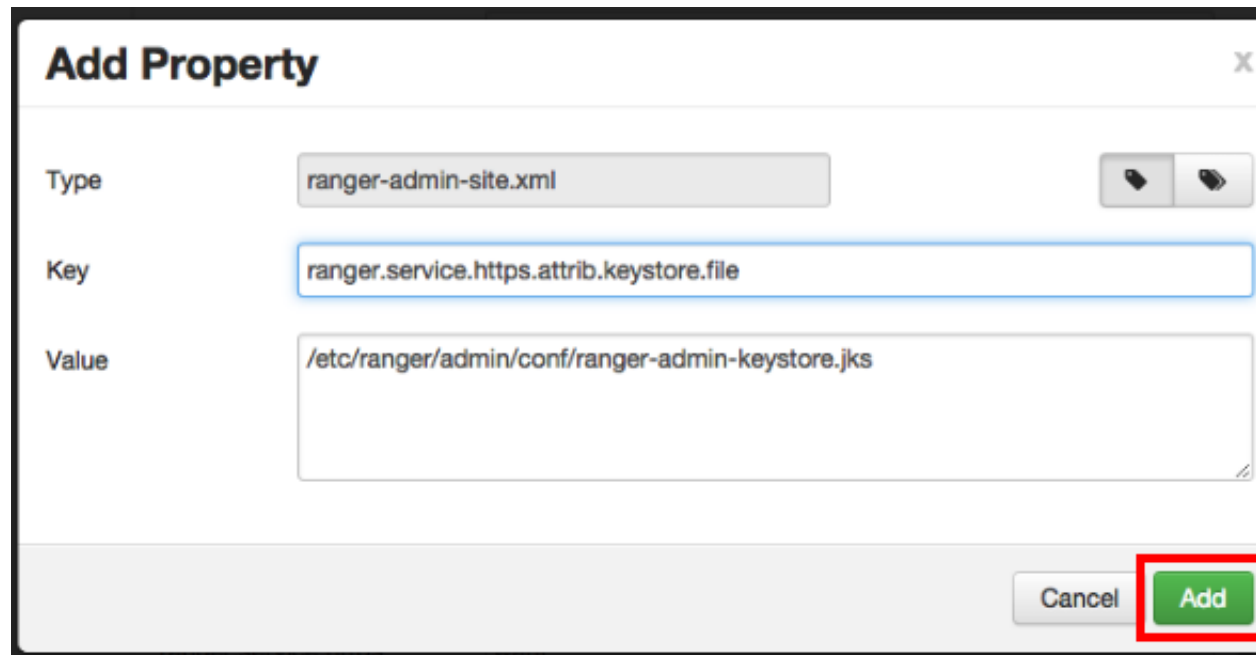
To add a Custom `ranger-admin-site` property:

- a. Select **Custom `ranger-admin-site`**, then click **Add Property**.



The screenshot displays a configuration interface for Ranger. It features several expandable sections: 'AD Settings', 'LDAP Settings', 'Advanced ranger-admin-site', 'Advanced ranger-env', 'Advanced ranger-ugsync-site', 'Custom admin-properties', 'Custom ranger-admin-site', 'Custom ranger-site', 'Custom ranger-ugsync-site', and 'Custom usersync-properties'. The 'Custom ranger-admin-site' section is expanded, and the 'Add Property ...' button is highlighted with a red rectangle. The 'AD Settings' section is also visible, showing two properties: 'ranger.ldap.ad.domain' with the value 'localhost' and 'ranger.ldap.ad.url' with the value 'ldap://ad.xasecure.net:389'. Each property has a lock icon and a refresh icon.

- b. On the Add Property pop-up, type the property name in the **Key** box, type the property value in the **Value** box, then click **Add**.



Add Property

Type: ranger-admin-site.xml

Key: ranger.service.https.attrib.keystore.file

Value: /etc/ranger/admin/conf/ranger-admin-keystore.jks

Cancel Add

5. Save your changes and start Ranger Admin.

When you attempt to access the Ranger Admin UI with the HTTPS protocol on the port specified by the `ranger.service.https.port` property, the browser should report that it does not trust the site. Click Proceed anyway and you should be able to access the Ranger Admin UI over HTTPS.

4.20.2.3. Configuring Ranger Usersync

1. Stop Ranger Usersync by selecting the **Ranger Usersync** link, then select **Started > Stop** next to Ranger Usersync.

The screenshot shows the Ambari web interface for a cluster named 'test_cluster'. At the top, there are navigation tabs for 'Dashboard', 'Services', 'Hosts', 'Alerts', and 'Admin'. The 'Alerts' tab is active, showing '16 alerts'. Below the navigation, the page title is 'c6403.ambari.apache.org' with a 'Back' link. There are tabs for 'Summary', 'Configs', 'Alerts', and 'Versions'. The 'Alerts' tab is selected, and a 'Host Actions' dropdown is visible in the top right.

The main content area is divided into two sections: 'Components' and 'Host Metrics'. The 'Components' section lists various services with their status and a dropdown menu for each. The 'Host Metrics' section shows various metrics like CPU Usage, Disk Usage, Load, Memory Usage, Network Usage, and Processes, all with 'No Data Available' placeholders.

The 'Components' list includes:

- Accumulo GC / Accumulo (Stopped)
- Accumulo Master / Accumulo (Stopped)
- Accumulo Monitor / Accumulo (Stopped)
- Accumulo Tracer / Accumulo (Stopped)
- App Timeline Server / YARN (Started)
- Atlas Metadata Server / Atlas (Started)
- DRPC Server / Storm (Started)
- Falcon Server / Falcon (Stopped)
- HBase Master / HBase (Stopped)
- History Server / MapReduce2 (Started)
- Hive Metastore / Hive (Started)
- HiveServer2 / Hive (Started)
- SmartSense HST Server / SmartSense (Stopped)
- Kafka Broker / Kafka (Started)
- Knox Gateway / Knox (Started)
- Metrics Collector / Ambari Metrics (Stopped)
- MySQL Server / Hive (Started)
- NameNode / HDFS (Started)
- Nimbus / Storm (Started)
- Oozie Server / Oozie (Started)
- Ranger Admin / Ranger (Started)
- Ranger Usersync / Ranger (Started)
- ResourceManager / YARN (Started) - **Stop** option is highlighted
- SNameNode / HDFS (Started)
- Spark History Server / Spark (Started)
- Storm UI Server / Storm (Started)

The dropdown menu for 'ResourceManager / YARN' is open, showing the following options:

- Restart
- Stop**
- Turn On Maintenance Mode

2. Check to see if `unixauthservice.jks` is in the `/etc/ranger/usersync/conf/` directory. If not, run the following commands in the CLI:

```
cd /etc/ranger/usersync/conf/
mkdir cert
keytool -genkeypair -keyalg RSA -alias selfsigned -keystore /etc/ranger/
usersync/conf/cert/unixauthservice.jks -keypass UnIx529p -storepass UnIx529p
-validity 3600 -keysize 2048 -dname 'cn=unixauthservice,ou=authenticator,o=
mycompany,c=US'
chown -R ranger:ranger /etc/ranger/usersync/conf/cert
chmod -R 400 /etc/ranger/usersync/conf/cert
```

3. Use the following CLI commands to create a truststore for the Ranger Admin's self-signed keystore. When prompted for a password, press the Enter key.

```
cd /etc/ranger/usersync/conf/
keytool -export -keystore /etc/ranger/admin/conf/ranger-admin-keystore.jks -
alias rangeradmin -file ranger-admin-trust.cerchown -R ranger:ranger /etc/
ranger/usersync/conf/cert
keytool -import -file ranger-admin-trust.cer -alias rangeradmintrust -
keystore mytruststore.jks -storepass changeit
chown ranger:ranger mytruststore.jks
```

4. Navigate back to Ranger and select **Configs > Advanced**, then click **Advanced ranger-usersync-site**. Set the following properties:

- `ranger.usersync.truststore.file` – Enter the path to the truststore file.
- `ranger.usersync.truststore.password` – Enter the truststore password.

The screenshot shows a configuration interface with two rows of properties:

- The first row is labeled `ranger.usersync.truststore.file` and has a text input field containing the path `/usr/hdp/current/ranger-usersync/conf/mytruststore.jks`. To the right of the input field are three icons: a lock, a green plus sign, and a blue refresh icon.
- The second row is labeled `ranger.usersync.truststore.password` and has two text input fields, both of which are masked with dots. To the right of the second input field is a lock icon.

5. Start Ranger Usersync by selecting the **Ranger Usersync** link on the Summary tab, then select **Stopped > Start** next to Ranger Usersync.

4.20.2.4. Configuring Ranger Plugins

The following section shows how to configure the Ranger HDFS plugin for SSL with a self-signed certificate. You can use the same procedure for other Ranger components. Additional steps required to configure the Ranger KMS plugin and server are provided in subsequent sections.



Note

- To ensure a successful connection after SSL is enabled, self-signed certificates should be imported to the Ranger Admin's trust store (typically `JDK cacerts`).
- The `ranger.plugin.<service>.policy.rest.ssl.config.file` property should be verified, for example:

```
ranger.plugin.hive.policy.rest.ssl.config.file ==> /etc/
hive/conf/conf.server/ranger-policymgr-ssl.xml
```

4.20.2.4.1. Configuring the Ranger HDFS Plugin for SSL

The following steps show how to configure the Ranger HDFS plugin for SSL. You can use the same procedure for other Ranger components.

1. Stop HDFS by selecting **Service Actions > Stop**.

The screenshot displays the Ambari interface for the HDFS service. The top navigation bar includes 'Ambari test_cluster', '0 ops', '13 alerts', and navigation tabs for 'Dashboard', 'Services', 'Hosts', 'Alerts', and 'Admin'. The left sidebar lists various services, with HDFS selected. The main content area is divided into 'Summary', 'Heatmaps', 'Configs', and 'Quick Links'. The 'Summary' section shows the following details:

- NameNode**: Started
- SNameNode**: Started
- DataNodes**: 1/1 Started
- NFSGateways**: 0/0 Started
- DataNodes Status**: 1 live / 0 dead / 0 decommissioning
- NameNode Uptime**: 21.82 days
- NameNode Heap**: 159.3 MB / 1011.3 MB (15.8% used)
- Disk Usage (DFS Used)**: 869.6 MB / 488.2 GB (0.17%)
- Disk Usage (Non DFS Used)**: 38.3 GB / 488.2 GB (7.84%)

The 'Service Actions' dropdown menu is open, showing options such as 'Start', 'Stop', 'Restart All', 'Restart DataNodes', 'Move NameNode', 'Move SNameNode', 'Enable NameNode HA', 'Run Service Check', 'Turn On Maintenance Mode', 'Rebalance HDFS', and 'Download Client Configs'. The 'Stop' option is highlighted with a red box.

The 'Metrics' section shows the following data:

NameNode GC count	NameNode GC time	NN Connection Load	NameNode Heap	NameNode Host Load
No Data Available	No Data Available	No Data Available	No Data Available	No Data Available

NameNode RPC	Failed disk volumes	Corrupted Blocks	Under Replicated Blocks	HDFS Space Utilization
No Data Available	n/a	0	481	n/a

2. Use the following CLI commands to change to the Ranger HDFS plugin directory and create a self-signed certificate.

```
cd /etc/hadoop/conf
keytool -genkey -keyalg RSA -alias rangerHdfsAgent -keystore ranger-plugin-keystore.jks -storepass myKeyFilePassword -validity 360 -keysize 2048
chown hdfs:hdfs ranger-plugin-keystore.jks
chmod 400 ranger-plugin-keystore.jks
```

When prompted, provide an identifiable string as the value for the "What is your first and last name?" question. then provide answers to the subsequent questions to create the keystore. When prompted for a password, press the Enter key.



Note

Important note: In the case where multiple servers talking to ranger admin for downloading policies for the same service/repository (e.g. HBase Master and Region servers, multiple NameNodes or Hive servers in an HA environment, be sure to use the repo name or a common string across all of the nodes (such as HbasePlugin, HdfsPlugin etc). (Note this and enter the same value in Common Name For Certificate field in the edit repository page in the Policy Manager UI).

3. Use the following CLI commands to create a truststore for the agent and add the Admin public key as a trusted entry. When prompted for a password, press the Enter key.

```
cd /etc/hadoop/conf
keytool -export -keystore /etc/ranger/admin/conf/ranger-admin-keystore.jks -alias rangeradmin -file ranger-admin-trust.cer
keytool -import -file ranger-admin-trust.cer -alias rangeradmintrust -keystore ranger-plugin-truststore.jks -storepass changeit
chown hdfs:hdfs ranger-plugin-truststore.jks
chmod 400 ranger-plugin-truststore.jks
```

4. Under Ranger Settings, provide the value in the **External URL** box in the format `https://<hostname of policy manager>:<https port>`.
5. Select **Advanced ranger-hdfs-policymgr-ssl** and set the following properties:
 - `xasecure.policymgr.clientssl.keystore` – Enter the location of the keystore created in the previous step.
 - `xasecure.policymgr.clientssl.keystore.password` – Enter the keystore password.
 - `xasecure.policymgr.clientssl.truststore` – Enter the location of the truststore created in the previous step.
 - `xasecure.policymgr.clientssl.truststore.password` – Enter the truststore password.

Advanced ranger-hdfs-policymgr-ssl

xasecure.policymgr. clientssl.keystore	<input type="text" value="/etc/hadoop/conf/ranger-plugin-keystore.jks"/>			
xasecure.policymgr. clientssl.keystore. credential.file	<input type="text" value="jceks://file{{credential_file}}"/>			
xasecure.policymgr. clientssl.keystore. password	<input type="password"/>	<input type="password"/>		
xasecure.policymgr. clientssl.truststore	<input type="text" value="/etc/hadoop/conf/ranger-plugin-truststore.jks"/>			
xasecure.policymgr. clientssl.truststore. credential.file	<input type="text" value="jceks://file{{credential_file}}"/>			
xasecure.policymgr. clientssl.truststore. password	<input type="password"/>	<input type="password"/>		

- 6. Select **Advanced ranger-hdfs-plugin-properties**, then select the **Enable Ranger for HDFS** check box.

Advanced ranger-hdfs-plugin-properties

Enable Ranger for HDFS	<input checked="" type="checkbox"/>			
Ranger repository config password	<input type="password"/>	<input type="password"/>		
Ranger repository config user	<input type="text" value="hadoop"/>			
common.name.for. certificate	<input type="text"/>			
hadoop.rpc.protection	<input type="text"/>			
Policy user for HDFS	<input type="text" value="ambari-qa"/>			

7. Click **Save** to save your changes.
8. Start HDFS by selecting **Service Actions > Start**.
9. Stop Ranger Admin by selecting **Service Actions > Stop**.
10. Use the following CLI commands to add the agent's self-signed cert to the Admin's trustedCACerts.

```
cd /etc/ranger/admin/conf
keytool -export -keystore /etc/hadoop/conf/ranger-plugin-keystore.jks
        -alias rangerHdfsAgent -file ranger-hdfsAgent-trust.cer -storepass
        myKeyFilePassword
keytool -import -file ranger-hdfsAgent-trust.cer -alias rangerHdfsAgentTrust
        -keystore <Truststore file used by Ranger Admin - can be the JDK cacerts> -
        storepass changeit
```

11. Restart Ranger Admin.
12. Log into the Ranger Policy Manager UI as the admin user. Click the **Edit** button of your repository (in this case, hadoopdev) and provide the CN name of the keystore as the value for **Common Name For Certificate**, then save your changes.
13. Start the HDFS service.
14. In the Policy Manager UI, select **Audit > Plugins**. You should see an entry for your repo name with HTTP Response Code 200.

4.20.2.4.2. Configuring the Ranger KMS Plugin for SSL

To configure the Ranger KMS (Key Management Service) plugin for SSL, use the procedure described in the previous section for HDFS, then perform the following additional step.

Log into the Policy Manager UI (as the keyadmin user) and click the **Edit** button of your KMS repository. Provide the CN name of the keystore as the value for Common Name For Certificate and save your changes. This property is not provided by default, so it must be added.

Ranger Access Manager Encryption

Service Manager Edit Service

Create Service

Service Details :

Service Name * cl1_kms

Description kms repo

Active Status Enabled Disabled

Config Properties :

KMS URL * kms://https@ip-172-31-26-219.ec2

Username * keyadmin

Password * *****

Add New Configurations

Name	Value
commonNameForCertificate	ip-172-31-26-219.ec2.internal

+

Test Connection

Save Cancel Delete

4.20.2.4.3. Configuring the Ranger KMS Server for SSL

Use the following steps to configure Ranger KMS (Key Management Service) Server for SSL.

1. Stop Ranger KMS by selecting **Service Actions > Stop**.
2. Use the following CLI commands to change to the Ranger KMS configuration directory and create a self-signed certificate.

```
cd /etc/ranger/kms/conf
keytool -genkey -keyalg RSA -alias rangerkms -keystore ranger-kms-keystore.
jks -storepass rangerkms -validity 360 -keysize 2048
chown kms:kms ranger-kms-keystore.jks
chmod 400 ranger-kms-keystore.jks
```

When prompted, provide an identifiable string as the value for the "What is your first and last name?" question. then provide answers to the subsequent questions to create the keystore. When prompted for a password, press the Enter key.

3. Select **Custom ranger-kms-site**, then add the following properties as shown below:

ranger.https.attrib.keystore.file

ranger.service.https.attrib.keystore.file (duplicate of above – workaround for now)

ranger.service.https.attrib.clientAuth

ranger.service.https.attrib.client.auth (duplicate of above – workaround for now)

ranger.service.https.attrib.keystore.keyalias

ranger.service.https.attrib.keystore.pass

ranger.service.https.attrib.ssl.enabled

ranger.service.https.port

Custom ranger-kms-site		
ranger.https.attrib.keystore.file	<input type="text" value="/etc/ranger/kms/conf/ranger-kms-keystore.jks"/>	+ -
ranger.service.https.attrib.client.auth	<input type="text" value="want"/>	+ -
ranger.service.https.attrib.clientAuth	<input type="text" value="false"/>	+ -
ranger.service.https.attrib.keystore.file	<input type="text" value="/etc/ranger/kms/conf/ranger-kms-keystore.jks"/>	+ -
ranger.service.https.attrib.keystore.keyalias	<input type="text" value="rangerkms"/>	+ -
ranger.service.https.attrib.keystore.pass	<input type="text" value="rangerkms"/>	+ -
ranger.service.https.attrib.ssl.enabled	<input type="text" value="true"/>	+ -
ranger.service.https.port	<input type="text" value="9393"/>	+ -
Add Property ...		

4. Under Advanced kms_env, update the value of kms_port to match the value of ranger.service.https.port.
5. Save your changes and restart Ranger KMS.

When you attempt to access the Ranger KMS UI with the HTTPS protocol on the port specified by the ranger.service.https.port property, the browser should report that it does not trust the site. Click Proceed anyway and you should be able to access the Ranger Admin UI over HTTPS.

6. Use the following CLI commands to export the Ranger KMS certificate.

```
cd /usr/hdp/<version>/ranger-kms/conf
keytool -export -keystore ranger-kms-keystore.jks -alias rangerkms -file
ranger-kms-trust.cer
```

7. Use the following CLI command to import the Ranger KMS certificate into the Ranger Admin truststore.

```
keytool -import -file ranger-kms-trust.cer -alias rangerkms -keystore
<Truststore file used by Ranger Admin - can be the JDK cacerts> -storepass
changeit
```

8. Use the following CLI command to import the Ranger KMS certificate into the Hadoop client truststore.

```
keytool -import -file ranger-kms-trust.cer -alias rangerkms -keystore /etc/
security/clientKeys/all.jks -storepass bigdata
```

9. Restart Ranger Admin and HDFS.

4.20.3. Configure Ranger Admin Database for SSL-Enabled MySQL

When an SSL-enabled database is configured for use with Ranger, you must add certain configurations to Ranger:

1. In Ambari>Ranger>Configs>Advanced>Custom admin-properties, add the following parameters:

- db_ssl_enabled=True
- db_ssl_required=True
- db_ssl_verifyServerCertificate=True
- javax_net_ssl_keyStore=/etc/ranger/admin/keystore
- javax_net_ssl_keyStorePassword=ranger
- javax_net_ssl_trustStore=/etc/ranger/admin/truststore
- javax_net_ssl_trustStorePassword=ranger

Change keystore and truststore file paths according to your environment.

If certificate verification is not required, you can set value `false` in property `db_ssl_verifyServerCertificate`. In this case, keystore and truststore file location need not to be valid and/or mandatory.

2. In Ambari>Ranger>Configs>Advanced>Custom ranger-admin-site, add the following parameters:

- ranger.db.ssl.enabled=true
- ranger.db.ssl.required=true

- `ranger.db.ssl.verifyServerCertificate=true`
- `ranger.keystore.file=/etc/ranger/admin/keystore`
- `ranger.keystore.password=ranger`

Change keystore file path according to your environment.

If certificate verification is not required, then you can set value `false` in property `ranger.db.ssl.verifyServerCertificate`. In this case, keystore and truststore file location need not to be valid and/or mandatory.

3. In Ambari>Ranger>Configs>Advanced>Advanced `ranger-admin-site`, add the following parameters:

- `ranger.truststore.file=/etc/ranger/admin/truststore`
- `ranger.truststore.password=password`

4. Install/restart Ranger.

4.21. Configure Non-Ambari Ranger SSL

4.21.1. Configuring Non-Ambari Ranger SSL Using Public CA Certificates

If you have access to Public CA issued certificates, use the following steps to configure non-Ambari Ranger SSL.

4.21.1.1. Configuring Ranger Admin

1. Use the following CLI command to stop Ranger Admin.

```
ranger-admin stop
```

2. Open the `ranger-admin-site.xml` file in a text editor.

```
vi /usr/hdp/current/ranger-admin/ews/webapp/WEB-INF/classes/conf/ranger-admin-site.xml
```

3. Update `ranger-admin-site.xml` as follows:

- `ranger.service.http.port` – Comment out the value for this property.
- `ranger.service.http.enabled` – Set the value of this property to `false`.
- `ranger.service.https.attrib.ssl.enabled` – Set the value of this property to `true`.
- `ranger.service.https.port` – Make sure that this port is available, or change the value to an available port number.

- `ranger.https.attrib.keystore.file` – Provide the location of the Public CA issued keystore file.
- `ranger.service.https.attrib.keystore.pass` – Enter the password for the keystore.
- `ranger.service.https.attrib.keystore.keyalias` – Enter the alias name for the keystore private key.
- `ranger.externalurl` – Set the value of this property in the format: `https://<hostname of policy manager>:<https port>`.
- Add or update the following properties with the values shown below:

```
<property>
  <name>ranger.service.https.attrib.clientAuth</name>
  <value>want</value>
</property>

<property>
  <name>ranger.service.https.attrib.client.auth</name>
  <value>want</value>
</property>

<property>
  <name>ranger.https.attrib.keystore.file</name>
  <value>/etc/ranger/admin/conf/ranger-admin-keystore.jks</value>
</property>

<property>
  <name>ranger.service.https.attrib.keystore.file</name>
  <value>/etc/ranger/admin/conf/ranger-admin-keystore.jks</value>
</property>
```

4. Save the changes to `ranger-admin-site.xml`, then use the following command to start Ranger Admin.

```
ranger-admin start
```

When you attempt to access the Ranger Admin UI with the HTTPS protocol on the port specified by the `ranger.service.https.port` property, the browser should report that it does not trust the site. Click Proceed anyway and you should be able to access the Ranger Admin UI over HTTPS.

4.21.1.2. Configuring Ranger Usersync

1. Use the following CLI command to stop the Ranger Usersync service.

```
ranger-usersync stop
```

2. Use the following commands to change to the Usersync install directory and open the `install.properties` file in a text editor.

```
cd /usr/hdp/current/ranger-usersync/
vi install.properties
```

3. Set the value of `POLICY_MGR_URL` in the format: `https://<hostname of policy manager>:<https port>` and save your changes.
4. Run the following commands to install the new settings.

```
cd /usr/hdp/current/ranger-usersync/  
./setup.sh
```

5. Use the following command to start the Ranger Usersync service.

```
ranger-usersync start
```

4.21.1.3. Configuring Ranger Plugins

This section shows how to configure the Ranger HDFS plugin for SSL. You can use the same procedure for other Ranger components.

1. Use the following CLI command to stop the NameNode.

```
su -l hdfs -c "/usr/hdp/current/hadoop-client/sbin/hadoop-daemon.sh stop  
namenode"
```

2. Open the HDFS `install.properties` file in a text editor.

```
vi /usr/hdp/<version>/ranger-hdfs-plugin/install.properties
```

3. Update `install.properties` as follows:

- `POLICY_MGR_URL` – Set this value in the format: `https://<hostname of policy manager>:<https port>`
 - `SSL_KEYSTORE_FILE_PATH` – The path to the location of the Public CA issued keystore file.
 - `SSL_KEYSTORE_PASSWORD` – The keystore password.
 - `SSL_TRUSTSTORE_FILE_PATH` – The truststore file path.
 - `SSL_TRUSTSTORE_PASSWORD` – The truststore password.
- Save the changes to the `install.properties` file.

4. Use the following command to see if `JAVA_HOME` is available.

```
echo $JAVA_HOME
```

5. If `JAVA_HOME` is not available, use the following command to set `JAVA_HOME` (Note that Ranger requires java 1.8).

```
export JAVA_HOME=<path for java 1.8>
```

6. Run the following commands to switch to the HDFS plugin install directory and run the `install agent` to update the plugin with the new configuration settings.

```
cd /usr/hdp/<version>/ranger-hdfs-plugin/  
./enable-hdfs-plugin.sh
```

7. Log into the Ranger Policy Manager UI as the admin user. Click the **Edit** button of your repository (in this case, `hadoopdev`) and provide the CN name of the keystore as the value for **Common Name For Certificate**, then save your changes.

8. Use the following command to start the NameNode.

```
su -l hdfs -c "/usr/hdp/current/hadoop-client/sbin/hadoop-daemon.sh start namenode"
```

9. In the Policy Manager UI, select **Audit > Plugins**. You should see an entry for your repo name with HTTP Response Code 200.

4.21.2. Configuring Non-Ambari Ranger SSL Using a Self Signed Certificate

If you do not have access to Public CA issued certificates, use the following steps to create a self-signed certificate and configure Non-Ambari Ranger SSL.

4.21.2.1. Configuring Ranger Admin

1. Use the following CLI command to stop Ranger Admin.

```
ranger-admin stop
```

2. Use the following commands to change to the Ranger Admin directory and create a self-signed certificate.

```
cd /etc/ranger/admin/conf
keytool -genkey -keyalg RSA -alias rangeradmin -keystore ranger-admin-keystore.jks -storepass xasecure -validity 360 -keysize 2048
chown ranger:ranger ranger-admin-keystore.jks
chmod 400 ranger-admin-keystore.jks
```

When prompted, provide the host name as the value for the "What is your first and last name?" question. then provide answers to the subsequent questions to create the keystore.



Note

When prompted for your password, press the Enter key. This will not work for Java keytool version 1.5.

3. Open the `ranger-admin-site.xml` file in a text editor.

```
vi /usr/hdp/current/ranger-admin/ews/webapp/WEB-INF/classes/conf/ranger-admin-site.xml
```

4. Update `ranger-admin-site.xml` as follows:

- `ranger.service.http.port` – Comment out the value for this property.
- `ranger.service.http.enabled` – Set the value of this property to `false`.
- `ranger.service.https.attrib.ssl.enabled` – Set the value of this property to `true`.
- `ranger.service.https.port` – Make sure that this port is available, or change the value to an available port number.

- `ranger.https.attrib.keystore.file` – Provide the location of the keystore file created previously: `/etc/ranger/admin/conf/ranger-admin-keystore.jks`.
- `ranger.service.https.attrib.keystore.pass` – Enter the password for the keystore (in this case, `xasecure`).
- `ranger.service.https.attrib.keystore.keyalias` – Enter the alias name for the keystore private key (in this case, `rangeradmin`).
- `ranger.externalurl` – Set the value of this property in the format: `https://<hostname of policy manager>:<https port>`.
- Add or update the following properties with the values shown below:

```
<property>
  <name>ranger.service.https.attrib.clientAuth</name>
  <value>want</value>
</property>

<property>
  <name>ranger.service.https.attrib.client.auth</name>
  <value>want</value>
</property>

<property>
  <name>ranger.https.attrib.keystore.file</name>
  <value>/etc/ranger/admin/conf/ranger-admin-keystore.jks</value>
</property>

<property>
  <name>ranger.service.https.attrib.keystore.file</name>
  <value>/etc/ranger/admin/conf/ranger-admin-keystore.jks</value>
</property>
```

5. Save the changes to `ranger-admin-site.xml`, then use the following command to start Ranger Admin.

```
ranger-admin start
```

When you attempt to access the Ranger Admin UI with the HTTPS protocol on the port specified by the `ranger.service.https.port` property, the browser should report that it does not trust the site. Click Proceed anyway and you should be able to access the Ranger Admin UI over HTTPS with the self-signed cert you just created.

4.21.2.2. Configuring Ranger Usersync

1. Use the following CLI command to stop the Ranger Usersync service.

```
ranger-usersync stop
```

2. Check to see if `unixauthservice.jks` is in the `/etc/ranger/usersync/conf/` directory. If not, run the following commands in the CLI:

```
cd /etc/ranger/usersync/conf/
mkdir cert
keytool -genkeypair -keyalg RSA -alias selfsigned -keystore /etc/ranger/
usersync/conf/cert/unixauthservice.jks -keypass UnIx529p -storepass UnIx529p
-validity 3600 -keysize 2048 -dname 'cn=unixauthservice,ou=authenticator,o=
mycompany,c=US'
chown -R ranger:ranger /etc/ranger/usersync/conf/cert
chmod -R 400 /etc/ranger/usersync/conf/cert
```

3. Use the following commands to change to the Usersync install directory and open the `install.properties` file in a text editor.

```
cd /usr/hdp/current/ranger-usersync/
vi install.properties
```

4. Set the value of `POLICY_MGR_URL` in the format: `https://<hostname of policy manager>:<https port>` and save your changes.
5. Use the following commands to create a truststore for the Ranger Admin's self-signed keystore. When prompted for a password, press the Enter key.

```
cd /etc/ranger/usersync/conf/
keytool -export -keystore /etc/ranger/admin/conf/ranger-admin-keystore.jks -
alias rangeradmin -file ranger-admin-trust.cerchown -R ranger:ranger /etc/
ranger/usersync/conf/cert
keytool -import -file ranger-admin-trust.cer -alias rangeradmintrust -
keystore mytruststore.jks -storepass changeit
chown ranger:ranger mytruststore.jks
```

6. Use the following commands to change to the Usersync conf directory and open the `ranger-ugsync-site.xml` file in a text editor.

```
cd /usr/hdp/current/ranger-usersync/conf/
vi ranger-ugsync-site.xml
```

Edit the following properties, then save your changes:

- `ranger.usersync.truststore.file` – Enter the path to the truststore file.
- `ranger.usersync.truststore.password` – Enter the truststore password.

7. Run the following commands to install the new settings.

```
cd /usr/hdp/current/ranger-usersync/
./setup.sh
```

8. Use the following command to start the Ranger Usersync service.

```
ranger-usersync start
```

4.21.2.3. Configuring Ranger Plugins

The following steps describe how to configure the Ranger HDFS plugin for SSL with a self-signed certificate in a non-Ambari cluster. You can use the same procedure for other Ranger components.

1. Use the following CLI command to stop the NameNode.

```
su -l hdfs -c "/usr/hdp/current/hadoop-client/sbin/hadoop-daemon.sh stop namenode"
```

2. Use the following commands to change to the Ranger HDFS plugin directory and create a self-signed certificate.

```
3. cd /etc/hadoop/conf
keytool -genkey -keyalg RSA -alias rangerHdfsAgent -keystore ranger-plugin-keystore.jks -storepass myKeyFilePassword -validity 360 -keysize 2048
chown hdfs:hdfs ranger-plugin-keystore.jks
chmod 400 ranger-plugin-keystore.jks
```

When prompted, provide an identifiable string as the value for the "What is your first and last name?" question. then provide answers to the subsequent questions to create the keystore. When prompted for a password, press the Enter key.



Note

Important note: In the case where multiple servers talking to ranger admin for downloading policies for the same service/repository (e.g. HBase Master and Region servers, multiple NameNodes or Hive servers in an HA environment, be sure to use the repo name or a common string across all of the nodes (such as HbasePlugin, HdfsPlugin etc). (Note this and enter the same value in Common Name For Certificate field in the edit repository page in the Policy Manager UI).

4. Use the following CLI commands to create a truststore for the agent and add the Admin public key as a trusted entry. When prompted for a password, press the Enter key.

```
cd /etc/hadoop/conf
keytool -export -keystore /etc/ranger/admin/conf/ranger-admin-keystore.jks -alias rangeradmin -file ranger-admin-trust.cer
keytool -import -file ranger-admin-trust.cer -alias rangeradmintrust -keystore ranger-plugin-truststore.jks -storepass changeit
chown hdfs:hdfs ranger-plugin-truststore.jks
chmod 400 ranger-plugin-truststore.jks
```

5. Open the HDFS `install.properties` file in a text editor.

```
vi /usr/hdp/<version>/ranger-hdfs-plugin/install.properties
```

6. Update `install.properties` as follows:

- `POLICY_MGR_URL` – Set this value in the format: `https://<hostname of policy manager>:<https port>`
 - `SSL_KEYSTORE_FILE_PATH` – The path to the location of the keystore file.
 - `SSL_KEYSTORE_PASSWORD` – The keystore password.
 - `SSL_TRUSTSTORE_FILE_PATH` – The truststore file path.
 - `SSL_TRUSTSTORE_PASSWORD` – The truststore password.
- Save the changes to the `install.properties` file.

7. Use the following command to see if `JAVA_HOME` is available.

```
echo $JAVA_HOME
```

8. If JAVA_HOME is not available , use the following command to set JAVA_HOME (Note that Ranger requires java 1.8).

```
export JAVA_HOME=<path for java 1.8>
```

9. Run the following commands to switch to the HDFS plugin install directory and run the install agent to update the plugin with the new configuration settings.

```
cd /usr/hdp/<version>/ranger-hdfs-plugin/  
./enable-hdfs-plugin.sh
```

10. Use the following command to stop Ranger Admin.

```
ranger-admin stop
```

11. Use the following commands to add the agent's self-signed cert to the Admin's trustedCACerts.

```
cd /etc/ranger/admin/conf  
keytool -export -keystore /etc/hadoop/conf/ranger-plugin-keystore.jks  
-alias rangerHdfsAgent -file ranger-hdfsAgent-trust.cer -storepass  
myKeyFilePassword  
keytool -import -file ranger-hdfsAgent-trust.cer -alias rangerHdfsAgentTrust  
-keystore <Truststore file used by Ranger Admin - can be the JDK cacerts> -  
storepass changeit
```

12. Use the following command to start Ranger Admin.

```
ranger-admin start
```

13. Log into the Ranger Policy Manager UI as the admin user. Click the **Edit** button of your repository (in this case, hadoopdev) and provide the CN name of the keystore as the value for **Common Name For Certificate**, then save your changes.

14. Use the following command to start the NameNode.

```
su -l hdfs -c "/usr/hdp/current/hadoop-client/sbin/hadoop-daemon.sh start  
namenode"
```

15. In the Policy Manager UI, select **Audit > Plugins**. You should see an entry for your repo name with HTTP Response Code 200.

4.22. Connecting to SSL-Enabled Components

This section explains how to connect to SSL enabled HDP Components.



Note

In order to access SSL enabled HDP Services through the Knox Gateway, additional configuration on the Knox Gateway is required, see [Configure SSL for Knox](#).

4.22.1. Connect to SSL Enabled HiveServer2 using JDBC

HiveServer2 implemented encryption with the Java SASL protocol's quality of protection (QOP) setting that allows data moving between a HiveServer2 over JDBC and a JDBC client to be encrypted.

From the JDBC client specify `sasl.sop` as part of the JDBC-Hive connection string, for example `jdbc:hive://hostname/dbname;sasl.qop=auth-int`. For more information on connecting to Hive, see [Data Integration Services with HDP, Moving Data into Hive: Hive ODBC and JDBC Drivers](#).



Tip

See [HIVE-4911](#) for more details on this enhancement.

4.22.2. Connect to SSL Enabled Oozie Server

On every Oozie client system, follow the instructions for the type of certificate used in your environment.

4.22.2.1. Use a Self-signed Certificate from Oozie Java Clients

When using a self-signed certificate, you must first install the certificate before the Oozie client can connect to the server.

1. Install the certificate in the keychain:
 - a. Copy or download the `.cert` file onto the client machine.
 - b. Run the following command (as root) to import the certificate into the JRE's keystore:

```
sudo keytool -import -alias tomcat -file path/to/certificate.cert -  
keystore <JRE_cacerts>
```

Where `$JRE_cacerts` is the path to the JRE's certs file. It's location may differ depending on the Operating System, but its typically called `cacerts` and located at `$JAVA_HOME/lib/security/cacerts`. It can be under a different directory in `$JAVA_HOME`. The default password is `changeit`.

Java programs, including the Oozie client, can now connect to the Oozie Server using the self-signed certificate.

2. In the connection strings change HTTP to HTTPS, for example, replace `http://oozie.server.hostname:11000/oozie` with `https://oozie.server.hostname:11443/oozie`.

Java does not automatically redirect HTTP addresses to HTTPS.

4.22.2.2. Connect to Oozie from Java Clients

In the connection strings change HTTP to HTTPS and adjust the port, for example, replace `http://oozie.server.hostname:11000/oozie` with `https://oozie.server.hostname:11443/oozie`.

Java does not automatically redirect HTTP addresses to HTTPS.

4.22.2.3. Connect to Oozie from a Web Browser

Use `https://oozie.server.hostname:11443/oozie` though most browsers should automatically redirect you if you use `http://oozie.server.hostname:11000/oozie`.

When using a Self-Signed Certificate, your browser warns you that it can't verify the certificate. Add the certificate as an exception.

5. Auditing in Hadoop

5.1. Using Apache Solr for Ranger Audits

Apache Solr is an open-source enterprise search platform. Apache Ranger can use Apache Solr to store audit logs, and Solr can also provide a search capability of the audit logs through the Ranger Admin UI.

It is recommended that Ranger audits be written to both Solr and HDFS. Audits to Solr are primarily used to enable search queries from the Ranger Admin UI. HDFS is a long-term destination for audits – audits stored in HDFS can be exported to any SIEM system, or to another audit store.

Apache Ranger uses Apache Solr to store audit logs and provides UI searching through the audit logs. Solr must be installed and configured before installing Ranger Admin or any of the Ranger component plugins. The default configuration for Ranger Audits to Solr uses the shared Solr instance provided under the [Ambari Infra](#) service. Solr is both memory and CPU intensive. If your production system has high volume of access requests, make sure that the Solr host has adequate memory, CPU, and disk space.

SolrCloud is the preferred setup for production usage of Ranger. [SolrCloud](#), which is deployed with the [Ambari Infra](#) service, is a scalable architecture that can run as a single node or multi-node cluster. It has additional features such as replication and sharding, which is useful for high availability (HA) and scalability. You should plan your deployment based on your cluster size. Because audit records can grow dramatically, plan to have at least 1 TB of free space in the volume on which Solr will store the index data. Solr works well with a minimum of 32 GB of RAM. You should provide as much memory as possible to the Solr process.

It is highly recommended to use SolrCloud with at least two Solr nodes running on different servers with [replication](#) enabled. You can use the information in this section to configure additional SolrCloud instances.

Configuration Options

- **Ambari Infra Managed Solr (default)** – Audits to Solr defaults to use the shared Solr instance provided under the [Ambari Infra](#) service. There are no additional configuration steps required for this option. [SolrCloud](#), which is deployed with the Ambari Infra service, is a scalable architecture which can run as a single node or multi-node cluster. This is the recommended configuration for Ranger. By default, a single-node SolrCloud installation is deployed when the Ambari Infra Service is chosen for installation. Hortonworks recommends that you install multiple Ambari Infra Solr Instances in order to provide distributed indexing and search for Atlas, Ranger, and LogSearch (Technical Preview). This can be accomplished by simply adding additional Ambari Infra Solr Instances to existing cluster hosts by selecting **Actions > Add Service** on the Ambari dashboard.
- **Externally Managed SolrCloud** – You can also install and manage an external [SolrCloud](#) that can run as single or multi-node cluster. It includes features such as replication and sharding, which are useful for high availability (HA) and scalability. With SolrCloud, customers need to plan the deployment based on the cluster size.

- **Externally Managed Solr Standalone** – Solr Standalone is NOT recommended for production use, and should be only used for testing and evaluation. Solr Standalone is a single instance of Solr that does not require ZooKeeper.



Warning

Solr Standalone is **NOT** recommended and support for this configuration will be deprecated in a future release.

- **SolrCloud for Kerberos** – This is the recommended configuration for SolrCloud in Kerberos environments.

The following sections describe how to install and configure Apache Solr for Ranger Audits:

- [Prerequisites \[501\]](#)
- [Installing Externally Managed SolrCloud \[501\]](#)
- [Configuring Externally Managed SolrCloud \[502\]](#)
- [Configuring Externally Managed Solr Standalone \[505\]](#)
- [Configuring SolrCloud for Kerberos \[506\]](#)

5.1.1. Prerequisites

Solr Prerequisites

- Ranger supports Apache Solr 5.2 or higher.
- Apache Solr requires the Java Runtime Environment (JRE) version 1.7 or higher.
- 1 TB free space in the volume where Solr will store the index data.
- 32 GB RAM.

SolrCloud Prerequisites

- SolrCloud supports replication and sharding. It is highly recommended that you use SolrCloud with at least two Solr nodes running on different servers with replication enabled.
- SolrCloud requires Apache ZooKeeper.
- SolrCloud with Kerberos requires Apache ZooKeeper and MIT Kerberos.

5.1.2. Installing Externally Managed SolrCloud

5.1.2.1. Installation and Configuration Steps

1. Run the following commands:

```
cd $HOME
git clone https://github.com/apache/incubator-ranger.git
cd incubator-ranger/security-admin/contrib/solr_for_audit_setup
```

2. Edit the `install.properties` file (see the instructions in the following sections).
3. Run the `./setup.sh` script.
4. Refer to `$SOLR_RANGER_HOME/install_notes.txt` for additional instructions.

5.1.2.2. Solr Installation

You can download the Solr package from [Apache Solr Downloads](#). Make sure that the Solr version is 5.2 or above. You can also use the Ranger `setup.sh` script to automatically download, install, and configure Solr. If you would like to use the `setup.sh` script to install Solr, set the following properties in the `install.properties` files, along with the settings from the one of the configuration options in the following sections.

Table 5.1. Solr `install.properties` Values for `setup.sh` script

Property Name	Value	Description
<code>SOLR_INSTALL</code>	<code>true</code>	When set to <code>true</code> , the <code>setup.sh</code> script will download the Solr package and install it.
<code>SOLR_DOWNLOAD_URL</code>	http://archive.apache.org/dist/lucene/solr/5.2.1/solr-5.2.1.tgz	It is recommended that you use one for Apache mirror sites to download the Solr package. You can choose a mirror site at http://lucene.apache.org/solr/mirrors-solr-latest-redirect.html
<code>SOLR_INSTALL_FOLDER</code>	<code>/opt/solr</code>	The Solr install folder.

5.1.3. Configuring Externally Managed SolrCloud



Note

Beginning with Ambari-2.4.0.0 and HDP-2.5.0, Ranger uses the [Ambari Infra](#) SolrCloud instance by default. Therefore, this procedure is only necessary for earlier versions, or if you are setting up additional external SolrCloud instances.

Use the following procedure to configure SolrCloud.

1. Use the following command to open the `install.properties` file in the `vi` text editor.

```
vi install.properties
```

Set the following property values, then save the changes to the `install.properties` file.

Table 5.2. Solr `install.properties` Values

Property Name	Value	Description
<code>JAVA_HOME</code>	<code><path_to_jdk></code> , for example: <code>/usr/jdk64/jdk1.8.0_40</code>	Provide the path to the JDK install folder. For Hadoop, you can check <code>/etc/hadoop/conf/hadoop-env.sh</code> for the value of <code>JAVA_HOME</code> . As noted previously, Solr only supports JDK 1.7 and higher.
<code>SOLR_USER</code>	<code>solr</code>	The Linux user used to run Solr.

Property Name	Value	Description
SOLR_INSTALL_FOLDER	/opt/lucidworks-hdpsearch/solr	The Solr installation directory.
SOLR_RANGER_HOME	/opt/solr/ranger_audit_server	The location where the Ranger-related configuration and schema files will be copied.
SOLR_RANGER_PORT	For HDP Search's Solr Instance: 8983 For Ambari Infra's Solr Instance: 8886	The Solr port for Ranger.
SOLR_DEPLOYMENT	solrcloud	The deployment type.
SOLR_ZK	<ZooKeeper_host>:2181/ ranger_audits	The Solr ZooKeeper host and port. It is recommended to provide a sub-folder to create the Ranger Audit related configurations so you can also use ZooKeeper for other Solr instances. Due to a Solr bug, if you are using a path (sub-folder), you can only specify one ZooKeeper host.
SOLR_SHARDS	1	If you want to distribute your audit logs, you can use multiple shards. Make sure the number of shards is equal or less than the number of Solr nodes you will be running.
SOLR_REPLICATION	1	It is highly recommend that you set up at least two nodes and replicate the indexes. This gives redundancy to index data, and also provides load balancing of Solr queries.
SOLR_LOG_FOLDER	/var/log/solr/ranger_audits	The folder for the Solr log files.
SOLR_MAX_MEM	2g	The memory allocation for Solr.

- Use the following command to run the set up script.

```
./setup.sh
```

- Run the following command **only once** from any node. This command adds the Ranger Audit configuration (including `schema.xml`) to ZooKeeper.

```
/opt/solr/ranger_audit_server/scripts/add_ranger_audits_conf_to_zk.sh
```

- Log in as the `solr` or `root` user and run the following command to start Solr on each node.

```
/opt/solr/ranger_audit_server/scripts/start_solr.sh
```

When Solr starts, a confirmation message appears.

```
Started Solr server on port 8983/8886 (pid=). Happy searching!
```

- Run the following command **only once** from any node. This command creates the Ranger Audit collection.

```
/opt/solr/ranger_audit_server/scripts/create_ranger_audits_collection.sh
```

- You can use a web browser to open the Solr Admin Console at the following address:

For HDP Search's Solr Instance:

```
http:<solr_host>:8983/solr
```

For Ambari Infra's Solr Instance:

```
http:<solr_host>:8886/solr
```



Note

You can use the following command to stop Solr:

```
/opt/solr/ranger_audit_server/scripts/stop_solr.sh
```

7. On the Ambari dashboard, select **Ranger > Configs > Ranger Audit**, then enable SolrCloud and External SolrCloud by clicking the **OFF** buttons. The button labels change to **ON** when SolrCloud and External SolrCloud are enabled.

The screenshot shows the Ambari dashboard interface for the 'Ranger Audit' configuration. The left sidebar lists various services, with 'Ranger' selected. The main content area is divided into two columns: 'Audit to Solr' and 'Audit to HDFS'. In the 'Audit to Solr' section, the 'SolrCloud' and 'External SolrCloud' toggle switches are highlighted with a red box, both showing 'ON'. Below them, the 'External SolrCloud kerberos' toggle is set to 'OFF'. The 'Destination HDFS Directory' is set to 'hdfs://c6406.ambari.apache.org:8020/ranger/ai'. The 'Ranger Audit' section also includes fields for 'ranger.audit.solr.zookeepers' (set to 'c6406.ambari.apache.org:2181/ranger_audits'), 'ranger.audit.solr.username' (set to 'ranger_solr'), and 'ranger.audit.solr.password'.

8. Set the value of the `ranger.audit.solr.zookeepers` property to `<host_name>:2181/ranger_audits`.
9. Select **Ranger > Configs > Advanced**, then select **Advanced ranger-env** and set the following properties:
 - `ranger_solr_replication_factor` – set this to the same value used in the `install.properties` file.
 - `ranger_solr_shards` – set this to the same value used in the `install.properties` file.
10. Click **Save**, then restart Ranger and all required services.

5.1.4. Configuring Externally Managed Solr Standalone



Warning

This configuration is NOT recommended for new installs of HDP-2.5 and is intended for non-production use. Support for this configuration will be deprecated in a future release.

Use the following procedure to configure Solr Standalone.

1. Use the following command to open the `install.properties` file in the vi text editor.

```
vi install.properties
```

Set the following property values, then save the changes to the `install.properties` file.

Table 5.3. Solr install.properties Values

Property Name	Value	Description
JAVA_HOME	<path_to_jdk>, for example: <code>/usr/jdk64/jdk1.8.0_60</code>	Provide the path to the JDK install folder. For Hadoop, you can check <code>/etc/hadoop/conf/hadoop-env.sh</code> for the value of <code>JAVA_HOME</code> . As noted previously, Solr only supports JDK 1.7 and higher.
SOLR_USER	<code>solr</code>	The Linux user used to run Solr.
SOLR_INSTALL_FOLDER	<code>/opt/solr</code>	The Solr installation directory.
SOLR_RANGER_HOME	<code>/opt/solr/ranger_audit_server</code>	The location where the Ranger-related configuration and schema files will be copied.
SOLR_RANGER_PORT	For HDP Search's Solr Instance: <code>8983</code>	The Solr port for Ranger.
SOLR_DEPLOYMENT	<code>standalone</code>	The deployment type.
SOLR_RANGER_DATA_FOLDER	<code>/opt/solr/ranger_audit_server/data</code>	The folder where the index data will be stored. The volume for this folder should have at least 1 TB free space for the index data, and should be backed up regularly.
SOLR_LOG_FOLDER	<code>/var/log/solr/ranger_audits</code>	The folder for the Solr log files.
SOLR_MAX_MEM	<code>2g</code>	The memory allocation for Solr.

2. Use the following command to run the Solr for Ranger setup script.

```
./setup.sh
```

3. To start Solr, log in as the `solr` or `root` user and run the following command.

```
/opt/solr/ranger_audit_server/scripts/start_solr.sh
```

When Solr starts, a confirmation message appears.

```
Started Solr server on port 8983/8886 (pid=). Happy searching!
```

4. You can use a web browser to open the Solr Admin Console at the following address:

For HDP Search's Solr Instance:

```
http:<solr_host>:8983/solr
```



Note

You can use the following command to stop Solr:

```
/opt/solr/ranger_audit_server/scripts/stop_solr.sh
```

5.1.5. Configuring SolrCloud for Kerberos



Note

Beginning with Ambari-2.4.0.0 and HDP-2.5.0, Ranger uses the [Ambari Infra SolrCloud](#) instance by default. Therefore, this procedure is only necessary for earlier versions, or if you are setting up additional SolrCloud instances.



Note

SolrCloud with Kerberos requires Apache ZooKeeper and MIT Kerberos. You should also review the other [SolrCloud Prerequisites](#).

Use the following steps to configure SolrCloud for Kerberos.

1. [Configure SolrCloud](#).
2. [Configure Kerberos for SolrCloud \[506\]](#)
3. [Configure SolrCloud for Kerberos \[507\]](#)

5.1.5.1. Configure Kerberos for SolrCloud

Use the following procedure to configure Kerberos for SolrCloud.

1. Create a principal "solr" in your KDC. You can make it host-specific or headless.
2. Log in as the root user to the KDC server and create the keytabs for users "solr" and HTTP.

```
kadmin.local
Authenticating as principal root/admin@EXAMPLE.COM with password.
kadmin.local: addprinc -randkey solr@EXAMPLE.COM
WARNING: no policy specified for solr@EXAMPLE.COM; defaulting to no policy
Principal "solr@EXAMPLE.COM" created.
kadmin.local: xst -k solr.service.keytab solr@EXAMPLE.COM
Entry for principal solr@EXAMPLE.COM with kvno 2, encryption type aes256-
cts-hmac-sha1-96 added to keytab WRFILE:solr.service.keytab.
Entry for principal solr@EXAMPLE.COM with kvno 2, encryption type aes128-
cts-hmac-sha1-96 added to keytab WRFILE:solr.service.keytab.
Entry for principal solr@EXAMPLE.COM with kvno 2, encryption type des3-cbc-
sha1 added to keytab WRFILE:solr.service.keytab.
Entry for principal solr@EXAMPLE.COM with kvno 2, encryption type arcfour-
hmac added to keytab WRFILE:solr.service.keytab.
Entry for principal solr@EXAMPLE.COM with kvno 2, encryption type des-hmac-
sha1 added to keytab WRFILE:solr.service.keytab.
Entry for principal solr@EXAMPLE.COM with kvno 2, encryption type des-cbc-
md5 added to keytab WRFILE:solr.service.keytab.
kadmin.local: quit
```

The example above creates a headless keytab for the "solr" service user. You should create one keytab per host. You should also create a principal for each host on which Solr is running. Use the procedure shown above, but use the principal name with the host. For example:

```
kadmin.local: addprinc -randkey solr/<SOLR_HOST_NAME>@EXAMPLE.COM
```

You will also need another keytab for Spnego. This is used by Solr to authenticate HTTP requests. Follow the process shown above, but replace "solr" with "HTTP". For example:

```
kadmin.local
kadmin.local: addprinc -randkey HTTP@EXAMPLE.COM
kadmin.local: xst -k HTTP.keytab HTTP@EXAMPLE.COM
kadmin.local: quit
```

3. After the keytabs are created, run the following commands to copy them to all of the hosts running Solr, chown to "solr", and chmod to 400.

```
mkdir -p /opt/solr/conf
#scp both the keytab files to the above folder
chown solr:solr /opt/solr/conf/solr.service.keytab
usermod -a -G hadoop solr
chmod 400 /opt/solr/conf/solr.service.keytab
chown solr:solr /opt/solr/conf/HTTP.keytab
chmod 400 /opt/solr/conf/HTTP.keytab
```



Note

The `usermod -a -G hadoop solr` command is required if you are using the HTTP (Spnego) keytab that is generated by Ambari.

5.1.5.2. Configure SolrCloud for Kerberos

Use the following procedure to configure SolrCloud for Kerberos.

1. Run the following commands:

```
cd /opt/solr
mkdir /opt/solr/conf
```

2. Create a new JAAS file in the `/opt/solr/conf` directory:

```
vi /opt/solr/conf/solr_jaas.conf
```

Add the following lines to the `solr_jaas.conf` file, but replace the REALM name `@EXAMPLE.COM` with your REALM.

```
Client {
    com.sun.security.auth.module.Krb5LoginModule required
    useKeyTab=true
    keyTab="/opt/solr/conf/solr.service.keytab"
    storeKey=true
    useTicketCache=true
    debug=true
    principal="solr@EXAMPLE.COM";
};
```

3. Copy the `solr_jaas.conf` file to all of the hosts on which Solr is running.
4. Edit the `solr.in.sh` file in the `<SOLR_INSTALL_HOME>/bin/` directory:

```
vi $SOLR_INSTALL_HOME/ranger_audit_server/scripts/solr.in.sh
```

Add the following lines at the end of the `solr.in.sh` file:

```
SOLR_JAAS_FILE=/opt/solr/conf/solr_jaas.conf
SOLR_HOST=`hostname -f`
ZK_HOST="$ZK_HOST1:2181,$ZK_HOST2:2181,$ZK_HOST3:2181/ranger_audits"
KERBEROS_REALM="EXAMPLE.COM"
SOLR_KEYTAB=/opt/solr/conf/solr.service.keytab
SOLR_KERB_PRINCIPAL=HTTP@${KERBEROS_REALM}
SOLR_KERB_KEYTAB=/opt/solr/conf/HTTP.keytab
SOLR_AUTHENTICATION_CLIENT_CONFIGURER="org.apache.solr.client.solrj.impl.
Krb5HttpClientConfigurer"
SOLR_AUTHENTICATION_OPTS=" -DauthenticationPlugin=org.apache.solr.security.
KerberosPlugin
-Djava.security.auth.login.config=${SOLR_JAAS_FILE} -Dsolr.kerberos.
principal=${SOLR_KERB_PRINCIPAL}
-Dsolr.kerberos.keytab=${SOLR_KERB_KEYTAB} -Dsolr.kerberos.cookie.domain=
${SOLR_HOST} -Dhost=${SOLR_HOST}
-Dsolr.kerberos.name.rules=DEFAULT"
```

5. Copy the `solr.in.sh` file to all of the hosts on which Solr is running.
6. Run the following command to enable Kerberos as the authentication scheme by updating the `security.json` file in ZooKeeper.

```
$SOLR_INSTALL_HOME/server/scripts/cloud-scripts/zkcli.sh -zkhost
$ZK_HOST:2181 -cmd put /ranger_audits/security.json '{"authentication":
{"class": "org.apache.solr.security.KerberosPlugin"}}'
```

7. Run the following commands to restart Solr on all hosts.

```
/opt/solr/ranger_audit_server/scripts/stop_solr.sh
/opt/solr/ranger_audit_server/scripts/start_solr.sh
```


- On the Ambari dashboard, select **Ranger > Configs > Ranger Audit**, then enable External SolrCloud Kerberos by clicking the **OFF** button. The button label changes to **ON** when External SolrCloud Kerberos is enabled.

The screenshot shows the Ambari Ranger Audit configuration page. The 'External SolrCloud kerberos' toggle switch is highlighted with a red box and is currently set to 'ON'. Other toggles for 'Audit to Solr', 'SolrCloud', and 'External SolrCloud' are also visible and set to 'ON'. The 'Destination HDFS Directory' is set to 'hdfs://c6406.ambari.apache.org:8020/ranger/ai'. Below the toggles, there are input fields for 'ranger.audit.solr.zookeepers', 'ranger.audit.solr.username', and 'ranger.audit.solr.password'.

- Click **Save**, then restart Ranger and all required services.

5.1.5.3. Connecting to Kerberos-enabled SolrCloud

To connect to Kerberos-enabled Solr from your local machine:

- On both Linux and Mac, copy the `/etc/krb5.conf` file from the Solr host to your local `/etc/krb5.conf`. If you already have a local `/etc/krb5.conf` file, merge the two files.
- Log in to the KDC host as root and run the following commands to create a KDC user:

```
kadmin.local
kadmin.local: addprinc $USERNAME@EXAMPLE.COM
kadmin.local: quit
```

3. Run the following command on your local machine.

```
kinit $USERNAME@EXAMPLE.COM
```

4. You can now use a browser to connect to the Solr URL.

5.2. Migrating Audit Logs from DB to Solr in Ambari Clusters

It is recommended that you store audits in both HDFS and Solr. Audit to DB is no longer recommended and the option is disabled in the Ambari UI. If your logs were previously stored on DB, you can migrate the logs to Solr.



Note

By default, Solr only indexes the last 30 days' logs.

Before you migrate your audit logs from DB to Solr, make sure your cluster meets the following requirements:

- Solr must be installed and running (see [Using Apache Solr for Ranger Audits](#)).
- All plug-ins must be upgraded and writing audit logs to Solr (i.e., plugins must not be writing audit logs to DB.)
- The DB server must be running, and the credentials used to connect Audit to DB must be available.
- Ranger must be running with the audit source as Solr, and the Solr URL must be configured.

To migrate your audit logs from DB to Solr, complete the following instructions:

1. Configure the properties `ranger.audit.source.type` and `ranger.audit.solr.urls`:

Property Name	Sample Value	Location
<code>ranger.audit.source.type</code>	<code>solr</code>	Ranger>Configs>Advanced>Advanced ranger-admin-site
<code>ranger.audit.solr.urls</code>	Syntax: <code>http://<solr_host>:<port>/solr/ranger_audits</code> Example: <code>http://192.168.0.2:8983/solr/ranger_audits</code> Example: <code>http://192.168.0.2:8886/solr/ranger_audits</code>	Ranger>Configs>Ranger Audit

2. Verify or enter the `ranger.jpa.audit.jdbc.url` value.

After upgrading Ranger and changing the audit log destination from DB to Solr, Ambari may not automatically populate the required property values. If necessary, you can add these as custom properties from Ambari.

- a. Select **Ranger>Configs>Advanced>Custom ranger-admin-site**, then click **Add Property....**
- b. Enter the following information on the **Add Property** pop-up:
 - **Type:** preloaded with the value `ranger-admin-site.xml`
 - **Key:** enter `ranger.jpa.audit.jdbc.url`
 - **Value:** enter the JDBC audit string for your DB platform:

Table 5.4. JDBC Audit String

DB Platform	Syntax	Example Value
MySQL	<code>jdbc:mysql://DB_HOST:PORT/</code>	<code>jdbc:mysql://c6401.ambari.apache.org:3306/ranger_audit</code>
Oracle	For Oracle SID: <code>jdbc:oracle:thin:@AUDIT_HOST:PORT:SID</code>	<code>jdbc:oracle:thin:@c6401.ambari.apache.org:1521:ORCL</code>
	For Oracle Service Name: <code>jdbc:oracle:thin:@//AUDIT_HOST[:PORT][/ServiceName]</code>	<code>jdbc:oracle:thin:@//c6401.ambari.apache.org:1521/XE</code>
PostgreSQL	<code>jdbc:postgresql://AUDIT_HOST/audit</code>	<code>jdbc:postgresql://c6401.ambari.apache.org:5432/ranger_audit</code>
MS SQL	<code>jdbc:sqlserver://AUDIT_HOST;databaseName=ranger_admin</code>	<code>jdbc:sqlserver://c6401.ambari.apache.org:1433;databaseName=ranger_admin</code>
SQLA	<code>jdbc:sqlanywhere:host=AUDIT_HOST</code>	<code>jdbc:sqlanywhere:host=c6401.ambari.apache.org:2638;databaseName=ranger_admin</code>

3. Restart Ranger Admin.

4. Navigate to the Ranger admin directory and run the following command:

```
$/path/to/java -Dlogdir=ews/logs -Dlog4j.configuration=db_patch.log4j.xml
-cp ews/webapp/WEB-INF/classes/conf:ews/webapp/WEB-INF/classes/lib/*:ews/webapp/WEB-INF/:ews/webapp/META-INF/:ews/webapp/WEB-INF/lib/*:ews/webapp/WEB-INF/classes/:ews/webapp/WEB-INF/classes/META-INF:/usr/share/java/mysql-connector-java.jar org.apache.ranger.patch.cliutil.DbToSolrMigrationUtil
```

If the script succeeds, it prints the following details on the screen:

- Processing batch 'n' of total 'noOfBatches' (Where each batch contains 10000 rows.)
- Total number of migrated audit logs.

If the script fails to migrate data, it returns the error: Migration process failed, Please refer `ranger_db_patch.log` file.

5.3. Manually Enabling Audit Settings in Ambari Clusters

It is recommended that Ranger audits be written to both Solr and HDFS. Audits to Solr are primarily used to enable queries from the Ranger Admin UI. HDFS is a long-term destination for audits; audits stored in HDFS can be exported to any SIEM system, or to another audit store.

Solr and HDFS audits are generally enabled as part of the standard Ambari installation procedure. This section describes how to manually update Ambari audit settings for Solr and HDFS.

5.3.1. Manually Updating Ambari Solr Audit Settings

You can save and store Ranger audits to Solr if you have installed and configured the Solr service in your cluster.



Note

If you enabled Solr Audits as part of the standard Ambari installation procedure, audits to Solr are activated automatically when Ranger is enabled for a plugin.

To save Ranger audits to Solr:

1. From the Ambari dashboard, select the Ranger service. Select **Configs > Advanced**, then scroll down and select **Advanced ranger-admin-site**. Set the following property value:

- `ranger.audit.source.type = solr`

2. On the Ranger Configs tab, select **Ranger Audit**. The SolrCloud button should be set to ON. The SolrCloud configuration settings are loaded automatically when the SolrCloud button is set from OFF to ON, but you can also manually update the settings.



Note

Audits to Solr requires that you have already [configured SolrCloud](#).

3. Restart the Ranger service.
4. After the Ranger service has been restarted, you will then need to make specific configuration changes for each plugin to ensure that the plugin's data is captured in Solr.
5. For example, if you would like to configure HBase for audits to Solr, perform the following steps:
 - Select the Audit to Solr checkbox in Advanced ranger-hbase-audit.
 - Enable the Ranger plugin for HBase.
 - Restart the HBase component.

6. Verify that the Ranger audit logs are being passed to Solr by opening one of the following URLs in a web browser:

```
http://{RANGER_HOST_NAME}:6080/index.html#!/reports/audit/bigData
```

For HDP Search's Solr Instance:

```
http://{SOLR_HOST}:8983/solr/ranger_audits
```

For Ambari Infra's Solr Instance:

```
http://{SOLR_HOST}:8886/solr/ranger_audits
```

5.3.2. Manually Updating HDFS Audit Settings (for Ambari installs)



Note

HDFS audits are enabled by default in the standard Ranger Ambari installation procedure, and are activated automatically when Ranger is enabled for a plugin.

The following steps show how to save Ranger audits to HDFS for HBase. You can use the same procedure for other components.

1. From the Ambari dashboard, select the HBase service. On the Configs tab, scroll down and select **Advanced ranger-hbase-audit**. Select the **Audit to HDFS** check box.
2. Set the HDFS path where you want to store audits in HDFS:

```
xasecure.audit.destination.hdfs.dir = hdfs://  
$NAMENODE_FQDN:8020/ranger/audit
```

Refer to the `fs.defaultFS` property in the **Advanced core-site** settings.



Note

For NameNode HA, `NAMENODE_FQDN` is the cluster name. In order for this to work, `/etc/hadoop/conf/hdfs-site.xml` needs to be linked under `/etc/<component_name>/conf`.

3. Enable the Ranger plugin for HBase.
4. Make sure that the plugin sudo user has permission on the HDFS Path:

```
hdfs://NAMENODE_FQDN:8020/ranger/audit
```

For example, we need to create a Policy for Resource : `/ranger/audit`, all permissions to user `hbase`.

5. Save the configuration updates and restart HBase.
6. Generate some audit logs for the HBase component.

7. Check the HDFS component logs on the NameNode:

```
hdfs://NAMENODE_FQDN:8020/ranger/audit
```



Note

For a secure cluster, use the following steps to enable audit to HDFS for Storm, Kafka, and Knox:

- In `core-site.xml` set the `hadoop.proxyuser.<component>.groups` property with value `" * "` or service user.
- For the Knox plugin there is one additional property to add to `core-site.xml`. Add `hadoop.proxyuser.<component>.users` property with value `" * "` or service user (i.e. `knox`).
- For Kafka and Knox, link to `/etc/hadoop/conf/core-site.xml` under `/etc/<component_name>/conf`. For Storm, link to `/etc/hadoop/conf/core-site.xml` under `/usr/hdp/<version>/storm/extlib-daemon/ranger-storm-plugin-impl/conf`.
- Verify the service user principal.
- Make sure that the component user has permissions on HDFS.

5.4. Enabling Audit Logging in Non-Ambari Clusters

It is recommended that Ranger audits be written to both Solr and HDFS. Audits to Solr are primarily used to enable queries from the Ranger Admin UI. HDFS is a long-term destination for audits; audits stored in HDFS can be exported to any SIEM system, or to another audit store.

To enable auditing for HDFS, perform the steps listed below.

1. Set the `XAAUDIT.HDFS.ENABLE` value to `"true"` for the component plug-in in the `install.properties` file, which can be found here:

```
/usr/hdp/<version>/ranger-<component>=plugin
```

2. Configure the NameNode host in the `XAAUDIT.HDFS.HDFS_DIR` field.
3. Create a policy in the HDFS service from the Ranger Admin for individual component users (`hive/hbase/knox/storm/yarn/kafka/kms`) to provide `READ` and `WRITE` permissions for the audit folder (i.e., for enabling Hive component to log Audits to HDFS, you need to create a policy for the `hive` user with `Read` and `WRITE` permissions for the audit directory).
4. Set the Audit to HDFS caches logs in the local directory, which can be specified in `XAAUDIT.HDFS.LOCAL_BUFFER_DIRECTORY` (this can be like `/var/log/<component>/**`), which is the path where the audit is stored for a short time. This is similar for archive logs that need to be updated.

To enable auditing reporting from the Solr database, perform the steps listed below.

1. Modify the following properties in the Ranger service `install.properties` to enable auditing to the Solr database in Ranger:

- `audit_store=solr`
- For HDP Search's Solr Instance: `http:<solr_host>:8983/solr/ranger_audits`

For Ambari Infra's Solr Instance: `http:<solr_host>:8886/solr/ranger_audits`
- `audit_solr_user=ranger_solr`
- `audit_solr_password-NONE`

2. Restart Ranger.

To enable auditing to the Solr database for a plug-in (e.g., HBase), perform the steps listed below.

1. Set the following properties in `install.properties` of the plug-in to begin audit logging to the Solr database:

- `XAAUDIT.SOLR.IS.ENABLED=true`
- `XAAUDIT.SOLR.ENABLE=true`
- For HDP Search's Solr Instance: `XAAUDIT.SOLR.URL= http://solr_host:8983/solr/ranger_audits`

For Ambari Infra's Solr Instance: `XAAUDIT.SOLR.URL= http://solr_host:8886/solr/ranger_audits`
- `XAAUDIT.SOLR.USER=ranger_solr`
- `XAAUDIT.SOLR.PASSWORD=NONE`
- `XAAUDIT.SOLR.FILE_SPOOL_DIR=/var/log/hadoop/hdfs/audit/solr/spool`

2. Enable the Ranger HBase plug-in.

3. Restart the HBase component.

5.5. Managing Auditing in Ranger

To explore options for auditing policies in Ranger, [access the Ranger console](#), then click **Audit** in the top menu.



There are five tabs on the Audit page:

- [Access \[517\]](#)
- [Admin \[518\]](#)
- [Login Sessions \[519\]](#)
- [Plugins \[520\]](#)
- [Plugin Status \[521\]](#)

5.5.1. View Operation Details

To view details on a particular operation, click the Policy ID, Operation name, or Session ID.

Operation : update ×

Policy ID : 2
 Added Deleted

Policy Name : hbase-test-1-20160202224128
 Repository Type : hbase
 Updated Date : 02/16/2016 09:51:42 AM PST
 Updated By : Mal

Policy Details :

Fields	Old Value	New Value
table	*	*.y

OK

Operation : create ×

Policy ID : 13

Policy Name : New-Service-1-20160211205602
 Repository Type :
 Created Date : 02/11/2016 12:56:02 PM PST
 Created By : Admin

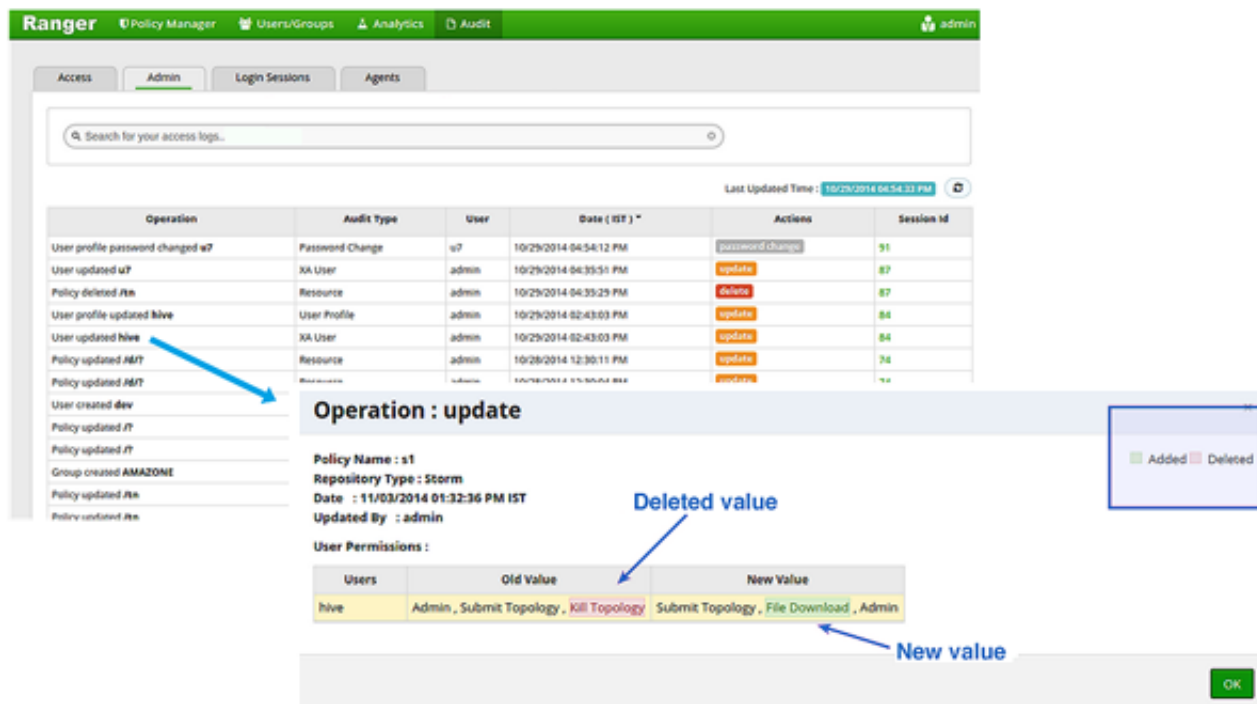
Policy Details :

Fields	New Value
Policy Name	New-Service-1-20160211205602
Policy Description	Default Policy for Service: New-Service
Policy Status	enabled

Users/Groups Permissions :

New Value
Groups: <empty>
Users: admin

OK



5.5.2. Differentiate Events from Multiple Clusters

To differentiate the audit events generated by multiple ephemeral clusters that are stored in the same shared service cluster, you can define a cluster name. The cluster name is visible in Ranger>Auditing>Access and Ranger>Auditing>Plugins.

1. From Ambari>component>Configs>Advanced>ranger-component-audit, define a value for `ranger.plugin.component.ambari.cluster.name=cluster_name`.
2. Restart Ranger and the component.

For example, in Ambari>HDFS>Configs>Advanced>ranger-hdfs-audit:
`ranger.plugin.hdfs.ambari.cluster.name=cluster_c6700`.

5.5.3. Access

Provides Service activity data for all Policies that have Audit set to **On**. The default service Policy is configured to log all user activity within the Service. This default policy does not contain user and group access rules.

You can filter the data based on the following criteria:

Table 5.5. Search Criteria

Search Criteria	Description
Access Enforcer	Ranger (ranger-acl) or Hadoop (hadoop-acl)
Access Type	Type of access user attempted (E.G., REVOKE, GRANT, OPEN, USE).

Search Criteria	Description
Client IP	IP address of the user system that tried to access the resource.
Result	Shows whether the operation was successful or not.
Service Name	The name of the service that the user tried to access.
Service Type	The type of the service that the user tried to access.
Start Date, End Date	Filters results for a particular date range.
User	Name of the user which tried to access the resource.
Cluster Name	Cluster name. Can be defined under Ambari>component>Configs>Advanced>ranger-component-audit file, using <code>ranger.plugin.component.ambari.cluster.name=cluster_name</code> .

The screenshot shows the Ranger Admin interface with the 'Admin' tab selected. A search bar is at the top with the text 'Search for your access audits...'. Below it, a table displays audit events. The table has the following columns: Policy ID, Event Time, User, Service Name / Type, Resource Name, Access Type, Result, Access Enforcer, Client IP, and Event Count. The table contains several rows of data, including events for policy 4 and 5, and a detailed view for policy 2 showing 'Attribute Details' for an 'expiry date' of 12/12/2012.

5.5.4. Admin

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

Operation	Audit Type	User	Date (PST) *	Actions	Session Id
Policy updated hbase-test-1-20160202224128	Ranger Policy	Mal	02/16/2016 09:51:42 AM	update	52509
Policy updated Example-Service-1-20160211205602	Ranger Policy	admin	02/11/2016 12:56:48 PM	update	52478
Service updated Example-Service	Ranger Service	admin	02/11/2016 12:56:34 PM	update	52478
Policy created New-Service-1-20160211205602	Ranger Policy	admin	02/11/2016 12:56:02 PM	create	52478
Service created New-Service	Ranger Service	admin	02/11/2016 12:56:02 PM	create	52478
Policy updated hbase-test-1-20160202224128	Ranger Policy	admin	02/11/2016 10:27:15 AM	update	52461
User updated Mal	XA User	admin	02/11/2016 10:26:06 AM	update	52461
Group created UX	XA Group	admin	02/11/2016 10:25:21 AM	create	52461
Policy created test-storm-1-2016021010740	Ranger Policy	admin	02/10/2016 05:07:40 PM	create	52391

You can filter the data based on the following criteria:

Table 5.6. Search Criteria

Search Criteria	Description
Action	These are operations performed on resources (actions like create, update, delete, password change).
Audit Type	There are three values Resource,asset and xa user according to operations performed on Service,policy and users.
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.
Session ID	The session count increments each time you try to login to the system
Start Date	Login time and date is stored for each session. A date range is used to filter the results for that particular date range.
User	Username who has performed create,update,delete operation.

5.5.5. Login Sessions

The Login Sessions tab logs the information related to the sessions for each login.

You can filter the data based on the following criteria:

Table 5.7. Search Criteria

Search Criteria	Description
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.

Search Criteria	Description
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 browser or library version used to login for the specific event (e.g. Mozilla, Java, Python)
Result	Logs whether or not the login was successful. Possible results can be Success, Wrong Password, Account Disabled, Locked, Password Expired or User Not Found.

The screenshot shows the Ranger web interface with the 'Login Sessions' tab selected. At the top, there are navigation tabs for 'Access', 'Admin', 'Login Sessions', and 'Plugins'. A search bar is present with the text 'Search for your login sessions...'. Below the search bar, the 'Last Updated Time' is shown as '02/09/2016 12:54:22 PM'. The main content is a table with the following columns: Session Id, Login Id, Result, Login Type, IP, User Agent, and Login Time (PST).

Session Id	Login Id	Result	Login Type	IP	User Agent	Login Time (PST)
52329	amb_ranger_admin	Success	Username/Password	192.168.64.101	Python-urllib/2.6	02/09/2016 12:50:32 PM
52328	admin	Success	Username/Password	192.168.64.101	Python-urllib/2.6	02/09/2016 12:50:32 PM
52327	admin	Success	Username/Password	192.168.64.101	Python-urllib/2.6	02/09/2016 12:50:32 PM
52326	admin	Success	Username/Password	192.168.64.1	Mozilla/5.0 (Macintosh; Intel ...	02/09/2016 12:39:38 PM
52325	amb_ranger_admin	Success	Username/Password	192.168.64.101	Python-urllib/2.6	02/09/2016 10:50:32 AM
52324	admin	Success	Username/Password	192.168.64.101	Python-urllib/2.6	02/09/2016 10:50:32 AM
52323	admin	Success	Username/Password	192.168.64.101	Python-urllib/2.6	02/09/2016 10:50:32 AM
52322	rangerusersync	Success	Username/Password	192.168.64.101	java/1.8.0_60	02/09/2016 10:21:22 AM
52321	rangerusersync	Success	Username/Password	192.168.64.101	java/1.8.0_60	02/09/2016 10:21:21 AM
52320	rangerusersync	Success	Username/Password	192.168.64.101	java/1.8.0_60	02/09/2016 10:21:21 AM
52319	rangerusersync	Success	Username/Password	192.168.64.101	java/1.8.0_60	02/09/2016 10:21:21 AM

5.5.6. Plugins

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

Table 5.8. Agents Search Criteria

Search Criteria	Description
Plugin IP	IP Address of the agent that tried to export the service.
Plugin ID	Name of the agent that tried to export the service.
HTTP Response Code	The HTTP code returned when trying to export the service.
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.
Service Name	The service name we are trying to export.
Cluster Name	Cluster name. Can be defined under Ambari>component>Configs>Advanced>ranger-component-

Search Criteria	Description
	audit file, using <code>ranger.plugin.component.ambari.cluster.name=cluster_name</code> .

Export Date (PST) *	Service Name	Plugin Id	Plugin IP	HTTP Response Code	Status
01/05/2017 09:13:09 AM	c6402_hive	hiveServer2@c6402.ambari.apache.org-c6402_hive	192.168.64.102	200	Policies synced to plugin
01/05/2017 09:12:29 AM	c6402_atlas	atlas@c6402.ambari.apache.org-c6402_atlas	192.168.64.102	200	Policies synced to plugin
01/05/2017 09:12:20 AM	c6402_hbase	hbaseRegional@c6402.ambari.apache.org-c6402_hbase	192.168.64.102	200	Policies synced to plugin
01/05/2017 09:12:13 AM	c6402_hbase	hbaseMaster@c6402.ambari.apache.org-c6402_hbase	192.168.64.102	200	Policies synced to plugin
01/05/2017 09:05:19 AM	c6402_hbase	hbaseRegional@c6402.ambari.apache.org-c6402_hbase	192.168.64.102	200	Policies synced to plugin
01/05/2017 09:05:05 AM	c6402_hbase	hbaseMaster@c6402.ambari.apache.org-c6402_hbase	192.168.64.102	200	Policies synced to plugin
01/05/2017 09:03:35 AM	c6402_hive	hiveServer2@c6402.ambari.apache.org-c6402_hive	192.168.64.102	200	Policies synced to plugin
01/05/2017 09:02:48 AM	c6402_hbase	hbaseRegional@c6402.ambari.apache.org-c6402_hbase	192.168.64.102	200	Policies synced to plugin
01/05/2017 09:02:13 AM	c6402_yarn	yarn@c6402.ambari.apache.org-c6402_yarn	192.168.64.102	200	Policies synced to plugin
01/05/2017 09:01:02 AM	c6402_hbase	hbaseMaster@c6402.ambari.apache.org-c6402_hbase	192.168.64.102	200	Policies synced to plugin
01/05/2017 08:59:37 AM	c6402_hadoop	hdfs@c6402.ambari.apache.org-c6402_hadoop	192.168.64.102	200	Policies synced to plugin

5.5.7. Plugin Status

This tab shows policies in effect for each plugin. Includes the relevant host info and when the plugin downloaded and started enforcing the policies.

Table 5.9. Plugin Status Search Criteria

Search Criteria	Description
Host Name	Host, e.g., c6401.ambari.apache.org.
Plugin IP	IP Address of the agent that uses the plugin.
Service Name	Name of the service that contains the policies, e.g., c6401_yarn.
Service Type	Component.

Service Name	Service Type	Host Name	Plugin IP	Policy (Time)			Tag (Time)		
				Active	Download	Last Update	Active	Download	Last Update
c6402_atlas	atlas	c6402.ambari.apache.org	192.168.64.102	01/05/2017 09:12:30 AM	01/05/2017 09:12:29 AM	01/05/2017 09:04:33 AM	--	--	--
c6402_hadoop	hdfs	c6402.ambari.apache.org	192.168.64.102	01/05/2017 08:59:39 AM	01/05/2017 08:59:37 AM	01/05/2017 08:59:30 AM	--	--	--
c6402_hbase	hbaseMaster	c6402.ambari.apache.org	192.168.64.102	01/05/2017 09:12:13 AM	01/05/2017 09:12:13 AM	01/05/2017 09:11:53 AM	--	--	--
c6402_hbase	hbaseRegional	c6402.ambari.apache.org	192.168.64.102	01/05/2017 09:12:20 AM	01/05/2017 09:12:20 AM	01/05/2017 09:11:53 AM	--	--	--
c6402_hive	hiveServer2	c6402.ambari.apache.org	192.168.64.102	01/05/2017 09:13:09 AM	01/05/2017 09:13:09 AM	01/05/2017 09:02:55 AM	--	--	--
c6402_yarn	yarn	c6402.ambari.apache.org	192.168.64.102	01/05/2017 09:02:14 AM	01/05/2017 09:02:13 AM	01/05/2017 09:02:04 AM	--	--	--

6. ACLs on HDFS

This guide describes how to use Access Control Lists (ACLs) on the Hadoop Distributed File System (HDFS). ACLs extend the HDFS permission model to support more granular file access based on arbitrary combinations of users and groups.

6.1. Configuring ACLs on HDFS

Only one property needs to be specified in the `hdfs-site.xml` file in order to enable ACLs on HDFS:

- **dfs.namenode.acls.enabled**

Set this property to "true" to enable support for ACLs. ACLs are disabled by default. When ACLs are disabled, the NameNode rejects all attempts to set an ACL.

Example:

```
<property>
  <name>dfs.namenode.acls.enabled</name>
  <value>true</value>
</property>
```

6.2. Using CLI Commands to Create and List ACLs

Two new sub-commands are added to FsShell: `setfacl` and `getfacl`. These commands are modeled after the same Linux shell commands, but fewer flags are implemented. Support for additional flags may be added later if required.

- **setfacl**

Sets ACLs for files and directories.

Example:

```
-setfacl [-bkR] {-m|-x} <acl_spec> <path>
-setfacl --set <acl_spec> <path>
```

Options:

Table 6.1. ACL Options

Option	Description
-b	Remove all entries, but retain the base ACL entries. The entries for User, Group, and Others are retained for compatibility with Permission Bits.
-k	Remove the default ACL.
-R	Apply operations to all files and directories recursively.
-m	Modify the ACL. New entries are added to the ACL, and existing entries are retained.
-x	Remove the specified ACL entries. All other ACL entries are retained.

Option	Description
--set	Fully replace the ACL and discard all existing entries. The <code>acl_spec</code> must include entries for User, Group, and Others for compatibility with Permission Bits.
<acl_spec>	A comma-separated list of ACL entries.
<path>	The path to the file or directory to modify.

Examples:

```
hdfs dfs -setfacl -m user:hadoop:rw- /file
hdfs dfs -setfacl -x user:hadoop /file
hdfs dfs -setfacl -b /file
hdfs dfs -setfacl -k /dir
hdfs dfs -setfacl --set user::rw-,user:hadoop:rw-,group::r--,other::r-- /
file
hdfs dfs -setfacl -R -m user:hadoop:r-x /dir
hdfs dfs -setfacl -m default:user:hadoop:r-x /dir
```

Exit Code:

Returns 0 on success and non-zero on error.

- **getfacl**

Displays the ACLs of files and directories. If a directory has a default ACL, `getfacl` also displays the default ACL.

Usage:

```
-getfacl [-R] <path>
```

Options:**Table 6.2. getfacl Options**

Option	Description
-R	List the ACLs of all files and directories recursively.
<path>	The path to the file or directory to list.

Examples:

```
hdfs dfs -getfacl /file
hdfs dfs -getfacl -R /dir
```

Exit Code:

Returns 0 on success and non-zero on error.

6.3. ACL Examples

Before the implementation of Access Control Lists (ACLs), the HDFS permission model was equivalent to traditional UNIX Permission Bits. In this model, permissions for each file or directory are managed by a set of three distinct user classes: Owner, Group, and Others. There are three permissions for each user class: Read, Write, and Execute. Thus, for any file system object, its permissions can be encoded in $3 \times 3 = 9$ bits. When a user attempts to access

a file system object, HDFS enforces permissions according to the most specific user class applicable to that user. If the user is the owner, HDFS checks the Owner class permissions. If the user is not the owner, but is a member of the file system object's group, HDFS checks the Group class permissions. Otherwise, HDFS checks the Others class permissions.

This model can sufficiently address a large number of security requirements. For example, consider a sales department that would like a single user – Bruce, the department manager – to control all modifications to sales data. Other members of the sales department need to view the data, but must not be allowed to modify it. Everyone else in the company (outside of the sales department) must not be allowed to view the data. This requirement can be implemented by running `chmod 640` on the file, with the following outcome:

```
-rw-r-----1 brucesales22K Nov 18 10:55 sales-data
```

Only Bruce can modify the file, only members of the sales group can read the file, and no one else can access the file in any way.

Suppose that new requirements arise. The sales department has grown, and it is no longer feasible for Bruce to control all modifications to the file. The new requirement is that Bruce, Diana, and Clark are allowed to make modifications. Unfortunately, there is no way for Permission Bits to address this requirement, because there can be only one owner and one group, and the group is already used to implement the read-only requirement for the sales team. A typical workaround is to set the file owner to a synthetic user account, such as "salesmgr," and allow Bruce, Diana, and Clark to use the "salesmgr" account via `sudo` or similar impersonation mechanisms. The drawback with this workaround is that it forces complexity onto end-users, requiring them to use different accounts for different actions.

Now suppose that in addition to the sales staff, all executives in the company need to be able to read the sales data. This is another requirement that cannot be expressed with Permission Bits, because there is only one group, and it is already used by sales. A typical workaround is to set the file's group to a new synthetic group, such as "salesandexecs," and add all users of "sales" and all users of "execs" to that group. The drawback with this workaround is that it requires administrators to create and manage additional users and groups.

Based on the preceding examples, you can see that it can be awkward to use Permission Bits to address permission requirements that differ from the natural organizational hierarchy of users and groups. The advantage of using ACLs is that it enables you to address these requirements more naturally, in that for any file system object, multiple users and multiple groups can have different sets of permissions.

Example 1: Granting Access to Another Named Group

To address one of the issues raised in the preceding section, we will set an ACL that grants Read access to sales data to members of the "execs" group.

- Set the ACL:

```
> hdfs dfs -setfacl -m group:execs:r-- /sales-data
```

- Run `getfacl` to check the results:

```
> hdfs dfs -getfacl /sales-data
# file: /sales-data
# owner: bruce
```



```
# group: sales
user::rw-
group:r--
group:execs:r--
mask:r--
other:---
```

- If we run the "ls" command, we see that the listed permissions have been appended with a plus symbol (+) to indicate the presence of an ACL. The plus symbol is appended to the permissions of any file or directory that has an ACL.

```
> hdfs dfs -ls /sales-data
Found 1 items
-rw-r-----+ 3 bruce sales 0 2014-03-04 16:31 /sales-data
```

The new ACL entry is added to the existing permissions defined by the Permission Bits. As the file owner, Bruce has full control. Members of either the "sales" group or the "execs" group have Read access. All others do not have access.

Example 2: Using a Default ACL for Automatic Application to New Children

In addition to an ACL enforced during permission checks, there is also the separate concept of a default ACL. A default ACL can only be applied to a directory – not to a file. Default ACLs have no direct effect on permission checks for existing child files and directories, but instead define the ACL that new child files and directories will receive when they are created.

Suppose we have a "monthly-sales-data" directory that is further subdivided into separate directories for each month. We will set a default ACL to guarantee that members of the "execs" group automatically get access to new subdirectories as they get created each month.

- Set a default ACL on the parent directory:

```
> hdfs dfs -setfacl -m default:group:execs:r-x /monthly-sales-data
```

- Make subdirectories:

```
> hdfs dfs -mkdir /monthly-sales-data/JAN
> hdfs dfs -mkdir /monthly-sales-data/FEB
```

- Verify that HDFS has automatically applied the default ACL to the subdirectories:

```
> hdfs dfs -getfacl -R /monthly-sales-data
# file: /monthly-sales-data
# owner: bruce
# group: sales
user::rwx
group:r-x
other:---
default:user::rwx
default:group:r-x
default:group:execs:r-x
default:mask:r-x
default:other:---

# file: /monthly-sales-data/FEB
# owner: bruce
```

```

# group: sales
user::rwx
group::r-x
group:execs:r-x
mask::r-x
other:---
default:user::rwx
default:group::r-x
default:group:execs:r-x
default:mask::r-x
default:other:---

# file: /monthly-sales-data/JAN
# owner: bruce
# group: sales
user::rwx
group::r-x
group:execs:r-x
mask::r-x
other:---
default:user::rwx
default:group::r-x
default:group:execs:r-x
default:mask::r-x
default:other:---

```

Example 3: Blocking Access to a Sub-Tree for a Specific User

Suppose there is a need to immediately block access to an entire sub-tree for a specific user. Applying a named user ACL entry to the root of that sub-tree is the fastest way to accomplish this without accidentally revoking permissions for other users.

- Add an ACL entry to block user Diana's access to "monthly-sales-data":

```
> hdfs dfs -setfacl -m user:diana:--- /monthly-sales-data
```

- Run `getfacl` to check the results:

```

> hdfs dfs -getfacl /monthly-sales-data
# file: /monthly-sales-data
# owner: bruce
# group: sales
user::rwx
user:diana:---
group::r-x
mask::r-x
other:---
default:user::rwx
default:group::r-x
default:group:execs:r-x
default:mask::r-x
default:other:---

```

It is important to keep in mind the order of evaluation for ACL entries when a user attempts to access a file system object:

- If the user is the file owner, the Owner Permission Bits are enforced.
- Else, if the user has a named user ACL entry, those permissions are enforced.

- Else, if the user is a member of the file's group or any named group in an ACL entry, then the union of permissions for all matching entries are enforced. (The user may be a member of multiple groups.)
- If none of the above are applicable, the Other Permission Bits are enforced.

In this example, the named user ACL entry accomplished our goal because the user is not the file owner and the named user entry takes precedence over all other entries.

6.4. ACLS on HDFS Features

POSIX ACL Implementation

ACLs on HDFS have been implemented with the POSIX ACL model. If you have ever used POSIX ACLs on a Linux file system, the HDFS ACLs work the same way.

Compatibility and Enforcement

HDFS can associate an optional ACL with any file or directory. All HDFS operations that enforce permissions expressed with Permission Bits must also enforce any ACL that is defined for the file or directory. Any existing logic that bypasses Permission Bits enforcement also bypasses ACLs. This includes the HDFS super-user and setting `dfs.permissions` to "false" in the configuration.

Access Through Multiple User-Facing Endpoints

HDFS supports operations for setting and getting the ACL associated with a file or directory. These operations are accessible through multiple user-facing endpoints. These endpoints include the FsShell CLI, programmatic manipulation through the `FileSystem` and `FileContext` classes, WebHDFS, and NFS.

User Feedback: CLI Indicator for ACLs

The plus symbol (+) is appended to the listed permissions of any file or directory with an associated ACL. To view, use the `ls -l` command.

Backward-Compatibility

The implementation of ACLs is backward-compatible with existing usage of Permission Bits. Changes applied via Permission Bits (`chmod`) are also visible as changes in the ACL. Likewise, changes applied to ACL entries for the base user classes (Owner, Group, and Others) are also visible as changes in the Permission Bits. Permission Bit and ACL operations manipulate a shared model, and the Permission Bit operations can be considered a subset of the ACL operations.

Low Overhead

The addition of ACLs will not cause a detrimental impact to the consumption of system resources in deployments that choose not to use ACLs. This includes CPU, memory, disk, and network bandwidth.

Using ACLs does impact NameNode performance. It is therefore recommended that you use Permission Bits, if adequate, before using ACLs.

ACL Entry Limits

The number of entries in a single ACL is capped at a maximum of 32. Attempts to add ACL entries over the maximum will fail with a user-facing error. This is done for two reasons: to simplify management, and to limit resource consumption. ACLs with a very high number of entries tend to become difficult to understand, and may indicate that the requirements are better addressed by defining additional groups or users. ACLs with a very high number of entries also require more memory and storage, and take longer to evaluate on each permission check. The number 32 is consistent with the maximum number of ACL entries enforced by the "ext" family of file systems.

Symlinks

Symlinks do not have ACLs of their own. The ACL of a symlink is always seen as the default permissions (777 in Permission Bits). Operations that modify the ACL of a symlink instead modify the ACL of the symlink's target.

Snapshots

Within a snapshot, all ACLs are frozen at the moment that the snapshot was created. ACL changes in the parent of the snapshot are not applied to the snapshot.

Tooling

Tooling that propagates Permission Bits will not propagate ACLs. This includes the `cp -p` shell command and `distcp -p`.

6.5. Use Cases for ACLs on HDFS

ACLs on HDFS supports the following use cases:

Multiple Users

In this use case, multiple users require Read access to a file. None of the users are the owner of the file. The users are not members of a common group, so it is impossible to use group Permission Bits.

This use case can be addressed by setting an access ACL containing multiple named user entries:

```
ACLs on HDFS supports the following use cases:
```

Multiple Groups

In this use case, multiple groups require Read and Write access to a file. There is no group containing all of the group members, so it is impossible to use group Permission Bits.

This use case can be addressed by setting an access ACL containing multiple named group entries:

```
group:sales:rw-  
group:execs:rw-
```

Hive Partitioned Tables

In this use case, Hive contains a partitioned table of sales data. The partition key is "country". Hive persists partitioned tables using a separate subdirectory for each distinct value of the partition key, so the file system structure in HDFS looks like this:

```
user
|-- hive
  |-- warehouse
  |-- sales
  |-- country=CN
  |-- country=GB
  |-- country=US
```

All of these files belong to the "salesadmin" group. Members of this group have Read and Write access to all files. Separate country groups can run Hive queries that only read data for a specific country, such as "sales_CN", "sales_GB", and "sales_US". These groups do not have Write access.

This use case can be addressed by setting an access ACL on each subdirectory containing an owning group entry and a named group entry:

```
country=CN
group: :rwx
group:sales_CN:r-x

country=GB
group: :rwx
group:sales_GB:r-x

country=US
group: :rwx
group:sales_US:r-x
```

Note that the functionality of the owning group ACL entry (the group entry with no name) is equivalent to setting Permission Bits.



Important

Storage-based authorization in Hive does not currently consider the ACL permissions in HDFS. Rather, it verifies access using the traditional POSIX permissions model.

Default ACLs

In this use case, a file system administrator or sub-tree owner would like to define an access policy that will be applied to the entire sub-tree. This access policy must apply not only to the current set of files and directories, but also to any new files and directories that are added later.

This use case can be addressed by setting a default ACL on the directory. The default ACL can contain any arbitrary combination of entries. For example:

```
default:user::rwx
default:user:bruce:rw-
default:user:diana:r--
default:user:clark:rw-
default:group::r--
```

```
default:group:sales::rw-
default:group:execs::rw-
default:others:---
```

It is important to note that the default ACL gets copied from the directory to newly created child files and directories at time of creation of the child file or directory. If you change the default ACL on a directory, that will have no effect on the ACL of the files and subdirectories that already exist within the directory. Default ACLs are never considered during permission enforcement. They are only used to define the ACL that new files and subdirectories will receive automatically when they are created.

Minimal ACL/Permissions Only

HDFS ACLs support deployments that may want to use only Permission Bits and not ACLs with named user and group entries. Permission Bits are equivalent to a minimal ACL containing only 3 entries. For example:

```
user::rw-
group:r--
others:---
```

Block Access to a Sub-Tree for a Specific User

In this use case, a deeply nested file system sub-tree was created as world-readable, followed by a subsequent requirement to block access for a specific user to all files in that sub-tree.

This use case can be addressed by setting an ACL on the root of the sub-tree with a named user entry that strips all access from the user.

For this file system structure:

```
dir1
|-- dir2
|   |-- dir3
|   |-- file1
|   |-- file2
|   |-- file3
```

Setting the following ACL on "dir2" blocks access for Bruce to "dir3","file1","file2," and "file3":

```
user:bruce:---
```

More specifically, the removal of execute permissions on "dir2" means that Bruce cannot access "dir2", and therefore cannot see any of its children. This also means that access is blocked automatically for any new files added under "dir2". If a "file4" is created under "dir3", Bruce will not be able to access it.

ACLs with Sticky Bit

In this use case, multiple named users or named groups require full access to a shared directory, such as "/tmp". However, Write and Execute permissions on the directory also give users the ability to delete or rename any files in the directory, even files created by other users. Users must be restricted so that they are only allowed to delete or rename files that they created.

This use case can be addressed by combining an ACL with the sticky bit. The sticky bit is existing functionality that currently works with Permission Bits. It will continue to work as expected in combination with ACLs.

7. Data Protection: HDFS Encryption

7.1. Ranger KMS Administration

The Ranger Key Management Service (Ranger KMS) is an open source, scalable cryptographic key management service supporting HDFS "data at rest" encryption.

Ranger KMS is based on the Hadoop KMS originally developed by the Apache community. The Hadoop KMS stores keys in a file-based Java keystore by default. Ranger extends the native Hadoop KMS functionality by allowing you to store keys in a secure database.

Ranger provides centralized administration of the key management server through the Ranger admin portal.

There are three main functions within the Ranger KMS:

1. **Key management.** Ranger admin provides the ability to create, update or delete keys using the Web UI or REST APIs. All [Hadoop KMS APIs](#) work with Ranger KMS using the keyadmin username and password.
2. **Access control policies.** Ranger admin also provides the ability to manage access control policies within Ranger KMS. The access policies control permissions to generate or manage keys, adding another layer of security for data encrypted in Hadoop.
3. **Audit.** Ranger provides full audit trace of all actions performed by Ranger KMS.

Ranger KMS along with HDFS encryption are recommended for use in all environments. In addition to secure key storage using a database, Ranger KMS is also scalable, and multiple versions of Ranger KMS can be run behind a load balancer.

For more information about HDFS encryption, see [HDFS "Data at Rest" Encryption](#).

7.1.1. Installing the Ranger Key Management Service

This section describes how to install the Ranger Key Management Service (KMS) using Ambari on a Kerberized cluster.

Prerequisites

Ranger KMS requires HDFS and Ranger to be installed and running on the cluster.

To install Ranger using Ambari, refer to the [Ranger Installation Guide](#). (For more information about the Ambari Add Service Wizard, see [Adding a Service in Apache Ambari Operations](#).)

To use 256-bit keys, install the Java Cryptography Extension (JCE) Unlimited Strength Jurisdiction Policy File on all hosts in the cluster. For installation information, see [Install the JCE](#). Make sure that the Java location is specified in the \$PATH environment variable.



Note

If you use the OpenJDK package, the JCE file is already built into the package.

7.1.1.1. Install Ranger KMS using Ambari (Kerberized Cluster)

To install Ranger KMS on a Kerberized cluster, complete the following steps.

1. Go to the Ambari Web UI, `http://<gateway-URL>:8080`.
2. From the Ambari dashboard, go to the **Actions** menu. Choose **Add Service**.
3. On the next screen, check the box next to **Ranger KMS**:

Add Service Wizard

<input checked="" type="checkbox"/>	HBase	1.1.1.2.3	A Non-relational distributed database, plus Phoenix, a high performance low latency applications.
<input checked="" type="checkbox"/>	Pig	0.15.0.2.3	Scripting platform for analyzing large datasets
<input type="checkbox"/>	Sqoop	1.4.6.2.3	Tool for transferring bulk data between Apache Hadoop and structured such as relational databases
<input checked="" type="checkbox"/>	Oozie	4.2.0.2.3	System for workflow coordination and execution of Apache Hadoop includes the installation of the optional Oozie Web Console which install the ExtJS Library.
<input checked="" type="checkbox"/>	ZooKeeper	3.4.6.2.3	Centralized service which provides highly reliable distributed coordination
<input type="checkbox"/>	Falcon	0.6.1.2.3	Data management and processing platform
<input checked="" type="checkbox"/>	Storm	0.10.0	Apache Hadoop Stream processing framework
<input type="checkbox"/>	Flume	1.5.2.2.3	A distributed service for collecting, aggregating, and moving large streaming data into HDFS
<input type="checkbox"/>	Accumulo	1.7.0.2.3	Robust, scalable, high performance distributed key/value store.
<input checked="" type="checkbox"/>	Ambari Metrics	0.1.0	A system for metrics collection that provides storage and retrieval collected from the cluster
<input type="checkbox"/>	Atlas	0.5.0.2.3	Atlas Metadata and Governance platform
<input checked="" type="checkbox"/>	Kafka	0.8.2.2.3	A high-throughput distributed messaging system
<input checked="" type="checkbox"/>	Knox	0.6.0.2.3	Provides a single point of authentication and access for Apache Hadoop cluster
<input type="checkbox"/>	Mahout	0.9.0.2.3	Project of the Apache Software Foundation to produce free implementations of distributed or otherwise scalable machine learning algorithms focusing on areas of collaborative filtering, clustering and classification
<input checked="" type="checkbox"/>	Ranger	0.5.0.2.3	Comprehensive security for Hadoop
<input checked="" type="checkbox"/>	Ranger KMS	0.5.0.2.3	Key Management Server
<input type="checkbox"/>	Slider	0.80.0.2.3	A framework for deploying, managing and monitoring existing distributed applications on YARN.
<input type="checkbox"/>	Spark	1.3.1.2.3	Apache Spark is a fast and general engine for large-scale data processing

4. Then, choose **Next**.
5. (Optional) In **Assign Masters**, if you wish to override the default host setting, specify the Ranger KMS host address. For example:

Add Service Wizard

HiveServer2:	vp-os-rh6-my-sec-nokms-1507
HBase Master:	vp-os-rh6-my-sec-nokms-1507
Oozie Server:	vp-os-rh6-my-sec-nokms-1507
ZooKeeper Server:	vp-os-rh6-my-sec-nokms-1507
ZooKeeper Server:	vp-os-rh6-my-sec-nokms-1507
ZooKeeper Server:	vp-os-rh6-my-sec-nokms-1507
Nimbus:	vp-os-rh6-my-sec-nokms-1507
DRPC Server:	vp-os-rh6-my-sec-nokms-1507
Storm UI Server:	vp-os-rh6-my-sec-nokms-1507
Kafka Broker:	vp-os-rh6-my-sec-nokms-1507
Kafka Broker:	vp-os-rh6-my-sec-nokms-1507
Kafka Broker:	vp-os-rh6-my-sec-nokms-1507
Kafka Broker:	vp-os-rh6-my-sec-nokms-1507
Metrics Collector:	vp-os-rh6-my-sec-nokms-1507
Ranger KMS Server:	vp-os-rh6-my-sec-nokms-1507
Knox Gateway:	vp-os-rh6-my-sec-nokms-150727-1736-2.openstacklocal (15.6 GB, 2 core) vp-os-rh6-my-sec-nokms-150727-1736-3.openstacklocal (15.6 GB, 2 core) vp-os-rh6-my-sec-nokms-150727-1736-4.openstacklocal (15.6 GB, 2 core) vp-os-rh6-my-sec-nokms-150727-1736-5.openstacklocal (15.6 GB, 2 core)
Ranger Admin:	vp-os-rh6-my-sec-nokms-1507
Ranger Usersync:	vp-os-rh6-my-sec-nokms-1507

← Back

6. In **Customize Services**, set required values (marked in red). Review other configuration settings, and determine whether you'd like to change any of the default values. (For more information about these properties, see [Ranger KMS Properties](#).)

a. Set the following required settings, marked in red in the "Advanced kms-properties" section:

- KMS_MASTER_KEY_PASSWD
- db_password
- db_root_password



Note

If do not wish to provide system Database Administrator (DBA) account details to the Ambari Ranger installer, you can use the `dba_script.py` Python script to create Ranger DB database users without exposing DBA account information to the Ambari Ranger installer. For more information, see [Setting up Database Users Without Sharing DBA Credentials](#).

Add Service Wizard

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

Customize Services

We have come up with recommended configurations for the services you selected. Customize them as

HDFS MapReduce2 YARN Tez Hive HBase Pig Oozie ZooKeeper Storm Amb

Knox Ranger **Ranger KMS** 3 Misc

Group **Ranger KMS Default (4)** Manage Config Groups Filter...

Advanced dbks-site

Advanced kms-env

Advanced kms-log4j

Advanced kms-properties 3

DB_FLAVOR	MYSQL		
KMS_MASTER_KEY_PASWORD	Type password	Retype Password	This is
REPOSITORY_CONFIG_PASSWORD	*****	*****	
REPOSITORY_CONFIG_USERNAME	keyadmin		
SQL_CONNECTOR_JAR	/usr/share/java/mysql-connector-java.jar		
db_host	vp-os-rh6-my-sec-nokms-150727-1736-5.openstacklocal		
db_name	rangerkms		
db_password	Type password	Retype Password	This is
db_root_password	Type password	Retype Password	This is
db_root_user	root		
db_user	rangerkms		

Also specify the username for `REPOSITORY_CONFIG_USERNAME`, so that Ranger will be able to connect to the Ranger KMS Server and look up keys for creating access policies. This user will need to be set to proxy into Ranger KMS in a Kerberos mode (steps included below).

- b. Add values for the following properties in the "Custom kms-site" section. These properties allow the specified system users (`hive`, `oozie`, and others) to proxy on behalf of other users when communicating with Ranger KMS. This helps individual services (such as Hive) use their own keytabs, but retain the ability to access Ranger KMS as the end user (use access policies associated with the end user).

- `hadoop.kms.proxyuser.hive.users`
- `hadoop.kms.proxyuser.oozie.users`
- `hadoop.kms.proxyuser.HTTP.users`
- `hadoop.kms.proxyuser.ambari.users`
- `hadoop.kms.proxyuser.yarn.users`
- `hadoop.kms.proxyuser.hive.hosts`
- `hadoop.kms.proxyuser.oozie.hosts`
- `hadoop.kms.proxyuser.HTTP.hosts`
- `hadoop.kms.proxyuser.ambari.hosts`
- `hadoop.kms.proxyuser.yarn.hosts`

- c. Add the following properties to the Custom KMS-site section of the configuration. These properties use the `REPOSITORY_CONFIG_USERNAME` specified in the first step in this section.

If you are using an account other than `keyadmin` to access Ranger KMS, replace "keyadmin" with the configured user for the Ranger KMS repository in Ranger admin:

- `hadoop.kms.proxyuser.keyadmin.groups=*`
- `hadoop.kms.proxyuser.keyadmin.hosts=*`
- `hadoop.kms.proxyuser.keyadmin.users=*`

Add Property

Type

kms-site.xml

Properties

key=value (one per line)

```
hadoop.kms.proxyuser.keyadmin.groups=*  
hadoop.kms.proxyuser.keyadmin.hosts=*  
hadoop.kms.proxyuser.keyadmin.users=*
```

d. Confirm settings of the following values in the "advanced kms-site" group:

- `hadoop.kms.authentication.type=kerberos`
- `hadoop.kms.authentication.kerberos.keytab=/etc/security/keytabs/spnego.service.keytab`
- `hadoop.kms.authentication.kerberos.principal=*`

7. Then, choose **Next**.

8. Review the default values on the **Configure Identities** screen. Determine whether you'd like to change any of the default values. Then, choose **Next**.

9. In **Review**, make sure the configuration values are correct. Ranger KMS will be listed under **Services**.

10. Then, choose **Deploy**.

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

12. The Summary screen displays the results. Choose **Complete**.

13. Restart the Ranger and Ranger KMS services.

7.1.1.1.1. Setting up Database Users Without Sharing DBA Credentials

If do not wish to provide system Database Administrator (DBA) account details to the Ambari Ranger installer, you can use the `dba_script.py` Python script to create Ranger DB database users without exposing DBA account information to the Ambari Ranger installer. You can then run the normal Ambari Ranger installation without specify a DBA user name and password.

To create Ranger DB users using the `dba_script.py` script:

1. Download the Ranger rpm using the yum install command.

```
yum install ranger-kms
```

2. You should see one file named `dba_script.py` in the `/usr/hdp/current/ranger-admin` directory.
3. Get the script reviewed internally and verify that your DBA is authorized to run the script.
4. Execute the script by running the following command:

```
python dba_script.py
```

5. Pass all values required in the argument. These should include `db flavor`, `JDBC jar`, `db host`, `db name`, `db user`, and other parameters.

- If you would prefer not to pass runtime arguments via the command prompt, you can update the `/usr/hdp/current/ranger-admin/install.properties` file and then run:

```
python dba_script.py -q
```

When you specify the `-q` option, the script will read all required information from the `install.properties` file

- You can use the `-d` option to run the script in "dry" mode. Running the script in dry mode causes the script to generate a database script.

```
python dba_script.py -d /tmp/generated-script.sql
```

Anyone can run the script, but it is recommended that the system DBA run the script in dry mode. In either case, the system DBA should review the generated script, but should only make minor adjustments to the script, for example, change the location of a particular database file. No major changes should be made that substantially alter the script – otherwise the Ranger install may fail.

The system DBA must then run the generated script.

6. Log in to the host where KMS is to be installed. Run the following commands to back up files:

```
cp /var/lib/ambari-agent/cache/common-services/RANGER_KMS/0.5.0.2.3/package/scripts/kms.py /var/lib/ambari-agent/cache/common-services/RANGER_KMS/0.5.0.2.3/package/scripts/kms.py.bak
cp /var/lib/ambari-server/resources/common-services/RANGER_KMS/0.5.0.2.3/package/scripts/kms.py /var/lib/ambari-server/resources/common-services/RANGER_KMS/0.5.0.2.3/package/scripts/kms.py.bak
```

7. In both of the `kms.py` files copied in the previous step, find and comment out the following line (shown here commented out).

```
#Execute(dba_setup, environment=env_dict, logoutput=True, user=params.kms_user)
```

8. Run the Ranger Ambari install procedure, but set **Setup Database and Database User to No** in the Ranger Admin section of the Customize Services screen.

7.1.1.1.2. Configure HDFS Encryption to use Ranger KMS Access

At this point, Ranger KMS should be installed and running. If you plan to use Ranger KMS for HDFS data at rest encryption, complete the following steps:

1. Create a link to `/etc/hadoop/conf/core-site.xml` under `/etc/ranger/kms/conf/`:

```
sudo ln -s /etc/hadoop/conf/core-site.xml /etc/ranger/kms/conf/core-site.xml
```

2. Configure HDFS to access Ranger KMS.

- a. In the left panel of the Ambari main menu, choose HDFS.
- b. Choose the **Configs** tab at the top of the page, and then choose the **Advanced** tab partway down the page.
- c. Specify the provider path (the URL where the Ranger KMS server is running) in the following two properties, if the path is not already specified:
 - In "Advanced core-site", specify `hadoop.security.key.provider.path`
 - In "Advanced hdfs-site", specify `dfs.encryption.key.provider.uri`

The Ranger KMS host is where Ranger KMS is installed. The Ranger KMS host name should have the following format:

```
kms://http@<kmshost>:9292/kms
```

3. Under Custom `core-site.xml`, set the value of the `hadoop.proxyuser.kms.groups` property to `*` or service user.
4. Restart the Ranger KMS service and the HDFS service.

7.1.1.1.3. Use a Kerberos Principal for the Ranger KMS Repository

In Ranger, all access policies are configured within a repository for each service. For more information, refer to the [Security Guide](#).

To manage access policies for Ranger KMS, a repository is needed with Ranger for the Ranger KMS service. Ambari creates the repository automatically using the repository config user and password provided.

The repository config user also needs to be created as a principal in Kerberos with a password. Use the following steps to use a Kerberos principal for the Ranger KMS repository.

1. Create system user `keyadmin` which should be sync in User Tabs in Ranger Admin.
2. Create principal `keyadmin@EXAMPLE.COM` with password `keyadmin`:

```
keyadmin.local -q 'addprinc -pw keyadmin keyadmin'
```

3. On the Add Service wizard Customize Services page, set the required values (marked in red).
4. Under `ranger-kms-properties`, set the principal and password in the `REPOSITORY_CONFIG_USERNAME` and `REPOSITORY_CONFIG_PASSWORD` fields.
5. To check logs, select **Audit to DB** under Advanced `ranger-kms-audit`.
6. Click **Next** to continue with the Ranger KMS Add Service wizard.

Advanced kms-properties

REPOSITORY_CONFIG_USERNAME	keyadmin@EXAMPLE.COM	⊕	↻	⊗
SQL_CONNECTOR_JAR	/usr/share/java/mysql-connector-java.jar	⊕	⊗	
db_host	mp-secranger-2406-3.openstacklocal	⊕	⊗	
db_user	rangerkms	⊕	⊗	
db_password	*****			
DB_FLAVOR	MYSQL	⊕	⊗	
db_root_user	root	⊕	⊗	
db_root_password	*****			
db_name	rangerkms	⊕	⊗	
KMS_MASTER_KEY_PASSWORD	*****			
REPOSITORY_CONFIG_PASSWORD	*****			

7.1.2. Store Master Key in a Hardware Security Module (HSM)

PCI compliance requires that keys are stored in Hardware Security Modules (HSMs) rather than a software KMS. For example, this is required for financial institutions working with customer credit/debit card terminals.



Note

You must have a separate partition for each KMS cluster.

To store keys in an HSM:

1. [Install the SafeNet Luna SA Client Software](#)
2. [Install Ranger KMS Hardware Security Module \(HSM\) \[541\]](#):
 - [Manually \(using the instructions on the Apache Wiki>Installing Ranger KMS HSM \(Manually\)\)](#)
 - [Install Ranger KMS HSM via Ambari with plain text password \[542\]](#)
 - [Install Ranger KMS HSM via Ambari with JCEKS \[543\]](#)
3. [Configure HSM High Availability \(HA\) \[544\]](#)
4. [HSM Migration \[547\]](#)
 - [Migrate HSM to Ranger DB \[547\]](#)
 - [Migrate Ranger DB to HSM \[548\]](#)
5. [Optional: Clear Objects from the HSM Partition \[548\]](#)

7.1.2.1. Install Ranger KMS Hardware Security Module (HSM)

Prerequisites

[Install the SafeNet Luna SA Client Software](#)



Note

You must have a separate partition for each KMS cluster.

About this Task

You can install Ranger KMS Hardware Security Module (HSM) in three ways:

- [Manually \(using the instructions on the Apache Wiki>Installing Ranger KMS HSM \(Manually\)\)](#)
- [Install Ranger KMS HSM via Ambari with plain text password \[542\]](#)

- [Install Ranger KMS HSM via Ambari with JCEKS \[543\]](#)

7.1.2.1.1. Install Ranger KMS HSM Manually

Prerequisites

[Install the SafeNet Luna SA Client Software](#)



Note

You must have a separate partition for each KMS cluster.

Steps

Refer to the instructions on the [Apache Wiki>Installing Ranger KMS HSM \(Manually\)](#).

7.1.2.1.2. Install Ranger KMS HSM via Ambari with plain text password

Prerequisites

[Install the SafeNet Luna SA Client Software](#)



Note

You must have a separate partition for each KMS cluster.

Steps

1. [Installing the Ranger Key Management Service \[532\]](#)
2. While configuring add the HSM related properties in Advanced dbks-site Menu (dbks-site.xml):
 - `ranger.ks.hsm.enabled=true`
 - `ranger.ks.hsm.partition.name=Partition Name`
 - `ranger.ks.hsm.partition.password=Partition Password`
 - `ranger.ks.hsm.type=LunaProvider`

3. Click on **Next** and follow the instructions to install Ranger KMS.

7.1.2.1.3. Install Ranger KMS HSM via Ambari with JCEKS

Prerequisites

[Install the SafeNet Luna SA Client Software](#)



Note

You must have a separate partition for each KMS cluster.

Steps

1. [Installing the Ranger Key Management Service \[532\]](#)
2. While configuring add the HSM related properties in Advanced dbks-site Menu (dbks-site.xml):
 - `ranger.ks.hsm.enabled=true`
 - `ranger.ks.hsm.partition.name=Partition Name`
 - `ranger.ks.hsm.partition.password=_`
 - `ranger.ks.hsm.partition.password.alias=ranger.kms.hsm.partition.password`

- `ranger.ks.hsm.type=LunaProvider`

Custom dbks-site

ranger.ks.hsm.enabled	true
ranger.ks.hsm.partition.name	HAHSM3
ranger.ks.hsm.partition.password	-
ranger.ks.hsm.partition.password.alias	ranger.kms.hsm.partition.password
ranger.ks.hsm.type	LunaProvider

[Add Property ...](#)

3. Click on **Next** and follow the instructions to install Ranger KMS.

Ranger KMS will fail to start (expected behavior).

4. Execute this command on the cluster where Ranger KMS is installed:

```
python /usr/hdp/current/ranger-kms/ranger_credential_helper.py -l "/usr/hdp/current/ranger-kms/cred/lib/*" -f /etc/ranger/kms/rangerkms.jceks -k ranger.kms.hsm.partition.password -v <Partition_Password> -c 1
```

5. Restart the KMS from Ambari.

7.1.2.2. Configure HSM High Availability (HA)

Prerequisites

You must have at least two Luna SA appliances with PED Authentication, or two with Password Authentication.

Steps

1. Set up appliances for HA:
 - a. Perform the network setup on both HA units: [Install the SafeNet Luna SA Client Software](#).
 - b. In `hsm showPolicies`, ensure that `Allow Cloning=on` and `Allow Network Replication=on`.
 - c. Initialize the HSMs on your Luna SA appliances. They must have the same cloning domain (i.e., must share the same red, domain PED Key if they are PED-authenticated) or they must share the same domain string if they are password-authenticated.

- d. Create a partition on each Luna SA. They do not need to have the same labels, but must have the same password.
 - e. Record the serial number of each partition created on each Luna SA (use partition show).
2. Register clients with Luna SA HA:
 - a. Proceed with normal client setup, [Prepare the Client for Network Trust Link](#).
 - b. Register your client computer with both Luna SAs.
 - c. Verify using `./vtl verify` command. It should show the numbers of partitions registered with client.
 3. Create the HA GroupNote for your client version:
 - **Version 5**
 - a. After creating partitions on (at least) two Luna appliances, and setting up Network Trust Links between those partitions and your client, use LunaCM to configure HA on your client:
 - i. Go to the directory: `/usr/safenet/lunaclient/bin/`
 - b. To add members in haadmin, create a new group on the client: `./vtl haAdmin newGroup -serialNum HA Group Number -label Groupname -password password`.

For example:

```
./vtl haAdmin newGroup -serialNum 156453092 -label myHAGroup -password S@fenet123
```
 - c. Add members into your haadmin: `./vtl haAdmin addMember -group HA Group Number -serialNum serial_number -password password`.

For example:

```
./vtl haAdmin addMember -group 1156453092 -serialNum 156451030 -password S@fenet123
```
 - d. Enable synchronization of HAadmin Members: `./vtl haAdmin synchronize -group HA Group Number -password password`.

For example:

```
./vtl haAdmin synchronize -enable -group 1156453092 -password S@fenet123
```
 - e. To Enable HAOnly: `./vtl haAdmin HAOnly -enable`.
 - f. Check haadmin status after synchronization: `./vtl haAdmin show`.

Note: After synchronization please verify kms master key copied to both partitions registered in hsm ha group. It takes time to copy master key to another partition.

- **Version 6**

- a. After creating partitions on (at least) two Luna appliances, and setting up Network Trust Links between those partitions and your client, use LunaCM to configure HA on your client:

- i. Go to directory: `/usr/safenet/lunaclient/bin/`.

- ii. Select Lunacm: `./lunacm`.

- b. To add members in hagroup, create a new group on the client: `haGroup creategroup -serialNumber serial number -l label -p password`.

For example:

```
lunacm:>haGroup creategroup -serialNumber 1047740028310 -l
HAHSM3 -p S@fenet123
```

- c. Use the `hagroup addmember` command to add new member into hagroup client: `hagroup addMember -group groupname -serialNumber serial number -password password`

Field descriptions:

- Label for the group (do NOT call the group just "HA"): *groupname*
- The serial number of the first partition OR the slot number of the first partition: *serial number*
- The password for the partition: *password*
- Lunacm also generates and assigns a Serial Number to the group itself.

For example:

```
lunacm:>hagroup addMember -group rkmsgroup -serialNumber
1047749341551 -password S@fenet123
```

- d. Use the `hagroup addmember` command to add another member to the HA group: `hagroup addMember -group groupname -serialNumber serial number -password password`.

For example:

```
lunacm:>hagroup addMember -serialNumber 1047740028310 -g
rkmslgroup -password S@fenet123
```

- e. Check group member in group using "hagroup listGroups" command: `hagroup listGroups`.

- f. Enable HAOnly: `hagroup HAOnly -enable`.
- g. Enable synchronization of HAGroup Members: `hagroup synchronize -group groupname -password password -enable`.

For example:

```
lunacm:>hagroup synchronize -group rkmslgroup -password
S@fenet123 -enable
```

- 4. After configuring HSM HA, to run Ranger KMS in HSM HA mode you must specify the virtual group name created above in `HSM_PARTITION_NAME` property of `install.properties` and setup and start Ranger KMS. Note: All other configuration for HSM in `install.properties` of Ranger KMS as mentioned in "Installing Ranger KMS HSM" will remain the same.

7.1.2.3. HSM Migration

If required, you can migrate from [HSM to Ranger DB](#) or [Ranger DB to HSM](#).

7.1.2.3.1. Migrate HSM to Ranger DB

Steps

1. If running, stop the Ranger KMS server.
2. Go to the Ranger KMS directory: `/usr/hdp/version/ranger-kms`.



Note

DB details must be correctly configured to which KMS needs migration to (located in the xml config file of Ranger KMS).

3. Run `./HSMK2DB.sh provider HSM_PARTITION_NAME`

For example:

```
./HSMK2DB.sh LunaProvider par19
```

4. Enter the partition password.
5. After the migration is completed: if you want to run Ranger KMS according to the new configuration (either with HSM enabled or disabled,) update the Ranger KMS properties if required.
6. Start Ranger KMS.

Note : After migration, when Ranger KMS is running with HSM disabled: from HSM, clear the Master Key object from the partition if it is not required as Master Key already being migrated to DB.

Deleting the master key is a destructive operation. If the master key is lost, there is potential data loss - data under encryption zones cannot be recovered. Therefore, it is a best practice to keep backups of the master key in DB as well as HSM.

7.1.2.3.2. Migrate Ranger DB to HSM

Steps

1. If running, stop the Ranger KMS server.
2. Go to the Ranger KMS directory: `/usr/hdp/version/ranger-kms`.



Note

DB details from which Ranger KMS needs migration must be correctly configured (located in the xml config file of Ranger KMS).

HSM details must be the KMS HSM to which we are migrating.

3. Run: `./DBMK2HSM.sh provider HSM_PARTITION_NAME`.

For example:

```
./DBMK2HSM.sh LunaProvider par19
```

4. Enter the partition password.
5. After the migration is completed: if you want to run Ranger KMS according to the new configuration (either with HSM enabled or disabled,) update the Ranger KMS properties if required.
6. Start Ranger KMS

Note: After migration, when Ranger KMS is running with HSM enabled: from DB table "ranger_masterkey", delete the Master Key row if it is not required as Master Key already being migrated to HSM.

7.1.2.4. Optional: Clear Objects from the HSM Partition

Steps

1. SSH to the HSM Appliance Server.

For example:

```
ssh admin@elab6.safenet-inc.com
```

2. Enter Password for the HSM Appliance Server when prompted.
3. Check the Partition Objects that you want to clear and enter the password for the partition when prompted:

```
Partition showContents -par partition_name
```

For example:

```
partition showContents -par par14
```




Note

All objects listed will be destroyed during step 3.

4. Clear the objects from HMS partition: `Partition clear -par partition_name`
5. Enter Password for Partition when prompted.

For example:

```
partition clear -par par14
```

7.1.3. Enable Ranger KMS Audit

Ranger KMS supports audit to DB, HDFS, and Solr. Solr is well-suited for short-term auditing and UI access (for example, one month of data accessible via quick queries in the Web UI). HDFS is typically used for archival auditing. They are not mutually exclusive; we recommend configuring audit to both Solr and HDFS.

First, make sure Ranger KMS logs are enabled:

1. Go to the Ambari UI: `http://<gateway>:8080`
2. Select `ranger-kms` from the service.
3. Click the Configs tab, and go to the accordion menu.
4. In the Advanced `ranger-kms-audit` list, set `xasecure.audit.is.enabled` to true.
5. Select "Audit to Solr" and/or "Audit to HDFS", depending on which database(s) you plan to use:

6. Save the configuration and restart the Ranger KMS service.

Next, check to see if the Ranger KMS Plugin is enabled:

1. Go to the Ranger UI: `http://<gateway>:6080`

2. Login with your keyadmin user ID and password (the defaults are `keyadmin`, `keyadmin`). The default repository will be added under KMS service.
3. Run a test connection for the service. You should see a 'connected successfully' pop-up message. If the connection is not successful, make sure that the configured user exists (in KDC for a secure cluster).
4. Choose the Audit > Plugin tab.
5. Check whether plugins are communicating. The UI should display `Http Response code 200` for the respective plugin.

The next two subsections describe how to save audit to Solr and HDFS.

7.1.3.1. Save Audits to Solr



Note

Saving audits to Solr requires that you have already [installed Solr](#) and [configured SolrCloud](#).

To save audits to Solr:

1. From the Ambari dashboard, select the Ranger service. Select **Configs > Advanced**, then scroll down and select **Advanced ranger-admin-site**. Set the following property value:
 - `ranger.audit.source.type = solr`
2. On the Ranger Configs tab, select **Ranger Audit**. The SolrCloud button should be set to ON. The SolrCloud configuration settings are loaded automatically when the SolrCloud button is set from OFF to ON, but you can also manually update the settings.
3. Restart the Ranger service.
4. Next, to enable Ranger KMS auditing to Solr, set the following properties in the Advanced `ranger-kms-audit` list:
 - a. Check the box next to `Enable audit to solr` in the Ranger KMS component.
 - b. Check the `Audit provider summary enabled` box, and make sure that `xasecure.audit.is.enabled` is set to `true`.
 - c. Restart Ranger KMS.



Note

Check audit logs on Ranger UI, to make sure that they are getting through Solr: `http://RANGER_HOST_NAME:6080/index.html#!/reports/audit/bigData` or:

For HDP Search's Solr Instance: `http:<solr_host>:8983/solr/ranger_audits`

For Ambari Infra's Solr Instance: `http:<solr_host>:8886/solr/ranger_audits`

7.1.3.2. Save Audits to HDFS

There are no configuration changes needed for Ranger properties.

To save Ranger KMS audits to HDFS, set the following properties in the Advanced ranger-kms-audit list.

Note: the following configuration settings must be changed in each Plugin.

1. Check the box next to `Enable Audit to HDFS` in the Ranger KMS component.
2. Set the HDFS path to the path of the location in HDFS where you want to store audits:

```
xasecure.audit.destination.hdfs.dir = hdfs://NAMENODE_FQDN:8020/ranger/audit
```

3. Check the `Audit provider summary enabled` box, and make sure that `xasecure.audit.is.enabled` is set to `true`.
4. Make sure that the plugin's root user (`kms`) has permission to access HDFS Path `hdfs://NAMENODE_FQDN:8020/ranger/audit`
5. Restart Ranger KMS.
6. Generate audit logs for the Ranger KMS.
7. **(Optional)** To verify audit to HDFS without waiting for the default sync delay (approximately 24 hours), restart Ranger KMS. Ranger KMS will start writing to HDFS after the changes are saved post-restart.

To check for audit data:

```
hdfs dfs -ls /ranger/audit/
```

To test Ranger KMS audit to HDFS, complete the following steps:

1. Under custom `core-site.xml`, set `hadoop.proxyuser.kms.groups` to `"*"` or to the service user.
2. In the custom `kms-site` file, add `hadoop.kms.proxyuser.keyadmin.users` and set its value to `"*"`. (If you are not using keyadmin to access Ranger KMS Admin, replace "keyadmin" with the user account used for authentication.)
3. In the custom `kms-site` file, add `hadoop.kms.proxyuser.keyadmin.hosts` and set its value to `"*"`. (If you are not using keyadmin to access Ranger KMS Admin, replace "keyadmin" with the user account used for authentication.)
4. Copy the `core-site.xml` to the component's class path (`/etc/ranger/kms/conf`)

OR

```
link to /etc/hadoop/conf/core-site.xml under /etc/ranger/kms/conf  
(ln -s /etc/hadoop/conf/core-site.xml /etc/ranger/kms/conf/core-site.xml)
```

5. Verify the service user principal. (For Ranger KMS it will be the `http` user.)
6. Make sure that the component user has permission to access HDFS. (For Ranger KMS the `http` user should also have permission.)

7.1.4. Enabling SSL for Ranger KMS

If you do not have access to Public CA-issued certificates, complete the following steps to create and configure self-signed certificates.



Note

The following examples contain sample values (folder locations, passwords, and filenames). Change these values according to your environment.

Considerations:

- Copy `keystore/truststore` files into a different location (e.g. `/etc/security/serverKeys`) than the `/etc/<component>/conf` folders.
- Make sure JKS file names are different from each other.
- Make sure correct permissions are applied.
- Make sure all passwords are secured.
- For the test connection to be successful after enabling SSL, self-signed certificates should be imported to the Ranger admin's trust store (typically JDK `cacerts`).
- Property `ranger.plugin.service.policy.rest.ssl.config.file` should be verified; for example:

```
ranger.plugin.kms.policy.rest.ssl.config.file ==> /etc/ranger/kms/
conf/ranger-policymgr-ssl.xml
```

To enable SSL:

1. Stop the Ranger KMS service:



2. Go to the Ranger KMS (and plugin) installation location, and create a self-signed certificate:

```
cd /etc/ranger/kms/conf/
```

```
keytool -genkey -keyalg RSA -alias rangerKMSAgent -keystore
<ranger-kms-ks> -storepass myKeyFilePassword -validity 360 -
keysize 2048
```

```
chown kms:kms <ranger-kms-ks>
```

```
chmod 400 <ranger-kms-ks>
```

where

<ranger-kms-ks> is the name of the Ranger KMS keystore (for example, ranger-plugin-keystore.jks)

3. Provide an identifiable string in response to the question "What is your first and last name?"

Important: In case multiple servers need to communicate with Ranger admin for downloading policies for the same service/repository, make sure to use the repo name or a common string across all nodes. Remember exactly what you entered, because this value will be required for the Common Name for Certificate field on the edit repository page in the policy manager UI.

To create the keystore, provide answers to the subsequent questions. **Note:** Press enter when prompted for a password.

4. Create a truststore for the Ranger KMS plugin, and add the public key of admin as a trusted entry into the truststore:

```
cd /etc/ranger/kms/conf/
```

```
keytool -export -keystore <ranger-admin-ks> -alias rangeradmin -file <cert-filename>
```

```
keytool -import -file <cert-filename> -alias rangeradmintrust -keystore <ranger-kms-ts> -storepass changeit
```

```
chown kms:kms <ranger-kms-ts>
```

```
chmod 400 <ranger-kms-ts>
```

where

<ranger-admin-ks> is the location of the Ranger Admin keystore (for example, /etc/ranger/admin/conf/ranger-admin-keystore.jks)

<ranger-kms-ts> is the name of the Ranger KMS plugin truststore (for example, ranger-plugin-truststore.jks)

<cert-filename> is the name of the Ranger Admin certificate file (for example, ranger-admin-trust.cer)

Note: Press enter when prompted for a password.

5. Update below properties available in Advanced `ranger-kms-policymgr-ssl`:
 - a. `xasecure.policymgr.clientssl.keystore`: Provide the location for the keystore that you created in the previous step.
 - b. `xasecure.policymgr.clientssl.keystore.password`: Provide the password for the keystore (`myKeyFilePassword`).

- c. `xasecure.policymgr.clientssl.truststore`: Provide the location for the truststore that you created in the previous step.
- d. `xasecure.policymgr.clientssl.truststore.password`: Provide the password for the truststore (changeit).

6. Add the plugin's self-signed cert into Admin's trustedCACerts:

```
cd /etc/ranger/admin/conf
```

```
keytool -export -keystore <ranger-kms-ks> -alias rangerKMSAgent
-file <cert-filename> -storepass myKeyFilePassword
```

```
keytool -import -file <cert-filename> -alias rangerkmsAgentTrust
-keystore <ranger-admin-ts> -storepass changeit
```

where

<ranger-kms-ks> is the path to the Ranger KMS keystore (for example, `/etc/ranger/kms/conf/ranger-plugin-keystore.jks`)

<cert-filename> is the name of the certificate file (for example, `ranger-kmsAgent-trust.cer`)

<ranger-admin-ts> is the name of the Ranger Admin truststore file (for example, the JDK cacerts file)

7. Log into the Policy Manager UI (as `keyadmin` user) and click on the Edit button of your KMS repository. Provide the CN name of the keystore for **Common Name For Certificate** (`commonNameForCertificate`), and save it. This property is not added by default.

The screenshot shows the 'Create Service' page in the Ranger Policy Manager UI. The page is titled 'Ranger Access Manager Encryption' and 'Service Manager Edit Service'. The 'Create Service' section is active. The 'Service Details' section includes fields for 'Service Name *' (value: d1_kms), 'Description' (value: kms repo), and 'Active Status' (radio buttons for 'Enabled' and 'Disabled', with 'Enabled' selected). The 'Config Properties' section includes fields for 'KMS URL *' (value: kms/https@ip-172-31-26-219.ec2), 'Username *' (value: keyadmin), and 'Password *' (masked with asterisks). Below these fields is a table for 'Add New Configurations' with columns 'Name' and 'Value'. The table contains one entry: 'commonNameForCertificate' with the value 'ip-172-31-26-219.ec2.internal'. At the bottom of the page are buttons for 'Test Connection', 'Save', 'Cancel', and 'Delete'.

Configuring the Ranger KMS Server

1. Go to the Ranger KMS config location and create a self-signed certificate:

```
cd /etc/ranger/kms/conf

keytool -genkey -keyalg RSA -alias rangerkms -keystore <ranger-
kms-ks> -storepass rangerkms -validity 360 -keysize 2048

chown kms:kms ranger-kms-keystore.jks

chmod 400 ranger-kms-keystore.jks
```

where

<ranger-kms-ks> is the name of the Ranger KMS keystore (for example, ranger-plugin-keystore.jks)

Provide an identifiable string in response to the question "What is your first and last name?" To create the keystore, provide answers to all subsequent questions to create the keystore **Note:** Press enter when prompted for a password.

2. Edit the following properties and values in Advanced `ranger-kms-site`:
 - `ranger.service.https.attrib.keystore.file`: Add file path of `ranger-kms-keystore.jks`
 - `ranger.service.https.attrib.client.auth`: `want`
 - `ranger.service.https.attrib.keystore.keyalias`: Add the alias used for creating `ranger-kms-keystore.jks`
 - `ranger.service.https.attrib.keystore.pass`: Add password used for creating `ranger-kms-keystore.jks`
 - `ranger.service.https.attrib.ssl.enabled`: `true`
3. Update `kms_port` in Advanced `kms-env` to 9393. Ambari will recommend the value to `{{ranger.service.https.port}}`.
4. Save your changes and start Ranger KMS.
5. In your browser (or from Curl) when you access the Ranger KMS UI using the HTTPS protocol on the `ranger.service.https.port` listed in Ambari, the browser should respond that it does not trust the site. Proceed, and you should be able to access Ranger KMS on HTTPS with the self-signed cert that you just created.
6. Export the Ranger KMS certificate:

```
cd /usr/hdp/<version>/ranger-kms/conf

keytool -export -keystore <ranger-kms-ks> -alias rangerkms -file
<cert-filename>
```

where

<ranger-kms-ks> is the name of the Ranger KMS keystore (for example, ranger-kms-keystore.jks)

<cert-filename> is the name of the certificate file (for example, ranger-kms-trust.cer)

7. Import the Ranger KMS certificate into the Ranger admin truststore:

```
keytool -import -file <cert-filename> -alias rangerkms -keystore <ranger-admin-ts> -storepass changeit
```

where

<cert-filename> is the name of the certificate file (for example, ranger-kms-trust.cer)

<ranger-admin-ts> is the name of the Ranger Admin truststore file (for example, JDK cacerts)



Note

Make sure Ranger Admin's truststore properties (ranger.truststore.file and ranger.truststore.password) are correctly configured in ranger-admin-site.xml.

8. Import the Ranger KMS certificate into the Hadoop client truststore:

```
keytool -import -file <cert-filename> -alias rangerkms -keystore <ts-filename> -storepass bigdata
```

where

<cert-filename> is the name of the certificate file (for example, ranger-kms-trust.cer)

<ts-filename> is the name of Hadoop client truststore file (for example, /etc/security/clientKeys/all.jks)

9. Restart Ranger Admin and Ranger KMS.

10. Login to Policy Manager UI with keyadmin credentials. Under default KMS Repo configuration, replace **KMS URL** configuration value with the new SSL-enabled KMS URL. For example:

Previous KMS URL = kms://http@internal host name:http_port/kms

New KMS URL = kms://https@internal host name:https_port/kms

11. Now in the Policy Manager UI > Audit > Plugin tab, you should see an entry for your service name with HTTP Response Code = 200.

7.1.5. Install Multiple Ranger KMS

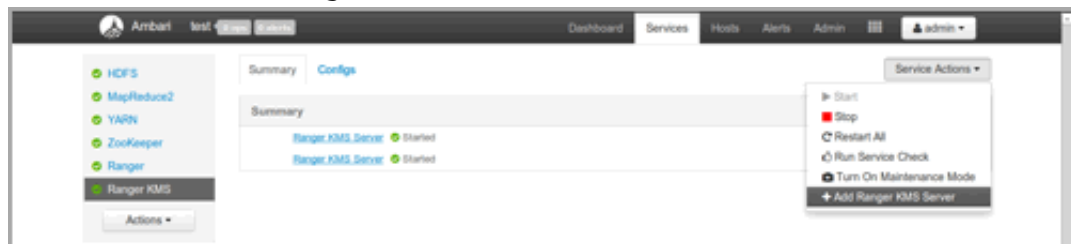
Multiple services can be set up for high availability of Ranger KMS. HDFS interacts with the active process.

Prerequisite: an instance with more than one node.

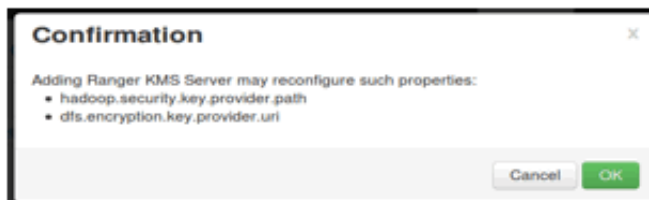
To install Ranger KMS on multiple nodes:

1. First install Ranger KMS on a single node (see [Installing the Ranger Key Management Service](#)).
2. Next, add the Ranger KMS service to another node.

In the Ambari Web UI for the additional node, go to Ranger KMS service # Summary # Service Actions # Add Ranger KMS server.



3. After adding Ranger KMS server, Ambari will show a pop-up message.
4. Press OK. Ambari will modify two HDFS properties, `hadoop.security.key.provider.path` and `dfs.encryption.key.provider.uri`.
5. Restart the HDFS service:



6. For the Ranger KMS service, go to the Advanced kms-site list and change the following property values:

```
hadoop.kms.cache.enable=false
```

```
hadoop.kms.cache.timeout.ms=0
```

```
hadoop.kms.current.key.cache.timeout.ms=0
```

```
hadoop.kms.authentication.signer.secret.provider=zookeeper
```

```
hadoop.kms.authentication.signer.secret.provider.zookeeper.connection.string
node1}:2181,{zookeeper-node2}:2181,{zookeeper-node3}:2181...
```

```
hadoop.kms.authentication.signer.secret.provider.zookeeper.auth.type=none
```

7. From Ambari>Ambari> Ranger KMS> Configs> Advanced> Custom kms-site, add the following property value:

```
hadoop.kms.authentication.zk-dt-secret-manager.enable=true
```

8. Save your configuration changes and restart the Ranger KMS service.

Next, check connectivity from Ranger admin for the newly-added Ranger KMS server:

1. Go to the Ranger UI: `http://<gateway>:6080`
2. Login with your keyadmin user ID and password (the defaults are `keyadmin`, `keyadmin`; these should be changed as soon as possible after installation). The default repository will be added under Ranger KMS service.
3. Under Config properties of the Ranger KMS URL, add the newly added Ranger KMS server FQDN. For example:

Previous Ranger KMS URL = `kms://http<internal host name>:9292/kms`

New Ranger KMS URL = `kms://http<RangerKMS-node1>;<RangerKMS-node2>;...:9292/kms`
4. Run a test connection for the service. You should see a 'connected successfully' message.
5. Choose the Audit > Plugin tab.
6. Check whether plugins are communicating. The UI should display HTTP Response Code = 200 for the respective plugin.

7.1.6. Using the Ranger Key Management Service

Ranger KMS can be accessed at the Ranger admin URL, `http://<hostname>:6080`. Note, however, that the login user for Ranger KMS is different than that for Ranger. Logging on as the Ranger KMS admin user leads to a different set of screens.

Role Separation

By default, Ranger admin uses a different admin user (`keyadmin`) to manage access policies and keys for Ranger KMS.

The person accessing Ranger KMS via the `keyadmin` user should be a different person than the administrator who works with regular Ranger access policies. This approach separates encryption work (encryption keys and policies) from Hadoop cluster management and access policy management.

7.1.6.1. Accessing the Ranger KMS Web UI

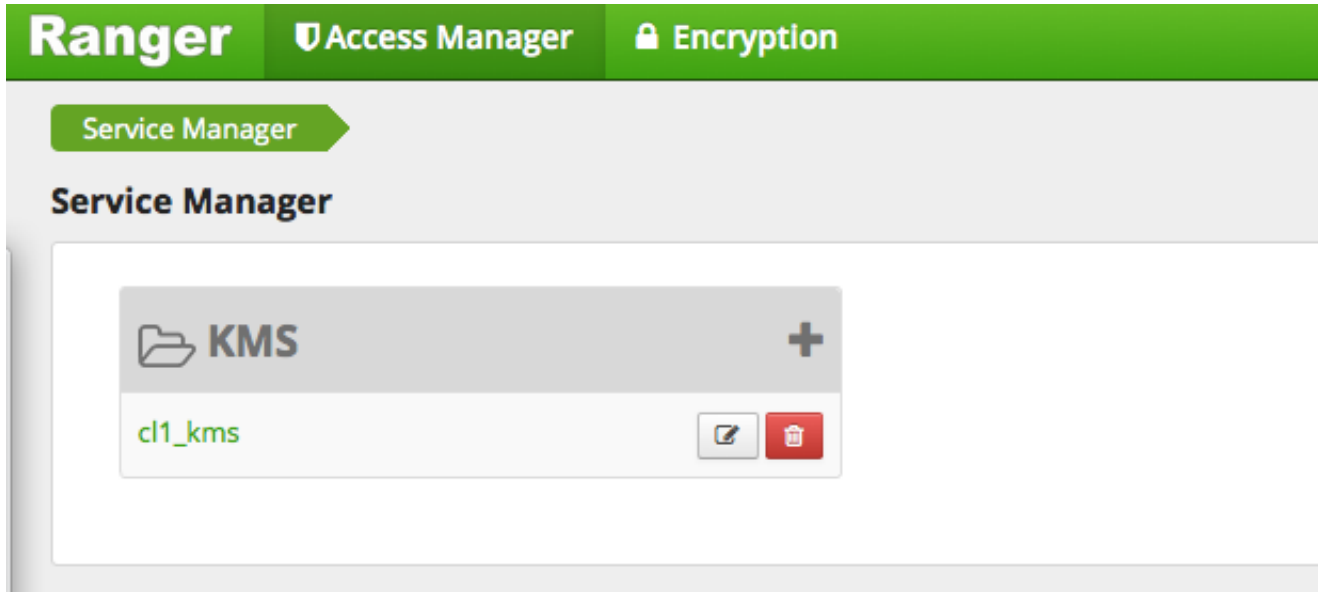
To access Ranger KMS, log in as user `keyadmin`, password `keyadmin`.



Important

Change the password after you log in.

After logging in, you will see the Service Manager screen. To view or edit Ranger KMS repository properties, click on the edit button next to the repository name:



You will see a list of service details and config properties for the repository:

Create Service

Service Details :

Service Name *

cl1_kms

Description

kms repo

Active Status

Enabled Disabled

Config Properties :

KMS URL *

kms://http@vp-os-rh6-my-sec-150

Username *

keyadmin@EXAMPLE.COM

Password *

.....

Add New Configurations

Name



Test Connection

Save

Cancel

Delete

7.1.6.2. Listing and Creating Keys

To list existing keys:

1. Choose the Encryption tab at the top of the Ranger Web UI screen.
2. Select the Ranger KMS service from the drop-down list.

The screenshot shows the Ranger web interface. At the top, there are navigation tabs: 'Ranger', 'Access Manager', and 'Encryption'. Below this, there is a 'KMS' breadcrumb and a 'Key Management' section. A 'Select Service' dropdown menu is open, showing 'cl1_kms' as the selected service. Below the dropdown is a search bar with the text 'Search for your k...'. Below the search bar is a table of keys.

Key Name	Cipher	Version	Attribu
sensitivefolder	AES/CTR/NoPadding	1	key.acl.name → Sensitiv
test	AES/CTR/NoPadding	1	key.acl.name → test
testkeyfromcli	AES/CTR/NoPadding	1	key.acl.name → testkeyf
testkeyfromui	AES/CTR/NoPadding	1	key.acl.name → testkeyf
testkeygmi	AES/CTR/NoPadding	1	key.acl.name → testkeyf
tk1	AES/CTR/NoPadding	1	key.acl.name → tk1

To create a new key:

1. Click on "Add New Key".
2. Add a valid key name.
3. Select the cipher name. Ranger supports AES/CTR/NoPadding as the cipher suite.
4. Specify the key length, 128 or 256 bits.
5. Add other attributes as needed, and then save the key.

Ranger Access Manager Encryption

KMS > cl1_kms > Key Create

Key Detail

Key Name *

Cipher

Length

Description

Attributes

Name	Value
<input type="text"/>	<input type="text"/>

7.1.6.3. Rolling Over an Existing Key

Rolling over (or "rotating") a key retains the same key name, but the key will have a different version. This operation re-encrypts existing file keys, but does not re-encrypt the actual file. Keys can be rolled over at any time.

After a key is rotated in Ranger KMS, new files will have the file key encrypted by the new master key for the encryption zone.

To rotate a key, click the edit button next to the key name in the list of keys, as shown in the following screen shot:

Policy ID	Policy Name	Status
1	cl1_kms-1-20150724233747	Enabled

Edit the key information, and then press Save.

When asked to confirm the rollover, click "OK":

Are you sure want to rollover?

Cancel OK

7.1.6.4. Deleting a Key



Warning

Deleting a key associated with an existing encryption zone will result in data loss.

To delete an existing key:

1. Choose the Encryption tab at the top of the Ranger Web UI screen.
2. Select Ranger KMS service from the drop-down list.
3. Click on the delete symbol next to the key.
4. You will see a confirmation pop-up window; confirm or cancel.

7.1.7. Ranger KMS Properties

This chapter describes configuration properties for the Ranger Key Management Service (KMS).

Table 7.1. Properties in Advanced dbks-site Menu (dbks-site.xml)

Property Name	Default Value	Description
ranger.ks.masterkey.credential.alias	ranger.ks.masterkey.password	Credential alias used for masterkey.
ranger.ks.jpa.jdbc.user	rangerkms	Database username used for operation.
ranger.ks.jpa.jdbc.url	jdbc:log4jdbc:mysql://localhost:3306/rangerkms	JDBC connection URL for database.
ranger.ks.jpa.jdbc.password	_ (default it's encrypted)	Database user's password.
ranger.ks.jpa.jdbc.driver	net.sf.log4jdbc.DriverSpy	Driver used for database.
ranger.ks.jpa.jdbc.dialect	org.eclipse.persistence.platform.database.MySQLPlatform	Dialect used for database.
ranger.ks.jpa.jdbc.credential.provider.path	/etc/ranger/kms/rangerkms.jceks	Credential provider path.
ranger.ks.jpa.jdbc.credential.alias	ranger.ks.jdbc.password	Credential alias used for password.
ranger.ks.jdbc.sqlconnectorjar	/usr/share/java/mysql-connector-java.jar	Driver jar used for database.
ranger.db.encrypt.key.password	_ (Default; it's encrypted)	Password used for encrypting the Master Key.
hadoop.kms.blacklist.DECRYPT_EEK	hdfs	Blacklist for decrypt EncryptedKey CryptoExtension operations. This can have multiple user IDs in a comma separated list. e.g. stormuser , yarn , hdfs.

Table 7.2. Properties in Advanced kms-env

Property Name	Default Value	Description
Kms User	kms	Ranger KMS process will be started using this user.
Kms Group	kms	Ranger KMS process will be started using this group.
LD library path		LD library path (basically used when the db flavor is SQLA). Example: /opt/sqlanywhere17/lib64
kms_port	9292	Port used by Ranger KMS.
kms_log_dir	/var/log/ranger/kms	Directory where the Ranger KMS log will be generated.

Table 7.3. Properties in Advanced kms-properties (install.properties)

Property Name	Default Value	Description
db_user	rangerkms	Database username used for the operation.
db_root_user		Database root username. Default is blank. Specify the root user.
db_root_password		Database root user's password. Default is blank. Specify the root user password.
db_password		Database user's password for the operation. Default is blank. Specify the Ranger KMS database password.
db_name	rangerkms	Database name for Ranger KMS.
db_host	<FQDN of instance where the Ranger KMS is installed>	Hostname where the database is installed. Note: Check the hostname for DB and change it accordingly.
SQL_CONNECTOR_JAR	/usr/share/java/mysql-connector.jar	Location of DB client library.
REPOSITORY_CONFIG_USERNAME	keyadmin	User used in default repo for Ranger KMS.
REPOSITORY_CONFIG_PASSWORD	keyadmin	Password for user used in default repo for Ranger KMS.
KMS_MASTER_KEY_PASSWD		Password used for encrypting the Master Key. Default value is blank. Set the master key to any string.
DB_FLAVOR	MYSQL	Database flavor used for Ranger KMS. Supported values: MYSQL, SQLA, ORACLE, POSTGRES, MSSQL

Table 7.4. Properties in Advanced kms-site (kms-site.xml)

Property Name	Default Value	Description
hadoop.security.keystore. JavaKeyStoreProvider.password	none	If using the JavaKeyStoreProvide, the password for the keystore file.
hadoop.kms.security. authorization.manager	org.apache.ranger. authorization.kms. authorizer.RangerKmsAuthorizer	Ranger KMS security authorizer.
hadoop.kms.key.provider.uri	dbks://http@localhost:9292/kms	URI of the backing KeyProvider for the KMS.
hadoop.kms.current.key. cache.timeout.ms	30000	Expiry time for the KMS current key cache, in milliseconds. This affects getCurrentKey operations.
hadoop.kms.cache.timeout.ms	600000	Expiry time for the KMS key version and key metadata cache, in milliseconds. This affects getKeyVersion and getMetadata.
hadoop.kms.cache.enable	true	Whether the KMS will act as a cache for the backing KeyProvider. When the cache is enabled, operations like getKeyVersion, getMetadata, and getCurrentKey will sometimes return cached data without consulting the backing KeyProvider. Cached values are flushed when keys are deleted or modified. Note: This setting is beneficial if Single KMS and single mode are used. If this

Property Name	Default Value	Description
		is set to true when multiple KMSs are used, or when the key operations are from different modes (Ranger UI, CURL, or <code>hadoop</code> command), it might cause inconsistency.
<code>hadoop.kms.authentication.type</code>	<code>simple</code>	Authentication type for the Ranger KMS. Can be either "simple" or "kerberos".
<code>hadoop.kms.authentication.signer.secret.provider.zookeeper.path</code>	<code>/hadoop-kms/hadoop-auth-signature-secret</code>	The ZooKeeper ZNode path where the Ranger KMS instances will store and retrieve the secret from.
<code>hadoop.kms.authentication.signer.secret.provider.zookeeper.kerberos.principal</code>	<code>kms/#HOSTNAME#</code>	The Kerberos service principal used to connect to ZooKeeper
<code>hadoop.kms.authentication.signer.secret.provider.zookeeper.kerberos.keytab</code>	<code>/etc/hadoop/conf/kms.keytab</code>	The absolute path for the Kerberos keytab with the credentials to connect to ZooKeeper.
<code>hadoop.kms.authentication.signer.secret.provider.zookeeper.connection.string</code>	<code>#HOSTNAME#:#PORT#,...</code>	The ZooKeeper connection string, a list of hostnames and port comma separated. For example: <FQDN for first instance>:2181,<FQDN for second instance>:2181
<code>hadoop.kms.authentication.signer.secret.provider.zookeeper.auth.type</code>	<code>kerberos</code>	ZooKeeper authentication type: 'none' or 'sasl' (Kerberos) The value "none" means the default authentication will be used (not, "no authentication is used.") Using "none" means: <ul style="list-style-type: none"> In a non-kerberized unsecure cluster: no security on the ZooKeeper connection In a kerberized secure cluster: Kerberos/SASL is used to connect to ZooKeeper with the Ranger KMS's "_Client_" Login Context The value "sasl" means that KMS would use an independent "ZKSignerSecretProviderClient" Login Context, but this is not yet supported on HDP.
<code>hadoop.kms.authentication.signer.secret.provider</code>	<code>random</code>	Indicates how the secret to sign authentication cookies will be stored. Options are 'random' (default), 'string', and 'zookeeper'. If you have multiple Ranger KMS instances, specify 'zookeeper'.
<code>hadoop.kms.authentication.kerberos.principal</code>	<code>HTTP/localhost</code>	The Kerberos principal to use for the HTTP endpoint. The principal must start with 'HTTP/' as per the Kerberos HTTP SPNEGO specification.
<code>hadoop.kms.authentication.kerberos.name.rules</code>	<code>DEFAULT</code>	Rules used to resolve Kerberos principal names.
<code>hadoop.kms.authentication.kerberos.keytab</code>	<code>\${user.home}/kms.keytab</code>	Path to the keytab with credentials for the configured Kerberos principal.

Property Name	Default Value	Description
hadoop.kms.audit.aggregation.window.ms	10000	Specified in ms. Duplicate audit log events within this aggregation window are quashed to reduce log traffic. A single message for aggregated events is printed at the end of the window, along with a count of the number of aggregated events.

Table 7.5. Properties in Advanced ranger-kms-audit (ranger-kms-audit.xml)

Property Name	Default Value	Description
Audit provider summary enabled		Enable audit provider summary.
xasecure.audit.is.enabled	true	Enable audit.
xasecure.audit.destination.solr.zookeepers	none	Specify solr zookeeper string.
xasecure.audit.destination.solr.urls	{{ranger_audit_solr_urls}}	Specify solr URL. Note: In Ambari this value is populated from the Ranger Admin by default.
xasecure.audit.destination.solr.batch.filespool.dir	/var/log/ranger/kms/audit/solr/spool	Directory for solr audit spool.
Audit to SOLR		Enable audit to solr.
xasecure.audit.destination.hdfs.dir	hdfs://NAMENODE_HOST:8020/ranger/audit	HDFS directory to write audit. Note: Make sure the service user has required permissions.
xasecure.audit.destination.hdfs.batch.filespool.dir	/var/log/ranger/kms/audit/hdfs/spool	Directory for HDFS audit spool.
Audit to HDFS		Enable hdfs audit.
xasecure.audit.destination.db.user	{{xa_audit_db_user}}	xa audit db user Note: In Ambari this value is populated from the Ranger Admin by default.
xasecure.audit.destination.db.password	encrypted (it's in encrypted format)	xa audit db user password Note: In Ambari this value is populated from the Ranger Admin by default.
xasecure.audit.destination.db.jdbc.url	{{audit_jdbc_url}}	Database JDBC URL for xa audit. Note: In Ambari the value for this is populated from the Ranger Admin by default.
xasecure.audit.destination.db.jdbc.driver	{{jdbc_driver}}	Database JDBC driver. Note: In Ambari this value is populated from the Ranger Admin by default.
xasecure.audit.destination.db.batch.filespool.dir	/var/log/ranger/kms/audit/db/spool	Directory for database audit spool.
Audit to DB		Enable audit to database.
xasecure.audit.credential.provider.file	jceks://file{{credential_file}}	Credential provider file.

Table 7.6. Properties in Advanced ranger-kms-policymgr-ssl

Property Name	Default Value	Description
xasecure.policymgr.clientssl.truststore.password	changeit	Password for the truststore.
xasecure.policymgr.clientssl.truststore	/usr/hdp/current/ranger-kms/conf/ranger-plugin-truststore.jks	jks file for truststore
xasecure.policymgr.clientssl.keystore.password	myKeyFilePassword	Password for keystore.
xasecure.policymgr.clientssl.keystore.credential.file	jceks://file{{credential_file}}	Java keystore credential file.
xasecure.policymgr.clientssl.keystore	/usr/hdp/current/ranger-kms/conf/ranger-plugin-keystore.jks	Java keystore file.
xasecure.policymgr.clientssl.truststore.credential.file	jceks://file{{credential_file}}	Java truststore file.

Table 7.7. Properties in Advanced ranger-kms-security

Property Name	Default Value	Description
ranger.plugin.kms.service.name	<default name for Ranger KMS Repo>	Name of the Ranger service containing policies for the KMS instance. Note: In Ambari the default value is <clusterName>_kms.
ranger.plugin.kms.policy.source.impl	org.apache.ranger.admin.client.RangerAdminRESTClient	Class to retrieve policies from the source.
ranger.plugin.kms.policy.rest.url	{{policymgr_mgr_url}}	URL for Ranger Admin.
ranger.plugin.kms.policy.rest.ssl.config.file	/etc/ranger/kms/conf/ranger-policymgr-ssl.xml	Path to the file containing SSL details for contacting the Ranger Admin.
ranger.plugin.kms.policy.pollIntervalMs	30000	Time interval to poll for changes in policies.
ranger.plugin.kms.policy.cache.dir	/etc/ranger/{{repo_name}}/policycache	Directory where Ranger policies are cached after successful retrieval from the source.

7.1.8. Troubleshooting Ranger KMS

Table 7.8. Troubleshooting Suggestions

Issue	Action
Not able to install Ranger KMS	Check to see if ranger admin is running, verify DB.
Not able to start Ranger KMS	Check the Ranger KMS log. If there is a message about illegal key size, make sure unlimited strength JCE is available.
Hadoop key commands fail	Make sure Ranger KMS client properties are updated in hdfs config.
Not able to create keys from Ranger UI	Make sure that the keyadmin user (or any custom user) configured in the KMS repository is added to proxy properties in the custom kms-site.xml file.

7.2. HDFS "Data at Rest" Encryption

Encryption is a form of data security that is required in industries such as healthcare and the payment card industry. Hadoop provides several ways to encrypt stored data.

- The lowest level of encryption is volume encryption, which protects data after physical theft or accidental loss of a disk volume. The entire volume is encrypted; this approach does not support finer-grained encryption of specific files or directories. In addition, volume encryption does not protect against viruses or other attacks that occur while a system is running.
- Application level encryption (encryption within an application running on top of Hadoop) supports a higher level of granularity and prevents "rogue admin" access, but adds a layer of complexity to the application architecture.
- A third approach, HDFS data at rest encryption, encrypts selected files and directories stored ("at rest") in HDFS. This approach uses specially designated HDFS directories known as "encryption zones."

This chapter focuses on the third approach, HDFS data at rest encryption. The chapter is intended as an introductory quick start to HDFS data at rest encryption. Content will be updated regularly.

7.2.1. HDFS Encryption Overview

HDFS data at rest encryption implements end-to-end encryption of data read from and written to HDFS. End-to-end encryption means that data is encrypted and decrypted only by the client. HDFS does not have access to unencrypted data or keys.

HDFS encryption involves several elements:

- **Encryption key:** A new level of permission-based access protection, in addition to standard HDFS permissions.
- **HDFS encryption zone:** A special HDFS directory within which all data is encrypted upon write, and decrypted upon read.
 - Each encryption zone is associated with an encryption key that is specified when the zone is created.
 - Each file within an encryption zone has a unique encryption key, called the "data encryption key" (DEK).
 - HDFS does not have access to DEKs. HDFS DataNodes only see a stream of encrypted bytes. HDFS stores "encrypted data encryption keys" (EDEKs) as part of the file's metadata on the NameNode.
 - Clients decrypt an EDEK and use the associated DEK to encrypt and decrypt data during write and read operations.
- **Ranger Key Management Service (Ranger KMS):** An open source key management service based on Hadoop's `KeyProvider` API.

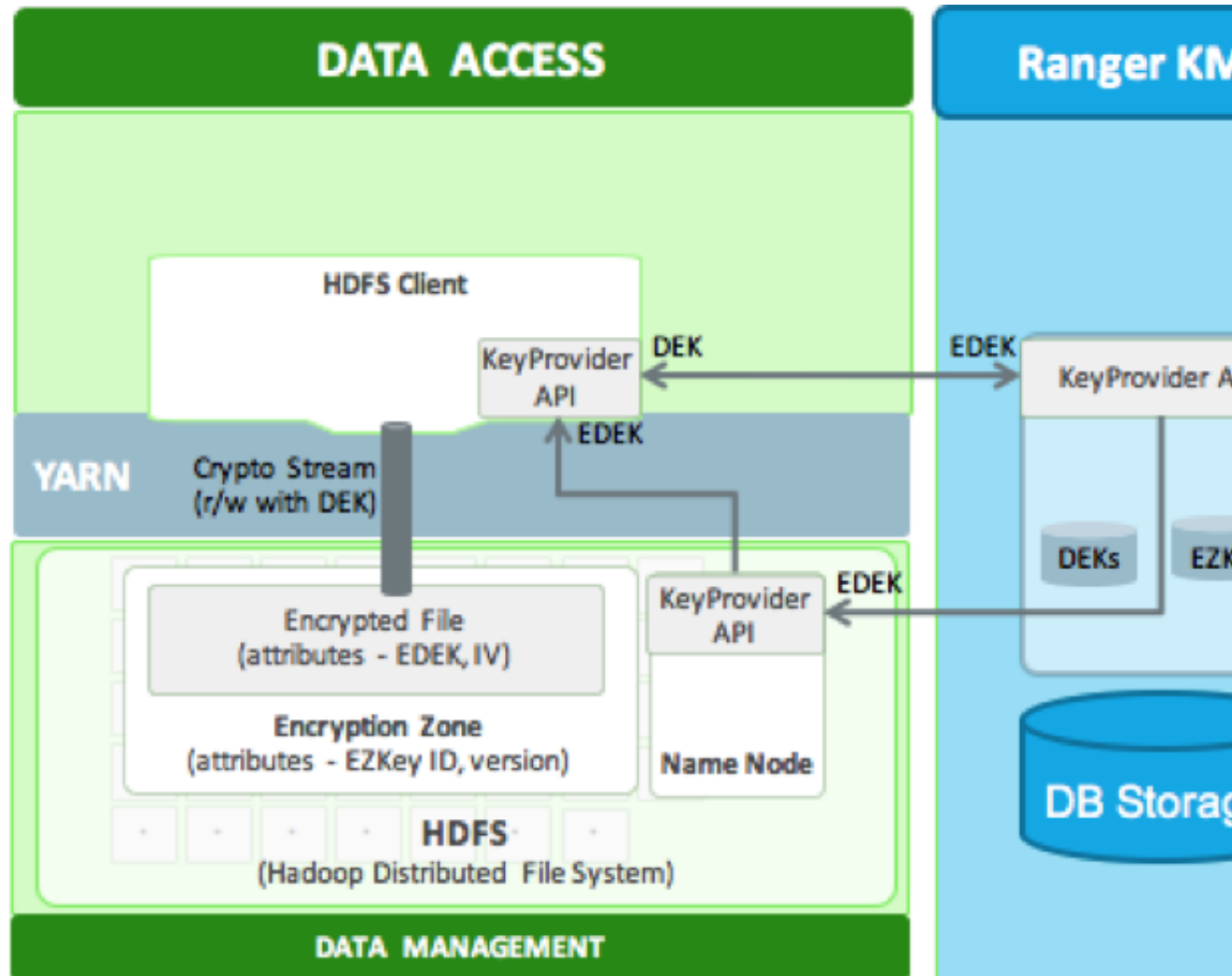
For HDFS encryption, the Ranger KMS has three basic responsibilities:

- Provide access to stored encryption zone keys.
- Generate and manage encryption zone keys, and create encrypted data keys to be stored in Hadoop.

- Audit all access events in Ranger KMS.

Note: This chapter is intended for security administrators who are interested in configuring and using HDFS encryption. For more information about Ranger KMS, see the [Ranger KMS Administration Guide](#).

Figure 7.1. HDFS Encryption Components



Role Separation

Access to the key encryption/decryption process is typically restricted to end users. This means that encrypted keys can be safely stored and handled by HDFS, because the HDFS admin user does not have access to them.

This role separation requires two types of HDFS administrator accounts:

- HDFS service user: the system-level account associated with HDFS (`hdfs` by default).
- HDFS admin user: an account in the `hdfs` supergroup, which is used by HDFS administrators to configure and manage HDFS.



Important

For clear segregation of duties, we recommend that you restrict use of the `hdfs` account to system/interprocess use. Do not provide its password to physical users. A (human) user who administers HDFS should only access HDFS through an admin user account created specifically for that purpose. For more information about creating an HDFS admin user, see [Creating an HDFS Admin User](#).

Other services may require a separate admin account for clusters with HDFS encryption zones. For service-specific information, see [Configuring HDP Services for HDFS Encryption](#).

7.2.2. Configuring and Starting the Ranger Key Management Service (Ranger KMS)

In a typical environment, a security administrator will set up the Ranger Key Management Service. For information about installing and configuring the Ranger KMS, see the [Ranger KMS Administration \[532\]](#).

7.2.3. Configuring and Using HDFS Data at Rest Encryption

After the Ranger KMS has been set up and the NameNode and HDFS clients have been configured, an HDFS administrator can use the `hadoop key` and `hdfs crypto` command-line tools to create encryption keys and set up new encryption zones.

The overall workflow is as follows:

1. Create an HDFS encryption zone key that will be used to encrypt the file-level data encryption key for every file in the encryption zone. This key is stored and managed by Ranger KMS.
2. Create a new HDFS folder. Specify required permissions, owner, and group for the folder.
3. Using the new encryption zone key, designate the folder as an encryption zone.
4. Configure client access. The user associated with the client application needs sufficient permission to access encrypted data. In an encryption zone, the user needs file/directory access (through Posix permissions or Ranger access control), as well as access for certain key operations. To set up ACLs for key-related operations, see the [Ranger KMS Administration Guide](#).

After permissions are set, Java API clients and HDFS applications with sufficient HDFS and Ranger KMS access privileges can write and read to/from files in the encryption zone.



Important

You should create a separate HDFS Admin user account for HDFS Data at Rest Encryption.

7.2.3.1. Prepare the Environment

HDP supports hardware acceleration with Advanced Encryption Standard New Instructions (AES-NI). Compared with the software implementation of AES, hardware acceleration offers an order of magnitude faster encryption/decryption.

To use AES-NI optimization you need CPU and library support, described in the following subsections.

7.2.3.1.1. CPU Support for AES-NI optimization

AES-NI optimization requires an extended CPU instruction set for AES hardware acceleration.

There are several ways to check for this; for example:

```
$ cat /proc/cpuinfo | grep aes
```

Look for output with flags and 'aes'.

7.2.3.1.2. Library Support for AES-NI optimization

You will need a version of the `libcrypto.so` library that supports hardware acceleration, such as OpenSSL 1.0.1e. (Many OS versions have an older version of the library that does not support AES-NI.)

A version of the `libcrypto.so` library with AES-NI support must be installed on HDFS cluster nodes and MapReduce client hosts – that is, any host from which you issue HDFS or MapReduce requests. The following instructions describe how to install and configure the `libcrypto.so` library.

RHEL/CentOS 6.5 or later

On HDP cluster nodes, the installed version of `libcrypto.so` supports AES-NI, but you will need to make sure that the symbolic link exists:

```
$ sudo ln -s /usr/lib64/libcrypto.so.1.0.1e /usr/lib64/  
libcrypto.so
```

On MapReduce client hosts, install the `openssl-devel` package:

```
$ sudo yum install openssl-devel
```

7.2.3.1.3. Verifying AES-NI Support

To verify that a client host is ready to use the AES-NI instruction set optimization for HDFS encryption, use the following command:

```
hadoop checknative
```

You should see a response similar to the following:


```
15/08/12 13:48:39 INFO bzip2.Bzip2Factory: Successfully loaded & initialized
native-bzip2 library system-native
14/12/12 13:48:39 INFO zlib.ZlibFactory: Successfully loaded & initialized
native-zlib library
Native library checking:
hadoop: true /usr/lib/hadoop/lib/native/libhadoop.so.1.0.0
zlib:   true /lib64/libz.so.1
snappy: true /usr/lib64/libsnappy.so.1
lz4:   true revision:99
bzip2: true /lib64/libbz2.so.1
openssl: true /usr/lib64/libcrypto.so
```

If you see `true` in the `openssl` row, Hadoop has detected the right version of `libcrypto.so` and optimization will work.

If you see `false` in this row, you do not have the correct version.

7.2.3.2. Create an Encryption Key

Create a "master" encryption key for the new encryption zone. Each key will be specific to an encryption zone.

Ranger supports AES/CTR/NoPadding as the cipher suite. (The associated property is listed under HDFS -> Configs in the Advanced `hdfs-site` list.)

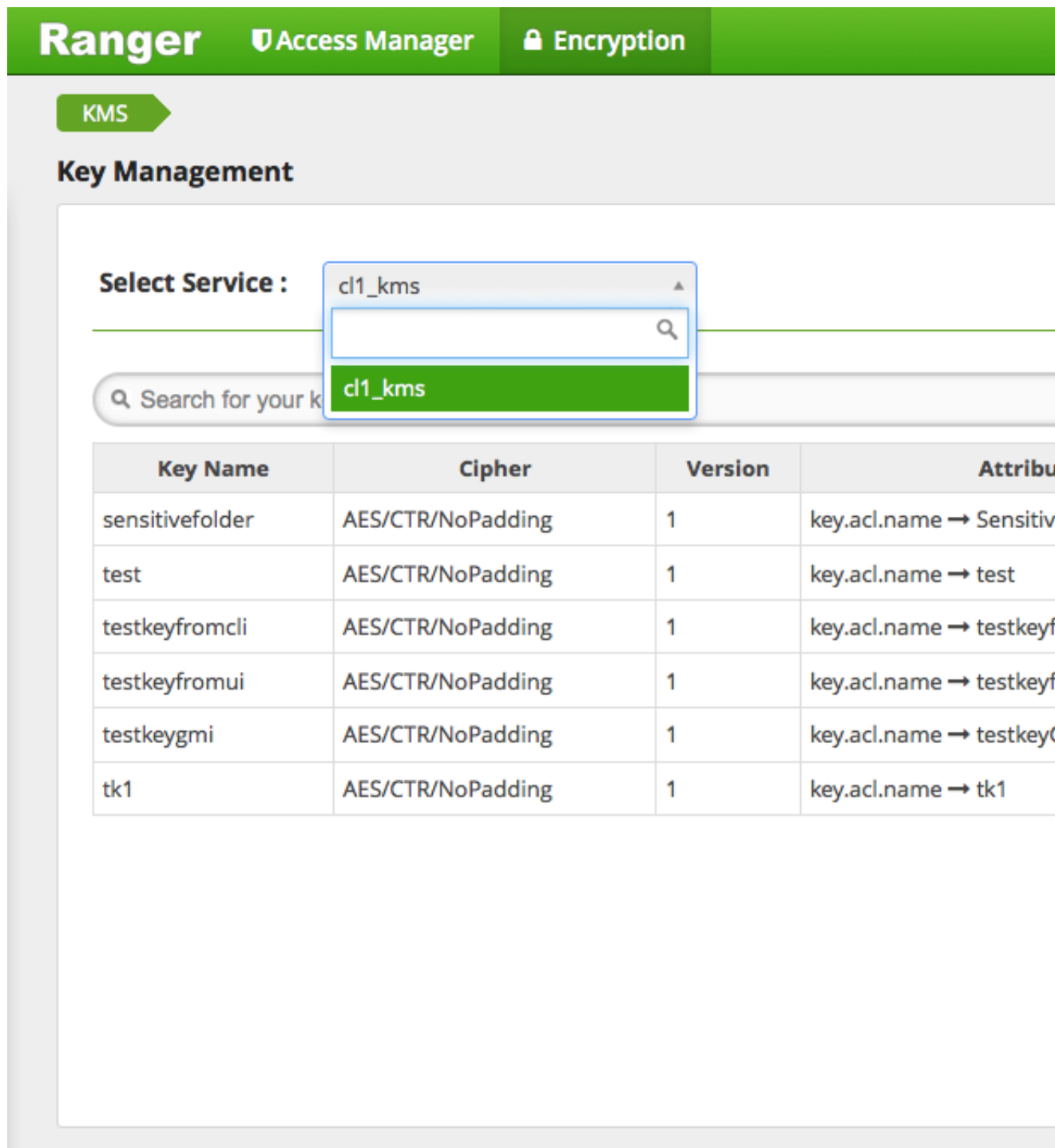
Key size can be 128 or 256 bits.

Recommendation: create a new superuser for key management. In the following examples, superuser `encr` creates the key. This separates the data access role from the encryption role, strengthening security.

Create an Encryption Key using Ranger KMS (Recommended)

In the Ranger Web UI screen:

1. Choose the Encryption tab at the top of the screen.
2. Select the KMS service from the drop-down list.



Select Service : cl1_kms

Search for your k

Key Name	Cipher	Version	Attribu
sensitivefolder	AES/CTR/NoPadding	1	key.acl.name → Sensitive
test	AES/CTR/NoPadding	1	key.acl.name → test
testkeyfromcli	AES/CTR/NoPadding	1	key.acl.name → testkeyf
testkeyfromui	AES/CTR/NoPadding	1	key.acl.name → testkeyf
testkeygmi	AES/CTR/NoPadding	1	key.acl.name → testkeyf
tk1	AES/CTR/NoPadding	1	key.acl.name → tk1

To create a new key:

1. Click on "Add New Key":
2. Add a valid key name.
3. Select the cipher name. Ranger supports AES/CTR/NoPadding as the cipher suite.
4. Specify the key length, 128 or 256 bits.

5. Add other attributes as needed, and then save the key.

Ranger

Access Manager

Encryption

KMS > cl1_kms > Key Create

Key Detail

Key Name *

Cipher

Length

Description

Attributes	Name	Value
	<input type="text"/>	<input type="text"/>

For information about rolling over and deleting keys, see [Using the Ranger Key Management Service](#).



Warning

Do not delete an encryption key while it is in use for an encryption zone. This will result in loss of access to data in that zone.

Create an Encryption Key using the CLI

The full syntax of the `hadoop key create` command is as follows:

```
[create <keyname> [-cipher <cipher>]
[-size <size>]
[-description <description>]
[-attr <attribute=value>]
[-provider <provider>]
[-help]]
```

Example:

```
# su - encr

# hadoop key create <key_name> [-size <number-of-bits>]
```

The default key size is 128 bits. The optional `-size` parameter supports 256-bit keys, and requires the Java Cryptography Extension (JCE) Unlimited Strength Jurisdiction Policy File on all hosts in the cluster. For installation information, see [Installing the JCE](#).

Example:

```
# su - encr

# hadoop key create key1
```

To verify creation of the key, list the metadata associated with the current user:

```
# hadoop key list -metadata
```

For information about rolling over and deleting keys, see [Using the Ranger Key Management Service](#).



Warning

Do not delete an encryption key while it is in use for an encryption zone. This will result in loss of access to data in that zone.

7.2.3.3. Create an Encryption Zone

Each encryption zone must be defined using an empty directory and an existing encryption key. An encryption zone cannot be created on top of a directory that already contains data.

Recommendation: use one unique key for each encryption zone.

Use the `crypto createZone` command to create a new encryption zone. The syntax is:

```
-createZone -keyName <keyName> -path <path>
```

where:

- `-keyName`: specifies the name of the key to use for the encryption zone.
- `-path` specifies the path of the encryption zone to be created. It must be an empty directory.



Note

The `hdfs` service account can create zones, but cannot write data unless the account has sufficient permission.

Recommendation: Define a separate user account for the HDFS administrator, and do not provide access to keys for this user in Ranger KMS.

Steps:

1. As HDFS administrator, create a new empty directory. For example:

```
# hdfs dfs -mkdir /zone_encr
```

2. Using the encryption key, make the directory an encryption zone. For example:

```
# hdfs crypto -createZone -keyName key1 -path /zone_encr
```

When finished, the NameNode will recognize the folder as an HDFS encryption zone.

3. To verify creation of the new encryption zone, run the `crypto -listZones` command as an HDFS administrator:

```
-listZones
```

You should see the encryption zone and its key. For example:

```
$ hdfs crypto -listZones
/zone-encr key1
```



Note

The following property (in the `hdfs-default.xml` file) causes `listZone` requests to be batched. This improves NameNode performance. The property specifies the maximum number of zones that will be returned in a batch.

```
dfs.namenode.list.encryption.zones.num.responses
```

The default is 100.

To remove an encryption zone, delete the root directory of the zone. For example:

```
hdfs dfs -rm -R /zone_encr
```

7.2.3.4. Copy Files from/to an Encryption Zone

To copy existing files into an encryption zone, use a tool like `distcp`.

Note: for separation of administrative roles, do not use the `hdfs` user to create encryption zones. Instead, designate another administrative account for creating encryption keys and zones. See [Appendix: Creating an HDFS Admin User \[590\]](#) for more information.

The files will be encrypted using a file-level key generated by the Ranger Key Management Service.

DistCp Considerations

`DistCp` is commonly used to replicate data between clusters for backup and disaster recovery purposes. This operation is typically performed by the cluster administrator, via an HDFS superuser account.

To retain this workflow when using HDFS encryption, a new virtual path prefix has been introduced, `/.reserved/raw/`. This virtual path gives super users direct access to the underlying encrypted block data in the file system, allowing super users to `distcp` data without requiring access to encryption keys. This also avoids the overhead of decrypting and re-encrypting data. The source and destination data will be byte-for-byte identical, which would not be true if the data were re-encrypted with a new EDEK.



Warning

When using `/.reserved/raw/` to `distcp` encrypted data, make sure you preserve extended attributes with the `-px` flag. This is necessary because encrypted attributes such as the EDEK are exposed through extended attributes; they *must* be preserved to be able to decrypt the file. For example:

```
sudo -u encr hadoop distcp -px hdfs:/cluster1-  
namenode:50070/.reserved/raw/apps/enczone hdfs:/cluster2-  
namenode:50070/.reserved/raw/apps/enczone
```

This means that if the `distcp` operation is initiated at or above the encryption zone root, it will automatically create a new encryption zone at the destination (if one does not already exist).

Recommendation: To avoid potential mishaps, first create identical encryption zones on the destination cluster.

Copying between encrypted and unencrypted locations

By default, `distcp` compares file system checksums to verify that data was successfully copied to the destination.

When copying between an unencrypted and encrypted location, file system checksums will not match because the underlying block data is different. In this case, specify the `-skipcrccheck` and `-update` flags to avoid verifying checksums.

7.2.3.5. Read and Write Files from/to an Encryption Zone

Clients and HDFS applications with sufficient HDFS and Ranger KMS permissions can read and write files from/to an encryption zone.

Overview of the client write process:

1. The client writes to the encryption zone.
2. The NameNode checks to make sure that the client has sufficient write access permissions. If so, the NameNode asks Ranger KMS to create a file-level key, encrypted with the encryption zone master key.
3. The Namenode stores the file-level encrypted data encryption key (EDEK) generated by Ranger KMS as part of the file's metadata, and returns the EDEK to the client.

4. The client asks Ranger KMS to decode the EDEK (to DEK), and uses the DEK to write encrypted data. Ranger KMS checks for permissions for the user before decrypting EDEK and producing the DEK for the client.

Overview of the client read process:

1. The client issues a read request for a file in an encryption zone.
2. The NameNode checks to make sure that the client has sufficient read access permissions. If so, the NameNode returns the file's EDEK and the encryption zone key version that was used to encrypt the EDEK.
3. The client asks Ranger KMS to decrypt the EDEK. Ranger KMS checks for permissions to decrypt EDEK for the end user.
4. Ranger KMS decrypts and returns the (unencrypted) data encryption key (DEK).
5. The client uses the DEK to decrypt and read the file.

The preceding steps take place through internal interactions between the DFSClient, the NameNode, and Ranger KMS.

In the following example, the `/zone_encr` directory is an encrypted zone in HDFS.

To verify this, use the `crypto -listZones` command (as an HDFS administrator). This command lists the root path and the zone key for the encryption zone. For example:

```
# hdfs crypto -listZones
/zone_encr key1
```

Additionally, the `/zone_encr` directory has been set up for read/write access by the `hive` user:

```
# hdfs dfs -ls /
...
drwxr-x--- - hive hive          0 2015-01-11 23:12 /zone_encr
```

The `hive` user can, therefore, write data to the directory.

The following examples use the `copyFromLocal` command to move a local file into HDFS.

```
[hive@blue ~]# hdfs dfs -copyFromLocal web.log /zone_encr
[hive@blue ~]# hdfs dfs -ls /zone_encr
Found 1 items
-rw-r--r--  1 hive hive          1310 2015-01-11 23:28 /zone_encr/web.log
```

The `hive` user can read data from the directory, and can verify that the file loaded into HDFS is readable in its unencrypted form.

```
[hive@blue ~]# hdfs dfs -copyToLocal /zone_encr/web.log read.log
[hive@blue ~]# diff web.log read.log
```



Note

For more information about accessing encrypted files from Hive and other components, see [Configuring HDP Services for HDFS Encryption](#).

Users without access to KMS keys will be able to see file names (via the `-ls` command), but they will not be able to write data or read from the encrypted zone. For example, the `hdfs` user lacks sufficient permissions, and cannot access the data in `/zone_encr`:

```
[hdfs@blue ~]# hdfs dfs -copyFromLocal install.log /zone_encr
copyFromLocal: Permission denied: user=hdfs, access=EXECUTE, inode="/
zone_encr":hive:hive:drwxr-x---
```

```
[hdfs@blue ~]# hdfs dfs -copyToLocal /zone_encr/web.log read.log
copyToLocal: Permission denied: user=hdfs, access=EXECUTE, inode="/
zone_encr":hive:hive:drwxr-x---
```

7.2.3.6. Delete Files from an Encryption Zone with Trash Enabled

The trash location for encrypted HDFS files is different than the default trash location for unencrypted files (`/user/$USER/.Trash/Current/OriginalPathToDeletedFile`).

When trash is enabled and an encrypted file is deleted, the file is moved to the `.Trash` subdirectory under the root of the encryption zone as `/EncryptionZoneRoot/.Trash/$USER/Current/OriginalPathToDeletedFile`. The file remains encrypted without additional decryption/re-encryption overhead during the move to trash. The move operation preserves the name of the user who executes the deletion, and the full path of the deleted file.

For example, if user `hdp-admin` deletes file `/zone_name/file1` using the following command:

```
hdfs dfs -rm /zone_name/file1
```

`file1` will remain encrypted, and it will be moved to the following location within the encryption zone:

```
/zone_name/.Trash/hdp-admin/Current/zone_name/file1
```

A trash checkpoint will be created for the `.Trash` subdirectory in each encryption zone. Checkpoints will be deleted/created according to the value of `fs.trash.checkpoint.interval` (number of minutes between trash checkpoints). A checkpoint for this example would be:

```
/zone_name/.Trash/hdp-admin/<CheckPointTimeStamp>/zone_name/
file1
```

For additional information, see Apache [HDFS-8831](#).

7.2.4. Configuring HDP Services for HDFS Encryption

HDFS data at rest encryption is supported on the following HDP components:



Important

You should create a separate Admin user account for HDFS Data at Rest Encryption for each of the following supported components.

- HBase

- Hive
- Hive on Tez
- MapReduce
- Oozie
- Spark
- Sqoop
- Storm
- WebHDFS
- YARN

HDFS data at rest encryption is not supported on the following components:

- Accumulo
- Falcon
- HDP Search

The remainder of this section describes scenarios and access considerations for accessing HDFS-encrypted files from supporting HDP components.

7.2.4.1. HBase

HBase stores all of its data under its root directory in HDFS, configured with `hbase.rootdir`. The only other directory that the HBase service will read or write is `hbase.bulkload.staging.dir`.

On HDP clusters, `hbase.rootdir` is typically configured as `/apps/hbase/data`, and `hbase.bulkload.staging.dir` is configured as `/apps/hbase/staging`. HBase data, including the root directory and staging directory, can reside in an encryption zone on HDFS.

The HBase service user needs to be granted access to the encryption key in the Ranger KMS, because it performs tasks that require access to HBase data (unlike Hive or HDFS).

By design, HDFS-encrypted files cannot be bulk-loaded from one encryption zone into another encryption zone, or from an encryption zone into an unencrypted directory. Encrypted files can only be copied. An attempt to load data from one encryption zone into another will result in a copy operation. Within an encryption zone, files can be copied, moved, bulk-loaded, and renamed.

7.2.4.1.1. Recommendations

- Make the parent directory for the HBase root directory and bulk load staging directory an encryption zone, instead of just the HBase root directory. This is because HBase bulk load operations need to move files from the staging directory into the root directory.
- In typical deployments, `/apps/hbase` can be made an encryption zone.

- Do not create encryption zones as subdirectories under `/apps/hbase`, because HBase may need to rename files across those subdirectories.
- The landing zone for unencrypted data should always be within the destination encryption zone.

7.2.4.1.2. Steps

On a cluster without HBase currently installed:

1. Create the `/apps/hbase` directory, and make it an encryption zone.
2. Configure `hbase.rootdir=/apps/hbase/data`.
3. Configure `hbase.bulkload.staging.dir=/apps/hbase/staging`.

On a cluster with HBase already installed, perform the following steps:

1. Stop the HBase service.
2. Rename the `/apps/hbase` directory to `/apps/hbase-tmp`.
3. Create an empty `/apps/hbase` directory, and make it an encryption zone.
4. `DistCp -skipcrccheck -update` all data from `/apps/hbase-tmp` to `/apps/hbase`, preserving user-group permissions and extended attributes.
5. Start the HBase service and verify that it is working as expected.
6. Remove the `/apps/hbase-tmp` directory.

7.2.4.1.3. Changes in Behavior after HDFS Encryption is Enabled

The HBase bulk load process is a MapReduce job that typically runs under the user who owns the source data. HBase data files created as a result of the job are then bulk loaded in to HBase RegionServers. During this process, HBase RegionServers move the bulk-loaded files from the user's directory and move (rename) the files into the HBase root directory (`/apps/hbase/data`). When data at rest encryption is used, HDFS cannot do a rename across encryption zones with different keys.

Workaround: run the MapReduce job as the `hbase` user, and specify an output directory that resides in the same encryption zone as the HBase root directory.

7.2.4.2. Hive

Recommendation: Store Hive data in an HDFS path called `/apps/hive`.

7.2.4.2.1. Configuring Hive Tables for HDFS Encryption

Before enabling encryption zones, decide whether to store your Hive tables across one zone or multiple encryption zones.

Single Encryption Zone

To configure a single encryption zone for your entire Hive warehouse:

1. Rename `/apps/hive` to `/apps/hive-old`
2. Create an encryption zone at `/apps/hive`
3. `distcp` all of the data from `/apps/hive-old` to `/apps/hive`.

To configure the Hive scratch directory (`hive.exec.scratchdir`) so that it resides inside the encryption zone:

1. Set the directory to `/apps/hive/tmp`.
2. Make sure that the permissions for `/apps/hive/tmp` are set to `1777`.

Multiple Encryption Zones

To access encrypted databases and tables with different encryption keys, configure multiple encryption zones.

For example, to configure two encrypted tables, `ez1.db` and `ez2.db`, in two different encryption zones:

1. Create two new encryption zones, `/apps/hive/warehouse/ez1.db` and `/apps/hive/warehouse/ez2.db`.
2. Load data into Hive tables `ez1.db` and `ez2.db` as usual, using `LOAD` statements. (For additional considerations, see "Loading Data into an Encrypted Table.")

7.2.4.2.2. Loading Data into an Encrypted Table

By design, HDFS-encrypted files cannot be moved or loaded from one encryption zone into another encryption zone, or from an encryption zone into an unencrypted directory. Encrypted files can only be copied.

Within an encryption zone, files can be copied, moved, loaded, and renamed.

Recommendations:

- When loading unencrypted data into encrypted tables (e.g., `LOAD DATA INPATH`), we recommend placing the source data (to be encrypted) into a landing zone within the destination encryption zone.
- An attempt to load data from one encryption zone into another will result in a copy operation. `Distcp` will be used to speed up the process if the size of the files being copied is higher than the value specified by the `hive.exec.copyfile.maxsize` property. The default limit is 32 MB.

Here are two approaches for loading unencrypted data into an encrypted table:

- To load unencrypted data into an encrypted table, use the `LOAD DATA ...` statement.

If the source data does not reside inside the encryption zone, the `LOAD` statement will result in a copy. If your data is already inside HDFS, though, you can use `distcp` to speed up the copying process.

- If the data is already inside a Hive table, create a new table with a `LOCATION` inside an encryption zone, as follows:

```
CREATE TABLE encrypted_table [STORED AS] LOCATION ... AS SELECT *
FROM <unencrypted_table>
```



Note

The location specified in the `CREATE TABLE` statement must be within an encryption zone. If you create a table that points `LOCATION` to an unencrypted directory, your data will not be encrypted. You must copy your data to an encryption zone, and then point `LOCATION` to that encryption zone.

If your source data is already encrypted, use the `CREATE TABLE` statement. Point `LOCATION` to the encrypted source directory where your data resides:

```
CREATE TABLE encrypted_table [STORED AS] LOCATION ... AS SELECT *
FROM <encrypted_source_directory>
```

This is the fastest way to create encrypted tables.

7.2.4.2.3. Encrypting Other Hive Directories

- **LOCALSCRATCHDIR** : The MapJoin optimization in Hive writes HDFS tables to a local directory and then uploads them to distributed cache. To enable encryption, either disable MapJoin (set `hive.auto.convert.join` to `false`) or encrypt the local Hive Scratch directory (`hive.exec.local.scratchdir`). **Performance note:** disabling MapJoin will result in slower join performance.
- **DOWNLOADED_RESOURCES_DIR**: Jars that are added to a user session and stored in HDFS are downloaded to `hive.downloaded.resources.dir`. If you want these Jar files to be encrypted, configure `hive.downloaded.resources.dir` to be part of an encryption zone. This directory needs to be accessible to the HiveServer2.
- **NodeManager Local Directory List**: Hive stores Jars and MapJoin files in the distributed cache, so if you'd like to use MapJoin or encrypt Jars and other resource files, the YARN configuration property NodeManager Local Directory List (`yarn.nodemanager.local-dirs`) must be configured to a set of encrypted local directories on all nodes.

Alternatively, to disable MapJoin, set `hive.auto.convert.join` to `false`.

7.2.4.2.4. Additional Changes in Behavior with HDFS-Encrypted Tables

- Users reading data from read-only encrypted tables must have access to a temp directory that is encrypted with at least as strong encryption as the table.
- By default, temp data related to HDFS encryption is written to a staging directory identified by the `hive-exec.stagingdir` property created in the `hive-site.xml` file associated with the table folder.
- As of HDP-2.6.0, Hive `INSERT OVERWRITE` queries require a Ranger URI policy to allow write operations, even if the user has write privilege granted through HDFS policy. To fix the failing Hive `INSERT OVERWRITE` queries:

1. Create a new policy under the Hive repository.
 2. In the dropdown where you see Database, select URI.
 3. Update the path (Example: /tmp/*)
 4. Add the users and group and save.
 5. Retry the insert query.
- When using encryption with Trash enabled, table deletion operates differently than the default trash mechanism. For more information see [Delete Files from an Encryption Zone with Trash Enabled \[580\]](#).

7.2.4.3. MapReduce on YARN

Recommendation: Make `/apps/history` a single encryption zone. History files are moved between the `intermediate` and `done` directories, and HDFS encryption will not allow you to move encrypted files across encryption zones.

7.2.4.3.1. Steps

On a cluster with MapReduce over YARN installed, create the `/apps/history` directory and make it an encryption zone.

If `/apps/history` already exists and is not empty:

1. Create an empty `/apps/history-tmp` directory
2. Make `/apps/history-tmp` an encryption zone
3. Copy (`distcp`) all data from `/apps/history` into `/apps/history-tmp`
4. Remove `/apps/history`
5. Rename `/apps/history-tmp` to `/apps/history`

7.2.4.4. Oozie

7.2.4.4.1. Recommendations

A new Oozie administrator role (`oozie-admin`) has been created in HDP 2.3.

This role enables role separation between the Oozie daemon and administrative tasks. Both the `oozie-admin` role and the `oozie` role must be specified in the `adminusers.txt` file. This file is installed in HDP 2.3 with both roles specified. Both are also defined in Ambari 2.1 as well. Modification is only required if administrators choose to change the default administrative roles for Oozie.

If `oozie-admin` is used as the Oozie administrator user in your cluster, then the role is automatically managed by ambari.

If you plan to create an Oozie admin user other than `oozie-admin`, add the chosen username to `adminusers.txt` under the `$OOZIE_HOME/conf` directory.

Here is a sample `adminusers.txt` file:

```
#
# Licensed to the Apache Software Foundation (ASF) under one
# or more contributor license agreements. See the NOTICE file
# distributed with this work for additional information
# regarding copyright ownership. The ASF licenses this file
# to you under the Apache License, Version 2.0 (the
# "License"); you may not use this file except in compliance
# with the License. You may obtain a copy of the License at
#
# http://www.apache.org/licenses/LICENSE-2.0
#
# Unless required by applicable law or agreed to in writing, software
# distributed under the License is distributed on an "AS IS" BASIS,
# WITHOUT WARRANTIES OR CONDITIONS OF ANY KIND, either express or implied.
# See the License for the specific language governing permissions and
# limitations under the License.
#
# Users should be set using following rules:
#
#   One user name per line
#   Empty lines and lines starting with '#' are ignored
#
oozie
oozie-admin
```

7.2.4.5. Sqoop

Following are considerations for using Sqoop to import or export HDFS-encrypted data.

7.2.4.5.1. Recommendations

- **For Hive:**

Make sure that you are using Sqoop with the `--target-dir` parameter set to a directory that is inside the Hive encryption zone. Specify the `-D` option after `sqoop import`.

For example:

```
sqoop import \  
-D sqoop.test.import.rootDir=<root-directory> \  
--target-dir <directory-inside-encryption-zone> \  
<additional-arguments>
```

- **For append or incremental import:**

Make sure that the `sqoop.test.import.rootDir` property points to the encryption zone specified in the `--target-dir` argument.

- **For HCatalog:**

No special configuration is required.

7.2.4.6. WebHDFS

7.2.4.6.1. Recommendations

WebHDFS is supported for writing and reading files to and from encryption zones.

7.2.4.6.1.1. Steps

To access encrypted files via WebHDFS, complete the following steps:

1. To enable WebHDFS in `hdfs-site.xml`, set the `dfs.webhdfs.enabled` property to `true`:

```
<property>
  <name>dfs.webhdfs.enabled</name>
  <value>true</value>
</property>
```

2. Make sure that you have separate HDFS administrative and service users, as described in [Creating an HDFS Admin User](#).
3. KMS supports a blacklist and a whitelist for key access (through `kms-acls.xml`).

By default the `hdfs` service user is included in the blacklist for `decrypt_eeek` operations. To support WebHDFS, the HDFS service user must not be on the key access blacklist. Remove the HDFS service user from the blacklist:

- a. To edit the blacklist using Ambari, go to Ranger KMS -> Configs, and search for "blacklist" or open the Advanced `dbks-site` list.
- b. Remove `hdfs` from the `hadoop.kms.blacklist.DECRYPT_EEK` property:

The screenshot shows the Ambari Web UI interface. On the left, a sidebar lists various services: HDFS, MapReduce2, YARN, Tez, Hive, HBase (with a red '4' badge), Pig, Oozie, ZooKeeper, Storm, Ambari Metrics (with a red '2' badge), Kafka, Kerberos, Knox, Ranger, and Ranger KMS (highlighted). Below the sidebar is an 'Actions' dropdown menu.

The main content area is titled 'Summary' and 'Configs'. It shows a group selection dropdown set to 'Ranger KMS Default (4)' with a 'Manage Config' link. Below this, there are version history cards for V4, V3, and V2, each showing 'admin' as the user and 'about a month ago' as the timestamp, with 'HDP-2.3' below. A green checkmark is visible under the V4 card.

A dark banner at the bottom of the version history section displays a refresh icon, a 'V4' label, a green checkmark, and the text 'admin authored on Fri, Jul 24, 2015 18:00'.

Below the banner, a section titled 'Advanced dbks-site' contains a configuration entry: 'hadoop.kms.blacklist.DECRYPT_EEK' with a text input field containing the value 'hdfs'.

c. Restart Ranger KMS.

4. The HDFS service user must have GENERATE_EEK and DECRYPT_EEK permissions. To add the permissions using the Ranger Web UI, select the Access Manager tab-> Resource Based Policies (the default Access Manager view). Select the key store, select the policy, and click the edit icon. In the Permissions column click the edit icon and check the boxes for GenerateEEK and DecryptEEK. Then click Save.

Ranger

Access Manager

Encryption

Service Manager > cl1_kms Policies > Create Policy

Create Policy

Policy Details :

Policy Name * enabled

Key Name *

Description

Audit Logging YES

User and Group Permissions :

Permissions	Select Group	Select User
	<input type="text" value="x hdfs"/>	<input type="text" value="x hdfs"/>
	<input type="button" value="+"/>	

5. Because the HDFS service user will have access to all keys, the HDFS service user should not be the administrative user. Specify a different administrative user in `hdfs-site.xml` for the administrative user.

For more information about operational tasks using Ranger KMS, see the [Ranger KMS Administration Guide](#).

7.2.5. Appendix: Creating an HDFS Admin User

To capitalize on the capabilities of HDFS data at rest encryption, you will need two separate types of HDFS administrative accounts:

- HDFS administrative user: an account in the `hdfs` supergroup that is used to manage encryption keys and encryption zones. Examples in this chapter use an administrative user account named `encr`.
- HDFS service user: the system-level account traditionally associated with HDFS. By default this is user `hdfs` in HDP. This account owns the HDFS DataNode and NameNode processes.



Important

This is a system-only account. Physical users should not be given access to this account.

Complete the following steps to create a new HDFS administrative user.

Note: These steps use sample values for group (`operator`) and user account (`opt1`).

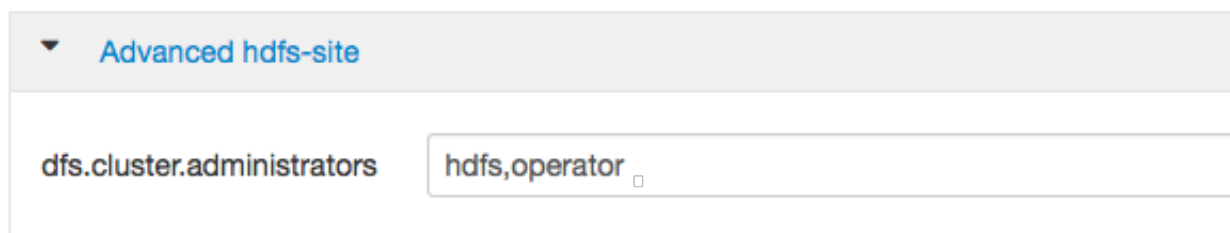
1. Create a new group called `operator`.
2. Add a new user (for example, `opt1`) to the group.
3. Add principal `opt1@EXAMPLE.COM` and create a keytab.
4. Login as `opt1`, and do a `kinit` operation.
5. In Ambari, replace the current value of `dfs.permissions.superusergroup` with the group name "operator".



Note

You can assign only one administrator group for the `dfs.permissions.superusergroup` parameter.

6. In Ambari, add `hdfs, operator` to `dfs.cluster.administrators`:



7. Add `opt1` to the KMS blacklist. Set the corresponding property in Ambari:

```
hadoop.kms.blacklist.DECRYPT_EEK=opt1
```

8. Restart HDFS.

Validation

Make sure the `opt1` account has HDFS administrative access:

```
hdfs dfsadmin -report
```

Make sure the `opt1` account cannot access encrypted files. For example, if `/data/test/file.txt` is in an encryption zone, the following command should return an error:

```
hdfs dfs -cat /data/test/file.txt
```

Additional Administrative User Accounts

If you plan to use HDFS data at rest encryption with YARN, we recommend that you create a separate administrative user account for YARN administration.

If you plan to use HDFS data at rest encryption with Oozie, refer to the [Oozie](#) section of this chapter.

8. Running DataNodes as Non-Root

This chapter describes how to run DataNodes as a non-root user.

8.1. Introduction

Historically, part of the security configuration for HDFS involved starting the DataNode as the root user, and binding to privileged ports for the server endpoints. This was done to address a security issue whereby if a MapReduce task was running and the DataNode stopped running, it would be possible for the MapReduce task to bind to the DataNode port and potentially do something malicious. The solution to this scenario was to run the DataNode as the root user and use privileged ports. Only the root user can access privileged ports.

You can now use Simple Authentication and Security Layer (SASL) to securely run DataNodes as a non-root user. SASL is used to provide secure communication at the protocol level.



Important

Make sure to execute a migration from using root to start DataNodes to using SASL to start DataNodes in a very specific sequence across the entire cluster. Otherwise, there could be a risk of application downtime.

In order to migrate an existing cluster that used root authentication to start using SASL instead, first ensure that HDP 2.2 or later has been deployed to all cluster nodes as well as any external applications that need to connect to the cluster. Only the HDFS client in versions HDP 2.2 and later can connect to a DataNode that uses SASL for authentication of data transfer protocol, so it is vital that all callers have the correct version before migrating. After HDP 2.2 or later has been deployed everywhere, update the configuration of any external applications to enable SASL. If an HDFS client is enabled for SASL, it can connect successfully to a DataNode running with either root authentication or SASL authentication. Changing configuration for all clients guarantees that subsequent configuration changes on DataNodes will not disrupt the applications. Finally, each individual DataNode can be migrated by changing its configuration and restarting. It is acceptable to temporarily have a mix of some DataNodes running with root authentication and some DataNodes running with SASL authentication during this migration period, because an HDFS client enabled for SASL can connect to both.

8.2. Configuring DataNode SASL

Use the following steps to configure DataNode SASL to securely run a DataNode as a non-root user:

1. Shut Down the DataNode

Shut down the DataNode using the applicable commands in [Controlling HDP Services Manually](#).

2. Enable SASL

Configure the following properties in the `/etc/hadoop/conf/hdfs-site.xml` file to enable DataNode SASL.

The `dfs.data.transfer.protection` property enables DataNode SASL. You can set this property to one of the following values:

- `authentication` – Establishes mutual authentication between the client and the server.
- `integrity` – in addition to authentication, it guarantees that a man-in-the-middle cannot tamper with messages exchanged between the client and the server.
- `privacy` – in addition to the features offered by authentication and integrity, it also fully encrypts the messages exchanged between the client and the server.

In addition to setting a value for the `dfs.data.transfer.protection` property, you must set the `dfs.http.policy` property to `HTTPS_ONLY`. You must also specify ports for the DataNode RPC and HTTP Servers.



Note

For more information on configuring SSL, see "Enable SSL on HDP Components" in the *HDP Security Guide*.

For example:

```
<property>
  <name>dfs.data.transfer.protection</name>
  <value>integrity</value>
</property>

<property>
  <name>dfs.datanode.address</name>
  <value>0.0.0.0:10019</value>
</property>

<property>
  <name>dfs.datanode.http.address</name>
  <value>0.0.0.0:10022</value>
</property>

<property>
  <name>dfs.http.policy</name>
  <value>HTTPS_ONLY</value>
</property>
```



Note

If you are already using the following encryption setting:

```
dfs.encrypt.data.transfer=true
```

This is similar to:

```
dfs.data.transfer.protection=privacy
```

These two settings are mutually exclusive, so you should not have both of them set. However, if both are set, `dfs.encrypt.data.transfer` will not be used.

3. Update Environment Settings

Edit the following setting in the `/etc/hadoop/conf/hadoop-env.sh` file, as shown below:

```
#On secure datanodes, user to run the datanode as after dropping privileges
export HADOOP_SECURE_DN_USER=
```

The `export HADOOP_SECURE_DN_USER=hdfs` line enables the legacy security configuration, and must be set to an empty value in order for SASL to be enabled.

4. Start the DataNode

Start the DataNode services using the applicable commands in [Controlling HDP Services Manually](#).

9. Addendum

This chapter collects supplemental documentation.

9.1. ZooKeeper ACLs Best Practices

Permissions for Secure Clusters

Introduction

As more and more components begin to rely on ZooKeeper within a Hadoop cluster, there are various permissions that need to be maintained to ensure the integrity and security of the znodes. These permissions are different from component to component.

Some components only use ZooKeeper when they are running in their component specific HA mode. Others have separate secure and unsecure ACLs defined and switch between which to enforce based on the component knowledge of whether the cluster is secured or not.

In general, it seems that the ACLs are pretty open and assume an unsecure cluster by default. These permissions need to be hardened for secure clusters in order to avoid inappropriate access or modification of this critical platform state.

This paper collects the required steps for tightening the ZooKeeper ACLs/permissions when provisioning a secure cluster to be used as a best practices guideline for ops and security management.

Unaffected Components

The following components require no action:

- Ambari
 - ZooKeeper Usage: Ambari does not use ZooKeeper; however it does install, configure, and manage it so that services running on the cluster can use it.
 - Default ACLs: None. Ambari does not create or use any znodes.
 - Security Best Practice ACLs/Permissions and Required Steps: None. Ambari does not create or use any znodes.
- Calcite
- DataFu
- Falcon
- Flume
 - HDP Flume currently does not depend upon ZooKeeper for any of its core operations. However, ZooKeeper is used by the HBase or Kafka connectors, as the respective client libraries need them.
 - There are no pre-created (i.e at install time) znodes that it depends upon.

- Hue
- Knox
- Mahout
- MapReduce
- Phoenix
 - ZooKeeper Usage: Phoenix does not use ZooKeeper on its own. All usages are covered in the HBase section.
 - Security Best Practice ACLs/Permissions and Required Steps: None. HBase correctly protects all ZNodes in ZooKeeper automatically.
- Pig
- Spark
- Sqoop
- Stargate/HBase RestServer
 - No ZooKeeper usage outside of normal HBase client usage.
- Tez
- Zeppelin

9.1.1. Accumulo

- **ZooKeeper Usage:**
 - `/accumulo` - Parent ZNode for all of Accumulo use in ZooKeeper
 - `/accumulo/$UUID` - Parent ZNode for a specific Accumulo instance
 - `/accumulo/instances` - Contains mappings of human-readable Accumulo names to the UUID
 - `/accumulo/$UUID/users` - Accumulo user database
 - `/accumulo/$UUID/problems` - Persisted advertisement of reported problems in Accumulo
 - `/accumulo/$UUID/root_tables` - The "root" Accumulo table (points to the Accumulo metadata table)
 - `/accumulo/$UUID/hdfs_reservations` - ZNode to coordinate unique directories in HFDS for bulk imports of Accumulo files to a table
 - `/accumulo/$UUID/gc` - Advertisement and leader election for Accumulo GarbageCollector

- `/accumulo/$UUID/table_locks` - RW-locks per Accumulo table
- `/accumulo/$UUID/fate` - Parent znode for Accumulo's FATE (distributed, multi-step transactions)
- `/accumulo/$UUID/tservers` - Advertisement and ephemeral znodes(keep-alive) for TabletServers
- `/accumulo/$UUID/tables` - The "database" of Accumulo tables (metadata)
- `/accumulo/$UUID/namespaces` - The "database" of Accumulo namespaces (metadata)
- `/accumulo/$UUID/next_file` - Coordinates unique name generation for files in HDFS
- `/accumulo/$UUID/config` - Dynamic configuration for Accumulo
- `/accumulo/$UUID/masters` - Advertisement and leader election for the Accumulo Master
- `/accumulo/$UUID/monitor` - Advertisement and leader election for the Accumulo Monitor
- `/accumulo/$UUID/bulk_failed_copyq` - Tracking files to bulk import which failed
- `/accumulo/$UUID/recovery` - Used to coordinate recovery of write-ahead logs
- **Default ACLs:**
 - All znodes not specified otherwise are world-readable and cdrwa 'accumulo'. Those below are not world-readable:
`/accumulo/$UUID/users/*`
- **Security Best Practice ACLs/Permissions and Required Steps:**
 - The user does not need to alter any ACLs in ZooKeeper. Accumulo protects all ZNodes automatically.

9.1.2. Ambari Solr

Ambari Solr is used by LogSearch, Ranger and Atlas.

- **ZooKeeper Usage:**
 - `/ambari-solr` - Solr node for storing collection, leaders, configuration, etc.
- **Default ACLs:**
 - `/ambari-solr` - world:anyone:cdrwa
- **Security Best Practice ACLs/Permissions and Required Steps:**

- /ambari-solr-world:anyone:r
- /ambari-solr-sasl:solr:cdrwa

9.1.3. Atlas

- **ZooKeeper Usage:**
 - /apache_atlas - Root zookeeper node which is configured for curator, under which nodes for leader election are created.
 - /apache_atlas/active_server_info - Znode used in HA environments for storing active server information.
 - /apache_atlas/setup_in_progress - Transient Znode used to ensure some setup steps are executed only from one instance. This gets deleted after use and should normally not be seen.
- **Default ACLs:**
 - All znodes have world:anyone:cdrwa by default.
- **Security Best Practice ACLs/Permissions and Required Steps:**
 - No user intervention is required for creating/using the Znodes. They are all managed internally by Atlas. Atlas exposes two configuration properties that define the auth and ACL - to use while creating these Znodes. Ambari should configure these correctly for a secure cluster. The recommended configuration is `atlas.server.ha.zookeeper.auth=sasl:atlas@<domain.com>` and `atlas.server.ha.zookeeper.acl=sasl:atlas@<domain.com>`, where `<domain.com>` should be replaced with the right value of the atlas service user principal. (Assuming atlas is the service user name). When set this way, the ACLs for all znodes will be `atlas.server.ha.zookeeper.acl=sasl:atlas@<domain.com>:cdrwa`. (Note we don't allow configuration of the permissions from Ambari).

9.1.4. HBase

- **ZooKeeper Usage:**
 - /hbase-unsecure - Default znode for unsecured clusters
 - /hbase-secure - Default znode used for secured clusters
- **Default ACLs:**
 - /hbase-unsecure - world:hbase:cdrwa
 - All children ZNodes are also world cdrwa
 - Open for global read, write protected: world:anyone:r, sasl:hbase:cdrwa
 - /hbase-secure

- `/hbase-secure/master`
- `/hbase-secure/meta-region-server`
- `/hbase-secure/hbaseid`
- `/hbase-secure/table`
- `/hbase-secure/rs`
- **No global read, r/w protected: `sasl:hbase:cdrwa:`**
 - `/hbase-secure/acl`
 - `/hbase-secure/namespace`
 - `/hbase-secure/backup-masters`
 - `/hbase-secure/online-snapshot`
 - `/hbase-secure/draining`
 - `/hbase-secure/replication`
 - `/hbase-secure/region-in-transition`
 - `/hbase-secure/splitWAL`
 - `/hbase-secure/table-lock`
 - `/hbase-secure/recovering-regions`
 - `/hbase-secure/running`
 - `/hbase-secure/tokenauth`
- **Security Best Practice ACLs/Permissions and Required Steps:**
 - HBase code determines which ACL to enforce based on the configured security mode of the cluster/hbase. Users are not expected to perform any modification of ZooKeeper ACLs on ZNodes and users should not alter any ACLs by hand.

9.1.5. HDFS/WebHDFS

- **ZooKeeper Usage:**
 - `hadoop-ha-hdfs zkfc automatic NameNode failover`
- **Default ACLs:**
 - `hadoop-ha-world: anyone: cdrwa`
- **Security Best Practice ACLs/Permissions and Required Steps:**

- `hadoop-ha - sasl: nn:cdrwa`
- Existing SmartSense rule recommends ACL of `sasl:nn:rwcd` for secured clusters. To set this:

1. Set `ha.zookeeper.acl` to `sasl:nn:rwcd`:

- Using Ambari:

Add `ha.zookeeper.acl` with value `sasl:nn:rwcd` in Configs>Advanced>Custom core-site.

- Manually:

Add this to `core-site.xml` as root user:

```
<property>
  <name>ha.zookeeper.acl</name>
  <value>sasl:nn:rwcd</value>
</property>
```

2. Add this `HADOOP_ZKFC_OPTS` export:

- Using Ambari:

In Configs > Advanced > Advanced hadoop-env > hadoop-env template, add the following:

```
export HADOOP_ZKFC_OPTS="Dzookeeper.sasl.client=true
zookeeper                               Dzookeeper.sasl.client.username=
etc/hadoop/conf/hdfs_jaas.conf          Djava.security.auth.login.config=/
                                          Dzookeeper.sasl.clientconfig=Client
                                          ${HADOOP_ZKFC_OPTS} "
```

- Manually:

Add this to `hadoop-env.sh` as root user:

```
export HADOOP_ZKFC_OPTS="Dzookeeper.sasl.client=true
zookeeper                               Dzookeeper.sasl.client.username=
etc/hadoop/conf/hdfs_jaas.conf          Djava.security.auth.login.config=/
                                          Dzookeeper.sasl.clientconfig=Client
                                          ${HADOOP_ZKFC_OPTS} "
```

3. On two Namenodes, create `/etc/hadoop/conf/hdfs_jaas.conf` as root user with the following contents:

```
Client {
    com.sun.security.auth.module.Krb5LoginModule required
    useKeyTab=true
    storeKey=true
    useTicketCache=false
    keyTab="/etc/security/keytabs/nn.service.keytab"
    principal="nn/<HOST>@EXAMPLE.COM" ;
};
```

nn/<HOST>@EXAMPLE.COM must be changed to the actual hostname and realm, e.g. nn/c6401.ambari.apache.org@EXAMPLE.COM. To get actual principal, on two Namenodes, run the command as hdfs user: `klist -k /etc/security/keytabs/nn.service.keytab`.

4. Stop the two ZKFCs.

5. On one of Namenodes, run the command as hdfs user: `hdfs zkfc -formatZK -force`.

6. Start the two ZKFCs.

One of two Namenodes may be stopped in the process, or standby Namenode may be transitioned to active one. Start the stopped namenode if any.

9.1.6. Hive/HCatalog

- **ZooKeeper Usage:**

- /hiveserver2 - The parent znode used by HiveServer2 when supporting dynamic service discovery. Each server instance creates an ephemeral znode under this namespace. Exposed via the hive config: `hive.server2.zookeeper.namespace`
- /hivedelegation/METASTORE - HA ONLY - The root path for token store data, used by Metastore servers to store delegation tokens. Exposed via hive config: `hive.cluster.delegation.token.store.zookeeper.znode`
- /hivedelegation/HIVESERVER2 - HA ONLY - The root path for token store data, used by HiveServer2 servers to store delegation tokens. Exposed via hive config: `hive.cluster.delegation.token.store.zookeeper.znode`
- /hive_zookeeper_namespace - Used by ZooKeeper-based implementation of Hive's LockMgr (`ZooKeeperHiveLockManager`) if used. This usage is writable-to by any user as it tries to co-ordinate locking among multiple users. Controlled by hive config: `hive.zookeeper.namespace`. In addition, which LockMgr we use is also controlled by hive config: `hive.lock.manager`. (Note also, that if ACID is used, we do not use a ZooKeeper-based lock manager)
- /llap-<sas1/unsecure>/user-<user_name> is used by LLAP to store cluster node locations. Should be writable by hive, readable by anyone. LLAP takes care of enforcing the ACLs for the secure path.
- /zkdtsm_<cluster_id>/ZKDTSMRoot/* is used by LLAP token/secret manager, in secure cluster only. Should only be accessible by hive. LLAP sets and validates the ACLs.

- **Default ACLs:**

- /hiveserver2 - world:anyone:r
- /hiveserver2 - sasl:hive:cdrwa
- /hivedelegation - world:anyone:r
- /hivedelegation - sasl:hive:cdrwa
- /hive_zookeeper_namespace - completely-open
- /llap-sasl/user-<user_name> - sasl:hive:cdrwa, world:anyone:r
- /llap-unsecure/user-<user_name> - world:anyone:cdrwa
- /zkdtm_<cluster_id>/ZKDTSMRoot/* - sasl:hive:cdrwa

Note that ACLs are considered recursively applied to nodes inside these roots - i.e., /hivedelegation/METASTORE, /hivedelegation/HIVESERVER2, or /hiveserver2/<first_server>.

- **Security Best Practice ACLs/Permissions and Required Steps:**

- /hiveserver2 - world:anyone:r
- /hiveserver2 - sasl:hive:cdrwa
- /hivedelegation - world:anyone:r
- /hivedelegation - sasl:hive:cdrwa
- /hive_zookeeper_namespace - completely-open
- /llap-sasl/user-<user_name> - sasl:hive:cdrwa, world:anyone:r
- /llap-unsecure/user-<user_name> - world:anyone:cdrwa
- /zkdtm_<cluster_id>/ZKDTSMRoot/* - sasl:hive:cdrwa

Note that ACLs are considered recursively applied to nodes inside these roots - i.e., /hivedelegation/METASTORE, /hivedelegation/HIVESERVER2, or /hiveserver2/<first_server>.

9.1.7. Kafka

- **ZooKeeper Usage:**

- /controller - Kafka Znode for controller leader election
- /brokers - Kafka Znode for broker metadata
- /kafka-acl - Kafka Znode for SimpleAclAuthorizer ACL storage
- /admin - Kafka admin tool metadata

- `/isr_change_notification` - Track changes to In Sync Replication
- `/controller_epoch` - Track movement of controller
- `/consumers` - Kafka Consumer list
- `/config` - Entity configuration
- **Default ACLs:**
 - N/A
- **Security Best Practice ACLs/Permissions and Required Steps:**
 - `/controller - world:anyone:r`
 - `/controller - sasl:kafka:cdrwa`
 - `/brokers - world:anyone:cdrwa`
 - `/kafka-acl - sasl:kafka:cdrwa`
 - `/admin - world:anyone:cdrwa`
 - `/isr_change_notification - world:anyone:r`
 - `/isr_change_notification - sasl:kafka:cdrwa`
 - `/controller_epoch - world:anyone:cdrwa`
 - `/consumers - world:anyone:cdrwa`
 - `/config - world:anyone:cdrwa`

When security is enabled `zookeeper.set.acl=true` should be in `kafkaConfig`. Which is not happening now. Users can add this using **Advanced Property** `zookeeper.set.acl` and add a new zkroot to `zookeepr.connect = "host.name:2181:/kafka"` to create new nodes as it won't update the ACLs on existing node. Alternatively, they can use `kafka.service.keytab` to log into zookeeper and set ACLs recursively.

9.1.8. Oozie

- **ZooKeeper Usage:**
 - Used to coordinate multiple Oozie servers.
- **Default ACLs:**

In a secure cluster, Oozie restricts the access to Oozie Znodes to the oozie principals only using Kerberos backed ACLs.

- `/oozie` - node that stores oozie server information in HA mode

Default ACLs:

- `/oozie-world:anyone:cdrwa`
- **Security Best Practice ACLs/Permissions and Required Steps:**
 - Set `oozie.zookeeper.secure` to `secure`

9.1.9. Ranger

- **ZooKeeper Usage:**
 - Ranger does not use ZooKeeper directly. Only if **Audit to Solr** is enabled and Solr is configured in SolrCloud mode, Solr nodes will need to access zookeeper node `/ranger_audits`.
`/ranger_audits`
- **Default ACLs:**
 - `/ranger_audits-world:anyone:cdrwa`
- **Security Best Practice ACLs/Permissions and Required Steps:**
 - Only Solr needs access to this Znode:
`/ranger_audits-sasl:solr:cdrwa`
 - After enabling SolrCloud, edit the Ranger collection path permission on Znode:
 1. SSH to the cluster where SolrCloud is present.
 2. Go to `/usr/hdp/<version>/zookeeper/bin`.
 3. Run `./zkCli.sh -server <FQDN SolrCloud host>:2181`
 4. After it connects, run: `ls /`
 5. Verify there is a folder for the Ranger Solr collection.
 6. Execute `getAcl /ranger_audits` and if the permission is for `world,anyone: cdrwa`, restrict the permission to `"sasl:solr:cdrwa"` using this command:
`setAcl /ranger_audits sasl:solr:cdrwa`.
 7. Repeat the above step for all clusters where SolrCloud is installed.

```
[zk: as-ha-27-3.openstacklocal:2181(CONNECTED) 0] ls /
[zookeeper, rmstore, ranger_audits]
[zk: as-ha-27-3.openstacklocal:2181(CONNECTED) 1] getAcl /ranger_audits
'world,'anyone
: cdrwa
[zk: as-ha-27-3.openstacklocal:2181(CONNECTED) 2] setAcl /ranger_audits
sasl:solr:cdrwa
cZxid = 0x200000037
ctime = Wed Jun 29 10:40:24 UTC 2016
mZxid = 0x200000037
mtime = Wed Jun 29 10:40:24 UTC 2016
pZxid = 0x200000056
```



```

cversion = 7
dataVersion = 0
aclVersion = 1
ephemeralOwner = 0x0
dataLength = 0
numChildren = 7
[zk: as-ha-27-3.openstacklocal:2181(CONNECTED) 3] getAcl /ranger_audits
'sasl','solr
: cdrwa
[zk: as-ha-27-3.openstacklocal:2181(CONNECTED) 4]

```

9.1.10. Ranger KMS/Hadoop KMS

- **ZooKeeper Usage:**

- If multiple instances of KMS are configured, both Ranger KMS and Hadoop KMS use zookeeper znode /hadoop-kms to store HTTP cookie signature secret. See “Http Authentication Signature” section [here](#).

```
/hadoop-kms - <HTTP cookie signature secret>
```

- **Default ACLs:**

- /hadoop-kms - world:anyone: cdrwa

- **Security Best Practice ACLs/Permissions and Required Steps:**

- /hadoop-kms - sasl:rangerkms: cdrwa
- Ranger KMS uses the user rangerkms. Only KMS needs access to this znode. This path (hadoop.kms.authentication.signer.secret.provider.zookeeper.path) can be configured in Ambari for Ranger KMS. Set the ACL using these steps:
 1. SSH to the cluster where Ranger KMS is present.
 2. Go to /usr/hdp/<version>/zookeeper/bin
 3. Run ./zkCli.sh -server <FQDN of Ranger KMS host>:2181"
 4. After it connects, run: ls /
 5. Verify there is a folder as specified in hadoop.kms.authentication.signer.secret.provider.zookeeper.path property of Ranger KMS configuration.
 6. Execute getAcl /hadoop-kms and if the permission is for world,anyone: cdrwa, restrict the permission to sasl:rangerkms: cdrwa using this command: setAcl /hadoop-kms sasl:rangerkms: cdrwa.
 7. Repeat the above step for all the clusters where Ranger KMS is installed.

```

[zk: dk-test-0706-3.openstacklocal:2181(CONNECTED) 0] getAcl /hadoop-kms
'world','anyone
: cdrwa
[zk: dk-test-0706-3.openstacklocal:2181(CONNECTED) 4] setAcl /hadoop-kms
sasl:rangerkms: cdrwa

```

```

cZxid = 0x20000001e
ctime = Tue Jun 07 12:22:58 UTC 2016
mZxid = 0x20000001e
mtime = Tue Jun 07 12:22:58 UTC 2016
pZxid = 0x20000001f
cversion = 1
dataVersion = 0
aclVersion = 1
ephemeralOwner = 0x0
dataLength = 0
numChildren = 1
[zk: dk-test-0706-3.openstacklocal:2181(CONNECTED) 5] getAcl /hadoop-kms
'sasl,'rangerkms
: cdrwa
[zk: dk-test-0706-3.openstacklocal:2181(CONNECTED) 6]

```

9.1.11. Slider

- **ZooKeeper Usage:**
 - `/services/slider/users/username/applicationName`- A node created by slider for use by an application if the “create.default.zookeeper.node” property is set to “true” in the Slider appConfig file. It is intended for use by the application so ACL modification would be application specific.
- **Default ACLs:**
 - `/services/slider/users/username/applicationName - world:anyone:r, world:user:cdrwa`
- **Security Best Practice ACLs/Permissions and Required Steps:**
 - The application node is created with the above ACLs during Slider application launch, so no further steps are required.

9.1.12. Storm

- **ZooKeeper Usage:**
 - `/storm` - All data for storm metadata, Storm's root znode
- **Default ACLs:**
 - `/storm - world:anyone:cr`
 - `/storm - sasl:storm-PRD1:cdrwa`

Where `-PRD1` comes from StormClient Principal and Ambari creates the principal with `storm-<cluster_name>`.
- **Security Best Practice ACLs/Permissions and Required Steps:**
 - `/storm - world:anyone:cr`
 - `/storm - sasl:storm-PRD1:cdrwa`

Where `-PRD1` comes from StormClient Principal and Ambari creates the principal with `storm-<cluster_name>`.

9.1.13. WebHCat

- **ZooKeeper Usage:**
 - `/templeton-hadoop` - WebHCat stores status of jobs that users can query in zookeeper (if ZooKeeperStorage is configured to find out the status of jobs - it can also use HDFS for this storage). WebHCat typically will create three znodes inside this root: "jobs", "overhead" and "created". This root node is exposed via config: `templeton.storage.root`. In addition, whether or not ZooKeeperStorage is used is configured by another config parameter: `templeton.storage.class`. Both these parameters are part of `webhcat-site.xml`. These nodes are altered from launcher map task as well, which runs as the end user.
- **Default ACLs:**
 - `/templeton-hadoop - world: anyone: cdrwa`
- **Security Best Practice ACLs/Permissions and Required Steps:**
 - `/templeton-hadoop - world: anyone: cdrwa`

9.1.14. YARN

- **ZooKeeper Usage:**
 - `/yarn-leader-election` - used for RM leader election
 - `/rmstore` - used for storing RM application state
- **Default ACLs:**
 - `/yarn-leader-election - world: anyone: cdrwa`
 - `/rmstore - world: anyone: cdrwa`
- **Security Best Practice ACLs/Permissions and Required Steps:**
 - `/yarn-leader-election - world: anyone: r`
 - `/yarn-leader-election - sasl: rm: rwcda`
 - `/rmstore - world: anyone: r`
 - `/rmstore - sasl: rm: rwcda`

9.1.15. YARN Registry

The YARN registry is a location into which statically and dynamically deployed applications can register service endpoints; client applications can look up these entries to determine the URLs and IPC ports with which to communicate with a service.

It is implemented as a zookeeper tree: services register themselves as `system services`, under the registry path `/system`, or `user services`, which are registered under `/users/USERNAME` where `USERNAME` is the name of the user registering the service.

As the purpose of the mechanism is to allow arbitrary clients to look up a service, the entries are always world readable. No secrets should be added to service entries.

In insecure mode, all registry paths are world readable and writeable: nothing may be trusted.

In a secure cluster, the registry is designed to work as follows:

1. Kerberos + SASL provides the identification and authentication.
2. `/system` services can only be registered by designated system applications (YARN, HDFS, etc)/
3. User-specific services can only be registered by the user deploying the application.
4. If a service is registered under a user's path, it may be trusted, and any published public information (such as HTTPS certifications) assumed to have been issued by the user.
5. All user registry entries should also be registered as world writeable with the list of system accounts defined in `hadoop.registry.system.accounts`; this is a list of ZK SASL-authenticated accounts to be given full access. This is needed to support system administration of the entries, especially automated deletion of old entries after application failures.
6. The default list of system accounts are `yarn`, `mapred`, `hdfs`, and `hadoop`; these are automatically associated with the Kerberos realm of the process interacting with the registry, to create the appropriate `sas1:account@REALM` ZK entries.
7. If applications are running from different realms, the configuration option `hadoop.registry.kerberos.realm` must be set to the desired realm, or `hadoop.registry.system.accounts` configured with the full realms of the accounts.
8. There is support for ZooKeeper `id:digest` authentication; this is to allow a user's short-lived YARN applications to register service endpoints without needing the Kerberos TGT. This needs active use by the launching application (which must explicitly create a user service node with an `id:digest` permission, or by setting `hadoop.registry.user.accounts`, to the list of credentials to be permitted.
9. System services must not use `id:digest` authentication —nor should they need to; any long-lived service already needs to have a kerberos keytab.
10. The per-user path for their user services, `/users/USERNAME`, is created by the YARN resource manager when users launch services, if the RM is launched with the option `hadoop.registry.rm.enabled` set to `true`.
11. When `hadoop.registry.rm.enabled` is `true`, the RM will automatically purge application and container service records when the applications and containers terminate.

12.Communication with ZK is over SASL, using the

`java.security.auth.login.config` system property to configure the binding.

The specific JAAS context to use can be set in `hadoop.registry.jaas.context` if the default value, `Client`, is not appropriate.

ZK Paths and Permissions:

All paths are world-readable; permissions are set up when the RM creates the root entry and user paths and `hadoop.registry.secure=true`.

Path	Role	Permissions
<code>/registry</code>	Base registry path	yarn, hdfs, mapred, hadoop : cdrwa
<code>/registry/system</code>	System services	yarn, hdfs, mapred, hadoop : cdrwa
<code>/registry/users</code>	Users	yarn, hdfs, mapred, hadoop : cdrwa
<code>/registry/users/USER</code>	The registry tree for the user <i>USER</i> .	USER: rwa yarn, hdfs, mapred, hadoop : cdrwa

Configuration options for secure registry access

Name	Recommended Value
<code>hadoop.registry.secure</code>	true
<code>hadoop.registry.rm.enabled</code>	true
<code>hadoop.registry.system.accounts</code>	sasl:yarn@, sasl:mapred@, sasl:hdfs@, sasl:hadoop@ Grants system accounts write access to the root registry paths. A tighter version would be <code>sasl:yarn@</code> which will only give the RM the right to manipulate these, or explicitly declare a realm, such as <code>sasl:yarn@EXAMPLE</code>
<code>hadoop.registry.kerberos.realm</code>	(empty) The Kerberos realm to use when converting the system accounts to full realms. If left empty, uses the realm of the user
<code>hadoop.registry.user.accounts</code>	(empty)
<code>hadoop.registry.client.auth</code>	kerberos How to authenticate with ZK. Alternative (insecure) options: anonymous, digest.
<code>hadoop.registry.jaas.context</code>	Client The JAAS context to use for registry clients to authenticate with ZooKeeper.

9.1.16. ZooKeeper

• ZooKeeper Usage:

- `/zookeeper` - node stores metadata of ZooKeeper itself.
- `/zookeeper/quota` stores quota information. In the Apache ZooKeeper 3.5 release line.

- `/zookeeper/config` stores dynamic reconfiguration information, but this is not applicable to HDP, which bases its ZooKeeper release off of the Apache ZooKeeper 3.4 release line.
- **Default ACLs:**
 - `/zookeeper - world: anyone: cdrwa`
- **Security Best Practice ACLs/Permissions and Required Steps:**

The following steps must be manually performed by users who are using the ZooKeeper quota feature. Components in HDP do not use this feature by default – most users do not need to execute the following commands.

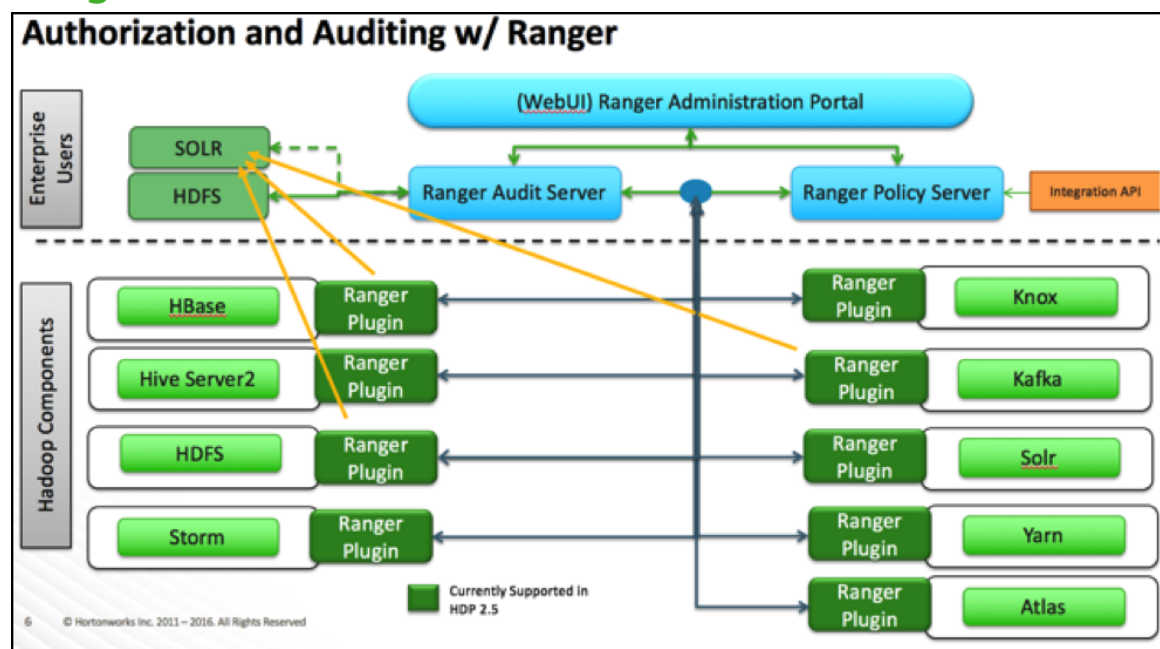
- `/zookeeper - sasl: zookeeper: cdrwa`
- `setAcl sasl:zookeeper:rwcd`

9.2. Ranger AD Integration

This chapter aims to provide some practical help on Ranger AD integration.

Ranger is at the core of governing access to multiple HDP/HDF components by providing the concept of a 'Ranger plugin'.

9.2.1. Ranger Architecture



When a Ranger plugin for a component (like HBase or HDFS) is activated, Ranger will be in full control of any access. There is a two-way communication between the Ranger plugin and Ranger (Admin) Policy Server (RPS):

1. **Plugins to RPS:** Ranger plugins regularly call the RPS to see if new policies were defined in the Ranger Administration Portal (RAP). Generally allow for 30 sec. for a policy to be updated.
2. **RPS to components:** The RPS queries the component for meta objects that live on the component to base policies upon (this provides the autocomplete and dropdown list when defining policies.)

The first communication channel (Plugins to RPS) is essential for the plugin to function whereas the second (RPS to components) is optional. It would still be possible to define and enforce policies if the second does not work, but you will not have autocomplete during policy definition.

Configuration details on both communication channels are configured on both Ambari configuration for the component and on the RAP.

Example for HDFS plugin:

The screenshot shows the configuration interface for the HDFS plugin in Ambari. It is divided into two sections: 'Advanced ranger-hdfs-plugin-properties' and 'Advanced ranger-hdfs-policymgr-ssl'.


Advanced ranger-hdfs-plugin-properties:

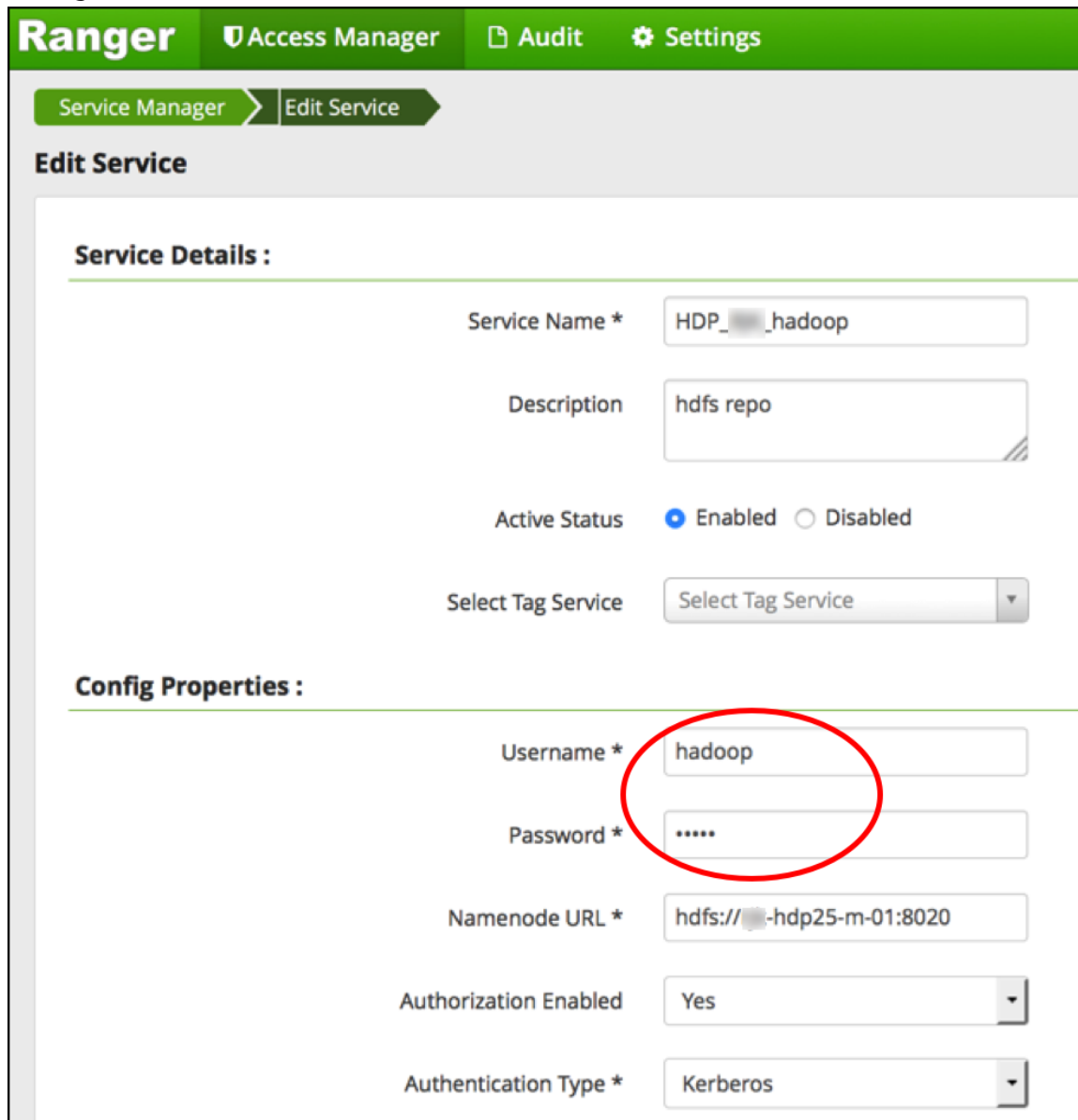
- Enable Ranger for HDFS: (with lock and refresh icons)
- Ranger repository config password: (with lock icon)
- Ranger repository config user: (with lock, status, and refresh icons)
- common.name.for.certificate: (with lock and status icons)
- hadoop.rpc.protection: (with lock, status, and refresh icons)
- Policy user for HDFS: (with lock, status, and refresh icons)

Advanced ranger-hdfs-policymgr-ssl:

- xasecure.policymgr.clientssl.keystore: (with status and refresh icons)
- xasecure.policymgr.clientssl.keystore.credential.file: (with status and refresh icons)
- xasecure.policymgr.clientssl.keystore.password: (with lock icon)
- xasecure.policymgr.clientssl.truststore: (with status and refresh icons)
- xasecure.policymgr.clientssl.truststore.credential.file: (with status and refresh icons)
- xasecure.policymgr.clientssl.truststore.password: (with lock icon)

The 'Ranger repository config user' is the one that involved the second communication channel (RPS to components) for getting metadata from HDFS (like HDFS folders) across.

The settings on the HDFS configuration have to match those set at the Ranger end (Access Manager > Resource Based Policies > HDFS > ):



Ranger Access Manager Audit Settings

Service Manager > Edit Service

Edit Service

Service Details :

Service Name * HDP_..._hadoop

Description hdfs repo

Active Status Enabled Disabled

Select Tag Service Select Tag Service

Config Properties :

Username * **hadoop**

Password *

Namenode URL * hdfs://...-hdp25-m-01:8020


Authorization Enabled Yes

Authentication Type * Kerberos

To verify if the paramount first communication channel (Plugins to RPS) works can be done by having a look in the RAP at Audit > Plugins:

The screenshot shows the Ranger Admin interface with the 'Plugins' tab selected. A search bar is at the top. Below it, a table displays synchronization logs. The table has columns for 'Export Date (CET) *', 'Service Name', 'Plugin Id', 'Plugin IP', 'Http Response Code', and 'Status'. The status for all entries is 'Policies synced to plugin'.

Export Date (CET) *	Service Name	Plugin Id	Plugin IP	Http Response Code	Status
12/13/2016 01:13:30 PM	HDP_hive	hiveServer2@-hdp25-m-02-HDP_hive	172.26.	200	Policies synced to plugin
12/13/2016 01:12:00 PM	HDP_hive	hiveServer2@-hdp25-m-02-HDP_hive	172.26.	200	Policies synced to plugin
12/13/2016 11:09:15 AM	HDP_atlas	atlas@-hdp25-m-01-HDP_atlas	172.26.	200	Policies synced to plugin
12/13/2016 11:00:14 AM	HDP_atlas	atlas@-hdp25-m-01-HDP_atlas	172.26.	200	Policies synced to plugin
12/13/2016 10:43:12 AM	HDP_atlas	atlas@-hdp25-m-01-HDP_atlas	172.26.	200	Policies synced to plugin
12/12/2016 10:58:18 PM	HDP_kafka	kafka@-hdp25-s-03-HDP_kafka	172.26.	200	Policies synced to plugin

To verify the second communication channel (RPS to components) press the 'Test Connection' button (Access Manager > Resource Based Policies > HDFS > ):

The screenshot shows the configuration page for HDFS. It includes several settings:

- Authorization Enabled: Yes
- Authentication Type: Kerberos
- hadoop.security.auth_to_local: RULE:[1:\$1@\$0](ambari-qa-hdp_rj)
- dfs.datanode.kerberos.principal: dn/-hdp25-m-01@FIELD.HORTC
- dfs.namenode.kerberos.principal: nn/-hdp25-m-01@FIELD.HORTC
- dfs.secondary.namenode.kerberos.principal: nn/-hdp25-m-01@FIELD.HORTC
- RPC Protection Type: Authentication
- Common Name for Certificate: (empty)

Under 'Add New Configurations', there is a table with the following entries:

Name	Value
ambari.service.check.user	ambari-qa
tag.download.auth.users	hdfs
policy.download.auth.users	hdfs

The 'Test Connection' button is circled in red. At the bottom, there are 'Save', 'Cancel', and 'Delete' buttons.

If the settings are right you'll get:



9.2.1.1. Ranger Audit

Ranger plugins furthermore send their audit event (whether access was granted or not and based on which policy) directly to the configured sink for audits, which can be HDFS, Solr or both. This is indicated by the yellow arrows in the architectural graph.

The audit access tab on the RAP (Audit > Access) is only populated if Solr is used as sink.

Policy ID	Event Time *	User	Service Name / Type	Resource Name / Type	Access Type	Result	Access Enforcer	Client IP	Event Count	Tags
--	12/14/2016 11:08:31 AM	spark	HDP_RJK_hadoop hdfs	hpaerk2-history path	READ_EXECUTE	Allowed	hadoop-act	172.26....	1	
--	12/14/2016 11:08:31 AM	spark	HDP_RJK_hadoop hdfs	hpaerk2-history path	WRITE	Allowed	hadoop-act	172.26....	1	
--	12/14/2016 11:08:31 AM	spark	HDP_RJK_hadoop hdfs	hpaerk-history path	READ_EXECUTE	Allowed	hadoop-act	172.26....	1	
--	12/14/2016 11:08:31 AM	spark	HDP_RJK_hadoop hdfs	hpaerk-history path	WRITE	Allowed	hadoop-act	172.26....	1	
--	12/14/2016 11:08:31 AM	spark	HDP_RJK_hadoop hdfs	hpaerk2-history path	WRITE	Allowed	hadoop-act	172.26....	1	
--	12/14/2016 11:08:31 AM	spark	HDP_RJK_hadoop hdfs	hpaerk-history path	WRITE	Allowed	hadoop-act	172.26....	1	
--	12/14/2016 11:08:26 AM	rangeragsync	HDP_RJK_kafka kafka	ATLAS_ENTITIES topic	describe	Denied	ranger-act	172.26....	11	
--	12/14/2016 11:08:26 AM	rangeragsync	HDP_RJK_kafka kafka	ATLAS_ENTITIES topic	describe	Denied	ranger-act	172.26....	7	
14	12/14/2016 11:08:25 AM	atlas	HDP_RJK_hbase hbase	atlas_qlanrm column-family	get	Allowed	ranger-act	172.26....	1	
8	12/14/2016 11:08:25 AM	atlas	HDP_RJK_kafka kafka	ATLAS_HOOK topic	consume	Allowed	ranger-act	172.26....	49	
--	12/14/2016 11:08:25 AM	rangeragsync	HDP_RJK_kafka kafka	ATLAS_ENTITIES topic	describe	Denied	ranger-act	172.26....	7	

This screen points out an important Ranger feature. When the plugin is enabled AND no specific policy is in place for access to some object, the plugin will fall back to enforcing the standard component level Access Control Lists (ACL's). For HDFS that would be the user : rwx / group : rwx / other : rwx ACL's on folders and files.

Once this defaulting to component ACL's happens the audit events show a ' - ' in the 'Policy ID' column instead of a policy number. If a Ranger policy was in control of allowing/denying the policy number is shown.

9.2.2. Ranger AD Integration

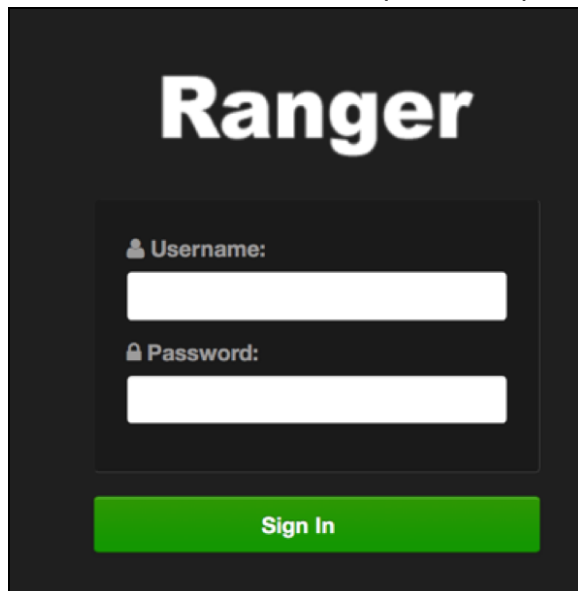
Rangers AD Integration has 2 levels:

1. Ranger UI authentication (which users may log on to Ranger itself?)
2. Ranger User / group sync (which users / groups to define policies for?)

The configuration of both is done entirely on Ambari.

9.2.2.1. Ranger UI Authentication

This is an extra AD level filter option on top of Kerberos authentication that maps to:



For working with AD there are 2 options for defining who can access the Ranger UI; LDAP or ACTIVE_DIRECTORY. There is not much difference between them, just another set of properties.

Some of the configuration is in fact shared with the configuration of Ranger usersync as can be seen by the property with formats like `ranger_ug_ldap_bind_dn`. These properties are provided at runtime only with the value of another property by that name.

ACTIVE_DIRECTORY

The configuration for it is on Ambari > Ranger > Configs > Advanced:

Authentication method

LDAP

ACTIVE_DIRECTORY

UNIX

NONE

HTTP enabled

AD Settings

ranger.ldap.ad.base.dn

ranger.ldap.ad.bind.dn

ranger.ldap.ad.bind.password

Domain Name (Only for AD)

ranger.ldap.ad.referral

ranger.ldap.ad.url

ranger.ldap.ad.user.searchfilter

The `ranger.ldap.ad.base.dn` determines the base of any search, so users not on this OU tree path can not be authenticated.

The `ranger.ldap.ad.user.searchfilter` is a dynamic filter that maps the user name in the Ranger Web UI login screen to `sAMAccountName`. For example, the AD `sAMAccountName` property has example values like `k.reshi` and `d.alora` so make sure to enter a matching value for 'Username' in the logon dialogue.

With `ACTIVE_DIRECTORY` it is not possible to limit the scope of users that can access Ranger UI any further by refining the `ranger.ldap.ad.user.searchfilter` even further to :

```
( & (memberOf=CN=Hdp_admins,OU=Company,OU=User
Accounts,OU=CorpUsers,DC=field,DC=hortonworks,DC=com)
(sAMAccountName={0}))
```

This does NOT work with the `ACTIVE_DIRECTORY` option.

LDAP

The other LDAP related properties do allow for more fine tuning:

The image shows two configuration panels from the Ranger interface. The top panel, 'Ranger Settings', includes:

- External URL: http://[redacted]-hdp25-m-02:6080
- Authentication method: LDAP (selected), ACTIVE_DIRECTORY, UNIX, NONE
- HTTP enabled: checked

 The bottom panel, 'LDAP Settings', includes:

- ranger.ldap.base.dn: OU=[redacted],OU=User Accounts,OU=CorpUsers,DC=field,DC=hortonworks,DC=com
- Bind User: {{ranger_ug_ldap_bind_dn}}
- Bind User Password: [redacted]
- ranger.ldap.group.roleattribute: cn
- ranger.ldap.referral: follow
- LDAP URL: {{ranger_ug_ldap_url}}
- ranger.ldap.user.dnpattern: DC=intentionally,DC=wrong
- User Search Filter: (&(objectclass=user)(memberOf=CN=Hdp_admins,OU=[redacted],OU=User Accounts,OU=...

There is 1 catch though; the `ranger.ldap.user.dnpattern` is evaluated first, so usually putting a value like:

```
CN={0},OU=London,OU=Company,OU=User
Accounts,OU=CorpUsers,DC=field,DC=hortonworks,DC=com
```

Would work, but has 2 by-effects; first users would have to log on with their 'long username' (like *'Kvothe Reshi / Denna Alora'*) which would also mean that policies would have to be updated using that long name in stead of the *k.reshi* short name variant.

Second traversing AD by DN patterns does not allow for applying group filters at all. In the syntax above only users directly in OU=London would be able to log on.

That adverse behavior can be worked around by intentionally putting a DN pattern (DC=intentionally,DC=wrong) in the `ranger.ldap.user.dnpattern` property AND a valid filter in **User Search Filter**:

```
(&(objectclass=user)(memberOf=CN=Hdp_admins,OU=Company,OU=User
Accounts,OU=CorpUsers,DC=field,DC=hortonworks,DC=com)
(sAMAccountName={0}))
```

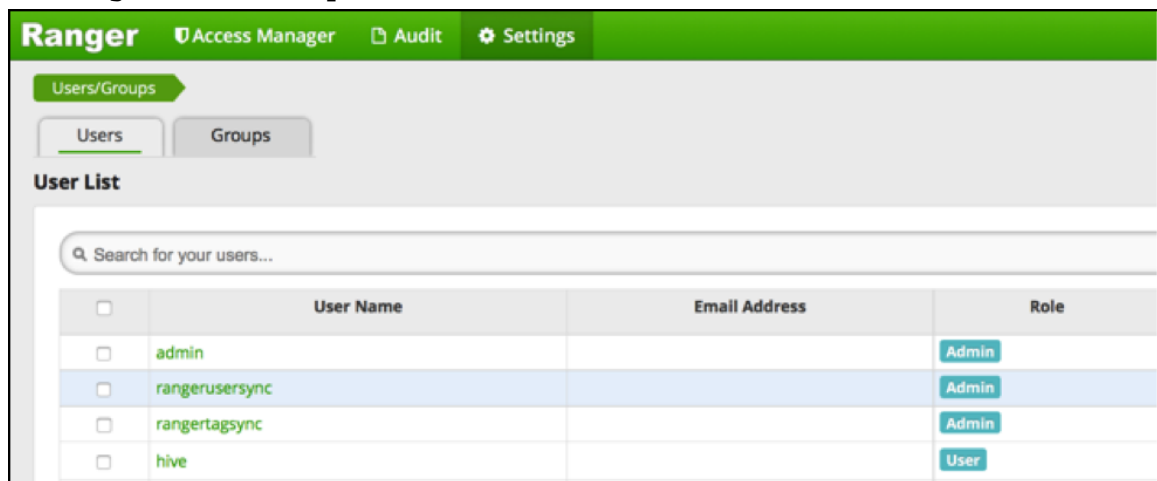
This works because the filter is only applied after the DN pattern query on AD does not return anything. If it does, then the **User Search Filter** is not applied.

Ranger has a very simple approach to the internal user list that is kept in a relational schema. That list contains all users that were synced with AD ever, and all those users can potentially log on to Ranger UI. But only admin users can really do anything policy related things on the Ranger UI (see next section).

Beware that all this is still only about authentication to Ranger. Someone from the 'Hdp_admins' group would still not have a Ranger admin role.

9.2.2.2. Ranger UI Authorization

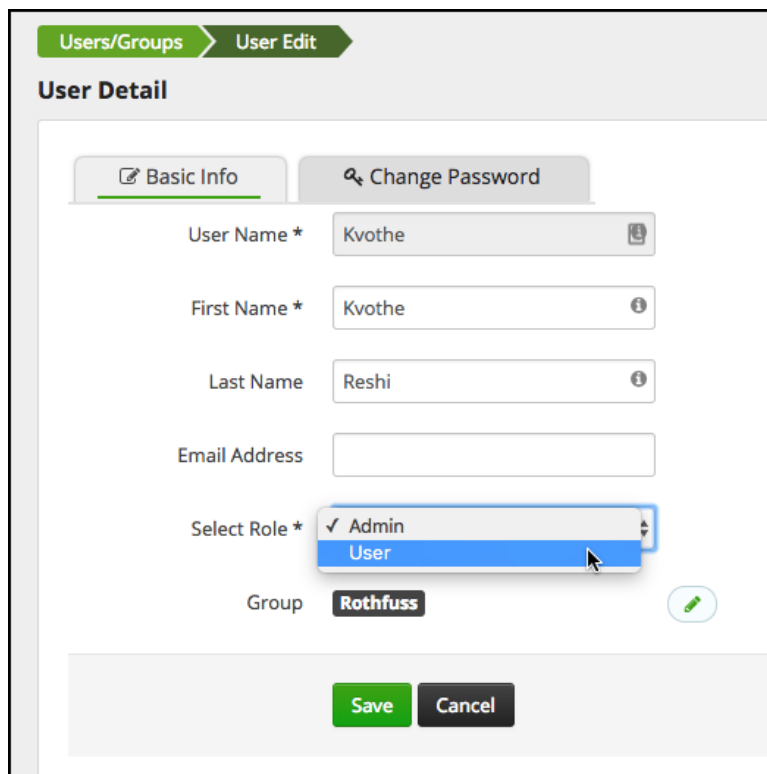
The Ranger authorization model is quite simple. It is maintained on the Ranger UI at Settings>Users/Groups :



The screenshot shows the Ranger UI interface. At the top, there is a green navigation bar with 'Ranger' and 'Access Manager', 'Audit', and 'Settings' tabs. Below this, there is a 'Users/Groups' section with 'Users' and 'Groups' sub-tabs. The 'Users' tab is active, showing a 'User List' table. The table has a search bar at the top and four columns: 'User Name', 'Email Address', and 'Role'. There are five rows of users listed, each with a checkbox on the left and a role button on the right.

<input type="checkbox"/>	User Name	Email Address	Role
<input type="checkbox"/>	admin		Admin
<input type="checkbox"/>	rangerusersync		Admin
<input type="checkbox"/>	rangertagsync		Admin
<input type="checkbox"/>	hive		User

A user can be either a normal user or an admin:

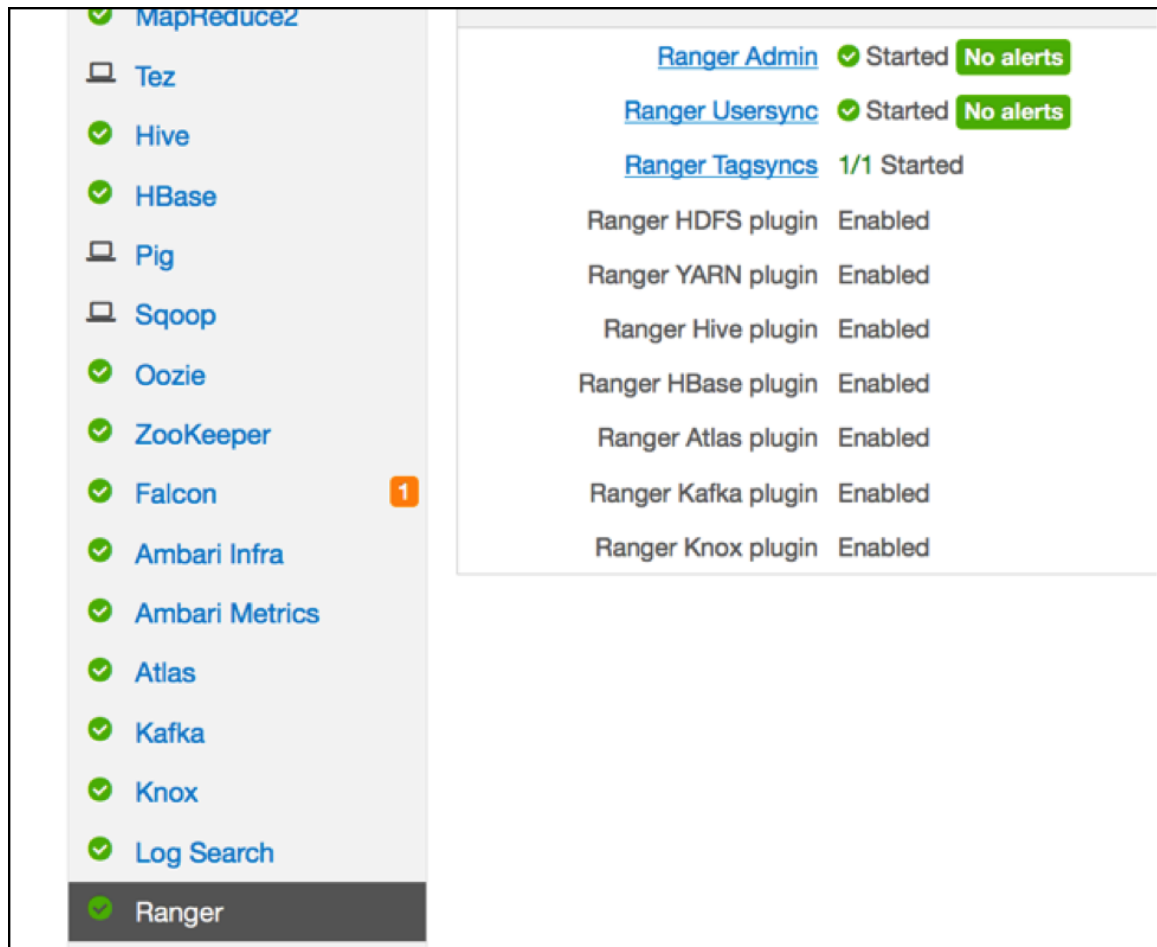


Only user with an Admin role can view or alter policies in Ranger.

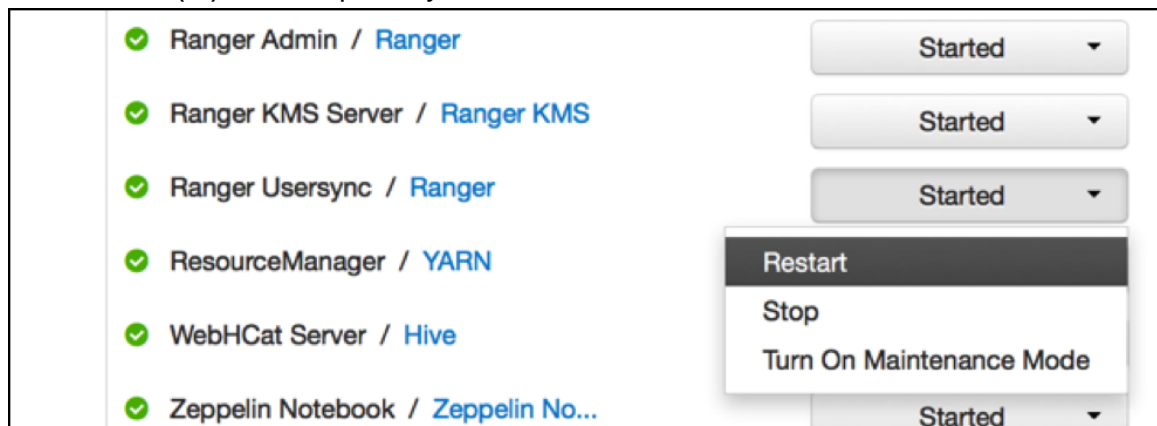
9.2.2.3. Ranger Usersync

A vital part of the Ranger architecture is the ability to get users and groups from the corporate AD to use in policy definitions.

Ranger usersync runs as separate daemon:



It can also be (re)started separately from Ambari:

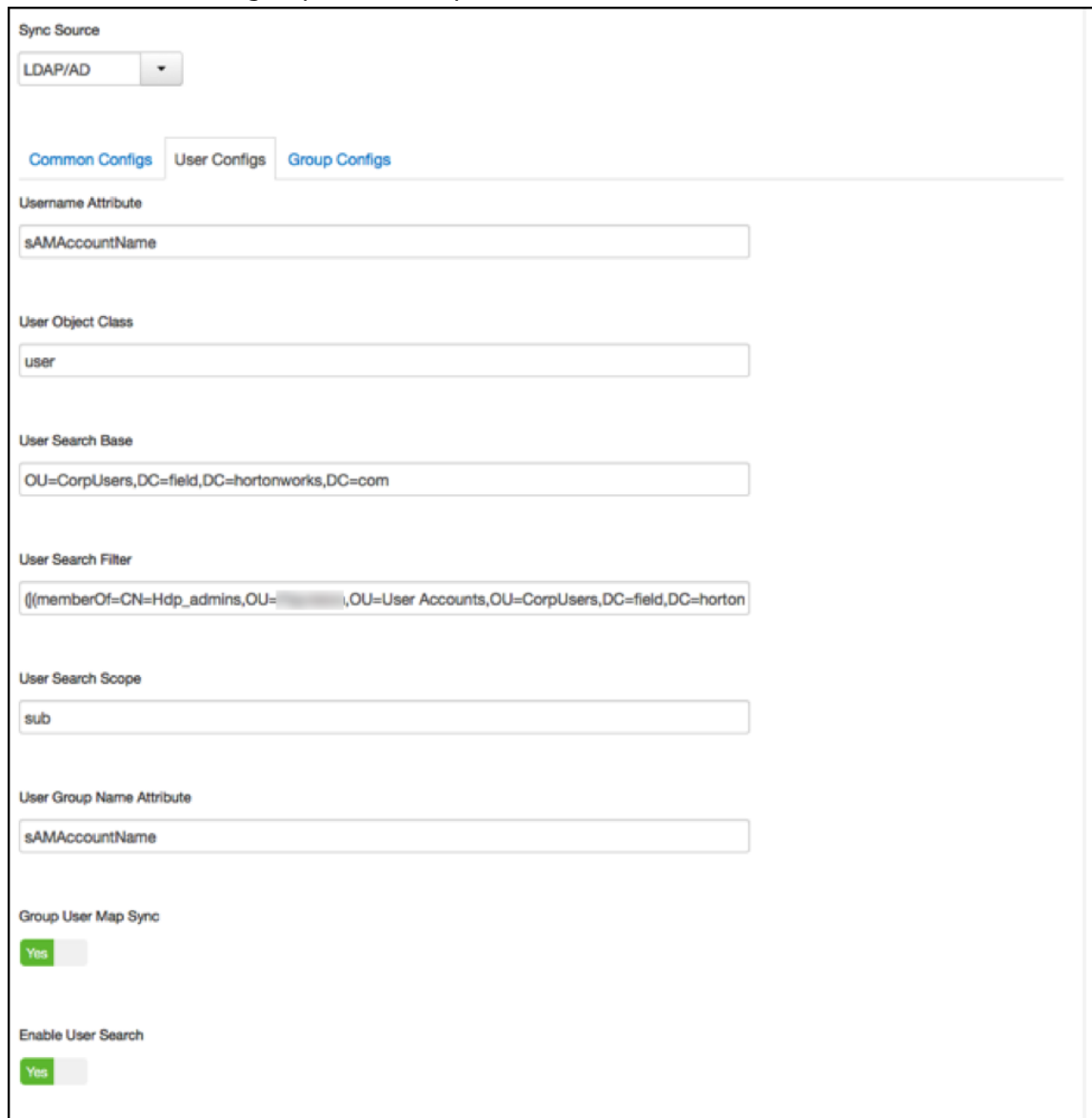


9.2.2.3.1. Configuration

Usersync has a lot of moving parts and can have very different outcomes. Two main sets of properties govern the way users and groups are synchronized.

Without **Enable Group Search First** (a setting on the tab **Group Configs**) the primary access pattern is user based and groups will only be searched/added based on the users it finds

first. In contrast, with **Enable Group Search First** enabled, the primary access pattern is group based (in turn based on the group search filter) and users will only be searched/added based on the group memberships it finds first



Sync Source

LDAP/AD

Common Configs User Configs Group Configs

Username Attribute

sAMAccountName

User Object Class

user

User Search Base

OU=CorpUsers,DC=field,DC=hortonworks,DC=com

User Search Filter

(|(memberOf=CN=Hdp_admins,OU=,OU=User Accounts,OU=CorpUsers,DC=field,DC=horton

User Search Scope

sub

User Group Name Attribute

sAMAccountName

Group User Map Sync

Yes

Enable User Search

Yes

Value of 'User Search Base':
 OU=CorpUsers,DC=field,DC=hortonworks,DC=com

Value of 'User Search Filter':
 (|(memberOf=CN=Hdp_admins,OU=Company,OU=User Accounts,OU=CorpUsers,DC=field,DC=hortonworks,DC=com) (memberOf=CN=Hdp_users,OU=Company,OU=User Accounts,OU=CorpUsers,DC=field,DC=hortonworks,DC=com))

Value of 'User Group Name Attribute':
 sAMAccountName

Ranger User Info

Enable User Sync

Sync Source
LDAP/AD

[Common Configs](#) [User Configs](#) [Group Configs](#)

Enable Group Sync

Group Member Attribute

Group Name Attribute

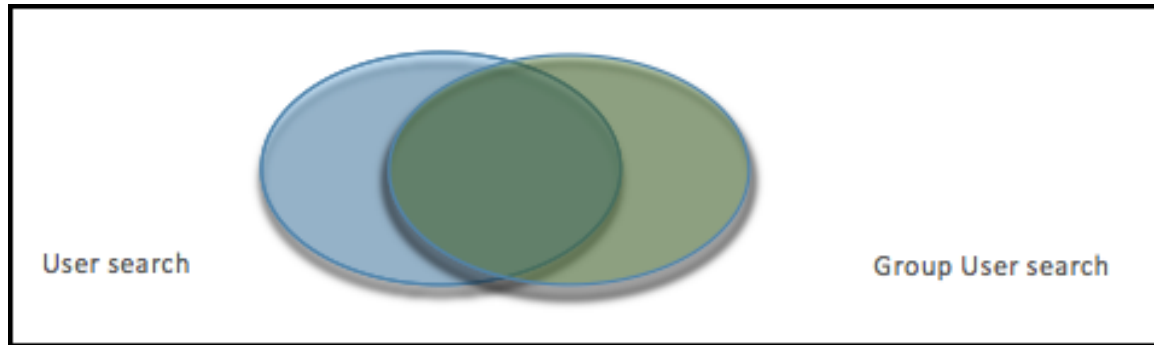
Group Object Class

Group Search Base

Group Search Filter

Enable Group Search First

Value of 'Group Search Base':
(|(CN=Hdp_users)(CN=Hdp_admins))



Beware that the filters on group level limit the returns on the user search and vice versa. In the graph above if the left oval would be the results of all users queried by the settings on the **User configs** and the right oval all users queried by **Group configs** settings, the eventual set of users that make it to the Ranger usersync is the overlap between the two.

Hortonworks therefore recommends to have the filters on both ends set exactly the same to potentially have a 100% overlap in the ovals.

In the example configuration given the scope of the usersync would be all members of both the groups 'Hdp_admins' and 'Hdp_users'.

The result in Ranger User list:

The screenshot shows the Ranger Admin interface. At the top, there are navigation links for 'Access Manager', 'Audit', and 'Settings'. Below that, there are tabs for 'Users/Groups', 'Users', and 'Groups'. The 'Users' tab is selected, and the 'User List' is displayed. A search bar is present with the placeholder text 'Search for your users...'. To the right of the search bar are buttons for 'Add New User', 'Set Visibility', and a red trash icon. The main content is a table with the following columns: 'User Name', 'Email Address', 'Role', 'User Source', 'Groups', and 'Visibility'. The table contains several rows of user data, including users like 'hdfs', 'sqoop', 'yarn', 'centos', 'mapred', 'knox', 'systemd-bus-proxy', 'amb_ranger_admin', 'slider', 'sssd', 'hdp', 'hdp_admins', and 'hdp_admins'.

	User Name	Email Address	Role	User Source	Groups	Visibility
<input type="checkbox"/>	hdfs		User	External	hadoop hdfs	Visible
<input type="checkbox"/>	sqoop		User	External	hadoop	Visible
<input type="checkbox"/>	yarn		User	External	hadoop	Visible
<input type="checkbox"/>	centos		User	External	centos admin wheel systemd-journal	Visible
<input type="checkbox"/>	mapred		User	External	hadoop	Visible
<input type="checkbox"/>	knox		User	External	hadoop	Visible
<input type="checkbox"/>	systemd-bus-proxy		User	External	systemd-bus-proxy	Visible
<input type="checkbox"/>	amb_ranger_admin		Admin	Internal	-	Visible
<input type="checkbox"/>	slider		User	External	slider	Visible
<input type="checkbox"/>	sssd		User	External	sssd	Visible
<input type="checkbox"/>	hdp		Admin	External	hdp_admins	Visible
<input type="checkbox"/>	hdp_admins		User	External	hdp_admins	Visible
<input type="checkbox"/>	hdp_admins		Admin	External	hdp_admins	Visible

Regarding the other switches on the user and group sync pages, best of both worlds is to have **Enable Group Search First** and **Enable User Search** enabled at the same time.

The logging of a run of the usersync daemon can be retrieved from `/var/log/ranger/usersync/usersync.log` on the server hosting Ranger Admin. A successful run might output logging like below:

```

[snatched: true, userSearchEnabled: true, ldap referral: ignore
08 Dec 2016 19:40:05 INFO UserGroupSync [UnixUserSyncThread] - Begin: Initial load of user/group from source=sink
08 Dec 2016 19:40:05 INFO LdapUserGroupBuilder [UnixUserSyncThread] - LdapUserGroupBuilder updateSink started
08 Dec 2016 19:40:05 INFO LdapUserGroupBuilder [UnixUserSyncThread] - Performing Group search first
08 Dec 2016 19:40:05 INFO LdapUserGroupBuilder [UnixUserSyncThread] - Adding Hdp_users to user
08 Dec 2016 19:40:05 INFO LdapUserGroupBuilder [UnixUserSyncThread] - Adding Hdp_users to user
08 Dec 2016 19:40:05 INFO LdapUserGroupBuilder [UnixUserSyncThread] - No. of members in the group Hdp_users = 2
08 Dec 2016 19:40:05 INFO LdapUserGroupBuilder [UnixUserSyncThread] - Adding Hdp_admins to user
08 Dec 2016 19:40:05 INFO LdapUserGroupBuilder [UnixUserSyncThread] - Adding Hdp_admins to user
08 Dec 2016 19:40:05 INFO LdapUserGroupBuilder [UnixUserSyncThread] - Adding Hdp_admins to user
08 Dec 2016 19:40:05 INFO LdapUserGroupBuilder [UnixUserSyncThread] - No. of members in the group Hdp_admins = 3
08 Dec 2016 19:40:05 INFO LdapUserGroupBuilder [UnixUserSyncThread] - LDAPUserGroupBuilder.getGroups() completed with group count: 2
08 Dec 2016 19:40:05 INFO LdapUserGroupBuilder [UnixUserSyncThread] - User search is enabled and hence computing user membership.
08 Dec 2016 19:40:05 INFO LdapUserGroupBuilder [UnixUserSyncThread] - Updating username for
08 Dec 2016 19:40:05 INFO LdapUserGroupBuilder [UnixUserSyncThread] - Updating username for
08 Dec 2016 19:40:05 INFO LdapUserGroupBuilder [UnixUserSyncThread] - Updating username for
08 Dec 2016 19:40:05 INFO LdapUserGroupBuilder [UnixUserSyncThread] - Updating username for
08 Dec 2016 19:40:05 INFO LdapUserGroupBuilder [UnixUserSyncThread] - Updating username for
08 Dec 2016 19:40:06 INFO LdapUserGroupBuilder [UnixUserSyncThread] - LDAPUserGroupBuilder.getUsers() completed with user count: 5
08 Dec 2016 19:40:06 INFO UserGroupSync [UnixUserSyncThread] - End: Initial load of user/group from source=sink
08 Dec 2016 19:40:06 INFO UserGroupSync [UnixUserSyncThread] - Done initializing user/group source and sink

```

From that log it clearly shows that the groups are synced first and that all users belonging to those groups are then retrieved according to its own settings, after which the user parts are enriched/overwritten by the returns from the user queries.

Beware:

If you don't enable **Enable User Search** that enrichment does NOT happen. Logging for such a run looks like this:

```

08 Dec 2016 18:24:28 INFO LdapUserGroupBuilder [UnixUserSyncThread] - LdapUserGroupBuilder initialization completed with -- ldapurl: ldap://wbi1.field.hortonworks.com:389, ldapbind: binduser@field.hortonworks.com, ldapbind
08 Dec 2016 18:24:28 INFO LdapUserGroupBuilder [UnixUserSyncThread] - LdapUserGroupBuilder updateSink started
08 Dec 2016 18:24:28 INFO LdapUserGroupBuilder [UnixUserSyncThread] - Performing Group search first
08 Dec 2016 18:24:28 INFO LdapUserGroupBuilder [UnixUserSyncThread] - Adding Hdp_users to user
08 Dec 2016 18:24:28 INFO LdapUserGroupBuilder [UnixUserSyncThread] - Adding Hdp_users to user
08 Dec 2016 18:24:28 INFO LdapUserGroupBuilder [UnixUserSyncThread] - No. of members in the group Hdp_users = 2
08 Dec 2016 18:24:28 INFO LdapUserGroupBuilder [UnixUserSyncThread] - Adding Hdp_admins to user
08 Dec 2016 18:24:28 INFO LdapUserGroupBuilder [UnixUserSyncThread] - Adding Hdp_admins to user
08 Dec 2016 18:24:28 INFO LdapUserGroupBuilder [UnixUserSyncThread] - No. of members in the group Hdp_admins = 3
08 Dec 2016 18:24:28 INFO LdapUserGroupBuilder [UnixUserSyncThread] - LDAPUserGroupBuilder.getGroups() completed with group count: 2
08 Dec 2016 18:24:28 INFO LdapUserGroupBuilder [UnixUserSyncThread] - User search is disabled and hence using the group member attribute for username.
08 Dec 2016 18:24:28 INFO LdapUserGroupBuilder [UnixUserSyncThread] - LongUserName:
08 Dec 2016 18:24:28 INFO LdapUserGroupBuilder [UnixUserSyncThread] - LongUserName:
08 Dec 2016 18:24:28 INFO LdapUserGroupBuilder [UnixUserSyncThread] - LongUserName:
08 Dec 2016 18:24:28 INFO LdapUserGroupBuilder [UnixUserSyncThread] - LongUserName:
08 Dec 2016 18:24:28 INFO LdapUserGroupBuilder [UnixUserSyncThread] - LongUserName:
08 Dec 2016 18:24:28 INFO LdapUserGroupBuilder [UnixUserSyncThread] - End: Initial load of user/group from source=sink
08 Dec 2016 18:24:28 INFO UserGroupSync [UnixUserSyncThread] - Done initializing user/group source and sink

```

The result in Ranger UI are other user names (LongUserName) derived from 'member' group attributes full DN. You get the long name 'James Kirk' in the Ranger userlist in stead of j.kirk.

Ranger does not treat those as one and the same user:

Username	Role	User Source
<input type="checkbox"/> Knox	User	External
<input type="checkbox"/> systemd-bus-proxy	User	External
<input type="checkbox"/> amb_ranger_admin	Admin	Internal
<input type="checkbox"/> slider	User	External
<input type="checkbox"/> sssd	User	External
<input type="checkbox"/> [blurred]	Admin	External
<input type="checkbox"/> [blurred]	User	External
<input type="checkbox"/> [blurred]	Admin	External
<input type="checkbox"/> [blurred]	User	External
<input type="checkbox"/> [blurred]	User	External
<input type="checkbox"/> hadoop	User	External
<input type="checkbox"/> rangerlookup	User	External
<input type="checkbox"/> [blurred]	User	External
<input type="checkbox"/> [blurred]	User	External
<input type="checkbox"/> [blurred]	User	External
<input type="checkbox"/> [blurred]	User	External
<input type="checkbox"/> [blurred]	User	External

Policies that were defined for user 'k.reshi' will not map to the user 'Kvothe Reshi' and vice versa. To prevent any confusion it is probably best to delete the long username versions from Rangers userlist.

Beware:

On the first page of Rangers user list there are lots of HDP system users. Most of them were put there by the Ranger installer and during the plugins installs:

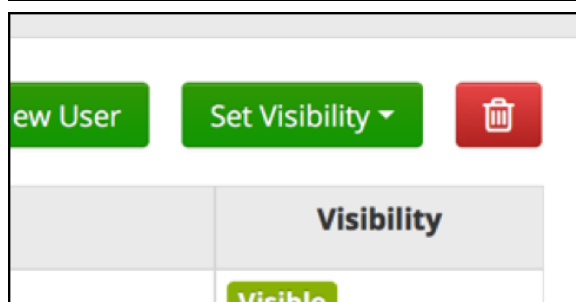
Username	Email Address	Role	User Source
<input type="checkbox"/> admin		Admin	Internal
<input type="checkbox"/> rangerusersync		Admin	Internal
<input type="checkbox"/> rangertagsync		Admin	Internal
<input type="checkbox"/> hive		User	External
<input type="checkbox"/> infra-solr		User	External
<input type="checkbox"/> atlas		User	External
<input type="checkbox"/> ams		User	External
<input type="checkbox"/> falcon		User	External
<input type="checkbox"/> systemd-network		User	External
<input type="checkbox"/> ranger		User	External
<input type="checkbox"/> kms		User	External
<input type="checkbox"/> polkitd		User	External
<input type="checkbox"/> nfsnobody		User	External
<input type="checkbox"/> spark		User	External
<input type="checkbox"/> hbase		User	External
<input type="checkbox"/> hcat		User	External
<input type="checkbox"/> zookeeper		User	External
<input type="checkbox"/> oozie		User	External
<input type="checkbox"/> tez		User	External
<input type="checkbox"/> zeppelin		User	External
<input type="checkbox"/> logsearch		User	External
<input type="checkbox"/> livy		User	External

Do NOT remove those system users!

There are basic access policies based on those system users designed to keep a Ranger governed HDP component working after Ranger is given all control over that components authorizations. Without those policies/users many HDP components will be in serious trouble.

9.2.2.4. Ranger User Management

<input type="checkbox"/>	Bast
<input checked="" type="checkbox"/>	Auri
<input type="checkbox"/>	Felurian
<input type="checkbox"/>	Cinder



User can be easily remove from Ranger by checking the username in the list and hit the red **Delete** button. Ranger takes care of referential integrity so that user will also be removed from any policy.

9.2.3. Issue Ranger Group Mapping

There is an issue with Ranger not being able to map a user on a group 'Hdp_admins' to a policy that allows/denies access to the group 'Hdp_admins'. The issue is on the capital characters that might be on a AD group name definition.

Most HDP components get the group information for a user via the SSSD daemon. When asked for the groups the user 'd.threpe' belongs to we get:

```
[centos@rjk-hdp25-m-01 ~]$ groups d.threpe
d.threpe : domain_users hdp_admins hadoop
```

So 'hdp_admins' all in lower case. Ranger does not treat this as the same value as 'Hdp_admins' which came via the group sync and was applied to some policies.

There is no way to make the group sync write or retrieve the group names all in lower case since there is no AD attribute that rewrites it in lowercase.

This issue can be worked around fortunately (till it gets solved). The solution is to define a local group in Ranger as a shadow group of a real group from AD, but then all in lower case:

<input type="checkbox"/>	system:bus proxy	Internal
<input type="checkbox"/>	slider	External
<input type="checkbox"/>	sssd	External
<input type="checkbox"/>	Hdp_users	External
<input type="checkbox"/>	Hdp_admins	External
<input type="checkbox"/>	hdp_admins	Internal
<input type="checkbox"/>	hdp_users	Internal

If we now create policies and use that lower case 'shadow' group literal the result is that policies are correctly mapped to the AD groups again:

The screenshot shows the Ranger Access Manager interface. At the top, there are navigation tabs for 'Access Manager', 'Audit', and 'Settings'. Below that, the breadcrumb path is 'Service Manager > HDP_atlas Policies'. The main heading is 'List of Policies : HDP_atlas'. There is a search bar with the placeholder text 'Search for your policy...'. Below the search bar is a table with the following columns: Policy ID, Policy Name, Status, Audit Logging, and Groups. The table contains five rows of policy data.

Policy ID	Policy Name	Status	Audit Logging	Groups
9	all - taxonomy	Enabled	Enabled	Hdp_admins hdp_users hdp_admins
10	all - operation	Enabled	Enabled	Hdp_admins hdp_admins hdp_users
11	all - type	Enabled	Enabled	Hdp_admins hdp_admins hdp_users
12	all - entity	Enabled	Enabled	Hdp_admins hdp_users
13	all - term	Enabled	Enabled	Hdp_admins hdp_users hdp_admins

*The 'Hdp_admins' entry does not have to be there, it is shown for clarification only. 'hdp_admins' is necessary to make it work.