

Hortonworks Data Platform

Hadoop Security Guide

(Mar 1, 2016)

Hortonworks Data Platform : Hadoop Security Guide

Copyright © 2012-2016 Hortonworks, Inc. Some rights reserved.

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

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

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



Except where otherwise noted, this document is licensed under
Creative Commons Attribution ShareAlike 3.0 License.
<http://creativecommons.org/licenses/by-sa/3.0/legalcode>

Table of Contents

1. HDP Security Overview	1
1.1. Understanding Data Lake Security	1
1.2. HDP Security Features	3
1.2.1. Administration	4
1.2.2. Authentication and Perimeter Security	5
1.2.3. Authorization	5
1.2.4. Audit	7
1.2.5. Data Protection	8
2. UI Authentication	9
2.1. Enabling Kerberos Authentication Using Ambari	9
2.1.1. Kerberos Overview	9
2.1.2. Hadoop and Kerberos Principals	10
2.1.3. Installing and Configuring the KDC	11
2.1.4. Enabling Kerberos Security	16
2.1.5. Kerberos Client Packages	20
2.1.6. Disabling Kerberos Security	20
2.1.7. Customizing the Attribute Template	20
2.2. Configuring Ambari Authentication with LDAP or AD	21
2.2.1. Configuring Ambari for LDAP or Active Directory Authentication	21
2.2.2. Configuring Ranger Authentication with UNIX, LDAP, or AD	26
2.2.3. Encrypting Database and LDAP Passwords in Ambari	35
2.3. Setting Up Kerberos Authentication for Non-Ambari Clusters	37
2.3.1. Preparing Kerberos	37
2.3.2. Configuring HDP for Kerberos	43
2.3.3. Setting up One-Way Trust with Active Directory	68
2.3.4. Configuring Proxy Users	70
2.4. Perimeter Security with Apache Knox	70
2.4.1. Apache Knox Gateway Overview	70
2.4.2. Configuring the Knox Gateway	72
2.4.3. Configuring a Hadoop Server for Knox	77
2.4.4. Configuring Authentication	81
3. Configuring Authorization in Hadoop	93
3.1. Installing Ranger Using Ambari	93
3.1.1. Overview	93
3.1.2. Installation Prerequisites	93
3.1.3. Ranger Installation	99
3.1.4. Enabling Ranger Plugins	136
3.1.5. Ranger Plugins - Kerberos Overview	161
3.2. Using Ranger to Provide Authorization in Hadoop	165
3.2.1. Opening and Closing the Ranger Console	165
3.2.2. Console Operations Summary	166
3.2.3. Configuring Services	167
3.2.4. Policy Management	184
3.2.5. Users/Groups and Permissions Administration	202
3.2.6. Reports Administration	212
3.2.7. Special Requirements for High Availability Environments	215
3.2.8. Adding a New Component to Apache Ranger	216
3.2.9. Developing a Custom Authorization Module	218

4. Data Protection: Wire Encryption	220
4.1. Enabling RPC Encryption	220
4.2. Enabling Data Transfer Protocol	221
4.3. Enabling SSL: Understanding the Hadoop SSL Keystore Factory	221
4.4. Creating and Managing SSL Certificates	223
4.4.1. Obtain a Certificate from a Trusted Third-Party Certification Authority (CA)	223
4.4.2. Create and Set Up an Internal CA (OpenSSL)	224
4.4.3. Installing Certificates in the Hadoop SSL Keystore Factory (HDFS, MapReduce, and YARN)	228
4.4.4. Using a CA-Signed Certificate	229
4.5. Enabling SSL for HDP Components	230
4.6. Enable SSL for WebHDFS, MapReduce Shuffle, and YARN	230
4.7. Enable SSL for HttpFS	233
4.8. Enable SSL on Oozie	234
4.8.1. Configure Oozie HCatalogJob Properties	235
4.9. Enable SSL on WebHBase and the HBase REST API	235
4.10. Enable SSL on HiveServer2	236
4.11. Enable SSL for Kafka Clients	237
4.11.1. Configuring the Kafka Broker	237
4.11.2. Configuring Kafka Producer and Kafka Consumer	239
4.12. Enable SSL for Accumulo	240
4.12.1. Generate a Certificate Authority	241
4.12.2. Generate a Certificate/Keystore Per Host	241
4.12.3. Configure Accumulo Servers	242
4.12.4. Configure Accumulo Clients	243
4.13. SPNEGO setup for WebHCat	243
4.14. Configure SSL for Hue	244
4.15. Configure SSL for Knox	244
4.15.1. Self-Signed Certificate with Specific Hostname for Evaluations	244
4.15.2. CA-Signed Certificates for Production	245
4.15.3. Setting Up Trust for the Knox Gateway Clients	245
4.16. Securing Phoenix	245
4.17. Set Up SSL for Ambari	246
4.17.1. Set Up Truststore for Ambari Server	247
4.18. Configure Ambari Ranger SSL	248
4.18.1. Configuring Ambari Ranger SSL Using Public CA Certificates	248
4.18.2. Configuring Ambari Ranger SSL Using a Self-Signed Certificate	260
4.19. Configure Non-Ambari Ranger SSL	275
4.19.1. Configuring Non-Ambari Ranger SSL Using Public CA Certificates	275
4.19.2. Configuring Non-Ambari Ranger SSL Using a Self Signed Certificate	278
4.20. Connecting to SSL-Enabled Components	282
4.20.1. Connect to SSL Enabled HiveServer2 using JDBC	283
4.20.2. Connect to SSL Enabled Oozie Server	283
5. Auditing in Hadoop	285
5.1. Using Apache Solr for Ranger Audits	285
5.1.1. Prerequisites	285
5.1.2. Installing Solr	286
5.1.3. Configuring Solr Standalone	286
5.1.4. Configuring SolrCloud	287

5.2. Manually Enabling Audit Settings in Ambari Clusters	289
5.2.1. Manually Updating Ambari Solr Audit Settings	289
5.2.2. Manually Updating Ambari HDFS Audit Settings	290
5.3. Enabling Audit Logging in Non-Ambari Clusters	291
5.4. Managing Auditing in Ranger	293
5.4.1. View Operation Details	293
5.4.2. Access	294
5.4.3. Admin	295
5.4.4. Login Sessions	296
5.4.5. Plugins	297
6. Data Protection: HDFS Encryption	298
6.1. Ranger KMS Administration Guide	298
6.1.1. Installing the Ranger Key Management Service	298
6.1.2. Enable Ranger KMS Audit	306
6.1.3. Enabling SSL for Ranger KMS	309
6.1.4. Install Multiple Ranger KMS	314
6.1.5. Using the Ranger Key Management Service	316
6.1.6. Ranger KMS Properties	320
6.1.7. Troubleshooting Ranger KMS	324
6.2. HDFS "Data at Rest" Encryption	324
6.2.1. HDFS Encryption Overview	324
6.2.2. Configuring and Starting the Ranger Key Management Service (Ranger KMS)	326
6.2.3. Configuring and Using HDFS Data at Rest Encryption	327
6.2.4. Configuring HDP Services for HDFS Encryption	335
6.2.5. Appendix: Creating an HDFS Admin User	342

List of Figures

3.1. Installing Ranger - Main Dashboard View	100
3.2. Installing Ranger - Add Service	101
3.3. Installing Ranger - Choose Service	102
3.4. Installing Ranger - Ranger Requirements	103
3.5. Installing Ranger Assign Masters	104
3.6. Knox Service Manager	164
3.7. Knox Service Edit	164
6.1. HDFS Encryption Components	326

List of Tables

2.1. UNIX Authentication Settings	27
2.2. Active Directory Authentication Settings	28
2.3. Active Directory Custom ranger-admin-site Settings	30
2.4. LDAP Authentication Settings	32
2.5. LDAP Custom ranger-admin-site Settings	34
2.6. Active Directory Authentication Settings	35
2.7. Service Principals	41
2.8. Service Keytab File Names	42
2.9. General core-site.xml, Knox, and Hue	45
2.10. core-site.xml Master Node Settings – Knox Gateway	46
2.11. core-site.xml Master Node Settings – Hue	46
2.12. hdfs-site.xml File Property Settings	48
2.13. yarn-site.xml Property Settings	52
2.14. mapred-site.xml Property Settings	54
2.15. hbase-site.xml Property Settings – HBase Server	55
2.16. hive-site.xml Property Settings	57
2.17. oozie-site.xml Property Settings	58
2.18. webhcat-site.xml Property Settings	59
2.19. Supported Hadoop Services	72
2.20. Apache Service Gateway Directores	73
2.21. gateway-site.xml Configuration Elements	91
3.1. Ranger DB Host	105
3.2. Driver Class Name	106
3.3. Ranger DB Username Settings	106
3.4. JDBC Connect String	106
3.5. DBA Credential Settings	107
3.6. UNIX User Sync Properties	116
3.7. LDAP/AD Common Configs	118
3.8. LDAP/AD User Configs	119
3.9. LDAP/AD Group Configs	121
3.10. UNIX Authentication Settings	123
3.11. LDAP Authentication Settings	124
3.12. AD Settings	128
3.13. LDAP Advanced ranger-ugsync-site Settings	132
3.14. AD Advanced ranger-ugsync-site Settings	133
3.15. Advanced ranger-ugsync-site Settings for LDAP and AD	133
3.16. HDFS Plugin Properties	162
3.17. Hive Plugin Properties	162
3.18. HBase Plugin Properties	163
3.19. Knox Plugin Properties	164
3.20. Knox Configuration Properties	164
3.21. Service Details	170
3.22. Config Properties	170
3.23. Service Details	171
3.24. Config Properties	172
3.25. Service Details	173
3.26. Config Properties	174
3.27. Service Details	175

3.28. Config Properties	175
3.29. Service Details	177
3.30. Config Properties	177
3.31. Service Details	179
3.32. Config Properties	179
3.33. Service Details	181
3.34. Config Properties	181
3.35. Service Details	183
3.36. Config Properties	183
3.37. Policy Details	186
3.38. User and Group Permissions	186
3.39. Policy Details	189
3.40. User and Group Permissions	189
3.41. Policy Details	191
3.42. User and Group Permissions	191
3.43. Policy Details	193
3.44. User and Group Permissions	194
3.45. Policy Details	195
3.46. User and Group Permissions	196
3.47. Policy Details	197
3.48. User and Group Permissions	197
3.49. Policy Details	199
3.50. User and Group Permissions	199
3.51. Knox User and Group Permissions	200
3.52. Policy Details	201
3.53. User and Group Permissions	202
4.1. Components that Support SSL	221
4.2. Configure SSL Data Protection for HDP Components	230
4.3. Configuration Properties in ssl-server.xml	232
5.1. Solr install.properties Values	286
5.2. Solr install.properties Values	288
5.3. Search Criteria	294
5.4. Search Criteria	296
5.5. Search Criteria	296
5.6. Agents Search Criteria	297
6.1. Properties in Advanced dbks-site Menu (dbks-site.xml)	320
6.2. Properties in Advanced kms-env	320
6.3. Properties in Advanced kms-properties (install.properties)	320
6.4. Properties in Advanced kms-site (kms-site.xml)	321
6.5. Properties in Advanced ranger-kms-audit (ranger-kms-audit.xml)	322
6.6. Properties in Advanced ranger-kms-policymgr-ssl	323
6.7. Properties in Advanced ranger-kms-security	323
6.8. Troubleshooting Suggestions	324

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 all the interactions do not 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; and the system 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 must be understood and addressed.

Hortonworks understands the importance of security and governance for every business. To ensure effective protection for our customers, we use 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. Understanding Data Lake Security

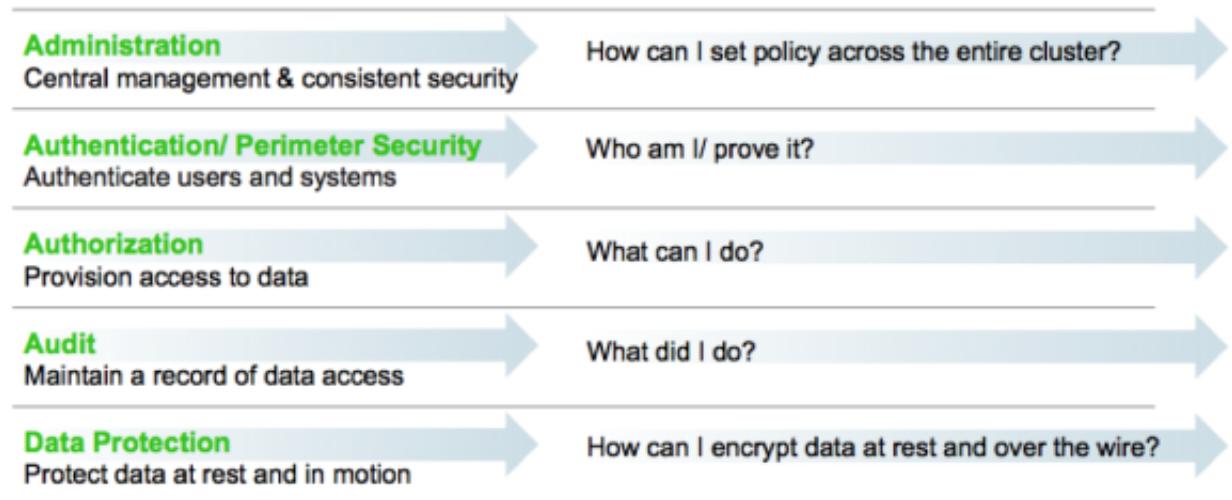
The general consensus in nearly every industry is that data is an essential new driver of competitive advantage. Hadoop plays a critical role in the modern data architecture by providing low-cost, large-scale data storage and processing. The successful Hadoop journey typically starts with data architecture optimization or new advanced analytic applications, which leads to the formation of a Data Lake. As new and existing types of data from sources such as 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 at will, using methods of their own choosing, further increasing the risk of a breach. Any breach of this enterprise-wide data can be catastrophic: privacy violations, regulatory infractions, or the compromise of vital corporate intelligence. To prevent damage to the company's business, customers, finances, and reputation, IT leaders must ensure that their Data Lake meets the same high standards of security as any legacy data environment.

Only as Secure as the Weakest Link

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



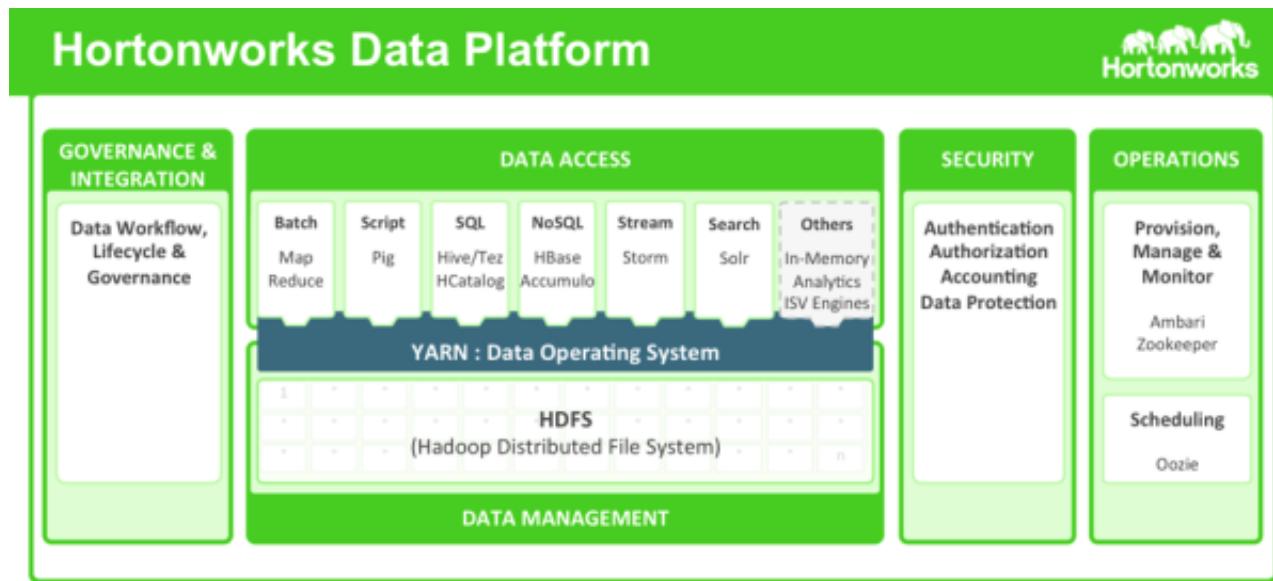
Requirements for Enterprise-Grade Security

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 will become a risk vector built into the very fabric of the company's Big Data environment. In this light, your Hadoop security strategy must address all five pillars, with a consistent implementation approach to ensure their effectiveness.

Needless to say, you can't achieve comprehensive protection across the Hadoop stack by using a hodgepodge 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 where new applications or data engines are always on the horizon in the form of new Open Source projects – a dynamic scenario that can quickly exacerbate any vulnerability.

Hortonworks helps customers maintain the high levels of protection for enterprise data by building centralized security administration and management into the infrastructure

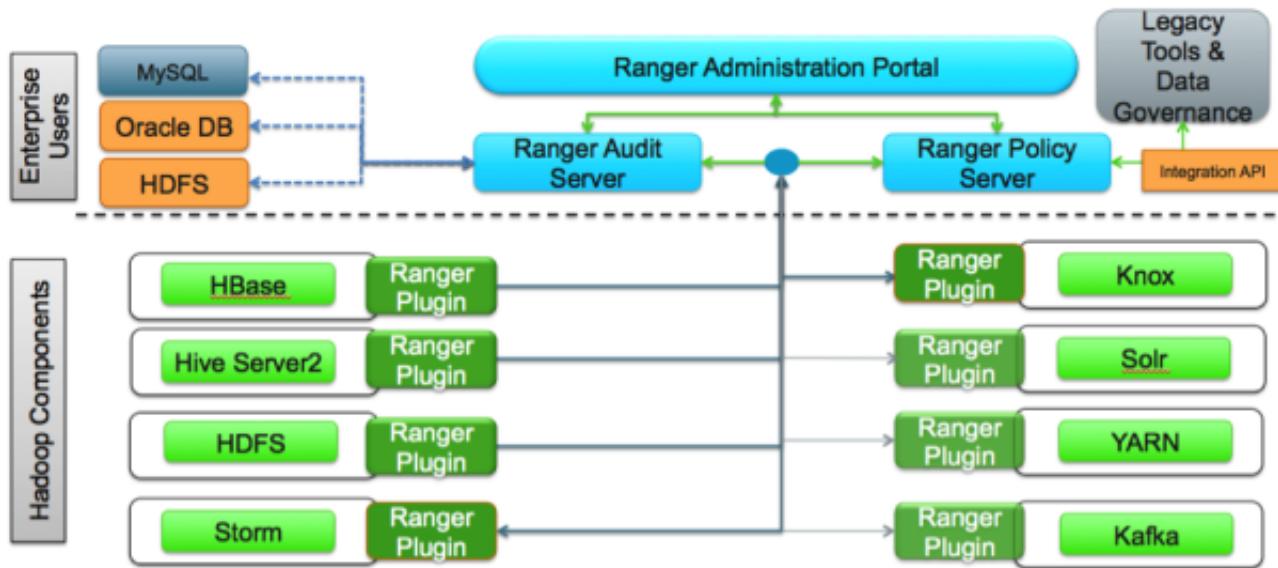
of the Hortonworks Data Platform. HDP provides an enterprise-ready data platform with rich capabilities spanning security, governance, and operations. HDP includes powerful data security functionality that works across component technologies and integrates with preexisting EDW, RDBMS and MPP systems. By implementing security at the platform level, Hortonworks ensures that security is consistently administered to all of the applications across the stack, and simplifies the process of adding or removing Hadoop applications.



The Hortonworks Data Platform

1.2. 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. Users can create and update policies, which are then stored in a policy database. Ranger plugins (light-weight Java programs) are embedded within the processes of each cluster component. For example, the Ranger plugin for Apache Hive is embedded within Hiveserver2.

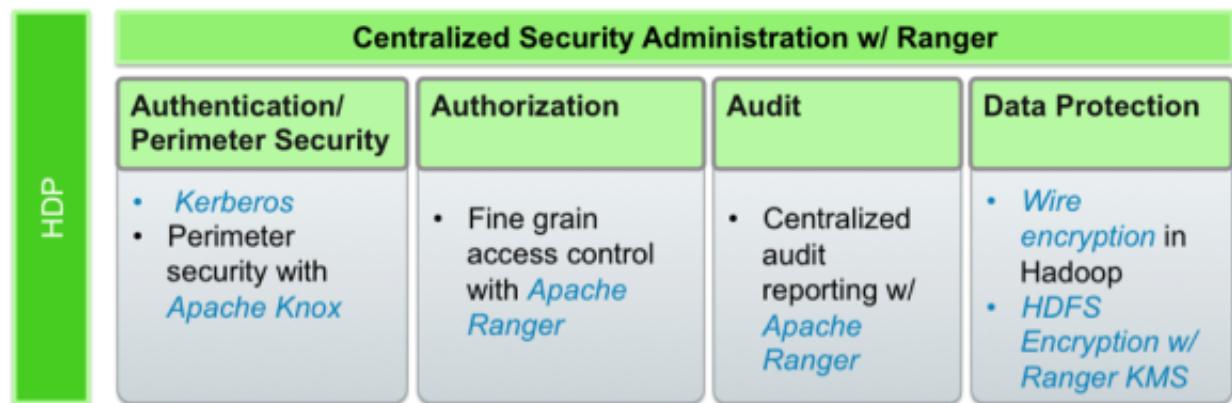


Apache Ranger Architecture

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

1.2.1. Administration

In order to deliver consistent security administration and management, Hadoop administrators require a centralized user interface that can be used 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 Admin Console

1.2.2. 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 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.

Single, simple point of access for a cluster

- Kerberos Encapsulation
- Single Hadoop access point
- REST API hierarchy
- Consolidated API calls
- Multi-cluster support

Central controls ensure consistency across one or more clusters

- Eliminates SSH "edge node"
- Central API management
- Central audit control
- Service level Authorization

Integrated with existing systems to simplify identity maintenance

- SSO Integration – Siteminder and OAM*
- LDAP & AD integration

Apache Knox Features

1.2.3. Authorization

Ranger manages fine-grained access control through a rich user interface that ensures consistent policy administration across Hadoop data access components. Security administrators have the flexibility to define security policies for a database, table and column, or a file, 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 highly 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 is able to enforce centrally administered policies for any method used to access the data lake.

Policy Details :

Policy Name *	Customer_Details_Table	<input checked="" type="radio"/> enabled
Hive Database *	xademo	<input checked="" type="radio"/> include
table *	customer_details	<input checked="" type="radio"/> include
Hive Column *	phone_number plan date	<input checked="" type="radio"/> include
Description	YES	
Audit Logging	<input checked="" type="radio"/>	

User and Group Permissions :

Permissions	Select Group	Select User	Delegate Admin
	Marketing	Mal Inara	<input type="checkbox"/>
<input checked="" type="checkbox"/> select	<input checked="" type="checkbox"/> Marketing	<input checked="" type="checkbox"/> Mal	<input checked="" type="checkbox"/> Add Permissions
<input type="checkbox"/> update	<input checked="" type="checkbox"/> Marketing	<input checked="" type="checkbox"/> Inara	<input type="checkbox"/>
<input type="checkbox"/> Create			
<input type="checkbox"/> Drop			
<input type="checkbox"/> Alter			
<input type="checkbox"/> Index			
<input type="checkbox"/> Lock			
<input type="checkbox"/> All			
<input type="checkbox"/> Select/Deselect All			

Ranger Security Policy Definitions

Ranger provides administrators with deep visibility into the security administration process that is required for auditing purposes. The combination of Ranger's rich user interface with

deep audit visibility makes it highly intuitive to use, enhancing productivity for security administrators.

The screenshot shows the Ranger interface for managing security policies. The top navigation bar includes 'Ranger', 'Access Manager', 'Audit', 'Settings', and a user icon labeled 'admin'. Below the navigation is a breadcrumb trail: 'Service Manager > sandbox_hive Policies'. The main title is 'List of Policies : sandbox_hive'. A search bar at the top right says 'Search for your policy...'. On the far right, there's a green button labeled 'Add New Policy'. The central part of the screen is a table with the following columns: Policy ID, Policy Name, Status, Audit Logging, Groups, Users, and Action. The table contains the following data:

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

Ranger Security Policy Overview

1.2.4. 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 these reasons, 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 called Apache Atlas. Apache Atlas is a set of core foundational governance services that enables enterprises to effectively and efficiently meet their compliance requirements within Hadoop, and also allows integration with the complete enterprise data ecosystem. These services include:

- Search and Lineage for datasets
- Metadata-driven data access control
- Indexed and searchable centralized auditing operational events
- Data lifecycle management – ingestion to disposition
- Metadata interchange with other tools

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

1.2.5. Data Protection

Data protection adds a robust layer of security by making data unreadable in transit over the network or at rest on a disk. HDP fully satisfies enterprise requirements for security and compliance by using transparent data encryption ([TDE](#)) to encrypt data for HDFS files, along with a Ranger-embedded open source Hadoop key management store (KMS). Ranger provides security administrators with the ability 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. With Hortonworks, our customers have the flexibility to leverage an open source key management store (KMS), or use enterprise-wide KMS solutions provided by the partner ecosystem.

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. UI 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](#)
- [Hadoop and Kerberos Principals](#)
- [Installing and Configuring the KDC](#)
- [Enabling Kerberos Security in Ambari](#)

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.

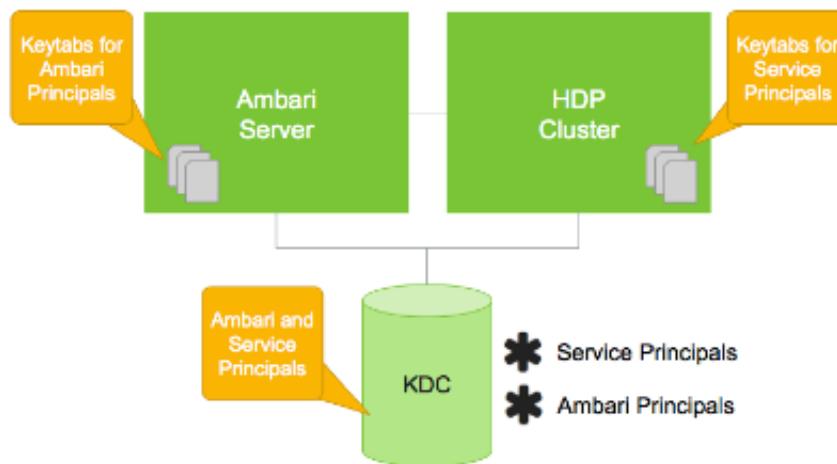
Because cluster resources (hosts or services) cannot provide a password each time to decrypt the TGT, they use a special file, called a **keytab**, which contains the resource principal's authentication credentials. The set of hosts, users, and services over which the Kerberos server has control is called 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. Hadoop and 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.



Principals and Keytabs

Principal and Keytab Naming Conventions

Asset	Convention	Example
Principals	\$service_component_name/\$FQDN@EXAMPLE.COM	nn/c6401.ambari.apache.org@EXAMPLE.COM
Keytabs	\$service_component_abbreviation.service.keytab	/etc/security/keytabs/nn.service.keytab

Note

In addition to the Hadoop **Service Principals**, Ambari itself also requires a set of **Ambari Principals** to perform service "smoke" checks and alert health checks.

Keytab files for the Ambari, or headless, principals reside on each cluster host, just as keytab files for the service principals.

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.

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](#)
- [Use an Existing Active Directory](#)
- [Use Manual Kerberos Setup](#)

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 in Ambari](#).

2.1.3.2. Use an Existing Active Directory

To use an existing Microsoft Active Directory 2008 and later domain 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 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 in Ambari](#).

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 in Ambari](#).

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 11

```
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 11

```
chkconfig rckrb5kdc on
```

```
chkconfig rckadmind on
```

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 11

```
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](#)
- [Running the Kerberos Wizard](#)

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.

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 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>

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

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. 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. (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. (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.**

- c. (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
```

- d. (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.**
- e. (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.
- f. (Optional) If your KDC has a password policy, expand the **Advanced kerberos-env** section and adjust the Password options.
- g. (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 Principal Name** 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.
- h. (Optional) If you need to customize the attributes for the principals Ambari will create, when using Active Directory, see the [Customizing the Attribute Template](#) for more information. When using MIT KDC, you can pass **Principal Attribute** options in the **Advanced kerberos-env** section. For example, you can set options related to pre-auth or max. renew life by passing:

```
-requires_pcreauth -maxrenewlife "7 days"
```

6. Proceed with the install.
7. 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.
8. 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**.

9. Confirm your configuration. You can optionally download a CSV file of the principals and keytabs that Ambari will automatically create.
10. Click **Next** to start the process.

11 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.



Note

If your cluster includes Storm, after enabling Kerberos, you must also [set up Ambari for Kerberos](#) for Storm Service Summary information to be displayed in Ambari Web. Otherwise, you will see n/a for Storm information such as Slots, Tasks, Executors and Topologies.

12. Exit the wizard when complete.

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.

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](#), 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 and continue the failure but removal of principals from the KDC will not be performed.

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	[[MD5 hash of the \$normalized_principal]]
\$principal_instance	c6401.ambari.apache.org
\$realm	EXAMPLE.COM
\$password	[[password]]

2.2. Configuring Ambari Authentication with LDAP or AD

2.2.1. 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 [section 3.1.4 - Specific Set of Users and Groups](#).

2.2.1.1. 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>	true or false	If true, use SSL when connecting to the LDAP or AD server.
<code>authentication.ldap.usernameAttribute</code>	[<code>uid</code>]	The attribute for username. Example: <code>uid</code>

Property	Values	Description
authentication.ldap.baseDn	[Distinguished Name]	The root Distinguished Name to search in the directory for users. Example: ou=people,dc=hadoop,dc=apache,dc=org
authentication.ldap.referral	[Referral method]	Determines if LDAP referrals should be followed, or ignored.
authentication.ldap.bindAnonymous	[Boolean] false	If true, bind to the LDAP or AD server anonymously
authentication.ldap.managerDn	[Full Distinguished Name]	If Bind anonymous is set to false, the Distinguished Name ("DN") for the manager. Example: uid=hdfs,ou=people,dc=hadoop,dc=apache,dc=org
authentication.ldap.managerPassword	[Password]	If Bind anonymous is set to false, the password for the manager
authentication.ldap.userObjectClass	[LDAP Object Class]	The object class that is used for users. Example: organizationalPerson
authentication.ldap.groupObjectClass	[LDAP Object Class]	The object class that is used for groups. Example: groupOfUniqueNames
authentication.ldap.groupMembershipAttribute	[LDAP Attribute]	The attribute for group membership. Example: uniqueMember
authentication.ldap.groupNameAttribute	[LDAP attribute]	The attribute for group name.

2.2.1.2. 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](#).

2.2.1.2.1. 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

2.2.1.3. 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 [section 3.1.4 - Specific Set of Users and Groups](#).

2.2.1.4. 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.

2.2.1.5. 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.

2.2.1.6. 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.

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

2.2.2.1. UNIX Authentication Settings

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

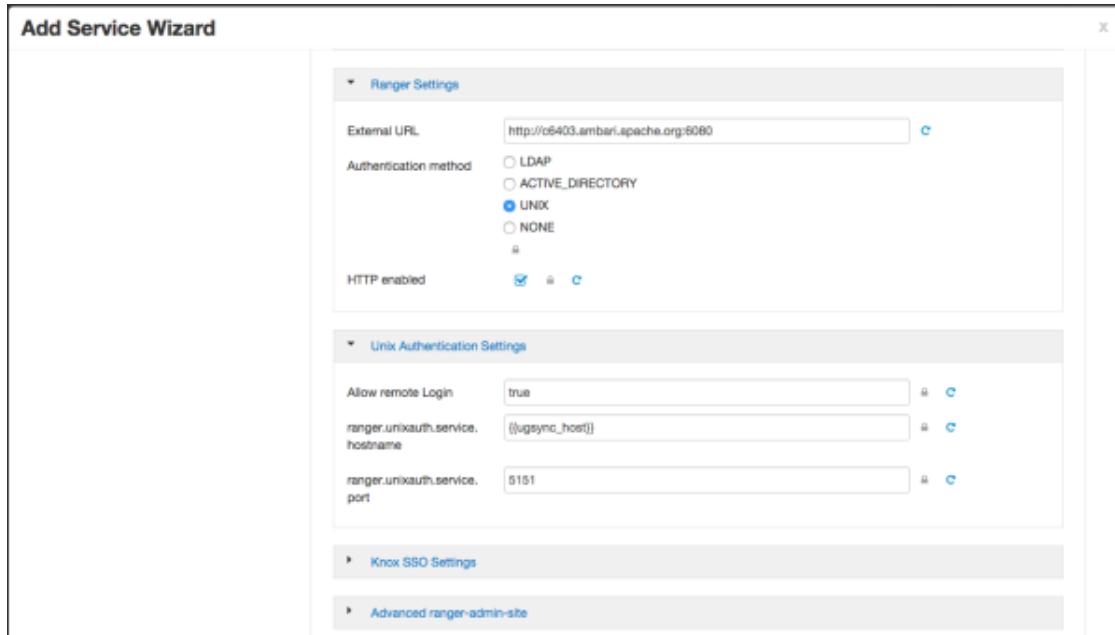


Table 2.1. UNIX Authentication Settings

Configuration Property
Allow remote Login
ranger.unixauth.service.hostname
ranger.unixauth.service.port

2.2.2.2. 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 Configuring Usersync Settings.

2.2.2.2.1. AD Settings

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

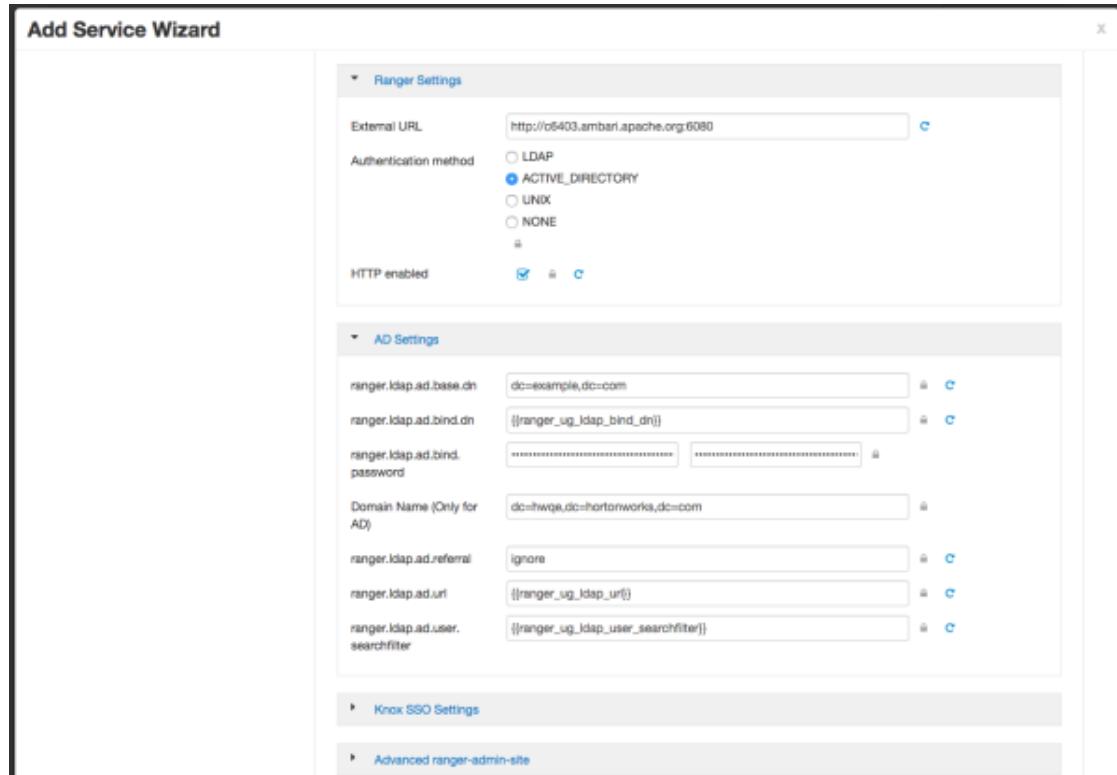


Table 2.2. Active Directory Authentication Settings

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.

2.2.2.2.2. 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**.

The screenshot shows a configuration interface for the Hortonworks Data Platform. The left side features a tree view of settings. At the top level, 'AD Settings' is expanded, showing two properties: 'ranger.ldap.ad.domain' set to 'localhost' and 'ranger.ldap.ad.url' set to 'ldap://ad.xasecure.net:389'. Below this, several other sections are listed: 'LDAP Settings', 'Advanced ranger-admin-site' (which is currently selected), 'Advanced ranger-env', 'Advanced ranger-ugsync-site', 'Custom admin-properties', 'Custom ranger-admin-site' (which is expanded), and 'Custom ranger-site', 'Custom ranger-ugsync-site', 'Custom usersync-properties'. The 'Add Property ...' button under 'Custom ranger-admin-site' is highlighted with a red box.

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 screenshot shows a modal dialog titled "Add Property". The "Type" field contains "ranger-site.xml". The "Key" field contains "ranger.ldap.ad.base_dn". The "Value" field contains "dc=example,dc=com". At the bottom right are "Cancel" and "Add" buttons.

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.

The screenshot shows a configuration interface for "Custom ranger-site". It lists four properties with their values and edit icons:

- 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

Each property row has a green plus sign and a red minus sign to its right.

Table 2.3. 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.

2.2.2.3. 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 Configuring Usersync Settings.

2.2.2.3.1. LDAP Settings

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

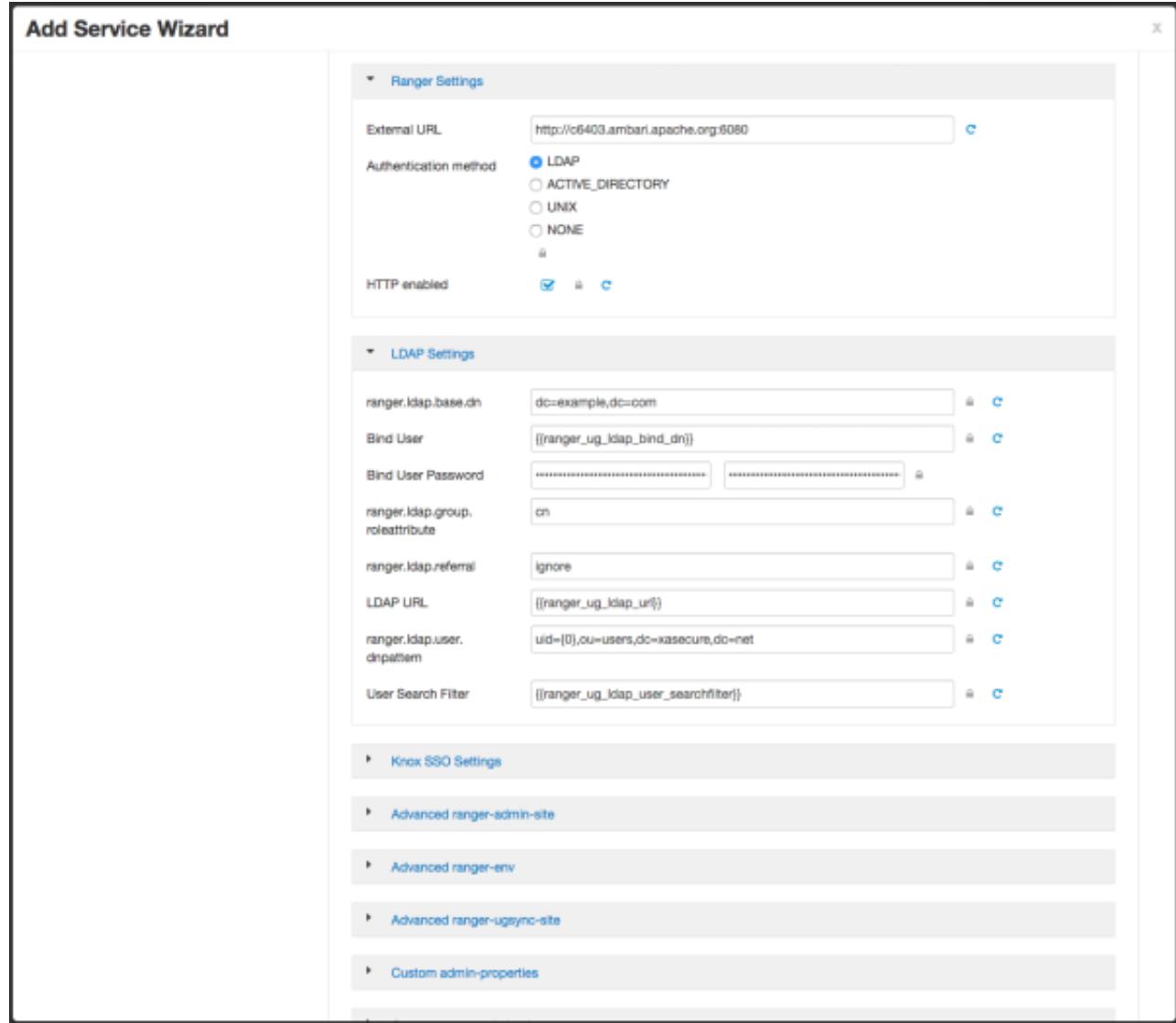


Table 2.4. LDAP Authentication Settings

Configuration Property Name	Description	Default Value	Example Value	Required?
ranger.ldap.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.
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.

2.2.2.3.2. 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 hierarchical configuration menu. At the top level, there are sections for 'AD Settings' and 'LDAP Settings'. Under 'AD Settings', two properties are listed: 'ranger.ldap.ad.domain' set to 'localhost' and 'ranger.ldap.ad.url' set to 'ldap://ad.xassecure.net:389'. Below these are sections for 'Advanced ranger-admin-site', 'Advanced ranger-env', 'Advanced ranger-ugsync-site', and 'Custom admin-properties'. A red box highlights the 'Custom ranger-admin-site' section, which contains a sub-item 'Add Property ...'. Under 'Custom ranger-admin-site', there are three more items: 'Custom ranger-site', 'Custom ranger-ugsync-site', and 'Custom usersync-properties'.

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 screenshot shows a 'Add Property' dialog box. The 'Type' field contains 'ranger-admin-site.xml'. The 'Key' field contains 'ranger.ldap.base_dn'. The 'Value' field contains 'dc=example,dc=com'. At the bottom right, there are 'Cancel' and 'Add' buttons, with 'Add' being highlighted by a red box.

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

The screenshot shows the 'Custom ranger-site' configuration page. It lists four properties with their values and edit icons:

- 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.referral: follow

An 'Add Property ...' link is located at the bottom left.

Table 2.5. 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.

2.2.2.3.3. Advanced ranger-admin-site Settings

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

Table 2.6. 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>

2.2.3. 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
```

2.2.3.1. Reset Encryption

There may be situations in which you want to:

- [Remove encryption entirely](#)
- [Change the current master key](#), 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.

2.2.3.2. 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.encrypted=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.

2.2.3.3. 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
```

2.3. Setting Up Kerberos Authentication for Non-Ambari Clusters

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

- [Preparing Kerberos](#)
- [Configuring HDP](#)
- [Configuring Secure HBase and ZooKeeper](#)
- [Configuring Hue](#)
- [Setting up One-Way Trust with Active Directory](#)
- [Configuring Proxy Users](#)

2.3.1. Preparing Kerberos

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

2.3.1.1. 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.

2.3.1.2. 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.

2.3.1.3. 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
```

2.3.1.4. 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 2.7. Service Principals

Service	Component	Mandatory Principal Name
HDFS	NameNode	nn/\$FQDN
HDFS	NameNode HTTP	HTTP/\$FQDN
HDFS	SecondaryNameNode	nn/\$FQDN
HDFS	SecondaryNameNode HTTP	HTTP/\$FQDN
HDFS	DataNode	dn/\$FQDN
MR2	History Server	jhs/\$FQDN
MR2	History Server HTTP	HTTP/\$FQDN
YARN	ResourceManager	rm/\$FQDN
YARN	NodeManager	nm/\$FQDN
Oozie	Oozie Server	oozie/\$FQDN
Oozie	Oozie HTTP	HTTP/\$FQDN
Hive	Hive Metastore	hive/\$FQDN
	HiveServer2	
Hive	WebHCat	HTTP/\$FQDN
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
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 2.8. 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		
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
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.

2.3.2. 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](#)
- [Adding Security Information to Configuration Files](#)
- [Configuring Secure HBase and ZooKeeper](#)
- [Configuring Hue](#)



Note

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

2.3.2.1. 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`.

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
```

2.3.2.2. 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
```

2.3.2.3. 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
```

2.3.2.3.1. core-site.xml

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

Table 2.9. General core-site.xml, Knox, and Hue

Property Name	Property Value	Description
<code>hadoop.security.authentication</code>	kerberos	Set the authentication type for the cluster. Valid values are: simple or kerberos.
<code>hadoop.rpc.protection</code>	authentication; integrity; privacy	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>	true	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/.*/	The mapping from Kerberos principal names to local OS user names. See Creating Mappings Between Principals

Property Name	Property Value	Description
	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	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>
```

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

Table 2.10. core-site.xml Master Node Settings – Knox Gateway

Property Name	Property Value	Description
<code>hadoop.proxyuser.knox.groups</code>	<code>users</code>	Grants proxy privileges for Knox user.
<code>hadoop.proxyuser.knox.hosts</code>	<code>\$knox_host_FQDN</code>	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 2.11. core-site.xml Master Node Settings – Hue

Property Name	Property Value	Description
<code>hue.kerberos.principal.shortname</code>	<code>hue</code>	Group to which all the Hue users belong. Use the wild card character to select multiple groups, for example <code>cli*</code> .
<code>hadoop.proxyuser.hue.groups</code>	<code>*</code>	Group to which all the Hue users belong. Use the wild card character to select multiple groups, for example <code>cli*</code> .

Property Name	Property Value	Description
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>

```

2.3.2.3.1.1. 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.

2.3.2.3.2. hdfs-site.xml

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

Table 2.12. 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.
<code>dfs.block.access.token.enable</code>	<code>true</code>	If true, access tokens are used as capabilities for accessing DataNodes. If false, no access tokens are checked on accessing DataNodes.
<code>dfs.namenode.kerberos.principal</code>	<code>nn/_HOST@EXAMPLE.COM</code>	Kerberos principal name for the NameNode.
<code>dfs.secondary.namenode.kerberos.principal</code>	<code>nn/_HOST@EXAMPLE.COM</code>	Kerberos principal name for the secondary NameNode.
<code>dfs.web.authentication.kerberos.principal</code>	<code>HTTP/_HOST@EXAMPLE.COM</code>	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.
<code>dfs.web.authentication.kerberos.keytab</code>	<code>/etc/security/keytabs/spnego.service.keytab</code>	The Kerberos keytab file with the credentials for the HTTP Kerberos principal used by Hadoop-Auth in the HTTP endpoint.
<code>dfs.datanode.kerberos.principal</code>	<code>dn/_HOST@EXAMPLE.COM</code>	The Kerberos principal that the DataNode runs as. "_HOST" is replaced by the real host name.
<code>dfs.namenode.keytab.file</code>	<code>/etc/security/keytabs/nn.service.keytab</code>	Combined keytab file containing the NameNode service and host principals.
<code>dfs.secondary.namenode.keytab.file</code>	<code>/etc/security/keytabs/nn.service.keytab</code>	Combined keytab file containing the NameNode service and host principals. <question?>
<code>dfs.datanode.keytab.file</code>	<code>/etc/security/keytabs/dn.service.keytab</code>	The filename of the keytab file for the DataNode.
<code>dfs.https.port</code>	<code>50470</code>	The HTTPS port to which the NameNode binds.
<code>dfs.namenode.https-address</code>	<code>Example: ip-10-111-59-170.ec2.internal:50470</code>	The HTTPS address to which the NameNode binds.
<code>dfs.datanode.data.dir.perm</code>	<code>750</code>	The permissions that must be set on the <code>dfs.data.dir</code> directories. The DataNode will not come up if all existing <code>dfs.data.dir</code> directories do not have this setting. If the directories do not exist, they will be created with this permission.
<code>dfs.cluster.administrators</code>	<code>hdfs</code>	ACL for who all can view the default servlets in the HDFS.

Property Name	Property Value	Description
dfs.namenode.kerberos.internal.spnego.principal	<code> \${dfs.web.authentication.kerberos.principal}</code>	
dfs.secondary.namenode.kerberos.internal.spnego.principal	<code> \${dfs.web.authentication.kerberos.principal}</code>	

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
```

2.3.2.3.3. yarn-site.xml

You must add the following information to the `yarn-site.xml` file on every host in your cluster:

Table 2.13. `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>

```

2.3.2.3.4. mapred-site.xml

You must add the following information to the `mapred-site.xml` file on every host in your cluster:

Table 2.14. mapred-site.xml Property Settings

Property Name	Property Value	Description
<code>mapreduce.jobhistory.keytab</code>	<code>/etc/security/keytabs/jhs.service.keytab</code>	Kerberos keytab file for the MapReduce JobHistory Server.
<code>mapreduce.jobhistory.principal</code>	<code>jhs/_HOST@TODO-KERBEROS-DOMAIN</code>	Kerberos principal name for the MapReduce JobHistory Server.
<code>mapreduce.jobhistory.webapp.address</code>	<code>TODO-JOBHISTORYNODE-HOSTNAME:19888</code>	MapReduce JobHistory Server Web UI host:port
<code>mapreduce.jobhistory.webapp.https.address</code>	<code>TODO-JOBHISTORYNODE-HOSTNAME:19889</code>	MapReduce JobHistory Server HTTPS Web UI host:port
<code>mapreduce.jobhistory.webapp.spnego-keytab-file</code>	<code>/etc/security/keytabs/spnego.service.keytab</code>	Kerberos keytab file for the spnego service.
<code>mapreduce.jobhistory.webapp.spnego-principal</code>	<code>HTTP/_HOST@TODO-KERBEROS-DOMAIN</code>	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>
```

2.3.2.3.5. 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. There are no default values; the following are only examples:

Table 2.15. hbase-site.xml Property Settings – HBase Server

Property Name	Property Value	Description
<code>hbase.master.keytab.file</code>	<code>/etc/security/keytabs/hm.service.keytab</code>	The keytab for the HMaster service principal.
<code>hbase.master.kerberos.principal</code>	<code>hm/_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/rs.service.keytab</code>	The keytab for the HRegionServer service principal.
<code>hbase.regionserver.kerberos.principal</code>	<code>rs/_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>	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>		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 <code>HTableDescriptor</code> .
<code>hbase.coprocessor.master.classes</code>		Comma-separated list of <code>org.apache.hadoop.hbase.coprocessor.MasterObserver</code> coprocessors that are loaded by default on the active HMaster process. For any implemented coprocessor methods, the listed classes will be called in order. After implementing your own <code>MasterObserver</code> , just put it in HBase's classpath and add the fully qualified class name here.

Following is the XML for these entries:

```
<property>
  <name>hbase.master.keytab.file</name>
  <value>/etc/security/keytabs/hm.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/rs.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>rs/_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></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></value>
    <description>A comma-separated list of
org.apache.hadoop.hbase.coprocessor.MasterObserver coprocessors that
are loaded by default on the active HMaster process. For any implemented
coprocessor methods, the listed classes will be called in order.
After implementing your own MasterObserver, just put it in HBase's
classpath and add the fully qualified class name here.
    </description>
</property>

```

2.3.2.3.6. hive-site.xml

Hive Metastore supports Kerberos authentication for Thrift clients only. HiveServer does not support Kerberos authentication for any clients.

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

Table 2.16. `hive-site.xml` Property Settings

Property Name	Property Value	Description
<code>hive.metastore.sasl.enabled</code>	true	If true, the Metastore Thrift interface will be secured with SASL and clients must authenticate with Kerberos.
<code>hive.metastore.kerberos.keytab.file</code>	<code>/etc/security/keytabs/hive.service.keytab</code>	The keytab for the Metastore Thrift service principal.
<code>hive.metastore.kerberos.principal</code>	<code>hive/_HOST@EXAMPLE.COM</code>	The service principal for the Metastore Thrift server. If <code>_HOST</code> is used as the hostname portion, it will be replaced with the actual hostname of the running instance.
<code>hive.metastore.cache.pinobjtypes</code>	<code>Table,Database,Type,FieldSchema,Order</code>	Comma-separated Metastore object types that should be pinned in the cache.

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>

<property>
    <name>hive.metastore.cache.pinobjtypes</name>
    <value>Table,Database,Type,FieldSchema,Order</value>
    <description>List of comma separated metastore object types that should be pinned in the cache</description>
</property>

```

2.3.2.3.7. oozie-site.xml

To the `oozie-site.xml` file, add the following information:

Table 2.17. 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/_HOST@EXAMPLE.COM	oozie/_HOST@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]h@.*EXAMPLE.COM)s/.*/hbase/ RULE:[2:\$1@\$0]([rs]r@.*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.

Property Name	Property Value	Description
oozie.service.ProxyUserService.proxyuser.knox.hosts	\$knox_host_FQDN	Identifies the Knox Gateway. Note only required when using a Knox Gateway.

2.3.2.3.8. webhcat-site.xml

To the `webhcat-site.xml` file, add the following information:

Table 2.18. 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.

2.3.2.3.9. 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 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 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
```

2.3.2.4. Configuring Secure HBase and ZooKeeper

Use the following instructions to set up secure HBase and ZooKeeper:

1. [Configure HBase Master](#)
2. [Create JAAS configuration files](#)
3. [Start HBase and ZooKeeper services](#)
4. [Configure secure client side access for HBase](#)
5. [Optional: Configure client-side operation for secure operation - Thrift Gateway](#)
6. [Optional: Configure client-side operation for secure operation - REST Gateway](#)
7. [Configure HBase for Access Control Lists \(ACL\)](#)

2.3.2.4.1. 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.

2.3.2.4.2. 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 your HBase Master host machine, create the `hbase-server.jaas` file under the `/etc/hbase/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/hbase.service.keytab"  
    principal="hbase/$HBase.Master.hostname";  
};
```

- On each of your RegionServer host machine, create the `regionserver.jaas` file under the `/etc/hbase/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/hbase.service.keytab"  
    principal="hbase/$RegionServer.hostname";  
};
```

- 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/regionserver.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`.

2.3.2.4.3. 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
```

2.3.2.4.4. 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>
```

2.3.2.4.5. 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.

2.3.2.4.6. 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.

2.3.2.4.7. 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 [here](#) 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').

2.3.2.5. Configuring Hue

To enable Hue to work with a HDP cluster configured for Kerberos, make the following changes to Hue and Kerberos.

1. Configure Kerberos as described in Setting Up Security for Manual Installs.

2. Create a principal for the Hue Server.

```
addprinc -randkey hue/$FQDN@EXAMPLE.COM
```

where `$FQDN` is the hostname of the Hue Server and EXAMPLE.COM is the Hadoop realm.

3. Generate a keytab for the Hue principal.

```
xst -k hue.service.keytab hue/$FQDN@EXAMPLE.COM
```

4. Place the keytab file on the Hue Server. Set the permissions and ownership of the keytab file.

```
/etc/security/keytabs/hue.service.keytab
chown hue:hadoop /etc/security/keytabs/hue.service.keytab
chmod 600 /etc/security/keytabs/hue.service.keytab
```

5. Confirm the keytab is accessible by testing kinit.

```
su - hue kinit -k -t /etc/security/keytabs/hue.service.keytab hue/
$FQDN@EXAMPLE.COM
```

6. 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 the kinit based on the OS.

```
# Path to kinit
# For RHEL/CentOS 6.x, kinit_path is /usr/bin/kinit
kinit_path=/usr/kerberos/bin/kinit
```

8. Set security_enabled=true for every component in hue.ini.

```
[[hdfs_clusters]], [[yarn_clusters]], [[liboozie]], [[hcatalog]]
```

9. Save the hue.ini file.

10. Restart Hue:

```
# /etc/init.d/hue start
```

2.3.3. 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.

2.3.3.1. 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.

2.3.3.2. 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')

2.3.4. 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.](#)

2.4. Perimeter Security with Apache Knox

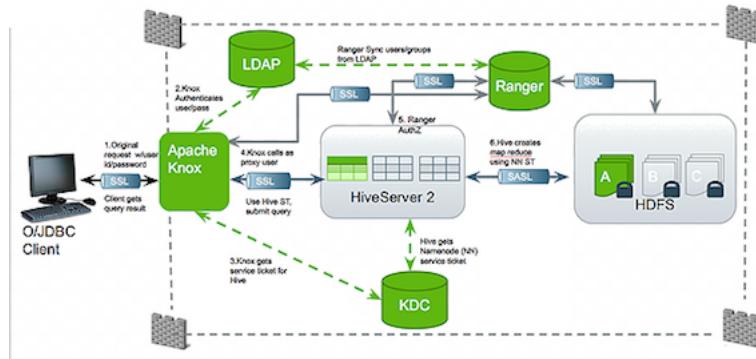
2.4.1. Apache Knox Gateway Overview

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.

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

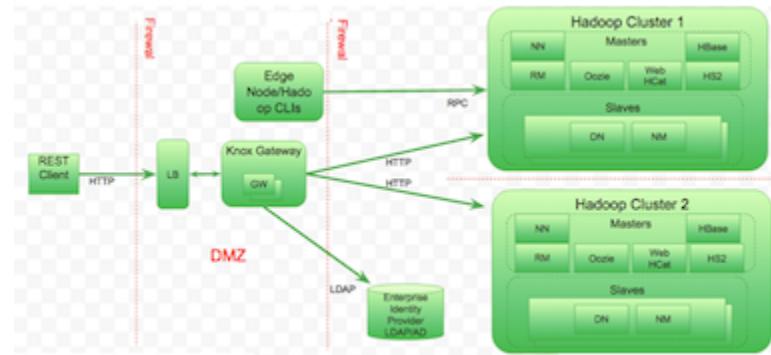
Knox can be used with both unsecured Hadoop clusters, and Kerberos secured clusters. In an enterprise solution that employs Kerberos secured clusters, the Apache Knox Gateway provides an enterprise security solution that:

- 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

2.4.1.1. 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

2.4.1.2. Supported Hadoop Services

Apache Knox Gateway supports the following Hadoop services versions in both Kerberized and Non-Kerberized clusters:

Table 2.19. Supported Hadoop Services

Service	Version
YARN	2.6.0
WebHDFS	2.6.0
WebHCat/Templeton	0.13.0
Oozie	4.1.0
HBase/Stargate	0.98.4
Hive (via WebHCat)	0.14.0
Hive (via JDBC)	0.14.0

2.4.1.3. 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:

- <http://knox.apache.org/books/knox-0-5-0/knox-0-5-0.html#Gateway+Samples>

2.4.2. 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
- Customize the Gateway Port and Path
- Manage the Master Secret

- Manually Redeploy Cluster Topologies
- Start and stop the Knox Gateway

2.4.2.1. 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 2.20. Apache Service Gateway Directores

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 informaiton 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 incubaiton at Apache.

- *SystemDrive/hadoop/logs knox*

This contains the output files from the Knox Gateway.

2.4.2.2. Customize the Gateway Port and Path

The Knox Gateway properties effect the URL used by the external clients to access the internal cluster. By default the port is set to 8443 and the context path is gateway.

To change the context path or port:

1. Edit `gateway-site.xml` and modify the following properties:

- `propertyname`
- `gateway.port namevalue`
- `gateway.portvalue`

where:

- `$gateway_port` is the HTTP port for the gateway (default port=8443)
- `$gateway` is the context path in the external URL (preconfigured value=gateway). For example, `https://knox.hortonworks.com:8443/hadoop/`, where `hadoop` is the context path.

2. Restart the gateway:

```
cd $gateway/bin/gateway.sh stop bin/gateway.sh start
```

The gateway loads the new configuration on startup.

2.4.2.3. 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 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 dateway 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.

2.4.2.4. 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 `$gateway/conf/topologies` directory and automatically reddeploys the cluster if any descriptor changes or a new one is added. (The corresponding deployment is in `$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).

2.4.2.5. 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.

2.4.3. 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](#)
- [Example Service Definitions](#)
- [Validating Service Connectivity](#)
- [Adding a New Service to the Knox Gateway](#)

2.4.3.1. 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 either WEBHDFS, WEBHCAT, WEBHBASE, OOZIE, HIVE, NAMENODE, or JOBTRACKER.
- <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.

2.4.3.2. 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>NAMENODE</role>
    <url>hdfs://namenode-host:8020</url>
</service>

<service>
    <role>JOBTRACKER</role>
    <url>rpc://jobtracker-host:8050</url>
</service>

<service>
    <role>RESOURCEMANAGER</role>
    <url>http://red3:8088/ws</url>
</service>

<service>
    <role>WEBHDFS</role>
    <url>http://localhost:50070/webhdfs</url>
</service>

<service>
    <role>WEBHCAT</role>
    <url>http://webcat-host:50111/templeton</url>
</service>

<service>
    <role>OOZIE</role>
    <url>http://oozie-host:11000/oozie</url>
</service>

<service>
    <role>WEBHBASE</role>
    <url>http://webhbase-host:60080</url>
</service>

<service>
    <role>HIVE</role>
    <url>http://hive-host:10001/cliservice</url>
</service>
```

2.4.3.3. 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:50070/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/ $webhcatt_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
```

2.4.3.4. 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.

2.4.3.4.1. 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
```

2.4.3.4.2. 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.

2.4.4. 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:

- [Setting up LDAP Authentication](#)
- [LDAP Authentication Caching](#)
- [Example Active Directory Configuration](#)
- [Example OpenLDAP Configuration](#)
- [Testing an LDAP Provider](#)
- [Setting up HTTP Header Authentication for Federation/SSO](#)
- [Example SiteMinder Configuration](#)

- [Testing HTTP Header Tokens](#)
- [Setting up 2-Way SSL Authentication](#)



Note

For information on how to configure an identity assertion provider, see [Configuring Identity Assertion](#).

2.4.4.1. 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.

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>
```

```
</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"
```

2.4.4.2. 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.shiro.cache.ehcache.EhCacheManager</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
```

2.4.4.3. 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>
        </provider>
    </gateway>
    <!-- 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>
    </param>
```

```
        <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://{{webhcatt_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>
```

2.4.4.4. Example OpenLDAP Configuration

```

<provider>
    <role>authentication</role>
    <name>ShiroProvider</name>
    <enabled>true</enabled>
    <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>mainLdapRealm.contextFactory</name>
        <value>$ldapContextFactory</value>
    </param>
```

```
</provider>
```

2.4.4.5. 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**.

2.4.4.6. Setting Up HTTP Header Authentication for Federation_SSO

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 0.6.0 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>preathe.ip.addresses</name>
    <value>$trusted_ip</value>
</param>
<param>
    <name>preathe.custom.header</name>
    <value>$user_field</value>
</param>
<param>
    <name>preathe.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.

2.4.4.7. 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>

```

2.4.4.8. 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**

2.4.4.9. 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 2.21. 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 by the client should be automatically trusted.	TRUE/FALSE	FALSE

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 refuse the connection request.

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 2.1 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 Security](#) in the [Ambari Security Guide](#).

3.1.2. Installation Prerequisites

Before you install Ranger, make sure your cluster meets the following requirements:

- It is recommended that you store audits in both HDFS and Solr, so you should [install Apache Solr](#).
- To ensure that LDAP/AD group level authorization is enforced in Hadoop, you should [set up Hadoop group mapping](#) for LDAP.
- A MySQL, Oracle, PostgreSQL, MS SQL, or SQL Anywhere 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](#)
 - [Configuring PostgreSQL for Ranger](#)
 - [Configuring Oracle for Ranger](#)
- 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:

- [Configuring Hadoop Group Mapping for LDAP/AD Using SSSD \(Recommended\)](#)
- [Configure Hadoop Group Mapping in core-site.xml](#)
- [Manually Create the Users and Groups in the Linux Environment](#)

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/dc=hadoop,dc=apache,dc=org</value>
</property>

<property>
<name>hadoop.security.group.mapping.ldap.url</name>
<value>ldap://localhost:389/dc=hadoop,dc=apache,dc=org</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>(&(|(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 MySQL for Ranger

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-jar.jar
```

3.1.2.3. Configuring PostgreSQL for Ranger

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.jar
```

6. Run the following command:

```
export HADOOP_CLASSPATH=${HADOOP_CLASSPATH}: ${JAVA_JDBC_LIBS}:/connector_jar
path
```

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 pg_hba.conf.

```
# 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
```

3.1.2.4. Configuring Oracle for Ranger

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:

```
cpojdbc7.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
```

3.1.3. Ranger Installation

To install Ranger using Ambari:

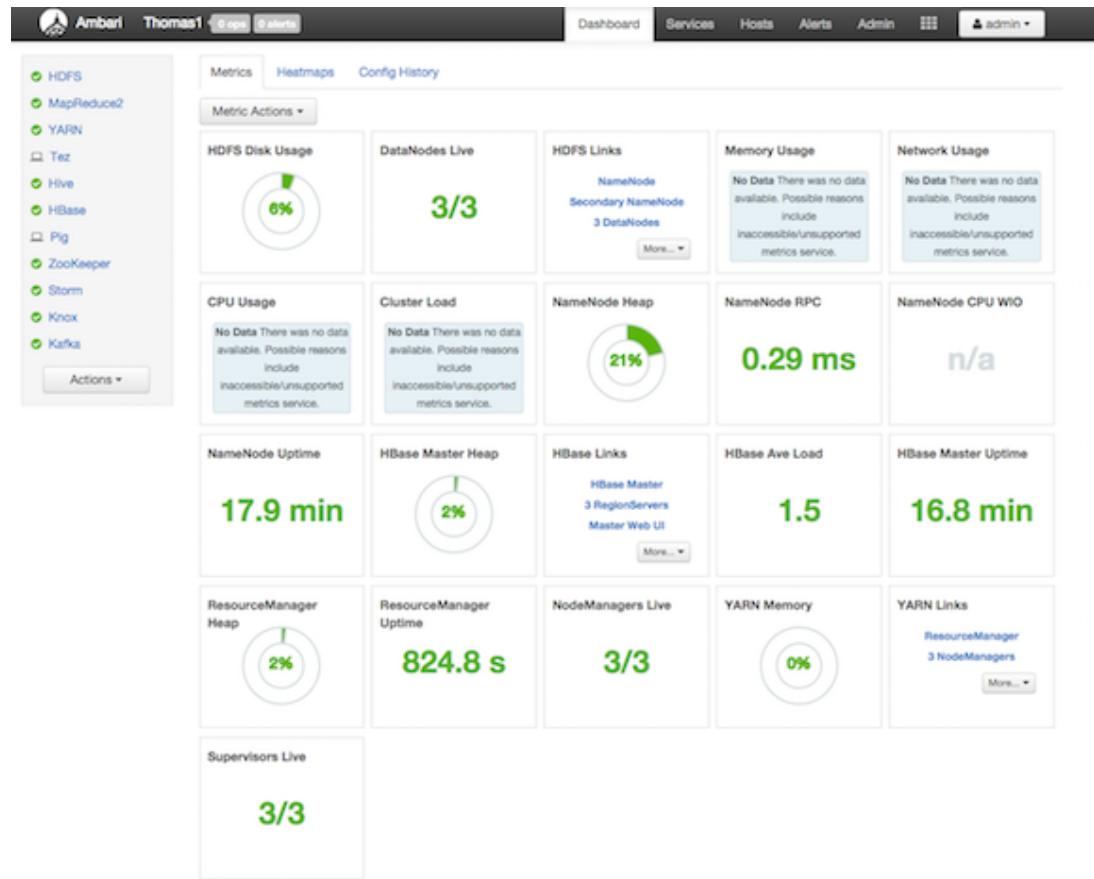
1. Start the Installation
2. Customize Services
3. Complete the Installation

Related Topics

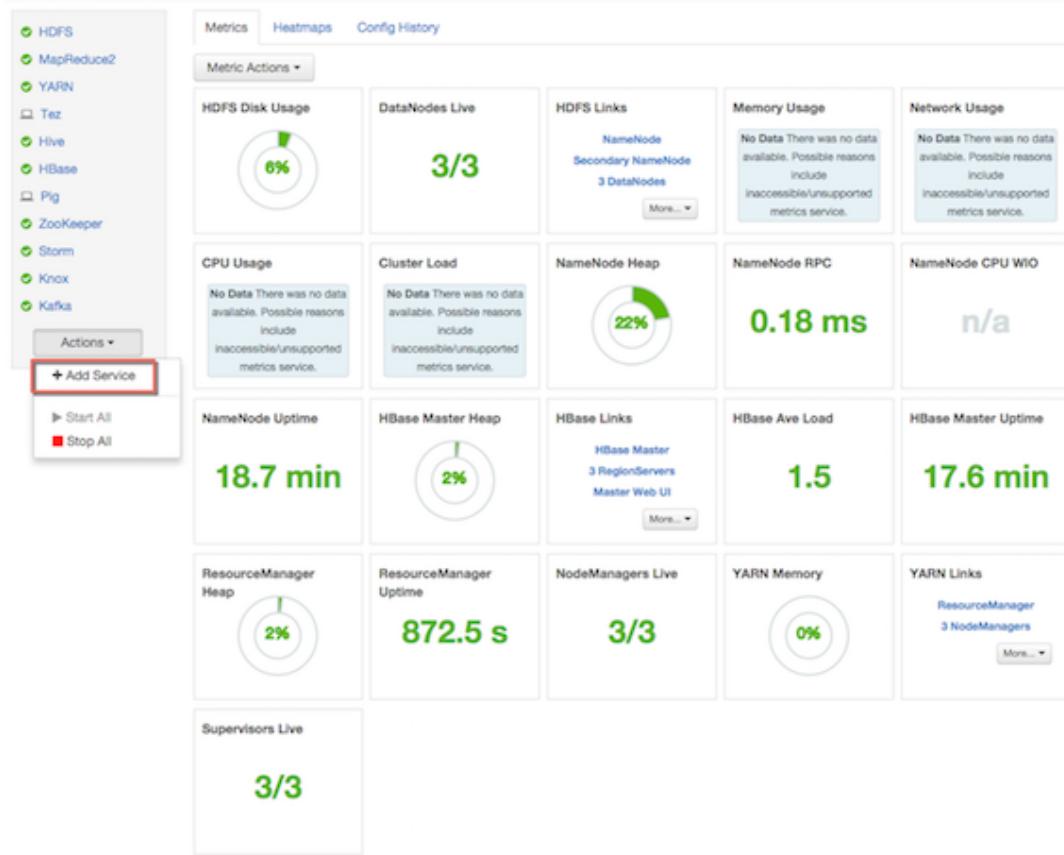
- [Setting up Database Users Without Sharing DBA Credentials](#)
- [Updating Ranger Admin Passwords](#)

3.1.3.1. Start the Installation

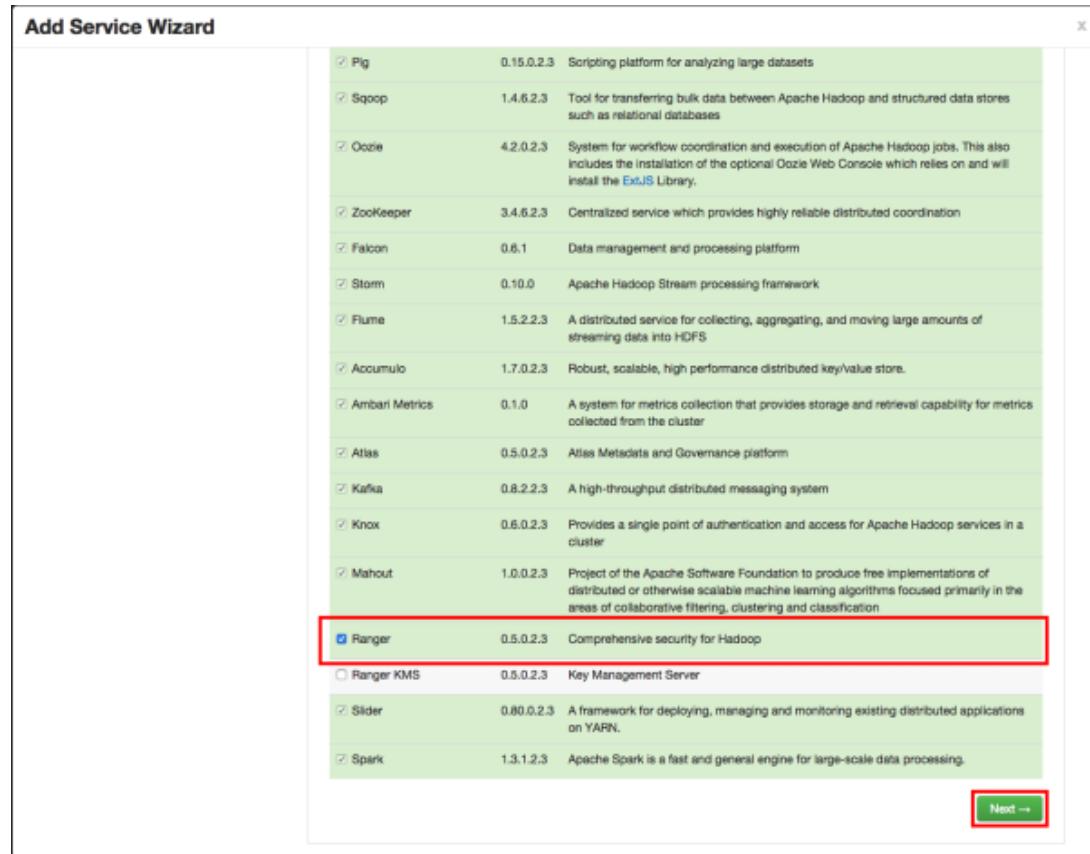
1. Log into your Ambari cluster with your designated user credentials. The main Ambari Dashboard page will be displayed.

Figure 3.1. Installing Ranger - Main Dashboard View

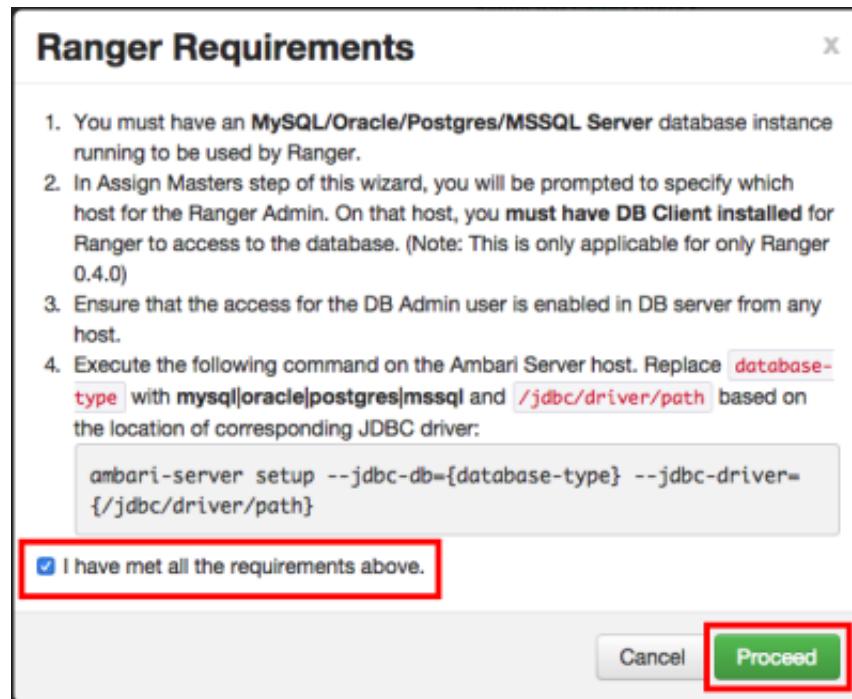
2. In the left navigation menu, click **Actions**, then select **Add Service**.

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

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).

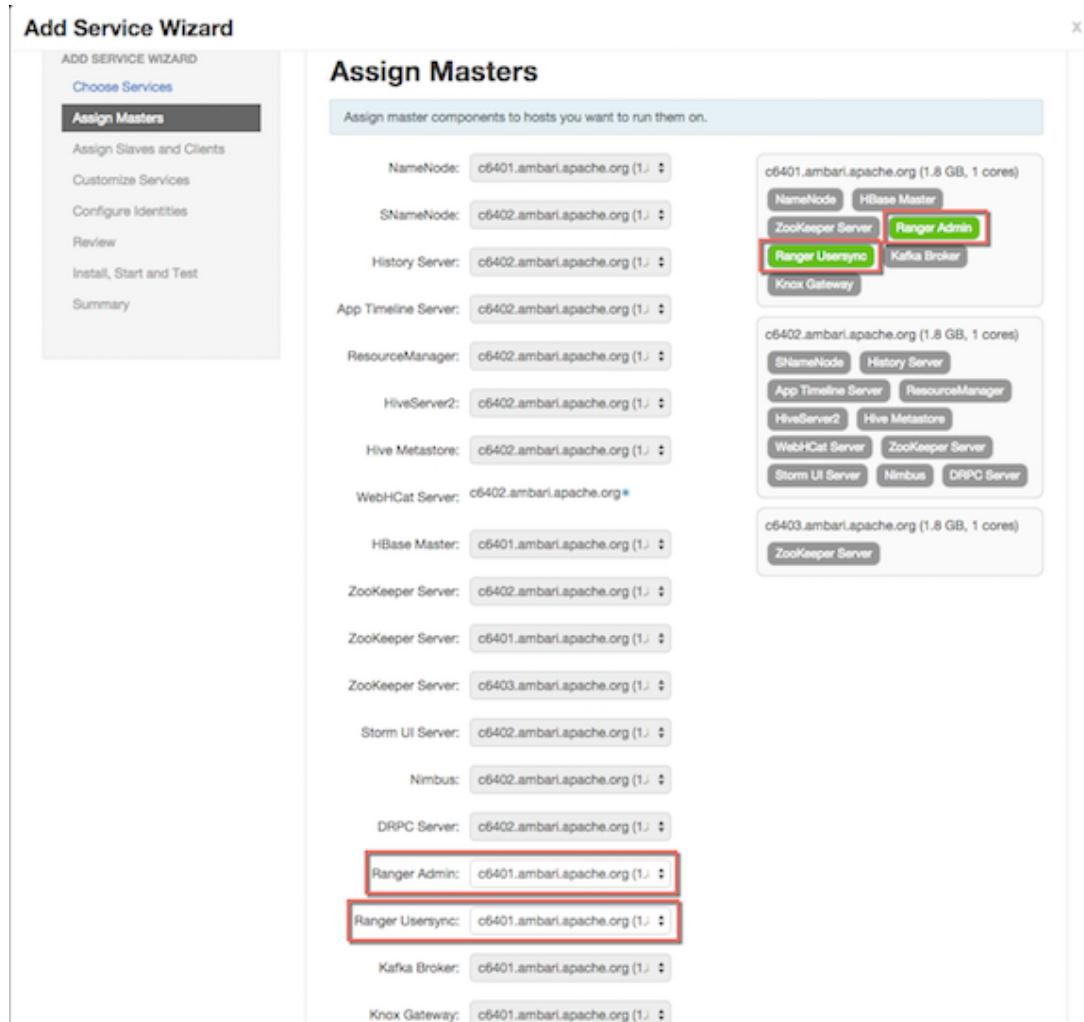
Make a note of the Ranger Admin host for use in subsequent installation steps. Click **Next** when finished to continue with the installation.



Note

The Ranger Admin and Ranger User Sync services must be installed on the same cluster node.

Figure 3.5. Installing Ranger Assign Masters



6. The Customize Services page appears. These settings are described in the next section.

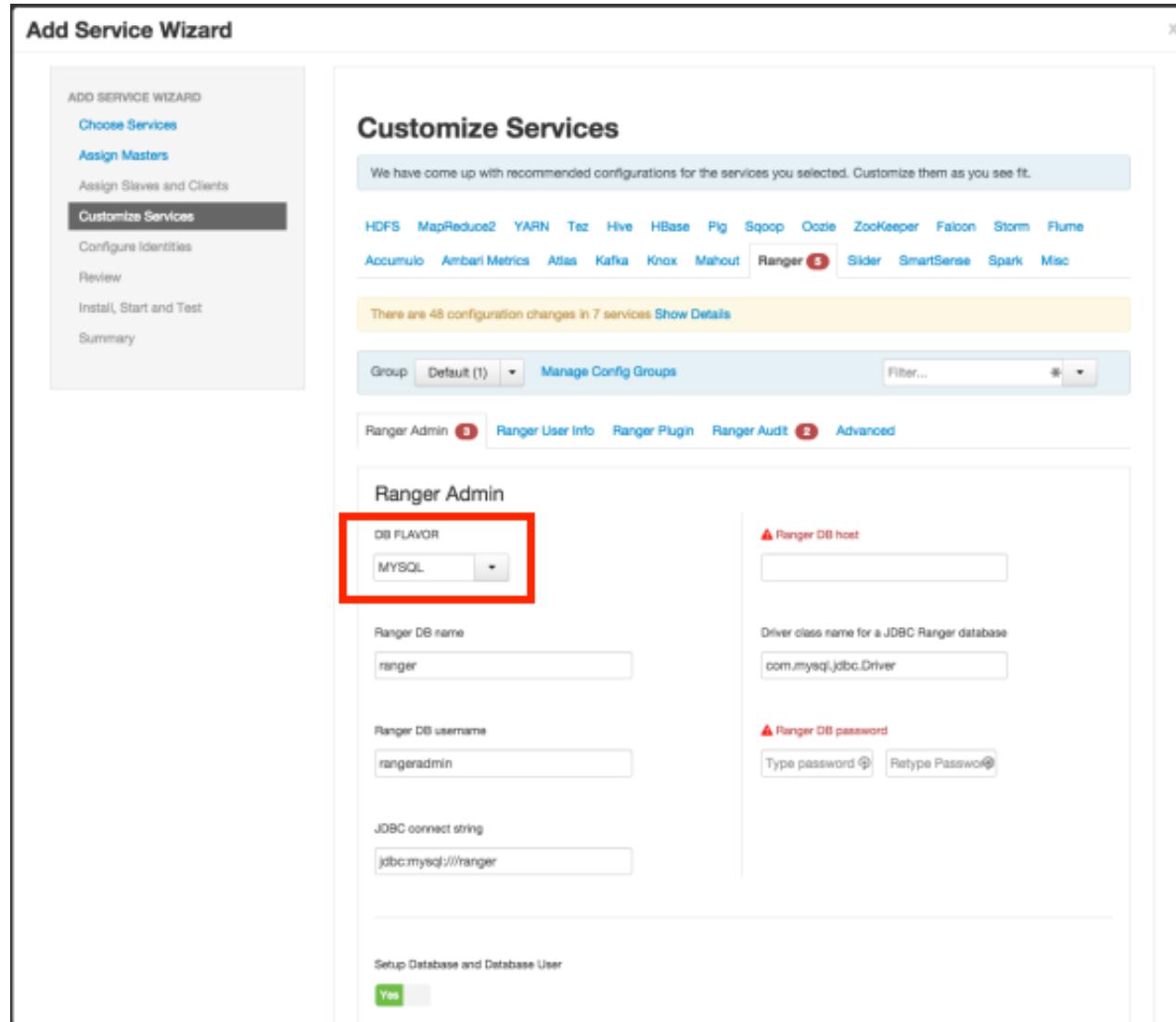
3.1.3.2. Customize Services

The next step in the installation process is to specify Ranger settings on the Customize Services page.

- [Ranger Admin Settings](#)
- [Ranger Audit Settings](#)
- [Configure Ranger User Sync](#)
- [Configure Ranger Authentication](#)

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
Oracle	<HOST:PORT:SID>	c6401.ambari.apache.org:1521:ORCL
	<HOST:PORT/Service>	c6401.ambari.apache.org:1521/XE
PostgreSQL	<HOST[:PORT]>	c6401.ambari.apache.org or c6401.ambari.apache.org:5432
MS SQL	<HOST[:PORT]>	c6401.ambari.apache.org

DB Flavor	Host	Example
		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	<code>jdbc:mysql://DB_HOST:PORT/db_name</code>	<code>jdbc:mysql://c6401.ambari.apache.org:3306/ranger_db</code>

DB Flavor	Syntax	Example Value
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 privileges to create database schemas and users.	root	root	Yes
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 Advanced

Ranger Admin

DB FLAVOR

MYSQL

Ranger DB name

ranger

Ranger DB username

rangeradmin

JDBC connect string

jdbc:mysql://c6401.ambari.apache.org/ranger

Ranger DB host

c6401.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

root

Database Administrator (DBA) password

.....

JDBC connect string for root user

jdbc:mysql://c6401.ambari.apache.org

Oracle – if the Oracle instance is running with a Service name.

Ranger Admin Ranger User Info Ranger Plugin Ranger Audit Advanced

Ranger Admin

DB FLAVOR	ORACLE	Ranger DB host
Ranger DB name	ranger	Driver class name for a JDBC Ranger database
Ranger DB username	rangeradmin	Ranger DB password
JDBC connect string	:oracle:thin:@//c6401.ambari.apache.org:1521/XE

Setup Database and Database User

Yes

Database Administrator (DBA) username	SYS AS SYSDBA	Database Administrator (DBA) password
JDBC connect string for root user	jdbc:oracle:thin:@//c6401.ambari.apache.org:1521

Oracle – if the Oracle instance is running with a SID.

[Ranger Admin](#)[Ranger User Info](#)[Ranger Plugin](#)[Ranger Audit](#)[Advanced](#)

Ranger Admin

DB FLAVOR

ORACLE

Ranger DB host

c6401.ambari.apache.org:1521:ORCL

Ranger DB name

ranger

Driver class name for a JDBC Ranger database

oracle.jdbc.driver.OracleDriver

Ranger DB username

rangeradmin

Ranger DB password

JDBC connect string

jdbc:oracle:thin:@c6401.ambari.apache.org:1521:orcl

Setup Database and Database User

Yes

Database Administrator (DBA) username

SYS AS SYSDBA

Database Administrator (DBA) password

JDBC connect string for root user

jdbc:oracle:thin:@c6401.ambari.apache.org:1521:orcl

PostgreSQL

Ranger Admin Ranger User Info Ranger Plugin Ranger Audit Advanced

Ranger Admin

DB FLAVOR	POSTGRES	Ranger DB host
Ranger DB name	ranger	Driver class name for a JDBC Ranger database
Ranger DB username	rangeradmin	Ranger DB password
JDBC connect string	jdbc:postgresql://c6401.ambari.apache.org:5432/r

Setup Database and Database User

Yes

Database Administrator (DBA) username	postgres	Database Administrator (DBA) password
JDBC connect string for root user	jdbc:postgresql://c6401.ambari.apache.org:5432

MS SQL

Ranger Admin Ranger User Info Ranger Plugin Ranger Audit Advanced

Ranger Admin

DB FLAVOR

MSSQL

Ranger DB name

ranger

Ranger DB username

rangeradmin

JDBC connect string

jdbc:sqlserver://c6401.ambari.apache.org:1433;databaseName=ranger

Ranger DB host

c6401.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

administrator

Database Administrator (DBA) password

.....

JDBC connect string for root user

jdbc:sqlserver://c6401.ambari.apache.org:1433;

SQL Anywhere

Ranger Admin Ranger User Info Ranger Plugin Ranger Audit Advanced

Ranger Admin

DB FLAVOR

SQL Anywhere

Ranger DB name

ranger

Ranger DB username

rangeradmin

JDBC connect string

=c6401.ambari.apache.org:2638;database=ranger

Ranger DB host

c6401.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

dba

Database Administrator (DBA) password

.....

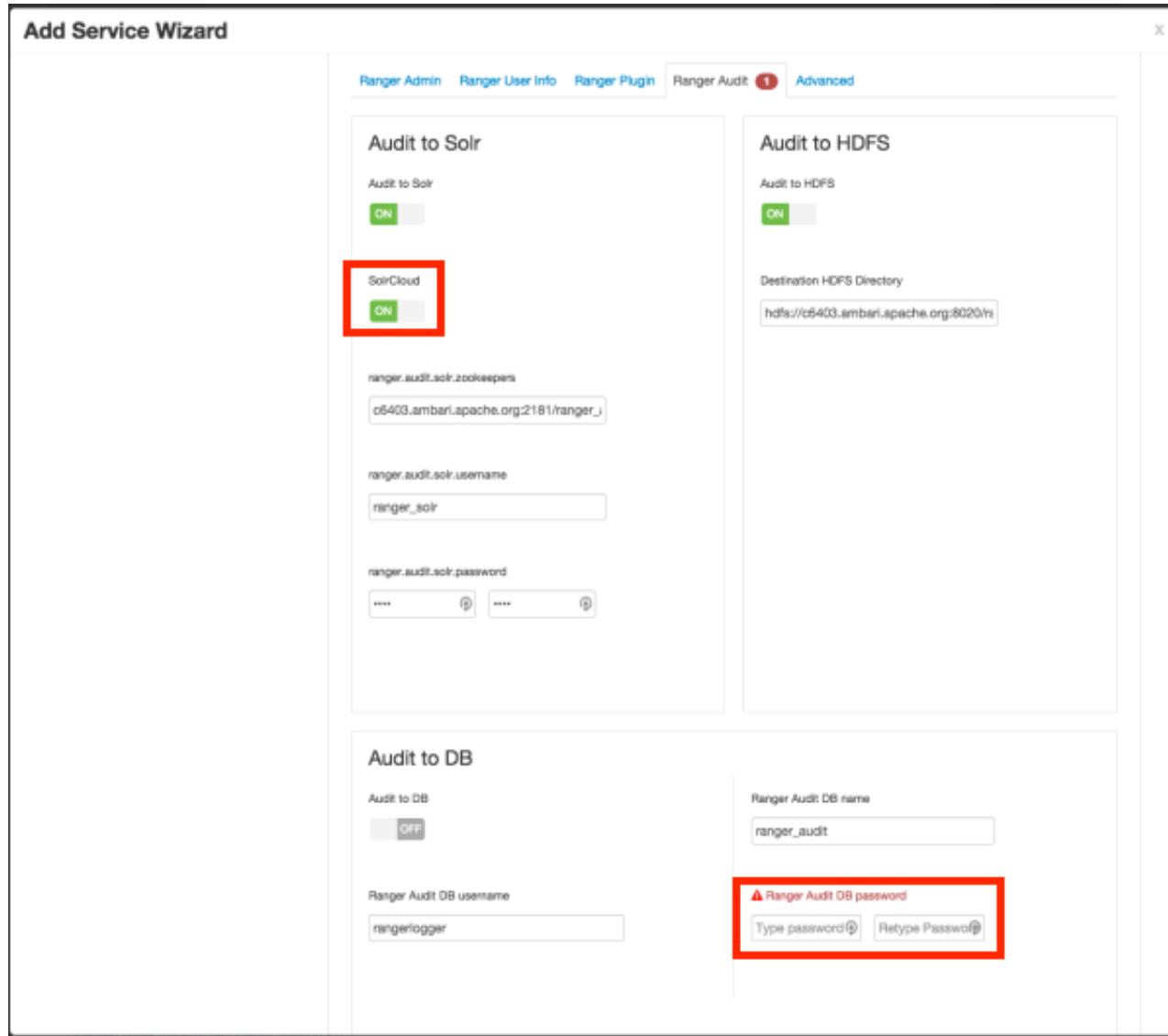
JDBC connect string for root user

jdbc:sqlanywhere:host=c6401.ambari.apache.org::

3.1.3.2.2. Ranger Audit Settings

1. On the Customize Services page, select the Ranger Audit tab.

It is recommended that you store audits in Solr and HDFS, and disable Audit to DB.



- 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.



Note

Audits to Solr requires that you have already [installed Solr](#) and [configured SolrCloud](#).

- Under Audit to DB, enter a password in the **Ranger Audit DB password** boxes. Audit to DB is set to **OFF** by default. (The password must be entered to preserve backward compatibility)

3.1.3.2.3. Configure Ranger User Sync

This section describes how to configure Ranger User Sync for either UNIX or LDAP/AD.

- Configuring Ranger User Sync for UNIX

- Configuring Ranger User Sync for LDAP/AD

3.1.3.2.3.1. Configuring Ranger User Sync for UNIX

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

The screenshot shows the 'Add Service Wizard' interface. On the left, a sidebar lists steps: 'Assign Masters', 'Assign Slaves and Clients', 'Customize Services' (which is selected), 'Configure Identities', 'Review', 'Install, Start and Test', and 'Summary'. The main area has a message: 'We have come up with recommended configurations for the services you selected. Customize them as you see fit.' Below this is a horizontal bar with service names: HDFS, MapReduce2, YARN, Tez, Hive, HBase, Pig, Sqoop, Oozie, ZooKeeper, Falcon, Storm, Flume, Accumulo, Ambari Metrics, Atlas, Kafka, Knox, Mahout, Ranger, Slider, SmartSense, Spark, and Misc. A yellow box indicates 'There are 50 configuration changes in 8 services' with a 'Show Details' link. Below this is a 'Manage Config Groups' section with tabs: Group, Default (1), Manage Config Groups, Filter..., Ranger Admin (selected), Ranger User Info, Ranger Plugin, Ranger Audit, and Advanced. The 'Ranger User Info' tab is active, showing settings for 'Enable User Sync' (set to 'Yes'), 'Sync Source' (set to 'UNIX'), 'Minimum User ID' (set to '500'), 'Password File' (set to '/etc/passwd'), and 'Group File' (set to '/etc/group'). At the bottom, a green bar says 'All configurations have been addressed.'

3.1.3.2.3.2. Configuring Ranger User Sync for LDAP/AD



Important

To ensure that LDAP/AD group level authorization is enforced in Hadoop, you should [set up Hadoop group mapping for LDAP/AD](#).



Note

You can use the [LDAP Connection Check tool](#) to determine User Sync settings for LDAP/AD.

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.		

The screenshot shows the 'Add Service Wizard' interface for configuring Ranger services. The main navigation bar at the top includes links for Accumulo, Ambari Metrics, Atlas, Kafka, Knox, Mahout, Ranger, Slider, SmartSense, Spark, and Misc. A message box indicates there are 51 configuration changes in 8 services, with a 'Show Details' link. Below the message, a 'Manage Config Groups' section allows users to switch between 'Group' and 'Default (1)' modes, and includes a 'Filter...' dropdown. The 'Ranger Admin' tab is active, while other tabs like 'Ranger User Info', 'Ranger Plugin', 'Ranger Audit', and 'Advanced' are visible. The 'Ranger User Info' configuration page is displayed, featuring sections for 'Enable User Sync' (set to 'Yes'), 'Sync Source' (set to 'LDAP/AD'), and 'User Configs'. The 'User Configs' tab is currently selected. Under 'User Configs', fields include 'LDAP/AD URL' (containing 'ldap://172.22.126.169:389') and 'Bind Anonymous' (set to 'No'). Other sections like 'Bind User' (containing 'cn=Manager,dc=qe,dc=hortonworks,dc=com') and 'Bind User Password' (with two masked input fields) are also present.

- 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.	No	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
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

Property	Description	Default Value	Sample Values
			of groupA or groupB: ((memberof=CN=GroupA,OU=groups,DC=example,DC=com)(memberof=CN=GroupB,OU=groups,DC=example,DC=com))
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

Add Service Wizard

Ranger User Info

Enable User Sync

Sync Source

[Common Configs](#) [User Configs](#) [Group Configs](#) Edit

Group User Map Sync

Username Attribute

User Object Class

User Search Base

User Search Filter

User Search Scope

User Group Name Attribute

- 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 from "User Group Name Attribute". In this case no additional group filters are applied. 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.	No	Yes
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	Search base for groups.		ou=groups,DC=example,DC=com
Group Search Filter	Optional additional filter constraining the groups selected for syncing.		Sample filter to retrieve all groups: cn=* Sample filter to retrieve only the groups whose cn is Engineering or Sales: ((cn=Engineering)(cn=Sales))

The screenshot shows the 'Add Service Wizard' interface for configuring a service. The current tab is 'Ranger User Info'. Key configuration options visible include:

- Enable User Sync:** Yes (selected)
- Sync Source:** LDAP/AD
- Enable Group Sync:** Yes (selected)
- Group Member Attribute:** member
- Group Name Attribute:** cn
- Group Object Class:** groupOfNames
- Group Search Base:** dc=qe,dc=hortonworks,dc=com
- Group Search Filter:** cn={}

3.1.3.2.4. Configure Ranger Authentication

This section describes how to configure Ranger authentication for UNIX, LDAP, and AD.

- [Configuring Ranger UNIX Authentication](#)
- [Configuring Ranger LDAP Authentication](#)
- [Configuring Ranger Active Directory Authentication](#)

3.1.3.2.4.1. 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 3.10. 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.	<code>{{{ugsync_host}}}</code>	
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.

Add Service Wizard

Ranger Settings

External URL:

Authentication method:

- LDAP
- ACTIVE_DIRECTORY
- UNIX
- NONE

HTTP enabled:

Unix Authentication Settings

Allow remote Login:	<input type="text" value="true"/>	
ranger.unixauth.service.hostname:	<input type="text" value="{{{ugsync_host}}}/"/>	
ranger.unixauth.service.port:	<input type="text" value="5151"/>	

Knox SSO Settings

Advanced ranger-admin-site

3.1.3.2.4.2. 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 3.11. 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 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}}	
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.		

Property	Description	Default Value	Example Value
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 .	<code>{ {ranger_ug_ldap_url} }</code>	<code>{ {ranger_ug_ldap_url} }</code>
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 password the user provided>	<code>uid={0},ou=users,dc=xasecure,dc=net</code>	<code>cn=ldapadmin,ou=Users,dc=example,dc=com</code>
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 .	<code>{ {ranger_ug_ldap_user_searchfilter} }</code>	<code>{ {ranger_ug_ldap_user_searchfilter} }</code>



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.

The screenshot shows the 'Add Service Wizard' interface for configuring a service. The 'Ranger Settings' tab is selected, displaying fields for 'External URL' (set to 'http://c6403.ambari.apache.org:6080'), 'Authentication method' (set to 'LDAP'), and 'HTTP enabled' (checkbox checked). Below it, the 'LDAP Settings' tab is selected, showing various configuration parameters for LDAP integration, such as 'ranger.ldap.base_dn' (set to 'dc=example,dc=com'), 'Bind User' (set to '[[ranger_ug_ldap_bind_dn]])', and 'Bind User Password' (redacted). Other settings include 'ranger.ldap.group.roleattribute' (set to 'cn'), 'ranger.ldap.referral' (set to 'ignore'), 'LDAP URL' (set to '[[ranger_ug_ldap_url]])', 'ranger.ldap.user.dnpattern' (set to 'uid=[0],ou=users,dc=kasecure,dc=net'), and 'User Search Filter' (set to '[[ranger_ug_ldap_user_searchfilter]])'. Below these tabs, several other tabs are listed: 'Knox SSO Settings', 'Advanced ranger-admin-site', 'Advanced ranger-env', 'Advanced ranger-ugsync-site', and 'Custom admin-properties'.

3.1.3.2.4.3. 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 3.12. 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 .	<code>{ {ranger_ug_ldap_bind_dn} }{ {ranger_ug_ldap_bind_dn} }</code>	
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 .	<code>{ {ranger_ug_ldap_url} }</code>	<code>{ {ranger_ug_ldap_url} }</code>
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 .	<code>{ {ranger_ug_ldap_user_searchfilter} }</code>	<code>{ {ranger_ug_ldap_user_searchfilter} }</code>



Note

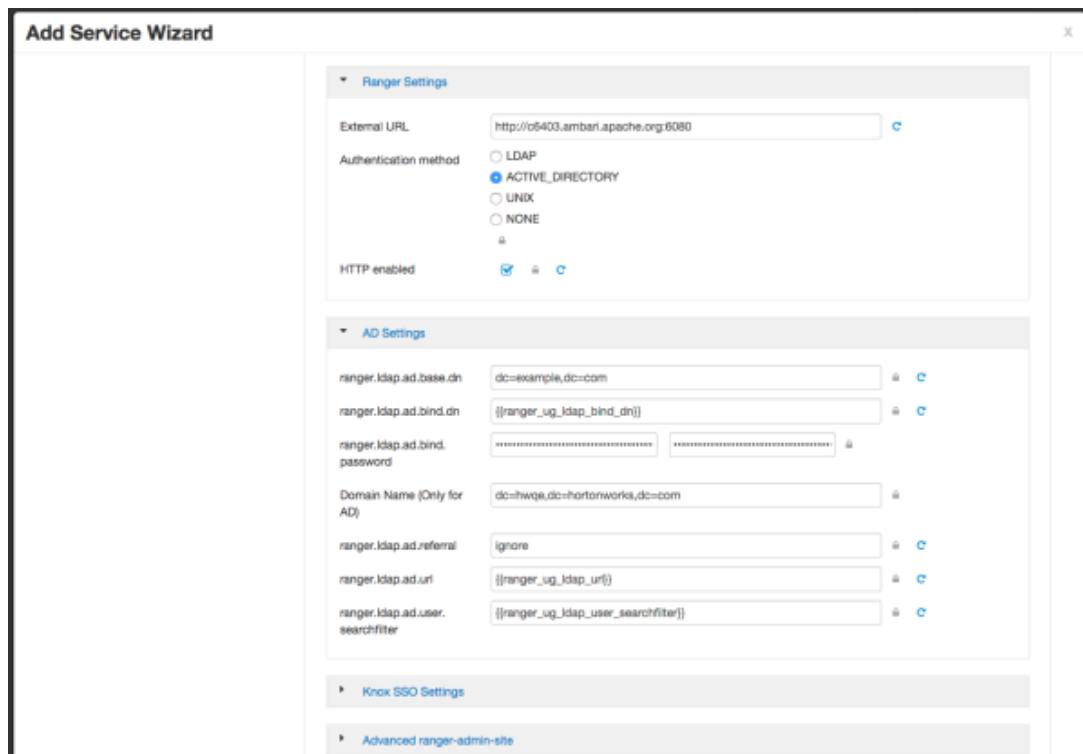
Properties with value `{ {xyz} }` are macro variables that are derived from other specified values in order to streamline the configuration process. Macro variables can be edited if required – if you need to restore the original value, click the Set Recommended symbol at the right of the property box.

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.

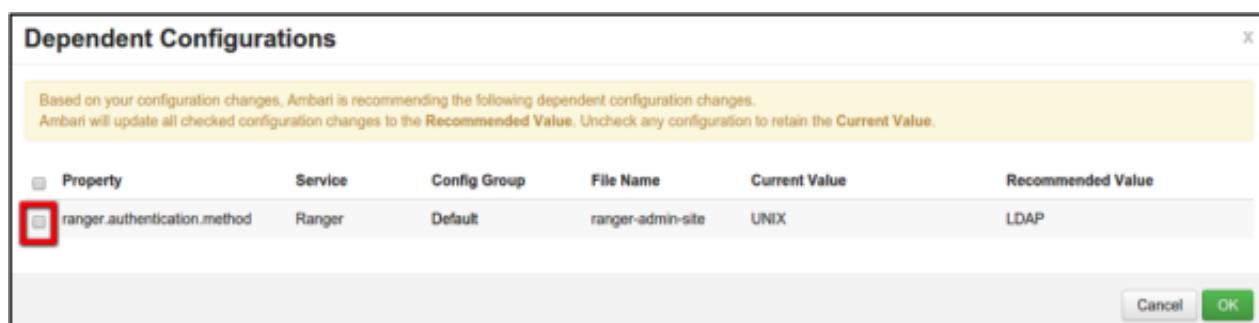
- When this property is set to `follow`, the AD service provider processes all of the normal entries first, and then follows the continuation references.

- 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.



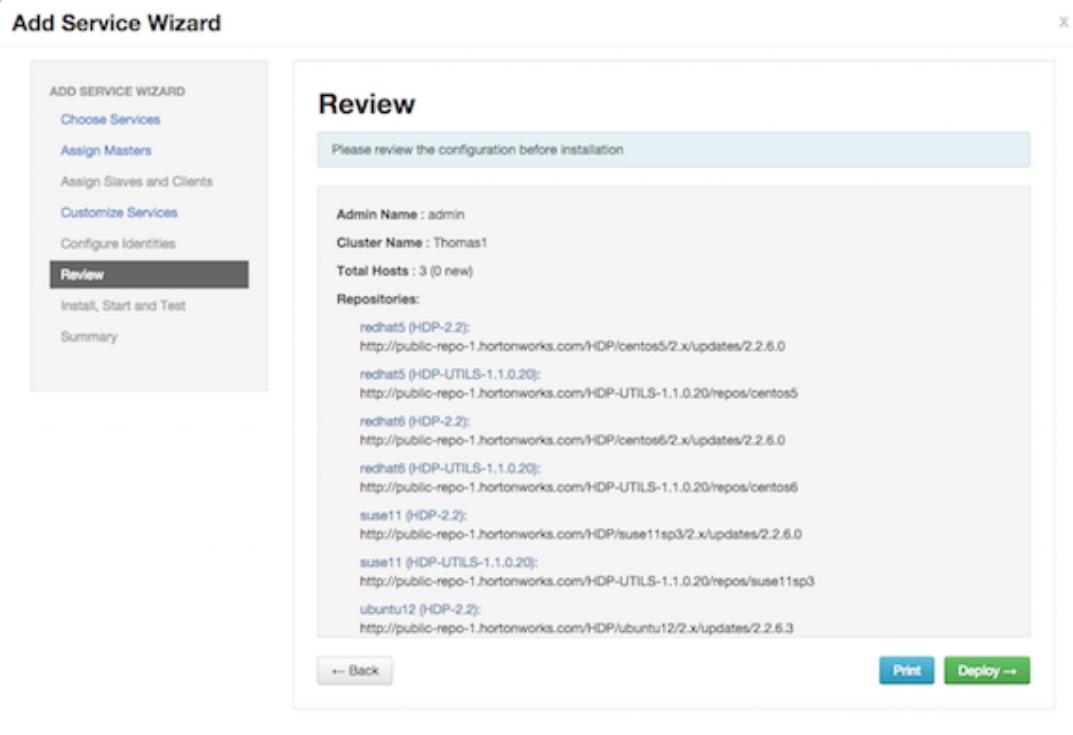
When you have finished configuring all of the Customize Services Settings, click **Next** at the bottom of the page to continue with the installation.

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**.

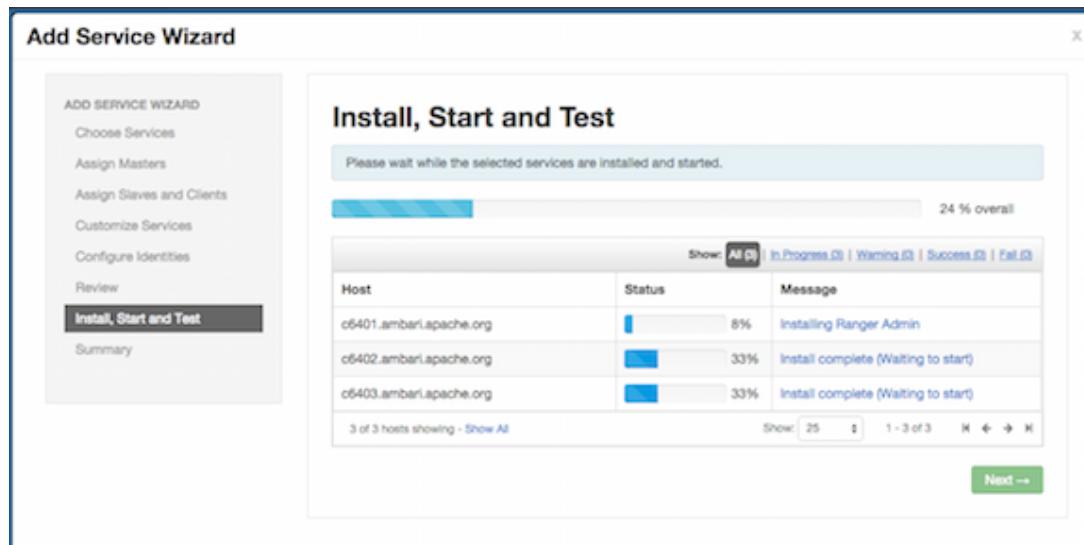


3.1.3.3. Complete the Ranger Installation

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.



2. When you click **Deploy**, Ranger is installed on the specified host on your Ambari server. A progress bar displays the installation progress.



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 [set up Hadoop group mapping](#) for LDAP.

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.ldap.bindkeystore		<input type="button" value=""/>	<input checked="" type="checkbox"/>		
ranger.usersync.ldap.ldapbindpassword	Type password	<input type="button" value=""/>	Retype Password <input type="button" value=""/>	<input type="button" value=""/>	<input checked="" type="checkbox"/>
ranger.usersync.group.memberattributename		<input type="button" value=""/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	
ranger.usersync.group.nameattribute		<input type="button" value=""/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	
ranger.usersync.group.objectclass		<input type="button" value=""/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	
ranger.usersync.group.searchbase		<input type="button" value=""/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	
ranger.usersync.group.searchenabled	false	<input type="button" value=""/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	
ranger.usersync.group.searchfilter		<input type="button" value=""/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	
ranger.usersync.group.searchscope		<input type="button" value=""/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	
ranger.usersync.group.usermodelsyncenabled	false	<input type="button" value=""/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	
ranger.usersync.ldap.searchBase	dc=hadoop,dc=apache,dc=org	<input type="button" value=""/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	
ranger.usersync.source.impl.class	org.apache.ranger.unixusersync.process.UnixUserGroupBuilder	<input type="button" value=""/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	
ranger.usersync.credstore.filename	/usr/hdp/current/ranger-usersync/conf/ugsync.jceks	<input type="button" value=""/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	
ranger.usersync.enabled	true	<input type="button" value=""/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	
ranger.usersync.filesource.file	/tmp/usergroup.txt	<input type="button" value=""/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	
ranger.usersync.filesource.text.delimiter	,	<input type="button" value=""/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	
ranger.usersync.keystore.file	/usr/hdp/current/ranger-usersync/conf/unixauthservice.jks	<input type="button" value=""/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	

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.

Table 3.13. LDAP Advanced ranger-ugsync-site Settings

Property Name	LDAP Value
ranger.usersync.ldap.bindkeystore	Set this to the same value as the ranger.usersync.credstore.filename property.

Property Name	LDAP Value
	i.e, the default value is /usr/hdp/current/ranger-usersync/conf/ugsync.jceks
ranger.usersync.ldap.bindalias	ranger.usersync.ldap.bindalias
ranger.usersync.source.impl.class	ldap

Table 3.14. 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.

Table 3.15. 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.ldapusersync.process.LdapUserGroupBuilder	
ranger.usersync.ldap.user.searchbase	ou=users, dc=example, dc=com	dc=example,dc=com
ranger.usersync.ldap.user.searchscope	sub	sub
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.usermodelsyncenabled *	false	false
ranger.usersync.group.searchbase *	ou=groups, dc=example, dc=com	dc=example,dc=com
ranger.usersync.group.searchscope *	sub	sub

Property Name	LDAP ranger-ugsync-site Value	AD ranger-ugsync-site Value
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

* 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 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-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.

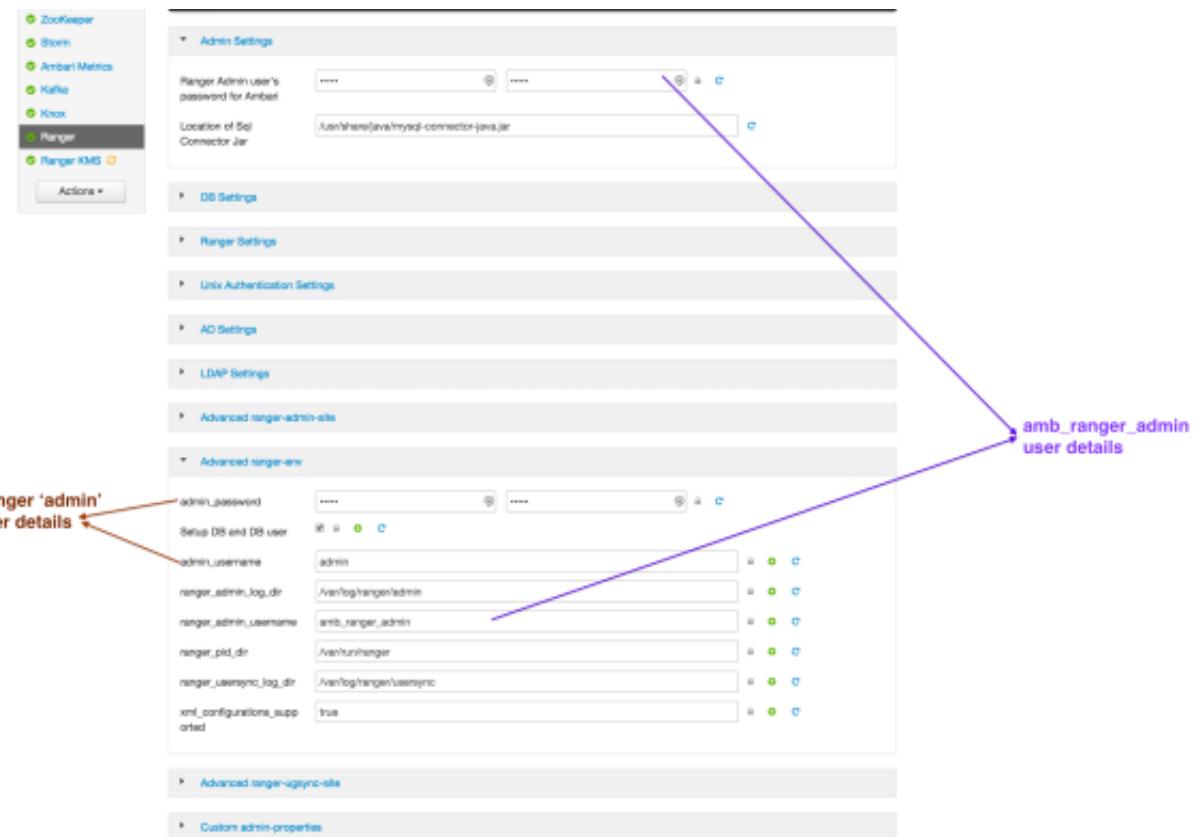
6. Run the Ranger Ambari install procedure, but set **Setup Database and Database User** to **No** in the Ranger Admin section of the Customize Services page.

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](#)
- [Hive](#)
- [HBase](#)
- [Kafka](#)
- [Knox](#)
- [YARN](#)
- [Storm](#)

3.1.4.1. HDFS

Use the following steps to enable the Ranger HDFS plugin.

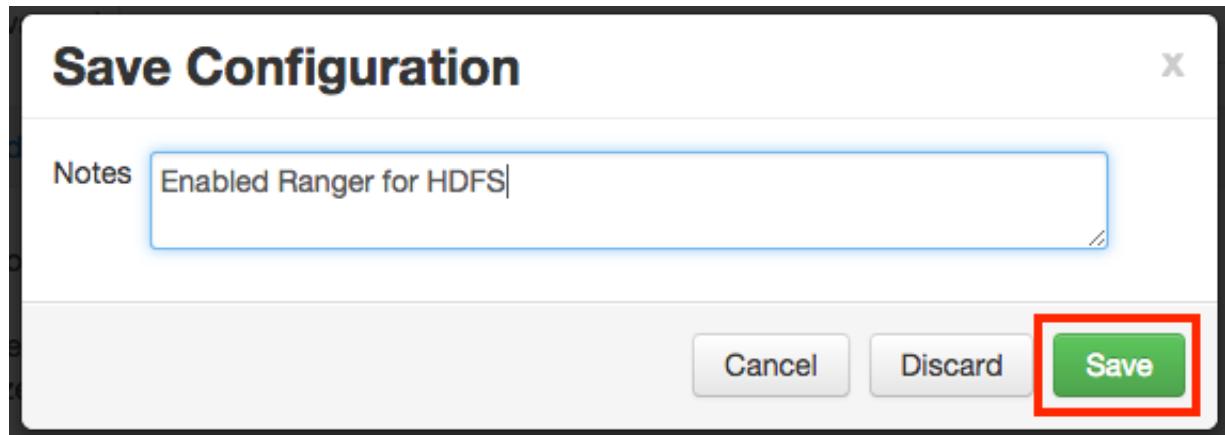
1. On the Ranger Configs page, select the **Ranger Plugin** tab.

The screenshot shows the Ambari interface for the cluster 'c6401_TEST'. The left sidebar lists various 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 currently selected. The main content area displays the 'Configs' tab for the 'Ranger Admin' section. It shows a configuration group named 'Default (1)' with three versions: V3 (admin, 6 days ago), V2 (admin, 7 days ago), and V1 (admin, 13 days ago). A message at the bottom indicates that 'admin authored on Wed, Feb 03, 2016 11:28'. Below this, the 'Ranger Plugin' tab is selected, showing the status of various Ranger Plugins: HDFS Ranger Plugin (ON), Hbase Ranger Plugin (ON), Knox Ranger Plugin (ON), YARN Ranger Plugin (ON), Storm Ranger Plugin (OFF), and Hive Ranger Plugin (ON).

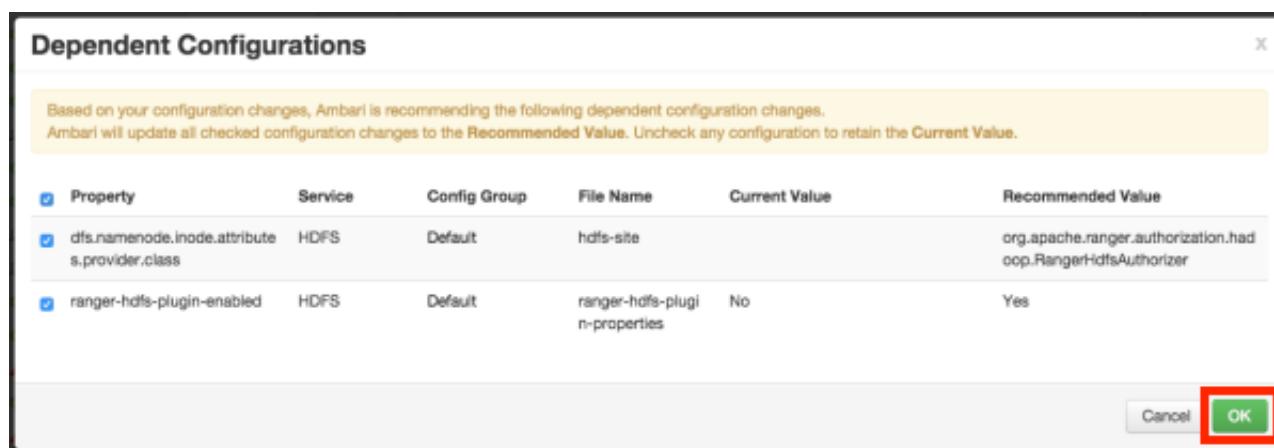
2. Under HDFS Ranger Plugin, select **On**, then click **Save** in the black menu bar.

This screenshot shows the 'Ranger Plugin' configuration page. The 'HDFS Ranger Plugin' button is highlighted with a red box. The other plugin status buttons are: Hbase Ranger Plugin (ON), Knox Ranger Plugin (ON), YARN Ranger Plugin (ON), Storm Ranger Plugin (OFF), and Kafka Ranger Plugin (ON). The 'ON' button for HDFS Ranger Plugin is highlighted with a red box.

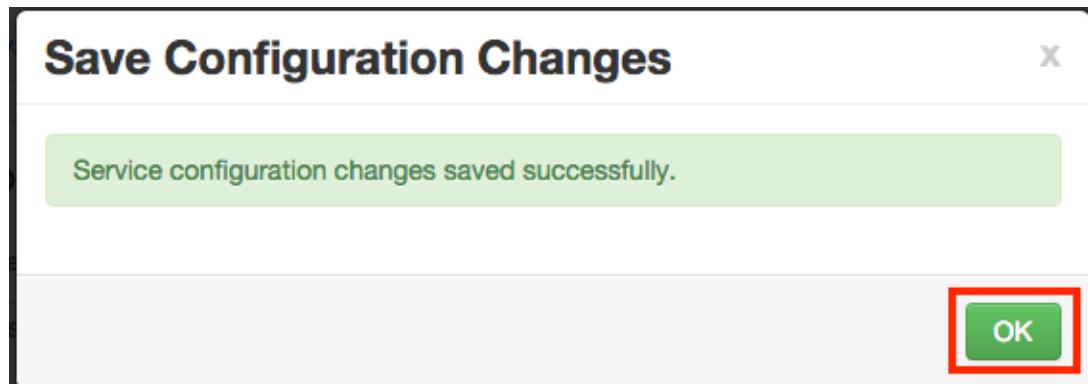
3. A Save Configuration pop-up appears. Type in a note describing the changes you just made, then click **Save**.



4. A Dependent Configuration pop-up appears. Click **OK** to confirm the configuration updates.



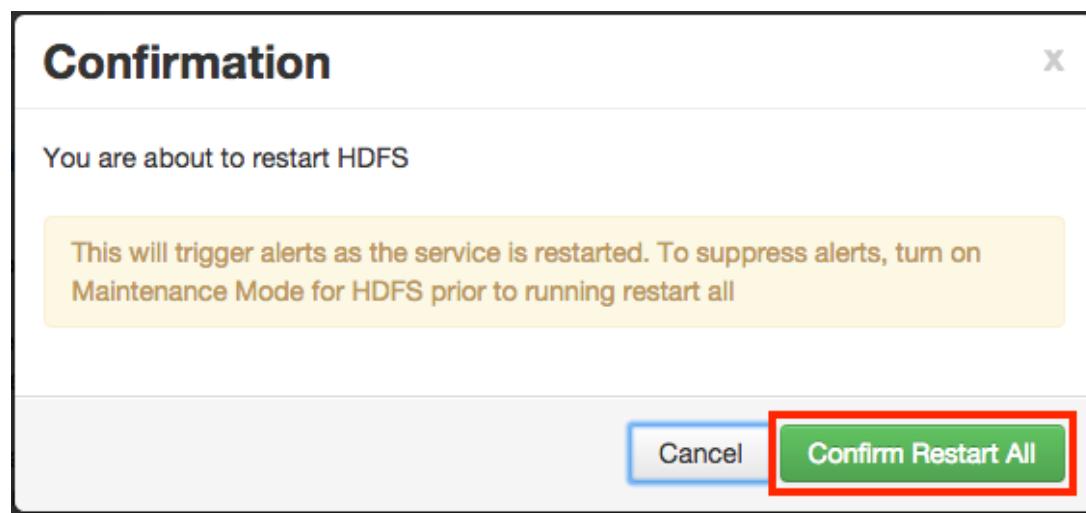
5. Click **OK** on the Save Configuration Changes pop-up.



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 UI for a cluster named 'test_cluster'. On the left, a sidebar lists various services: HDFS (green), MapReduce2 (orange), YARN (blue), Tez (grey), Hive (green), HBase (red), Pig (grey), Sqoop (grey), Oozie (green), ZooKeeper (green), Falcon (red), Storm (green), Flume (red), Accumulo (red), Ambari Metrics (green), Atlas (green), Kafka (green), Knox (green), Mahout (grey), Ranger (green), Slider (grey), SmartSense (green), and Spark (green). In the center, the 'Configs' tab is selected under the 'HDFS' service. A message at the top says 'Restart Required: 4 Components on 1 Host'. Below it, a 'Manage Config Groups' section shows three configurations: V3 (2 minutes ago), V2 (2 hours ago), and V1 (15 days ago). A 'Restart' button is visible above the configuration list, with a red box highlighting the 'Restart All Affected' option. The main area contains two panels: 'NameNode' and 'DataNode', each with several configuration parameters and their current values.

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.

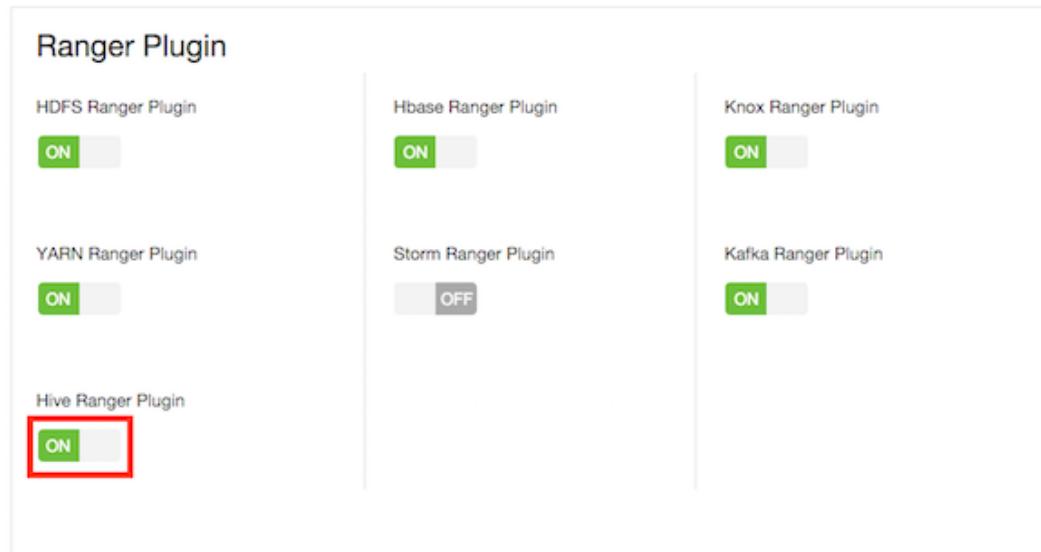
3.1.4.2. Hive

Use the following steps to enable the Ranger Hive plugin.

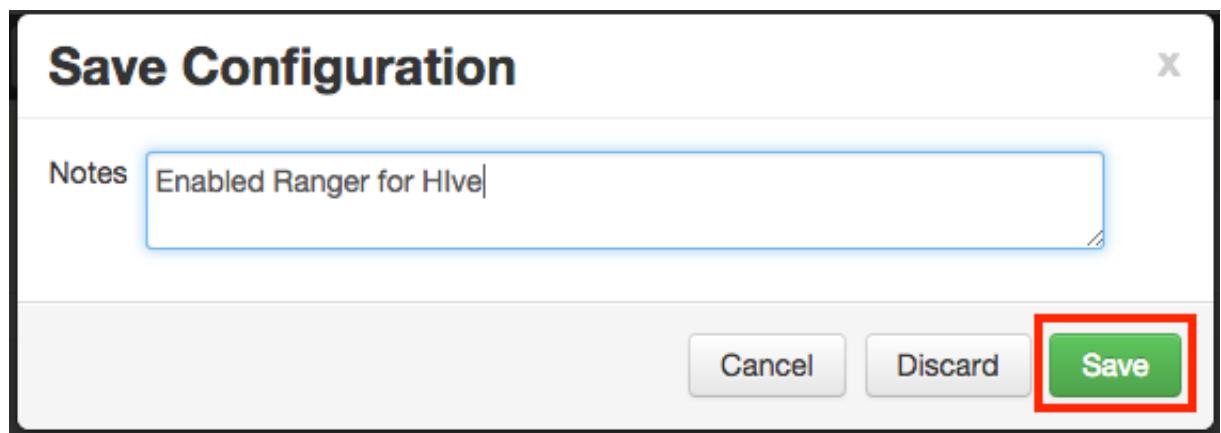
1. On the Ranger Configs page, select the **Ranger Plugin** tab.

The screenshot shows the Ambari interface for the cluster 'c6401_TEST'. The left sidebar lists various 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 currently selected. The main content area displays the 'Configs' tab for the 'Default (1)' group. A configuration for 'V3' is selected, showing it was authored by 'admin' on 'Wed, Feb 03, 2016 11:28'. Below this, the 'Ranger Plugin' tab is active, showing the status of several 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), and Hive Ranger Plugin (ON). At the bottom of the page, there are '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.

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

Cancel **OK**

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

Save Configuration Changes

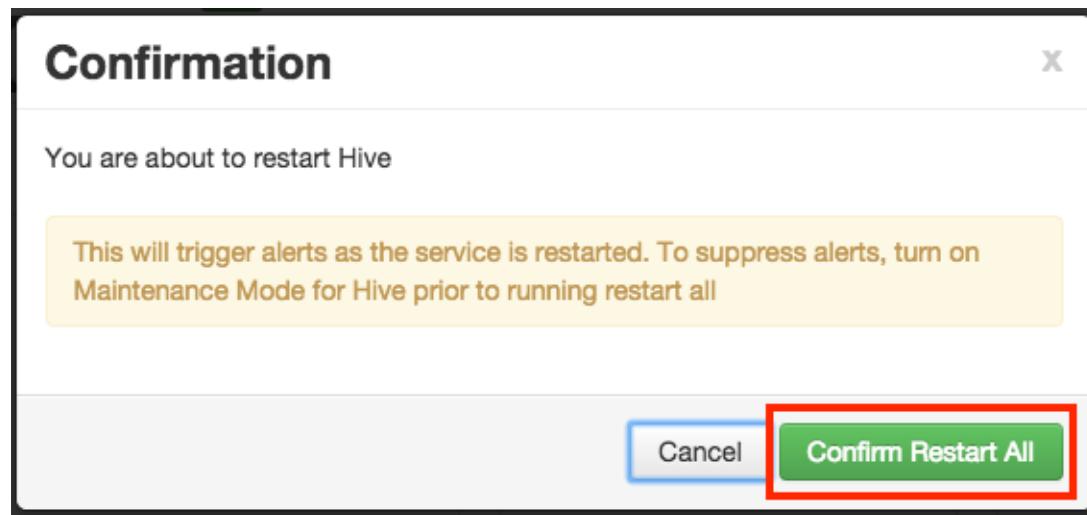
Service configuration changes saved successfully.

OK

- Select **Hive** in the navigation menu, then select **Restart > Restart All Affected** to restart the Hive service and load the new configuration.

The screenshot shows the Ambari interface for a test cluster. On the left, a sidebar lists various services: HDFS, MapReduce2, YARN, Tez, Hive (selected), HBase, Pig, Sqoop, Oozie, ZooKeeper, Falcon, Storm, Flume, Accumulo, Ambari Metrics, Atlas, Kafka, Knox, Mahout, Ranger, Slider, SmartSense, and Spark. In the center, the 'Configs' tab is selected under the 'Summary' tab. A yellow banner at the top states 'Restart Required: 6 Components on 1 Host'. Below it, a 'Manage Config Groups' section shows three groups: V3 (admin, about a minute ago, HDP-2.3), V2 (admin, 5 hours ago, HDP-2.3), and V1 (admin, 15 days ago, HDP-2.3). A modal dialog is open, showing settings for ACID Transactions, Interactive Query, and Security. The 'Security' tab includes options for Choose Authorization (set to Ranger), Run as end user instead of Hive user (set to False), HiveServer2 Authentication (set to None), and Use SSL (set to False). In the top right of the main area, there are 'Service Actions' buttons for 'Restart' and 'Restart All Affected', with the latter being highlighted by a red box.

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

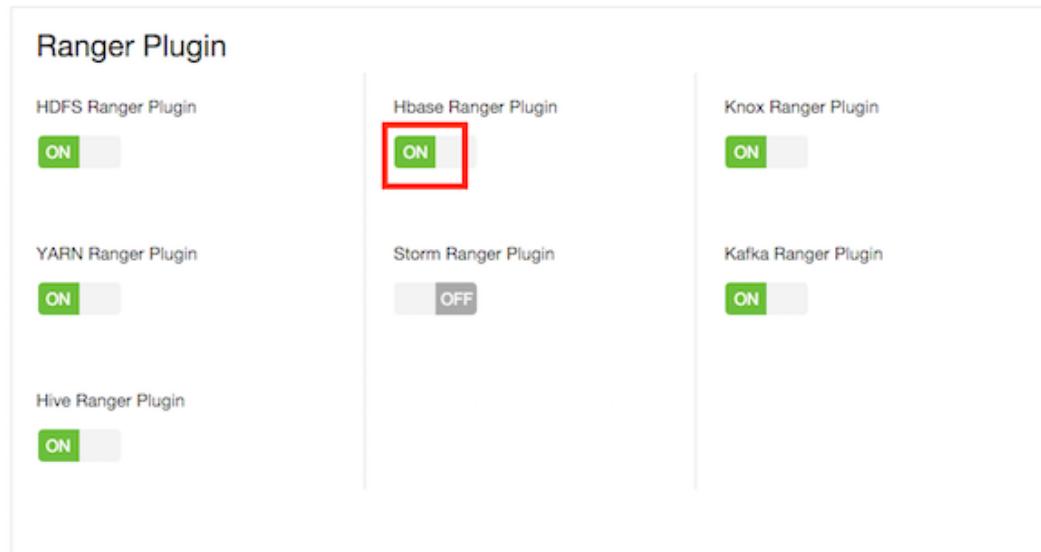
When Hive is configured with Ranger, and specifically XASecure Authorizer, you may only grant and revoke privileges.

Use the following steps to enable the Ranger HBase plugin.

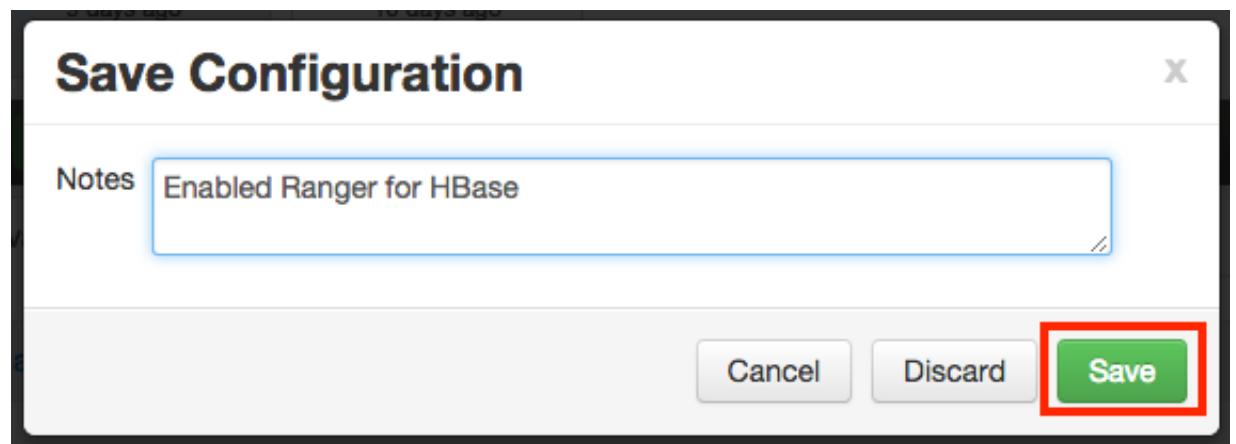
1. On the Ranger Configs page, select the **Ranger Plugin** tab.

The screenshot shows the Ambari interface for the cluster 'c6401_TEST'. The top navigation bar includes 'Dashboard', 'Services', 'Hosts', 'Alerts', 'Admin', and a user dropdown for 'admin'. The left sidebar lists various services: HDFS, MapReduce2, YARN, Tez, Hive, HBase (selected), Pig, Swoop, Oozie, ZooKeeper, Falcon, Storm, Flume, Accumulo, Ambari Metrics, Atlas, Kafka, Knox, Mahout, Ranger (selected), Slider, SmartSense, and Spark. The main content area has tabs for 'Summary' and 'Configs'. Under 'Configs', it shows a 'Group' section with 'Default (1)' and a 'Manage Config Groups' button. Below this is a list of configurations: V3 (admin, 6 days ago, HDP-2.3), V2 (admin, 7 days ago, HDP-2.3), and V1 (admin, 13 days ago, HDP-2.3). A message below the configurations states 'V3 ✓ admin authored on Wed, Feb 03, 2016 11:28'. At the bottom of this section are 'Discard' and 'Save' buttons. Below these buttons is a 'Ranger Admin' tab, followed by 'Ranger User Info', 'Ranger Plugin' (which is the active tab), 'Ranger Audit', and 'Advanced'. The 'Ranger Plugin' section contains six toggle switches: 'HDFS Ranger Plugin' (ON), 'Hbase Ranger Plugin' (ON), 'Knox Ranger Plugin' (ON), 'YARN Ranger Plugin' (ON), 'Storm Ranger Plugin' (OFF), and 'Kafka Ranger Plugin' (ON). The 'Actions' button is located at the bottom left of the main content area.

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.

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.RangerAuthorizationCoprocessor
<input checked="" type="checkbox"/> hbase.coprocessor.master.classes	HBase	Default	hbase-site		org.apache.ranger.authorization.hbase.RangerAuthorizationCoprocessor
<input checked="" type="checkbox"/> hbase.coprocessor.region.classes	HBase	Default	hbase-site	org.apache.hadoop.hbase.security.access.SecureBulkLoadEndpoint,org.apache.ranger.authorization.hbase.RangerAuthorizationCoprocessor	org.apache.hadoop.hbase.security.access.SecureBulkLoadEndpoint,org.apache.ranger.authorization.hbase.RangerAuthorizationCoprocessor
<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

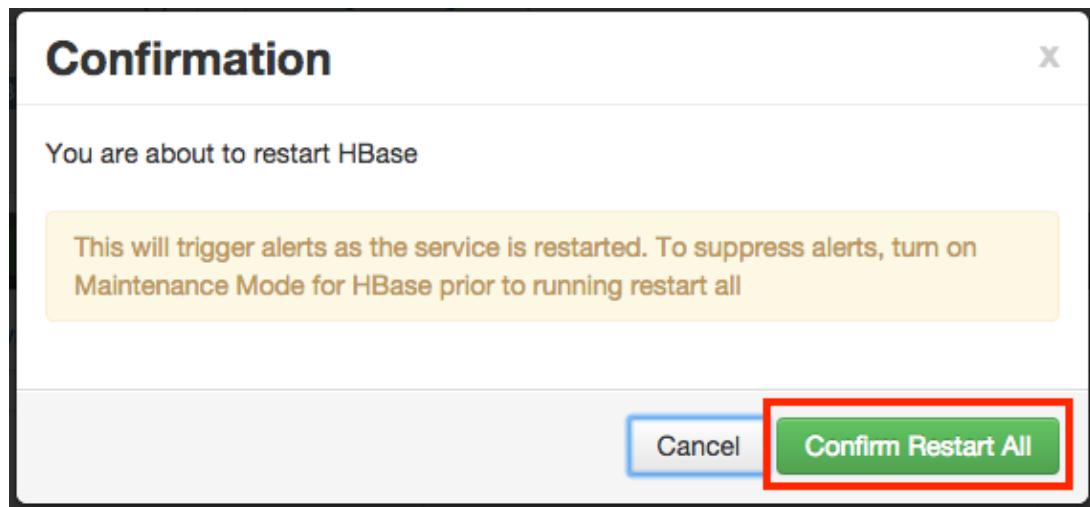
Service configuration changes saved successfully.

OK

6. Select **HBase** in the navigation menu, then select **Restart > Restart All Affected** to restart the HBase service and load the new configuration.

The screenshot shows the Ambari interface for managing HBase configurations. On the left, there's a sidebar with various service icons. The main area has tabs for 'Summary', 'Heatmaps', and 'Configs'. A message at the top says 'Restart Required: 3 Components on 1 Host'. Below it, a 'Manage Config Groups' section shows three versions (V3, V2, V1) of a configuration group named 'HBase Default (1)'. A red box highlights the 'Restart All Affected' button. At the bottom, there are sections for 'HBase Master' and 'RegionServer' with various configuration parameters like 'Maximum Memory' and 'Number of Handlers per RegionServer'.

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



- 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.

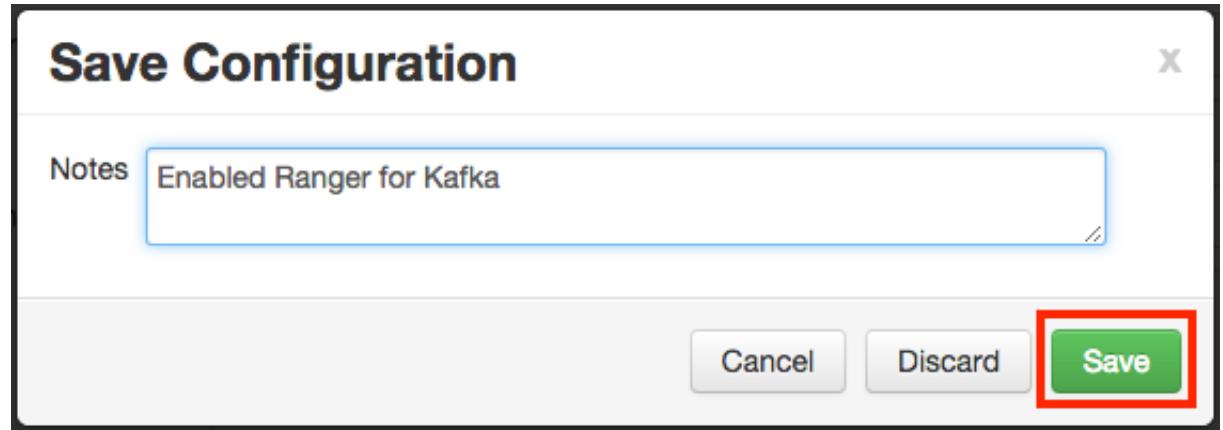
- On the Ranger Configs page, select the **Ranger Plugin** tab.

The screenshot shows the Ambari interface for the cluster 'c6401_TEST'. The left sidebar lists various 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 selected. The main panel displays the 'Ranger Plugin' configuration. It includes sections for HDFS, Hbase, and Knox Ranger Plugins, all set to 'ON'. Below these are sections for YARN, Storm, and Kafka Ranger Plugins. The 'Kafka Ranger Plugin' button is highlighted with a red box. At the bottom of the page, there is a note: 'admin authored on Wed, Feb 03, 2016 11:28' and buttons for 'Discard' and 'Save'.

2. Under Kafka Ranger Plugin, select **On**, then click **Save** in the black menu bar.

This screenshot shows the same 'Ranger Plugin' configuration page as the previous one, but with a red box drawn around the 'ON' button for the 'Kafka Ranger Plugin'. The other buttons ('ON' for HDFS, Hbase, and Knox) are also green. The 'Kafka Ranger Plugin' button is specifically highlighted with a red border.

3. A Save Configuration pop-up appears. Type in a note describing the changes you just made, then click **Save**.



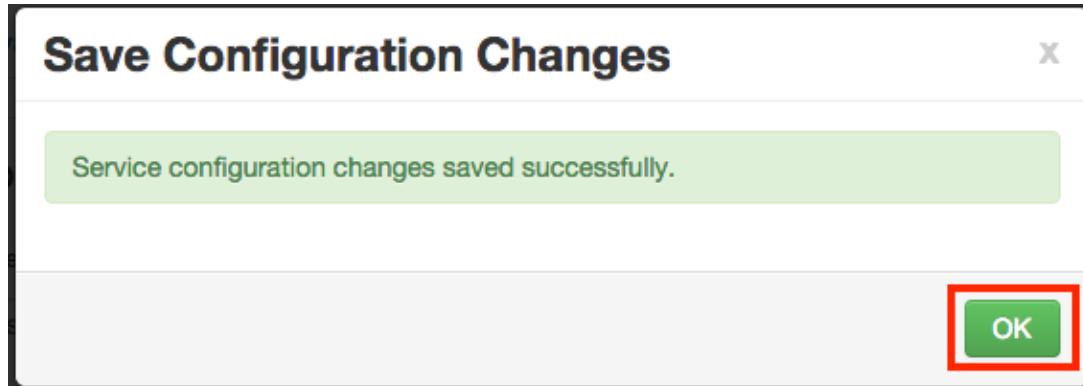
4. A Dependent Configuration pop-up appears. Click **OK** to confirm the configuration updates.

A screenshot of a 'Dependent Configurations' dialog box. It shows a table of configuration changes:

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.

At the bottom right of the dialog box, there are 'Cancel' and 'OK' buttons, with 'OK' also highlighted with a red border.

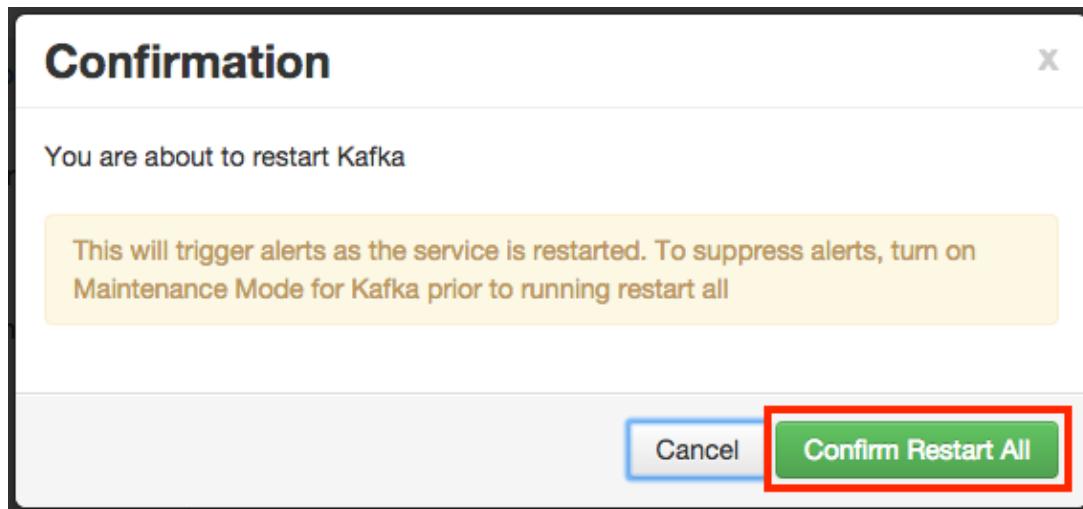
5. Click **OK** on the Save Configuration Changes pop-up.



6. Select **Kafka** in the navigation menu, then select **Restart > Restart All Affected** to restart the Kafka service and load the new configuration.

A screenshot of the Ambari web interface. On the left, a sidebar lists various services: HDFS, MapReduce2, YARN, Tez, Hive, HBase, Pig, Sqoop, Oozie, ZooKeeper, Falcon, Storm, Flume, Accumulo, Ambari Metrics, Atlas, and Kafka. The "Kafka" service is selected. The main content area shows the "Configs" tab for the Kafka Broker. It displays configuration parameters like "Kafka Broker host" (c6403.ambari.apache.org), "zookeeper.connect" (c6403.ambari.apache.org:2181), "log.dirs" (/kafka-logs), "log.retention.hours" (168), "log.roll.hours" (168), and "listeners" (PLAINTEXT://localhost:6667). Above the configuration table, a yellow banner states "Restart Required: 1 Component on 1 Host". To the right of the configuration table, there is a "Service Actions" section with a "Restart" dropdown menu. A red rectangular box highlights the "Restart All Affected" button in this menu. Below the configuration table, there are sections for "Advanced kafka-broker", "Advanced kafka-env", and "Advanced kafka-log4j".

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.

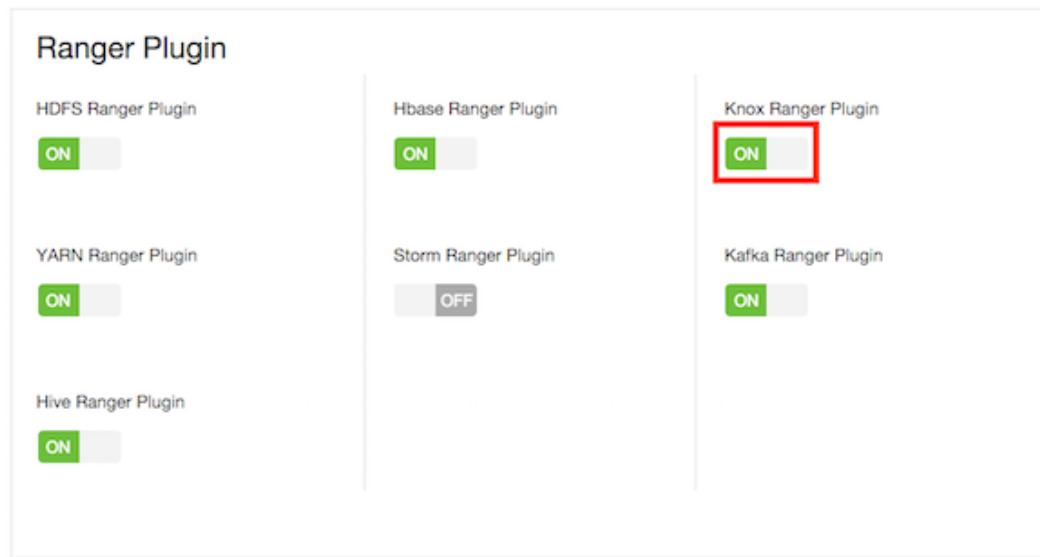
3.1.4.5. Knox

Use the following steps to enable the Ranger Knox plugin.

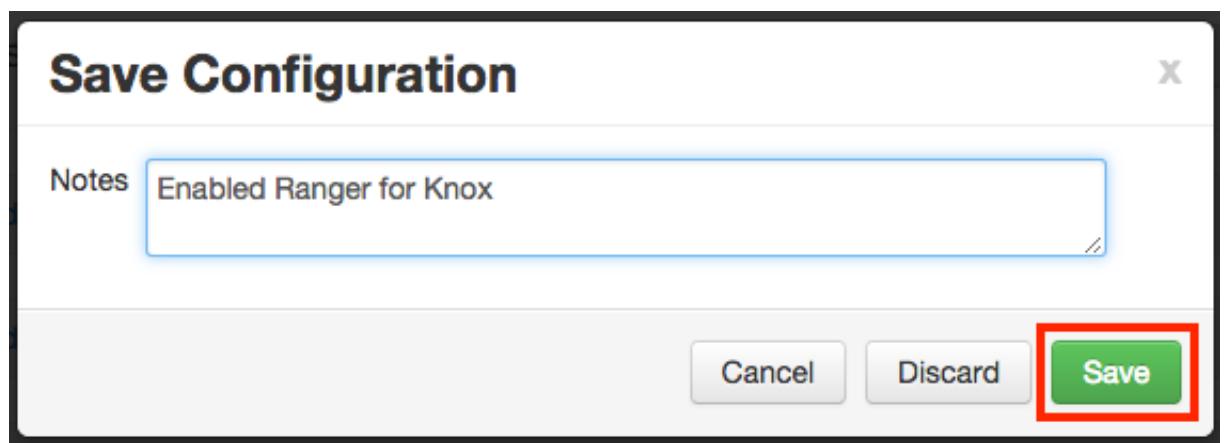
1. On the Ranger Configs page, select the **Ranger Plugin** tab.

A screenshot of the Ambari interface showing the Ranger Configs page for cluster c6401_TEST. The left sidebar shows various services like HDFS, MapReduce2, YARN, Tez, Hive, etc., with the Ranger service selected. The main area has tabs for Summary, Configs, and Ranger Plugin (which is currently selected). Under the Ranger Plugin tab, there are several sub-plugins listed with their status: HDFS Ranger Plugin (ON), Hbase Ranger Plugin (ON), Knox Ranger Plugin (ON), YARN Ranger Plugin (ON), Storm Ranger Plugin (OFF), and Hive Ranger Plugin (ON). At the bottom of the main area, there are "Discard" and "Save" buttons.

2. Under Knox Ranger Plugin, select **On**, then click **Save** in the black menu bar.



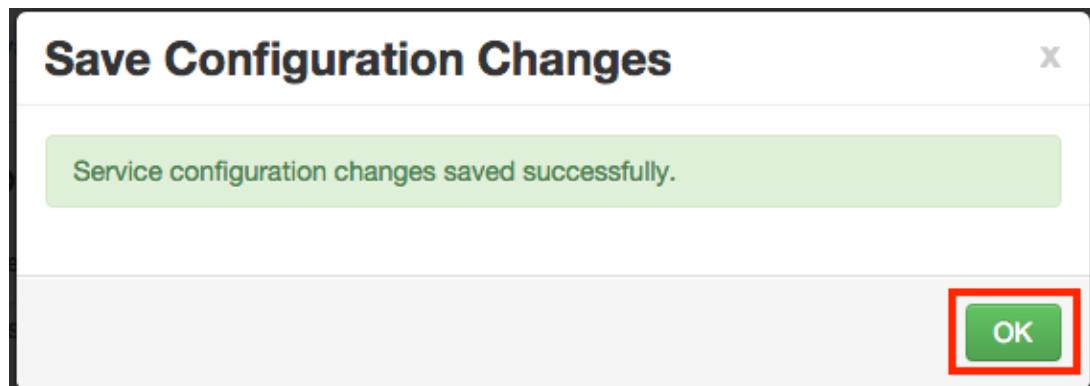
3. A Save Configuration pop-up appears. Type in a note describing the changes you just made, then click **Save**.



4. A Dependent Configuration pop-up appears. Click **OK** to confirm the configuration updates.

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.						
Property	Service	Config Group	File Name	Current Value	Recommended Value	
<input checked="" type="checkbox"/> ranger-knox-plugin-enabled	Knox	Default	ranger-knox-plugin-properties	No	Yes	
<input checked="" type="checkbox"/> content	Knox	Default	topology	<dependency> <gateway> <provider> <role>authentication</role> <chain> <chainProvider>chain</chainProvider> <enable> <true></enable> <parent> <name>sessionTimeout</name> <value>30</value> </parent> <parent> <name>main.ipcRealm</name> <value>org.apache.hadoop.gateway.sso.ipcrealm.Knox.ipcRealm</value> </parent> <parent> <name>main.idapRealm.userOnTemplate</name> <value>ou=people,dc=hadoop,dc=apache,dc=org</value> </parent> <parent> <name>main.ipcRealm</name> <value>(dap://)(knox_host_name)[3333]</value> </parent> <parent> <name>main.ipcRealm.comsFactory.authenticationMechanism</name> <value>simple</value> </parent> <parent> <name>vars</name> <value>authBasic</value> </parent> <parent> <name>vars</name> <value>authBasic</value> </parent> <parent> <name>vars</name> <value>authBasic</value> </parent>	<dependency> <gateway> <provider> <role>authentication</role> <chain> <chainProvider>chain</chainProvider> <enable> <true></enable> <parent> <name>sessionTimeout</name> <value>30</value> </parent> <parent> <name>main.ipcRealm</name> <value>org.apache.hadoop.gateway.sso.ipcrealm.Knox.ipcRealm</value> </parent> <parent> <name>main.idapRealm.userOnTemplate</name> <value>ou=people,dc=hadoop,dc=apache,dc=org</value> </parent> <parent> <name>main.ipcRealm</name> <value>(dap://)(knox_host_name)[3333]</value> </parent> <parent> <name>main.ipcRealm.comsFactory.authenticationMechanism</name> <value>simple</value> </parent> <parent> <name>vars</name> <value>authBasic</value> </parent> <parent> <name>vars</name> <value>authBasic</value> </parent> <parent> <name>vars</name> <value>authBasic</value> </parent>	

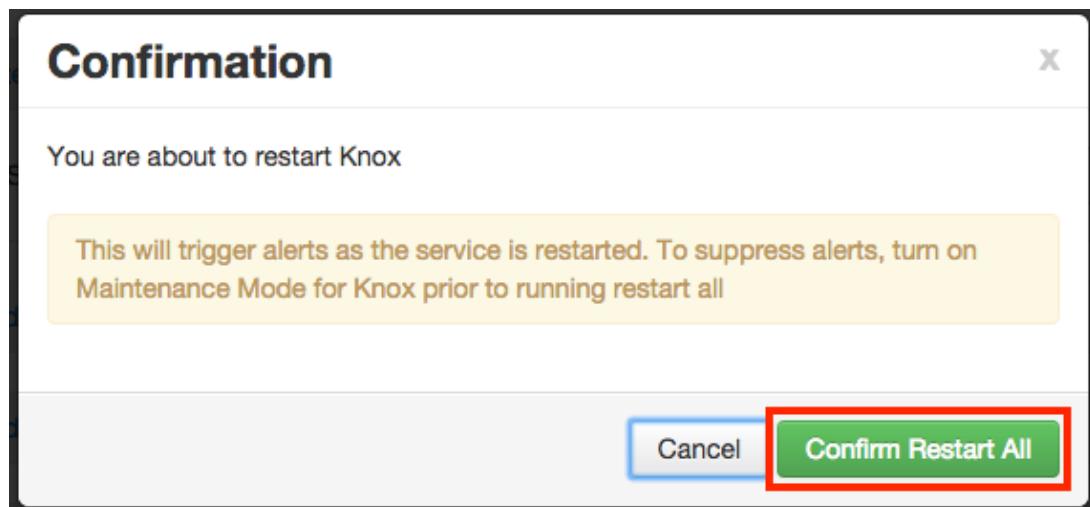
5. Click **OK** on the Save Configuration Changes pop-up.



6. Select **Knox** in the navigation menu, then select **Restart > Restart All Affected** to restart the Knox service and load the new configuration.

The screenshot shows the Ambari interface for a 'test_cluster'. On the left, a sidebar lists various services: HDFS, MapReduce2, YARN, Tez, Hive, HBase, Pig, Sqoop, Oozie, ZooKeeper, Falcon, Storm, Flume, Accumulo, Ambari Metrics, Atlas, Kafka, and Knox. The 'Knox' service is selected and highlighted in orange. The main content area is titled 'Knox Gateway' and contains configuration sections for 'Knox Gateway host' (set to c6403.ambari.apache.org) and 'Knox Master Secret'. At the top right, there is a 'Service Actions' dropdown with a 'Restart' option. Below it, a button labeled 'Restart All Affected' is highlighted with a red box.

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



- 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.

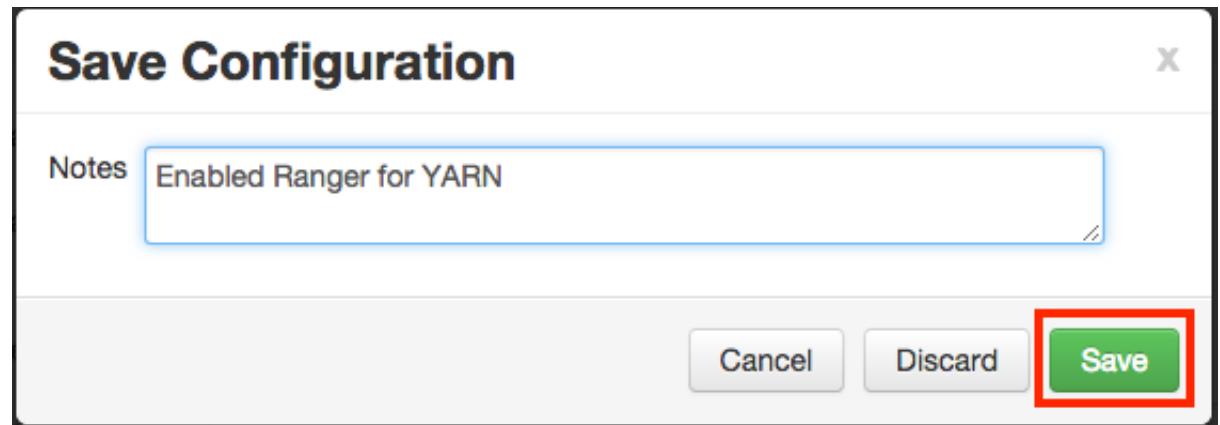
- On the Ranger Configs page, select the **Ranger Plugin** tab.

The screenshot shows the Ambari interface for the cluster 'c6401_TEST'. The left sidebar lists various 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 selected. The main panel displays the 'Ranger Plugin' configuration. It includes sections for HDFS, Hbase, and Knox Ranger Plugins, all set to 'ON'. The YARN Ranger Plugin is also set to 'ON'. The Storm Ranger Plugin is set to 'OFF'. The Hive Ranger Plugin is set to 'ON'. A black menu bar at the top has tabs for Summary, Configs, Quick Links, Service Actions, Hosts (1), Alerts, Admin, and a user dropdown for 'admin'. A note in the center says 'admin authored on Wed, Feb 03, 2016 11:28' with 'Discard' and 'Save' buttons.

2. Under YARN Ranger Plugin, select **On**, then click **Save** in the black menu bar.

This screenshot shows the same 'Ranger Plugin' configuration page as the previous one, but with a red box highlighting the 'ON' button for the YARN Ranger Plugin. The other settings remain the same: HDFS, Hbase, and Knox Ranger Plugins are ON; Storm Ranger Plugin is OFF; and Hive Ranger Plugin is ON.

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.

A screenshot of a 'Dependent Configurations' dialog box. It shows a table with the following data:

Property	Service	Config Group	File Name	Current Value	Recommended Value
ranger-knox-plugin-enabled	Knox	Default	ranger-knox-plugin-properties	Yes	No
content	Knox	Default	topology	<topology> <gateway> <provider> <role>authentication</role> <name> apacheShiroProvider</name> <enable>0</enable><enabled>1</enabled> <param> <name> sessionTimeout</name> <value> 60</value> </param> <param> <name> main_idapRealm</name> <value> apache.hadoop.gateway.s Kerberos.Knox_idapRealm</value> </param> <param> <name> main_idapRealm_useDirTemplate</name> <value>0</value> </param> <param> <name> main_idapRealm_dn</name> <value>ou=people,dc=hadoop,dc=apache,dc=org</value> </param> <param> <name> main_idapRealm_ipPort</name> <value>33389</value> </param> <param> <name> main_idapRealm_contextFactory</name> <value>simple</value> </param> <param> <name> authBasic</value> </param> <param> <name> authMechanisms</value> </param>	<topology> <gateway> <provider> <role>authentication</role> <name> apacheShiroProvider</name> <enable>1</enable><enabled>1</enabled> <param> <name> sessionTimeout</name> <value> 60</value> </param> <param> <name> main_idapRealm</name> <value> apache.hadoop.gateway.s Kerberos.Knox_idapRealm</value> </param> <param> <name> main_idapRealm_useDirTemplate</name> <value>0</value> </param> <param> <name> main_idapRealm_dn</name> <value>ou=people,dc=hadoop,dc=apache,dc=org</value> </param> <param> <name> main_idapRealm_ipPort</name> <value>33389</value> </param> <param> <name> main_idapRealm_contextFactory</name> <value>simple</value> </param> <param> <name> authBasic</value> </param> <param> <name> authMechanisms</value> </param>

In the bottom right corner of the dialog, there are 'Cancel' and 'OK' buttons, with 'OK' being highlighted with a red border.

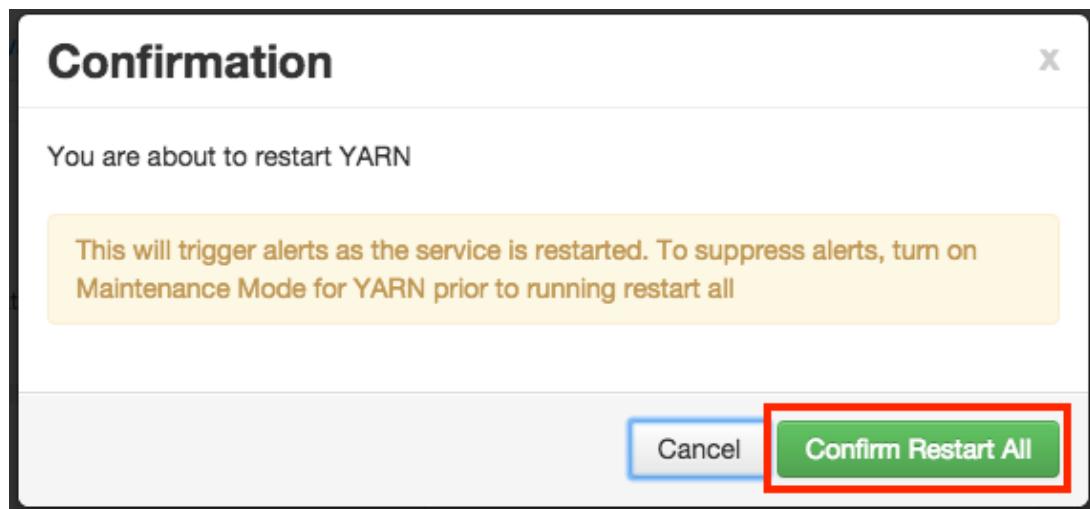
5. Click **OK** on the Save Configuration Changes pop-up.

A screenshot of a 'Save Configuration Changes' dialog box. It displays a message: 'Service configuration changes saved successfully.' In the bottom right corner, there is an 'OK' button, which is highlighted with a red border.

6. Select **YARN** in the navigation menu, then select **Restart > Restart All Affected** to restart the YARN service and load the new configuration.

The screenshot shows the Ambari interface for managing Hadoop services. The left sidebar lists various services: HDFS, MapReduce2, YARN (selected), Tez, Hive, HBase, Pig, Sqoop, Oozie, ZooKeeper, Falcon, Storm, Flume, Accumulo (6 alerts), Ambari Metrics (2 alerts), Atlas, Kafka, Knox, Mahout, Ranger, Slider, and Spark. The main content area is for the YARN service. It includes tabs for Summary, Heatmaps, and Configs. A prominent yellow banner at the top states "Restart Required: 4 Components on 1 Host". Below this are "Group" and "YARN Default (1)" dropdowns, and "Manage Config Groups" buttons. A large "Restart" button is highlighted with a red box. To its right is a "Restart All Affected" button, also highlighted with a red box. The "YARN Features" section contains "Node Labels" and "Pre-emption" options, both currently set to "Disabled". In the center, there are sections for "Memory" and "CPU", each with sub-sections for Node and Container configurations. Memory usage is shown with green progress bars, and CPU scheduling is set to "Disabled".

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



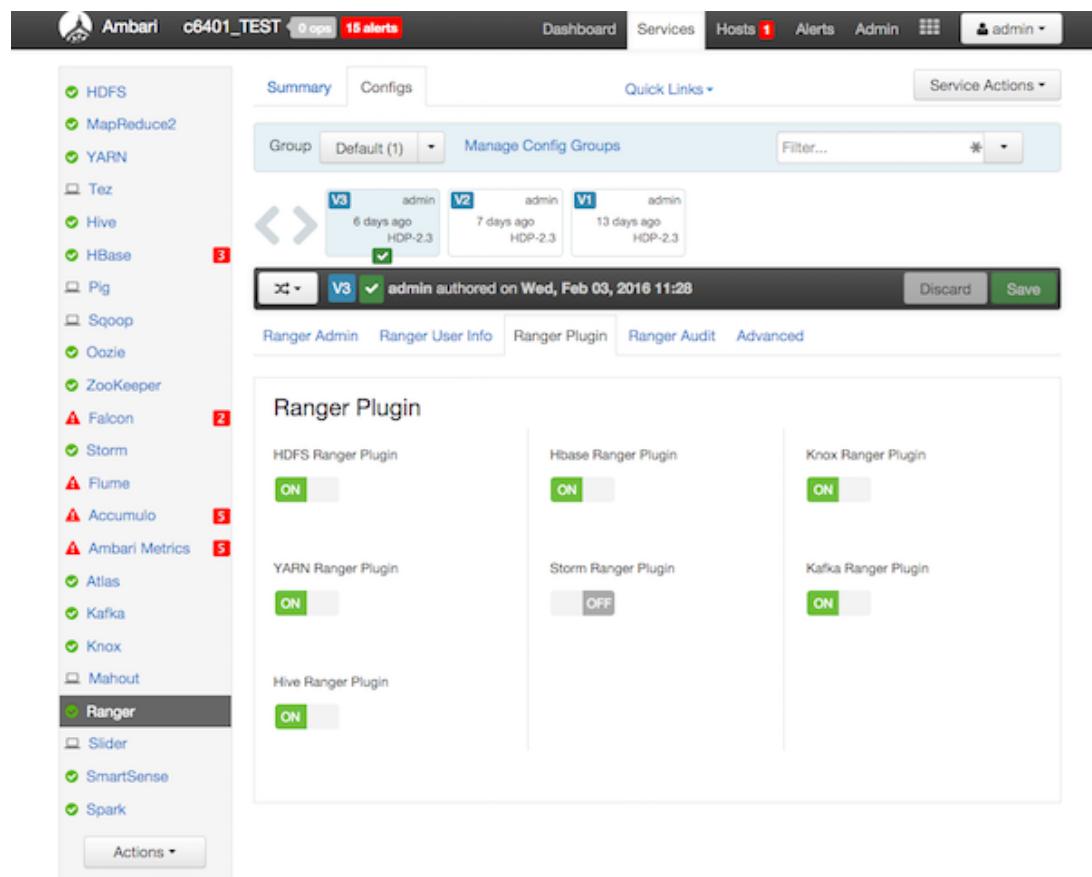
8. After YARN restarts, the Ranger plugin for YARN will be enabled. Other components may also require a restart.

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 Security](#) in the [Ambari Security Guide](#).

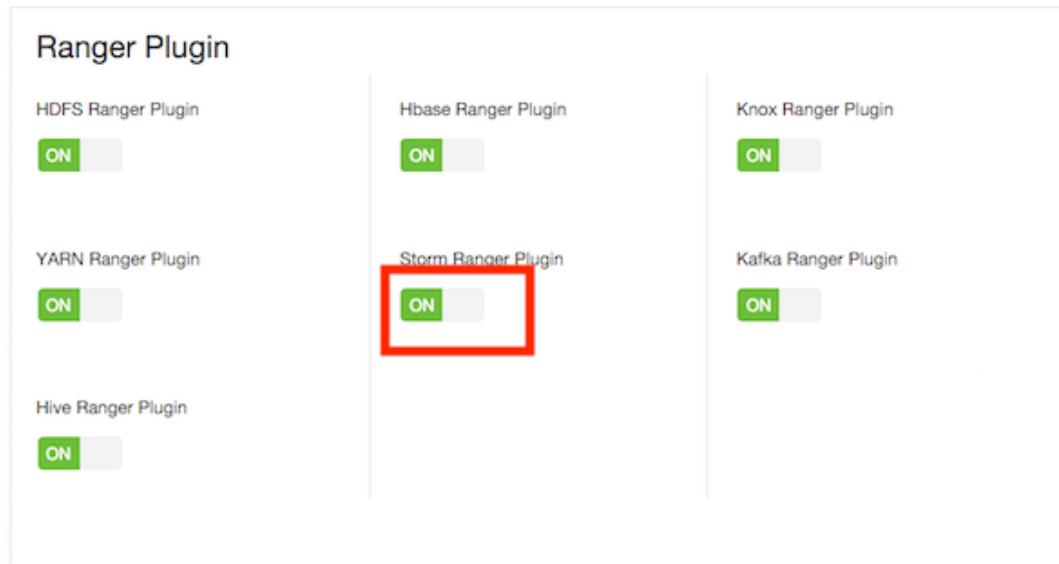
Use the following steps to enable the Ranger Storm plugin.

1. On the Ranger Configs page, select the **Ranger Plugin** tab.

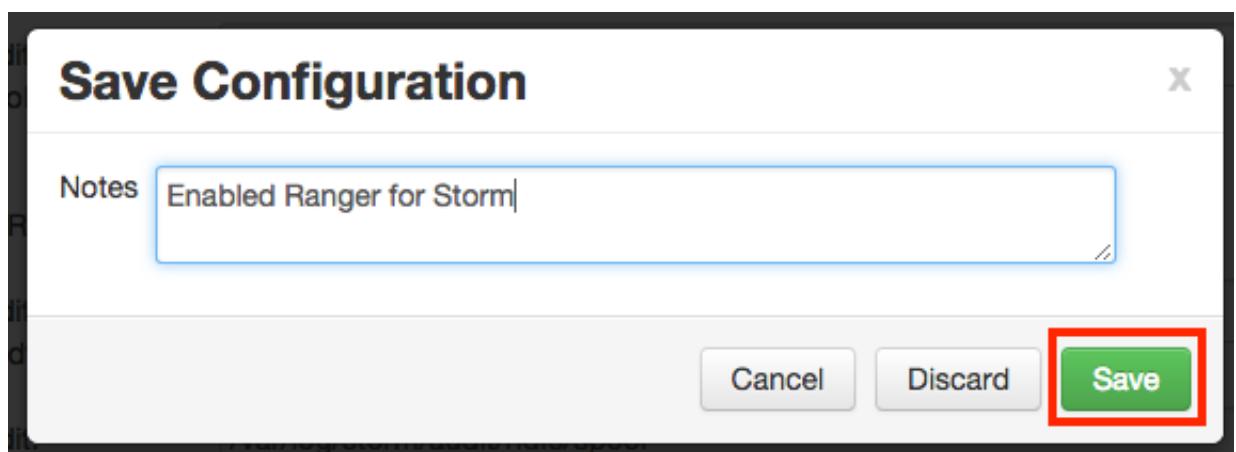


The screenshot shows the Ambari interface for managing configurations. The top navigation bar includes 'Ambari', the cluster name 'c6401_TEST', and status indicators for '0 ops' and '15 alerts'. The main menu on the left lists various Hadoop components: HDFS, MapReduce2, YARN, Tez, Hive, HBase, Pig, Sqoop, Oozie, ZooKeeper, Falcon, Storm, Flume, Accumulo, Ambari Metrics, Atlas, Kafka, Knox, Mahout, and Ranger. The 'Ranger' item is currently selected. The central area displays the 'Configs' tab for the 'Default (1)' group. A modal window is open over the main content, specifically for the 'Storm' configuration. This modal has tabs for 'Ranger Admin', 'Ranger User Info', 'Ranger Plugin' (which is selected), 'Ranger Audit', and 'Advanced'. Under the 'Ranger Plugin' tab, there are six configuration items with 'ON' or 'OFF' status indicators: HDFS Ranger Plugin (ON), Hbase Ranger Plugin (ON), Knox Ranger Plugin (ON), YARN Ranger Plugin (ON), Storm Ranger Plugin (OFF), and Hive Ranger Plugin (ON). At the bottom of the modal is a black menu bar with 'Discard' and 'Save' buttons, where 'Save' is highlighted in green.

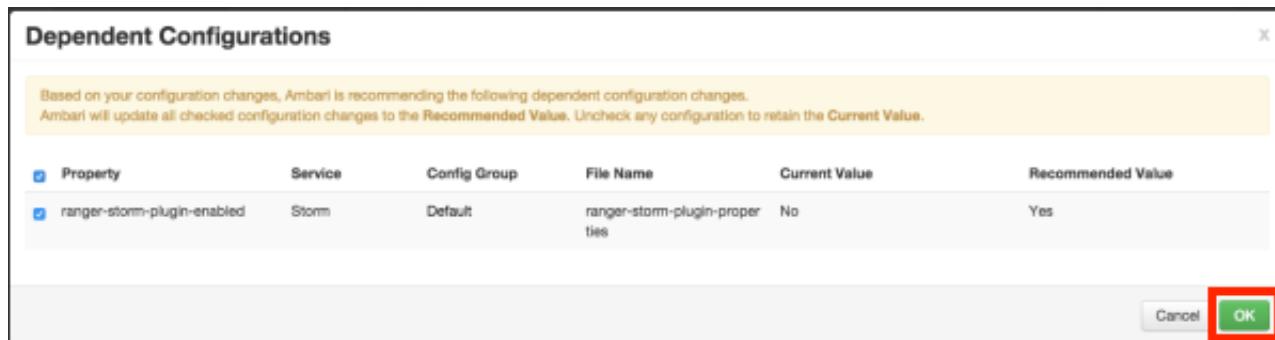
2. Under Storm Ranger Plugin, select **On**, then click **Save** in the black menu bar.



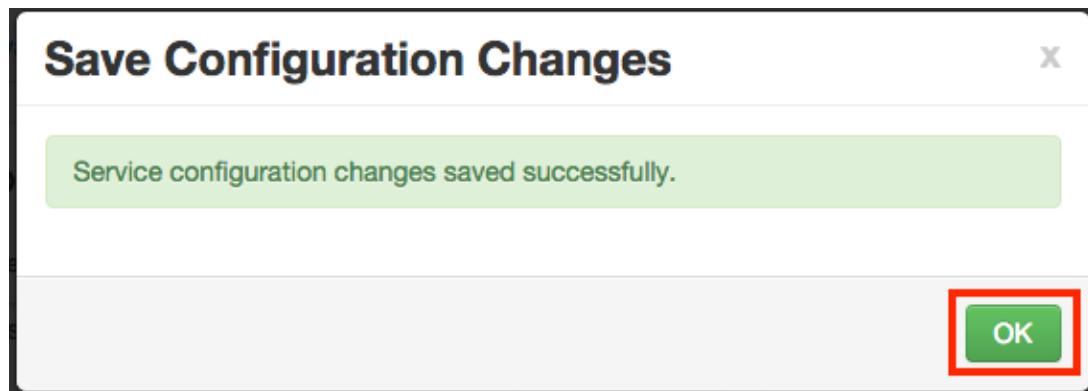
3. A Save Configuration pop-up appears. Type in a note describing the changes you just made, then click **Save**.



4. A Dependent Configuration pop-up appears. Click **OK** to confirm the configuration updates.



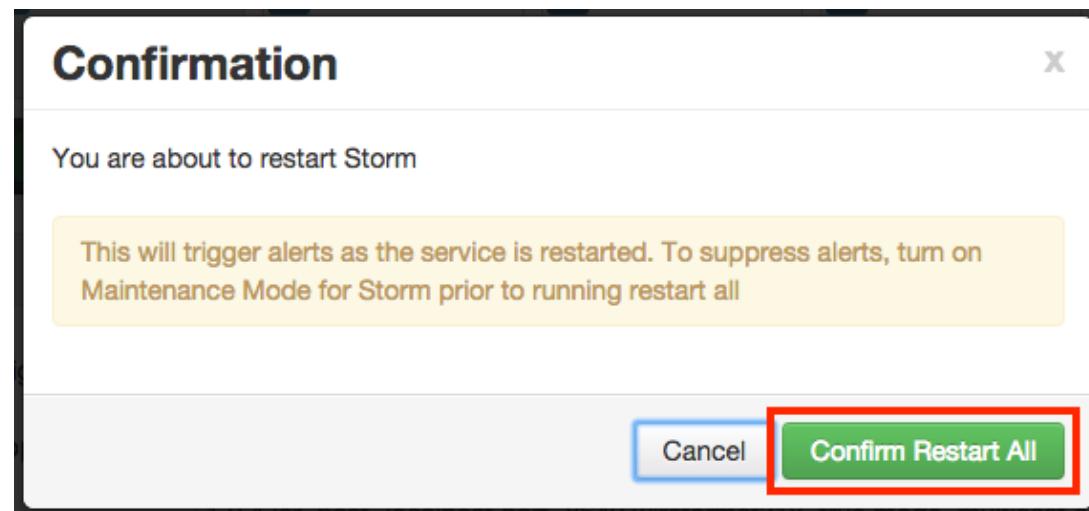
5. Click **OK** on the Save Configuration Changes pop-up.



6. Select **Storm** in the navigation menu, then select **Restart > Restart All Affected** to restart the Storm service and load the new configuration.

A screenshot of the Ambari UI interface. The left sidebar shows a list of services: HDFS, MapReduce2, YARN, Tez, Hive, HBase, Pig, Sqoop, Oozie, ZooKeeper, Falcon, and Storm. The "Storm" service is selected. The main content area is titled "Configs" and shows a configuration group named "Storm Default (1)". A yellow banner at the top right says "Restart Required: 4 Components on 1 Host". Below this, a table lists four components (V5, V4, V3, V2) with their last modified times and administrators. To the right of the table is a "Service Actions" dropdown with a "Restart" option, and within that, a "Restart All Affected" button, which is highlighted with a red rectangle. At the bottom of the configuration table, there are "Discard" and "Save" buttons. On the left side of the main content area, there is a section titled "Nimbus" containing various configuration parameters like "nimbus.reassign", "nimbus.childopts", etc., each with input fields and status indicators.

7. Click **Confirm Restart All** on the confirmation pop-up to confirm the Storm restart.



8. After Storm restarts, the Ranger plugin for Storm 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 to ensure you can use the different Ranger plugins on a Kerberos cluster. These plugins are:

1. [HDFS](#)
2. [Hive](#)
3. [HBase](#)
4. [Knox](#)



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.

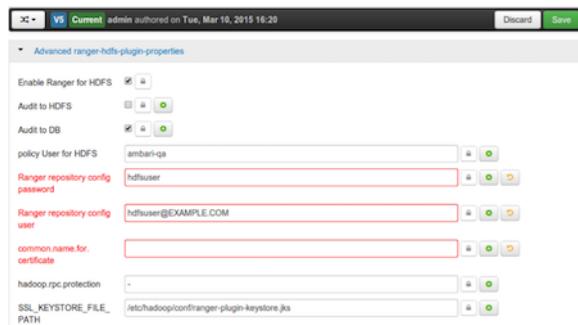


Table 3.16. HDFS Plugin Properties

Configuration Property Name	Value
Ranger service config user	rangerhdfslookup@example.com
Ranger service 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 HBase 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.
4. Click the **Config** tab and navigate to *advanced ranger-hive-plugin-properties*.
5. Update the following properties with the values listed in the table below.

Table 3.17. Hive Plugin Properties

Configuration Property Name	Value
Ranger service config user	rangerhivelookup@example.com
Ranger service config password	rangerhivelookup

Configuration Property Name	Value
common.name.for.certificate	blank

6. 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.

1. Create the system (OS) user `rangerhbaselookup`. Make sure this user is synced to Ranger Admin (under *users/groups* tab in the Ranger Admin UI).
2. Create a Kerberos principal for `rangerhbaselookup` by entering the following command:
 - `kadmin.local -q 'addprinc -pw rangerhbaselookup rangerhbaselookup@example.com'`
3. Navigate to the HBase service.
4. Click the **Config** tab and go to *advanced ranger-hbase-plugin-properties*.
5. Update the following properties with the values listed in the table below.

Table 3.18. HBase Plugin Properties

Configuration Property Name	Value
Ranger service config user	<code>rangerhbaselookup@example.com</code>
Ranger service config password	<code>rangerhbaselookup</code>
common.name.for.certificate	blank

6. After updating these properties, click **Save** and then restart the HBase service.

3.1.5.4. Knox

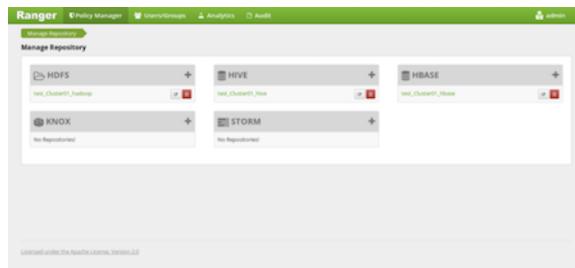
To enable the Ranger Knox plugin on a Kerberos-enabled cluster, perform the steps described below.

1. 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).
2. 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 3.19. 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. Knox Service Manager

9. Click the repository (clusterName_hadoop) **Edit** option under the HDFS box.

Figure 3.7. Knox Service Edit

10. Update the following properties listed in the table below under the Config Properties section:

Table 3.20. Knox Configuration Properties

Configuration Property Name	Value
fs.default.name	hdfs

Configuration Property Name	Value
hadoop.rpc.protection	blank
common.name.for.certificate	blank

11.Click **Named Test Connection**. You should see a *Connected Successfully* dialog box appear.

12.Click **Save**.

3.2. Using Ranger to Provide Authorization in Hadoop

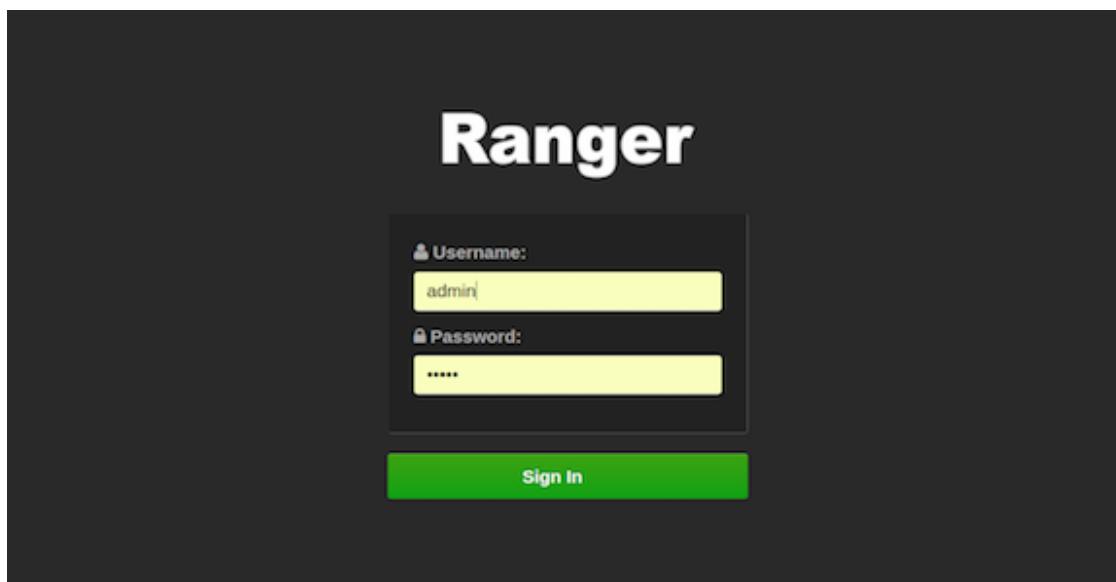
Once a user has been authenticated, their access rights must be determined. Authorization defines access rights to resources, or privileges of use (permissions) for users. For example, a user may be allowed to create a policy and view reports, but not allowed to edit users and groups. Ranger authorization controls this access rights process for:

- Configuring Services
- Policy Management
- Users/Groups and Permissions Administration
- Reports Administration

For more information on Ranger authorization, see the [Authorization](#) overview.

3.2.1. 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 username and password, then click **Sign In**.



Ranger Console Home Page

The screenshot shows the Ranger Console Home Page. At the top, there is a navigation bar with tabs for 'Ranger', 'Access Manager', 'Audit', and 'Settings'. The 'Service Manager' tab is selected. Below the navigation bar, there is a section titled 'Service Manager' with a green header. This section contains eight groups, each with a plus sign icon to its right:

- HDFS**: Contains 'test-hdfs' and 'c6401_TEST_hadoop'.
- HBASE**: Contains 'hbase-test' and 'c6401_TEST_hbase'.
- HIVE**: Contains 'c6401_TEST_hive'.
- YARN**: Contains 'c6401_TEST_yarn'.
- KNOX**: Contains 'c6401_TEST_knox'.
- STORM**: Contains 'c6401_TEST_storm'.
- SOLR**: Contains 'c6401_TEST_solr'.
- KAFKA**: Contains 'c6401_TEST_kafka'.

Ranger Login Console

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

The screenshot shows the Ranger Console Home Page with a red arrow pointing to the user name 'admin' in the top right corner of the navigation bar.

To log out of the Ranger Console, click your user name in the top menu, then select **Log Out**.

The screenshot shows the Ranger Console Home Page with a red arrow pointing to the 'Log Out' button in the top right corner of the navigation bar.

3.2.2. Console Operations Summary

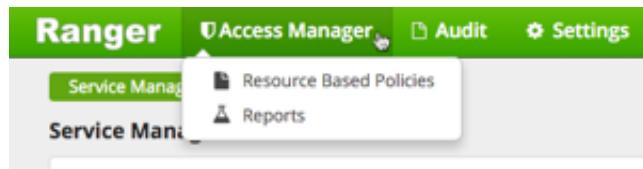
The Ranger console controls four types of functions:

- **The Service Manager** - (visible upon user login) creates, administers, and shows reports on service policies.

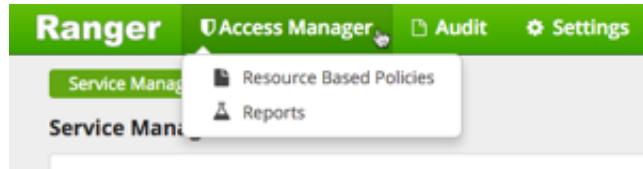
The screenshot shows the Ranger Console Home Page. At the top, there is a navigation bar with tabs for 'Ranger', 'Access Manager', 'Audit', and 'Settings'. The 'Service Manager' tab is selected. Below the navigation bar, there is a section titled 'Service Manager' with a green header. This section contains eight groups, each with a plus sign icon to its right:

- HDFS**: Contains 'test-hdfs' and 'c6401_TEST_hadoop'.
- HBASE**: Contains 'hbase-test' and 'c6401_TEST_hbase'.
- HIVE**: Contains 'c6401_TEST_hive'.
- YARN**: Contains 'c6401_TEST_yarn'.
- KNOX**: Contains 'c6401_TEST_knox'.
- STORM**: Contains 'c6401_TEST_storm'.
- SOLR**: Contains 'c6401_TEST_solr'.
- KAFKA**: Contains 'c6401_TEST_kafka'.

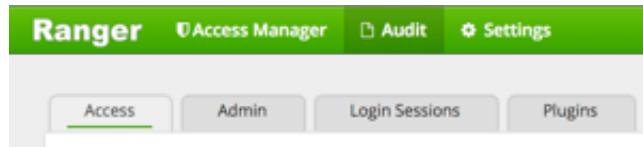
The Access Manager menu includes the Resource Based Policies and Reports menu options.



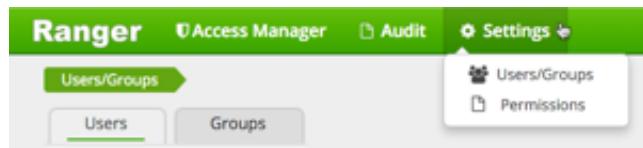
- **Resource Based Policies** - creates, administers, and shows reports on service policies. Shows the Access Manager/Service Manager.



- **The Audit tab** - monitors user activity at the resource level, and conditional auditing based on users, groups, or time. The Audit page includes the Access, Admin, Login Sessions, and Plugins tabs.



- **The Settings tab** - manages users and groups, and assigns policy permissions to users and groups. The dropdown menu includes the Users/Groups and Permissions menu options.



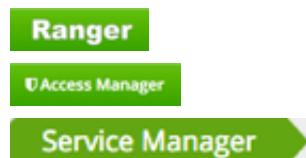
3.2.3. Configuring Services

The Ranger Access Manager is open by default in the Ranger Console. To return to the Access Manager from any tab in the Ranger Console, go to the top left corner of the console and click **Ranger**, **Access Manager**, **Service Manager**, or **Access Manager>Resource Based Policies**.



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 service to Resource Based Policies, click the



icon in the applicable box on the Service Manager page. Enter the required configuration settings, then click **Add**



).

- To edit a service, click



to the right of the entry for that service. Resource Based Policies displays an expanded view of that service, including a list of the policies it contains, their status, and the groups designated to administer those policies.

- To delete a service from Resource Based Policies, click



to the right of the entry for that service. Deleting a service also deletes all the policies within that service.

Ranger Service Manager Console

This section describes how to configure services in:

- [HBase](#)
- [HDFS](#)
- [Hive](#)
- [Kafka](#)
- [Knox](#)
- [Solr](#)
- [Storm](#)

- YARN

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



icon next to HBase.

The Create Service page appears.

The screenshot shows the 'Create Service' page in the Ranger Service Manager. The top navigation bar includes 'Ranger', 'Access Manager', 'Audit', and 'Settings'. Below it, a breadcrumb trail shows 'Service Manager > Create Service'. The main form is titled 'Create Service' and contains two sections: 'Service Details:' and 'Config Properties:'.

Service Details :

- Service Name *: A text input field.
- Description: A text area.
- Active Status: Radio buttons for 'Enabled' (selected) and 'Disabled'.

Config Properties :

- Username *: A text input field.
- Password *: A text input field.
- hadoop.security.authentication *: A dropdown menu set to 'Simple'.
- hbase.master.kerberos.principal: A text input field.
- hbase.security.authentication *: A dropdown menu set to 'Simple'.
- hbase.zookeeper.property.clientPort *: A text input field containing '2181'.
- hbase.zookeeper.quorum *: A text input field.
- zookeeper.znode.parent *: A text input field containing '/hbase'.
- Common Name for Certificate: A text input field.
- Add New Configurations: A table with columns 'Name' and 'Value'. It has one row with a red delete button and a '+' button below it.

At the bottom are 'Test Connection' and 'Add' / 'Cancel' buttons.

2. Enter the following information on the Create Service page:

Table 3.21. 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.

Table 3.22. 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



icon next to HDFS.

The Create Service page appears.

2. Enter the following information on the Create Service page:

Table 3.23. 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.

Table 3.24. 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	<p>hdfs://NAMENODE_FQDN:8020</p> <p>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.</p> <p>This field was formerly named fs.defaultFS.</p>
Authorization Enabled	Authorization involves restricting access to resources. If enabled, user need authorization credentials.
Authentication Type	<p>The type of authorization in use, as noted in the hadoop configuration file core-site.xml; either simple or Kerberos. (Required only if authorization is enabled).</p> <p>This field was formerly named hadoop.security.authorization.</p>
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	<p>The name of the certificate.</p> <p>This field is interchangeably named Common Name For Certificate and Ranger Plugin SSL CName in Create Service pages.</p>
Add New Configurations	Add any other new configuration(s).

3. Click **Test Connection**.

 Test Connection

4. Click **Add**.

 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



next to Hive.

The Create Service page appears.

A screenshot of the 'Create Service' page in the Ranger UI. The top navigation bar includes 'Ranger', 'Access Manager', 'Audit', and 'Settings'. Below it, a breadcrumb trail shows 'Service Manager > Create Service'. The main form has two sections: 'Service Details' and 'Config Properties'. In 'Service Details', there are fields for 'Service Name *' (empty), 'Description' (empty), and 'Active Status' (radio buttons for 'Enabled' and 'Disabled' with 'Enabled' selected). In 'Config Properties', there are fields for 'Username *' (empty), 'Password *' (empty), 'jdbc.driverClassName *' (set to 'org.apache.hive.jdbc.HiveDriver'), 'jdbc.url *' (empty), and 'Common Name for Certificate' (empty). Below these is a table for 'Add New Configurations' with columns 'Name' and 'Value', showing one entry with a delete button. At the bottom are 'Test Connection' and 'Add' (green) and 'Cancel' buttons.

2. Enter the following information on the Create Service page:

Table 3.25. 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.

Table 3.26. 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
jdbc.url	The complete connection URL, including port and database name. (Default port: 10000.) For example, on the sandbox, jdbc:hive2://sandbox:10000/.
Common Name For Certificate	<p>The name of the certificate.</p> <p>This field is interchangeably named Common Name For Certificate and Ranger Plugin SSL CName in Create Service pages.</p>
Add New Configurations	Add any other new configuration(s).

3. Click **Test Connection**.



4. Click **Add**.



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



next to Kafka.

The Create Service page appears.

2. Enter the following information on the Create Service page:

Table 3.27. 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.

Table 3.28. 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**.

 Test Connection

4. Click **Add**.

 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



next to Knox.

The Create Service page appears.

2. Enter the following information on the Create Service page:

Table 3.29. 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.

Table 3.30. 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.

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**.

 Test Connection

4. Click **Add**.

 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



next to Solr.

The Create Service page appears.

2. Enter the following information on the Create Service page:

Table 3.31. 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.

Table 3.32. 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	<code>http://Solr_host:6083</code>
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**.

A rectangular button with a thin gray border and a white background. Inside, the text "Test Connection" is centered in a small, black, sans-serif font.

4. Click **Add**.

A small, rounded rectangular button with a green background and white text. The word "Add" is written in a white, sans-serif font.

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



next to Storm.

The Create Service page appears.

2. Enter the following information on the Create Service page:

Table 3.33. 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.

Table 3.34. 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	Hostname of nimbus format, in the form: http:// <i>ipaddress</i> :8080. This field was formerly named nimbus.url.
Common Name For Certificate	The name of the certificate.

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**.

A rectangular button with a thin gray border and a white background, containing the text "Test Connection" in a small, dark font.

4. Click **Add**.

A small, rounded rectangular button with a green background and white text, containing the word "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



next to YARN.

The Create Service page appears.

2. Enter the following information on the Create Service page:

Table 3.35. 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.

Table 3.36. 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.
Authentication Type	The type of authorization in use, as noted in the hadoop configuration file core-site.xml; either simple or Kerberos. (Required only if authorization is enabled).

Field name	Description
	This field was formerly named hadoop.security.authorization.
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. Policy Management

To examine the policies associated with each service, go to the service where the service resides and click



The Ranger Service Manager view appears and an expanded view of that service displays, with the policies listed beneath. The policy view includes a search window.

- To add a new policy to the service, click **Add New Policy**

).

The form looks slightly different, depending on the type of service to which you are adding the policy.

- To edit a policy, click



to the right of the entry for that service. Resource Based Policies displays an expanded view of that policy.

- To delete a policy, click



the

Delete icon to the right of the entry for that service.

The screenshot shows the Ranger Service Manager interface. At the top, there's a navigation bar with 'Ranger', 'Access Manager', 'Audit', 'Settings', and a user icon for 'admin'. Below the navigation is a breadcrumb trail: 'Service Manager > c6401_TEST_knox Policies'. A modal window titled 'Add new policy' is open on the right, containing a search bar and a green 'Add New Policy' button. In the center, there's a table titled 'List of Policies : c6401_TEST_knox' with one row visible. The table columns are: Policy ID (9), Policy Name (c6401_TEST_knox-1-20160203193206), Status (Enabled), Audit Logging (Enabled), Groups, Users, and Action. The 'Action' column contains three icons: a pencil for 'Edit policy', a trash can for 'Delete policy', and a refresh symbol. Red arrows point from the text labels 'Edit policy', 'Delete policy', and 'Add New Policy' to their respective icons in the screenshot.

This section describes the requirements for policy creation in

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

3.2.4.1. 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 screenshot shows the Service Manager interface for HBase. At the top, there's a header with a folder icon and the text 'HBASE'. Below it is a list of services:

- 'hbase-test' is selected, indicated by a cursor icon over its name.
- 'c6401_TEST_hbase'

Each service entry has two buttons on the right: a pencil icon for editing and a trash can icon for deleting.

The List of Policies page appears.

The screenshot shows the 'List of Policies : Example-Service' page. The top navigation bar includes tabs for 'Ranger', 'Access Manager', 'Audit', and 'Settings', with 'admin' selected. Below the navigation is a breadcrumb trail: 'Service Manager > Example-Service Policies'. The main area is titled 'List of Policies : Example-Service'. It features a search bar and a green 'Add New Policy' button. A table lists policies with columns: Policy ID, Policy Name, Status, Audit Logging, Groups, Users, and Action. One row is visible, showing Policy ID 13, Policy Name 'Example-Service-1-20160211205602', Status 'Enabled', Audit Logging 'Enabled', Groups empty, Users 'admin', and Action buttons.

2. Click **Add New Policy**.

Add New Policy

The Create Policy console appears.

The screenshot shows the 'Create Policy' page in the Ranger Access Manager. At the top, there are tabs for 'Access Manager', 'Audit', and 'Settings', with 'admin' selected. The main area is titled 'Create Policy'.

Policy Details:

- Policy Name: (Enabled)
- HBase Table: (Include)
- HBase Column-family: (Include)
- HBase Column: (Include)
- Description:

Audit Logging: YES

User and Group Permissions:

Permissions	Select Group	Select User	Permissions	Delegate Admin
Add	Select Group	Select User	Add Permissions	X
+ Add				

At the bottom are 'Add' and 'Cancel' buttons.

3. Complete the Create Policy page as follows:

Table 3.37. 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.
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 3.38. User and Group Permissions

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
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).

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

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

4. Click **Add**.



3.2.4.2. Provide User Access to HBase Database Tables from the Command Line

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

- GRANT

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

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

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

- REVOKE

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

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



Note

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

3.2.4.3. Create an HDFS Policy

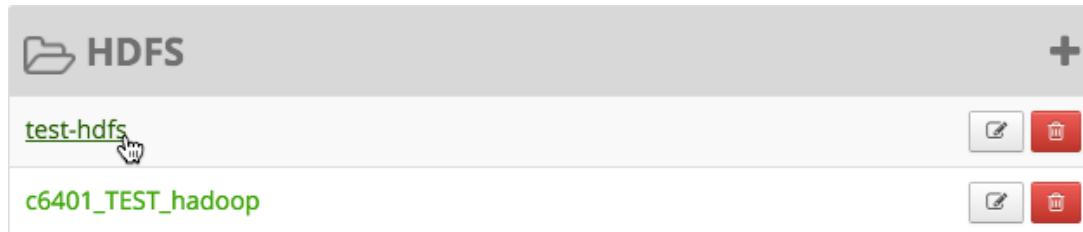
Through configuration, Apache Ranger enables both Ranger policies and HDFS permissions to be checked for a user request. When the NameNode receives a user request, the Ranger

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.

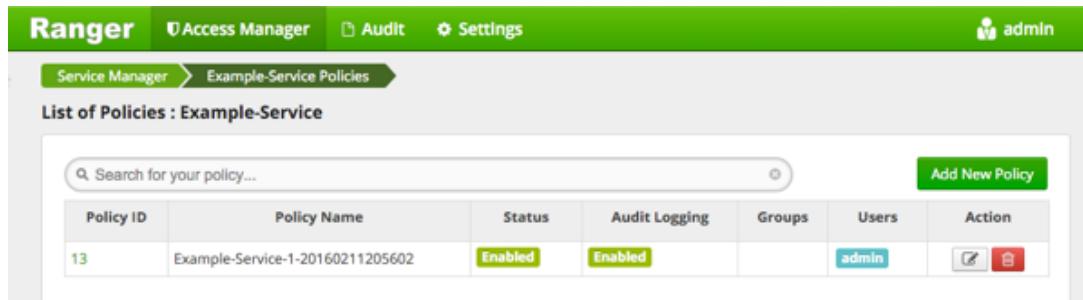
To add a new policy to an existing HDFS service:

1. On the Service Manager page, select an existing service under HDFS.



The screenshot shows the 'HDFS' service in the Service Manager. It lists two services: 'test-hdfs' and 'c6401_TEST_hadoop'. Each service has edit and delete icons next to it.

The List of Policies page appears.



The screenshot shows the 'List of Policies : Example-Service' page. It includes a search bar, an 'Add New Policy' button, and a table with columns: Policy ID, Policy Name, Status, Audit Logging, Groups, Users, and Action. One policy is listed: Policy ID 13, Policy Name 'Example-Service-1-20160211205602', Status 'Enabled', Audit Logging 'Enabled', Groups empty, Users 'admin', and Action buttons.

2. Click **Add New Policy**.



The Create Policy page appears.

The screenshot shows the 'Create Policy' interface in the Ranger web UI. At the top, there are tabs for 'Access Manager', 'Audit', and 'Settings', with 'Access Manager' being the active tab. A navigation breadcrumb shows 'Service Manager > test-hdfs Policies > Create Policy'. The main form is titled 'Create Policy' and contains a 'Policy Details' section with fields for 'Policy Name' (disabled), 'Resource Path' (recursive), 'Description', and 'Audit Logging' (YES). Below this is a 'User and Group Permissions' section with a table for managing permissions, and buttons for 'Add' and 'Cancel'.

3. Complete the Create Policy page as follows:

Table 3.39. 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 3.40. User and Group Permissions

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).
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.

Label	Description
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).

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

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

4. Click **Add**.



3.2.4.4. 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.

A screenshot of the Service Manager interface. At the top, there's a header with a folder icon and the word "HIVE". Below it, a list shows a service named "c6401 TEST hive" with a green link-like style. To the right of the service name are two small icons: a pencil for editing and a trash can for deleting. A cursor arrow points towards the service name.

The List of Policies page appears.

A screenshot of the "List of Policies : Example-Service" page. The top navigation bar includes tabs for "Ranger", "Access Manager", "Audit", and "Settings", with "admin" selected. Below the navigation is a breadcrumb trail: "Service Manager > Example-Service Policies". The main area is titled "List of Policies : Example-Service". It features a search bar with placeholder text "Search for your policy..." and an "Add New Policy" button. A table lists one policy entry:

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 console appears.

3. Complete the Create Policy page as follows:

Table 3.41. Policy Details

Field	Description
Policy Name	Enter an appropriate policy name. This name cannot be duplicated across the system. This field is mandatory.
Hive Database	Select the appropriate database. Multiple databases can be selected for a particular policy. This field is mandatory.
Table/UDF Drop-down	To continue adding a table-based policy, keep Table selected. To add a User Defined Function (UDF), select UDF.
Hive Column	For the selected database, select table(s) for which the policy will be applicable.
Description	(Optional) Describe the purpose of the policy.
Audit Logging	Specify whether this policy is audited. (De-select to disable auditing).

Table 3.42. User and Group Permissions

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
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).

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

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

4. Click **Add**.



3.2.4.5. 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 is `grant select on table default.hivesmoke22074 to user user1;`

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

- REVOKE

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

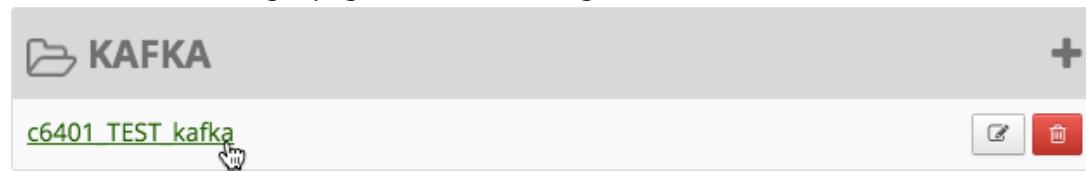
For example, to revoke the SELECT rights of user1 to the table default.hivesmoke22074, the command is `revoke select on table default.hivesmoke22074 from user user1;`

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

3.2.4.6. Create a Kafka Policy

To add a new policy to an existing Kafka service:

- On the Service Manager page, select an existing service under Kafka.



The List of Policies page appears.

The screenshot shows the 'List of Policies : Example-Service' page. The header includes tabs for Service Manager, Example-Service Policies, and a breadcrumb path: Service Manager > Example-Service Policies. The main content is a table titled 'List of Policies' with columns: Policy ID, Policy Name, Status, Audit Logging, Groups, Users, and Action. There is one row in the table with Policy ID 13, Policy Name 'Example-Service-1-20160211205602', Status 'Enabled', Audit Logging 'Enabled', Groups empty, Users 'admin', and Action buttons.

- Click Add New Policy.

Add New Policy

The Create Policy console appears.

The screenshot shows the 'Create Policy' page. The header includes tabs for Service Manager, c6401_TEST_kafka Policies, and Create Policy. The main content is divided into two sections: 'Policy Details:' and 'User and Group Permissions:'. In 'Policy Details:', there are fields for Policy Name (set to 'enabled'), Topic (set to 'include'), and Description. In 'User and Group Permissions:', there are tabs for Permissions, Select Group, Select User, Policy Conditions, Permissions, and Delegate Admin. Below these tabs are 'Add Conditions' and 'Add Permissions' buttons. At the bottom of the page are 'Add' and 'Cancel' buttons.

- Complete the Create Policy page as follows:

Table 3.43. 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.

Field	Description
Audit Logging	Specify whether this policy is audited. (De-select to disable auditing).

Table 3.44. User and Group Permissions

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).
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).

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

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

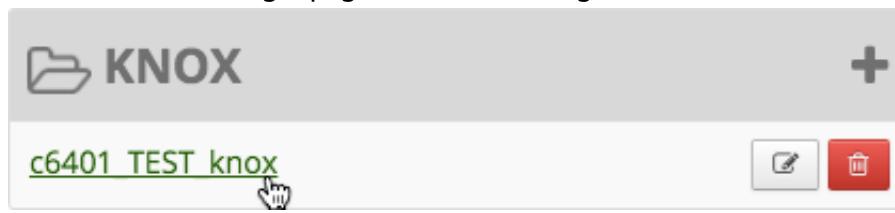
4. Click **Add**.



3.2.4.7. Create a Knox Policy

To add a new policy to an existing Knox service:

1. On the Service Manager page, select an existing service under Knox.



The List of Policies page appears.

2. Click Add New Policy.

Add New Policy

The Create Policy console appears.

3. Complete the Create Policy page as follows:

Table 3.45. 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.

Field	Description
Audit Logging	Specify whether this policy is audited. (De-select to disable auditing).

Table 3.46. User and Group Permissions

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).
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).

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.

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

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

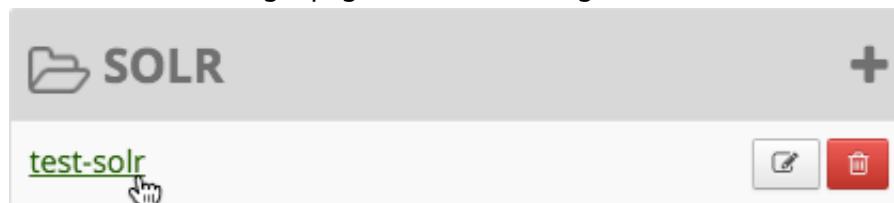
4. Click **Add**.



3.2.4.8. Create a Solr Policy

To add a new policy to an existing Solr service:

1. On the Service Manager page, select an existing service under Solr.



The List of Policies page appears.

The screenshot shows the Ranger UI with the title 'List of Policies : Example-Service'. At the top right is the user 'admin'. Below the title is a search bar and an 'Add New Policy' button. A table lists policies with columns: Policy ID, Policy Name, Status, Audit Logging, Groups, Users, and Action. One policy is listed: 'Example-Service-1-20160211205602' with status 'Enabled' and audit logging 'Enabled'. The 'Action' column for this row contains two icons: a pencil and a trash can.

2. Click Add New Policy.

Add New Policy

The Create Policy console appears.

The screenshot shows the 'Create Policy' page. At the top right is the user 'admin'. The page has sections for 'Policy Details' and 'User and Group Permissions'. In 'Policy Details', fields include 'Policy Name' (set to 'enabled'), 'Solr Collection' (set to 'include'), 'Description' (empty), and 'Audit Logging' (set to 'YES'). In 'User and Group Permissions', there are tabs for 'Permissions', 'Select Group' (with 'Select Group' and '+'), 'Select User' (with 'Select User' and '+'), 'Policy Conditions' (with 'Add Conditions' and '+'), 'Permissions' (with 'Add Permissions' and '+'), and 'Delegate Admin' (with a checkbox and a red 'X'). At the bottom are 'Add' and 'Cancel' buttons.

3. Complete the Create Policy page as follows:

Table 3.47. 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	http://host_ip:6083/solr
Description	(Optional) Describe the purpose of the policy.
Audit Logging	Specify whether this policy is audited. (De-select to disable auditing).

Table 3.48. User and Group Permissions

Label	Description
Select Group	Specify the group to which this policy applies. To designate the group as an Administrator for the chosen

Label	Description
	resource, specify Admin permissions. (Administrators can create child policies based on existing policies).
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).

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

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

4. Click **Add**.



3.2.4.9. Create a Storm Policy

To add a new policy to an existing Storm service:

1. On the Service Manager page, select an existing service under Storm.

The screenshot shows the Service Manager interface. At the top, there's a header with a folder icon and the word 'STORM'. Below it, a list of services includes 'test-storm' which is highlighted with a green underline and has a cursor pointing at it. To the right of the service names are two icons: a pencil for editing and a trash can for deleting.

The List of Policies page appears.

The screenshot shows the 'List of Policies : Example-Service' page. The top navigation bar includes 'Ranger', 'Access Manager', 'Audit', 'Settings', and a user icon labeled 'admin'. Below the navigation is a breadcrumb trail: 'Service Manager > Example-Service Policies'. The main area is titled 'List of Policies : Example-Service'. It features a search bar and an 'Add New Policy' button. A table lists policies with columns: Policy ID, Policy Name, Status, Audit Logging, Groups, Users, and Action. One row is visible, showing Policy ID 13, Policy Name 'Example-Service-1-20160211205602', Status 'Enabled', Audit Logging 'Enabled', Groups empty, Users 'admin', and Action buttons.

2. Click **Add New Policy**.



The Create Policy console appears.

The screenshot shows the 'Create Policy' page in the Ranger Access Manager. At the top, there are tabs for Service Manager, test-storm Policies, and Create Policy. The main area is titled 'Create Policy'. It has sections for 'Policy Details' (Policy Name, Storm Topology, Description, Audit Logging) and 'User and Group Permissions' (with tables for Select Group, Select User, Permissions, and Delegate Admin). Buttons for 'Add' and 'Cancel' are at the bottom.

3. Complete the Create Policy page as follows:

Table 3.49. 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 3.50. User and Group Permissions

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).
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).

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 3.51. Knox User and Group Permissions

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

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

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

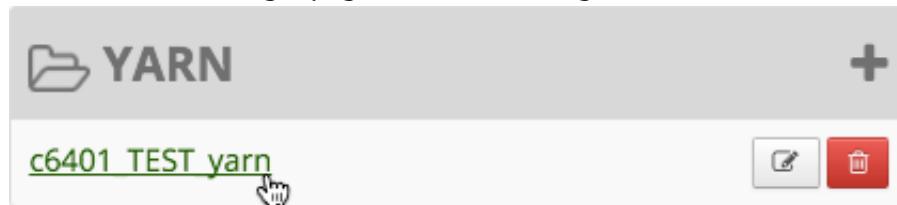
4. Click **Add**.



3.2.4.10. 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.

The screenshot shows the Ranger Access Manager interface. At the top, there are tabs for 'Ranger', 'Access Manager', 'Audit', and 'Settings'. The 'Access Manager' tab is selected. Below the tabs, a breadcrumb navigation shows 'Service Manager > Example-Service Policies > List of Policies : Example-Service'. The main area is titled 'List of Policies : Example-Service'. It features a search bar and a table with columns: Policy ID, Policy Name, Status, Audit Logging, Groups, Users, and Action. One row in the table is highlighted, showing Policy ID 13, Policy Name 'Example-Service-1-20160211205602', Status 'Enabled', Audit Logging 'Enabled', Groups and Users both containing 'admin', and Action buttons. A prominent green 'Add New Policy' button is located at the bottom right of the table.

2. Click Add New Policy.



The Create Policy console appears.

The screenshot shows the 'Create Policy' page. At the top, there are tabs for 'Ranger', 'Access Manager', 'Audit', and 'Settings'. The 'Access Manager' tab is selected. Below the tabs, a breadcrumb navigation shows 'Service Manager > c6401_TEST_yarn Policies > Create Policy'. The main area is titled 'Create Policy'. It has two main sections: 'Policy Details' and 'User and Group Permissions'. In the 'Policy Details' section, there are fields for 'Policy Name' (with a required asterisk), 'Queue' (with a required asterisk), 'Description', and 'Audit Logging' (set to 'YES'). In the 'User and Group Permissions' section, there is a table with columns: 'Permissions', 'Select Group' (with a dropdown menu 'Select Group'), 'Select User' (with a dropdown menu 'Select User'), 'Permissions' (with a 'Add Permissions' button and a '+' icon), and 'Delegate Admin' (with a checkbox). At the bottom of the page are 'Add' and 'Cancel' buttons.

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.
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 3.53. User and Group Permissions

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).
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).

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

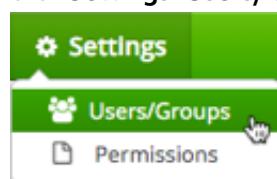
- * indicates zero or more occurrences of characters
- ? indicates a single character

4. Click **Add**.



3.2.5. Users/Groups and Permissions Administration

To examine the list of users and groups who can access the Ranger portal or its services, click **Settings>Users/Groups** at the top of the Ranger Console.



The User/sGroup view displays:

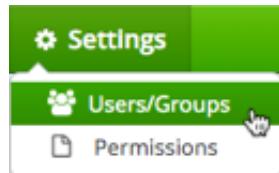
- Internal users who can log in to the Ranger portal; created by the Ranger console Service Manager
- External users who can access services controlled by the Ranger portal; created at other systems like Active Directory, LDAP or UNIX, and synched with those systems
- Admins who are the only users with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

	Group Name	Group Source	Visibility
public	Internal	Visible	
group01	Internal	Visible	
group02	Internal	Visible	
group03	Internal	Visible	
group04	Internal	Visible	
group05	Internal	Visible	
group06	Internal	Visible	
group07	Internal	Visible	

3.2.5.1. Add a User

To add a new user to the user list:

1. Click **Settings>Users/Groups**.



The Users/Groups page appears.

	User Name	Email Address	Role	User Source	Groups	Visibility	Status
admin			Admin	Internal	--	Visible	Enabled
rangerusersync			Admin	Internal	--	Visible	Enabled
testuser			User	External	--	Visible	Enabled
user0001			User	External	group01	Visible	Enabled
user0002			User	External	group01	Visible	Enabled
user0003			User	External	group01	Visible	Enabled
user0004			User	External	group01	Visible	Enabled
user0005			User	External	group01	Visible	Enabled
user0006			User	External	group01	Visible	Enabled
user0007			User	External	group01	Visible	Enabled
user0008			User	External	group01	Visible	Enabled
user0009			User	External	group01	Visible	Enabled
user0010			User	External	group01	Visible	Enabled

2. Click **Add New User**.



The Ranger User Detail page appears.

The screenshot shows the 'User Detail' form for creating a new user in Ranger. The form includes fields for User Name, New Password, Password Confirm, First Name, Last Name, Email Address, and Select Role (set to Admin). The Group field is marked with a red asterisk and contains the placeholder 'Please select'. A green 'Save' button and a black 'Cancel' button are at the bottom.

3. Add the appropriate user details, then click **Save**.

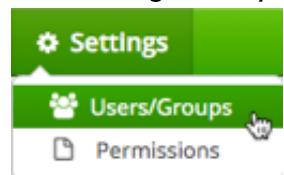
Save

The user is immediately added to the list.

3.2.5.2. Edit a User

To edit a user:

1. Click **Settings>Users/Groups**.



The Users/Groups page opens to the Users tab.

	Group Name	Group Source	Visibility
public	Internal	Visible	
group01	Internal	Visible	
group02	Internal	Visible	
group03	Internal	Visible	
group04	Internal	Visible	
group05	Internal	Visible	
group06	Internal	Visible	
group07	Internal	Visible	

2. Choose to edit yourself or another user:

User Name	Email Address	Role	User Source	Groups	Visibility	Status
admin		Admin	Internal	--	Visible	Enabled
rangerusersync		Admin	Internal	--	Visible	Enabled
testuser		User	External	--	Visible	Enabled
user0001		User	External	group01	Visible	Enabled
user0002		User	External	group01	Visible	Enabled
user0003		User	External	group01	Visible	Enabled
user0004		User	External	group01	Visible	Enabled
user0005		User	External	group01	Visible	Enabled
user0006		User	External	group01	Visible	Enabled

- To edit another user: in the User Name column, select the user you wish 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.

Users/Groups > User Edit

User Detail

Basic Info Change Password

User Name *

First Name *

Last Name

Email Address

Select Role *

Group

You can change everything except the User Name when editing an internal user.

Users/Groups > User Edit

User Detail

User Name *

First Name

Last Name

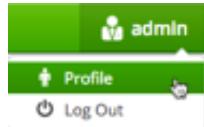
Email Address

Select Role *

Group

You can only change the user role when editing an external user.

- To edit your user profile, click **Username>Profile**.



The User Profile page appears.

The screenshot shows the 'User Profile' page with the 'Basic Info' tab selected. It contains fields for First Name (Admin), Last Name (Last Name), Email Address (Email Address), and a dropdown for Select Role (Admin). At the bottom are 'Save' and 'Cancel' buttons.

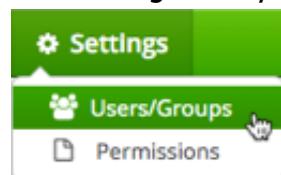
3. Edit the appropriate details and click **Save**.



3.2.5.3. Add a Group

To add a group:

1. Click **Settings>Users/Groups**.



The Users/Groups page opens to the Users tab.

User Name	Email Address	Role	User Source	Groups	Visibility	Status
admin		Admin	Internal	--	Visible	Enabled
rangerusersync		Admin	Internal	--	Visible	Enabled
testuser		User	External	--	Visible	Enabled
user0001		User	External	group01	Visible	Enabled
user0002		User	External	group01	Visible	Enabled
user0003		User	External	group01	Visible	Enabled
user0004		User	External	group01	Visible	Enabled
user0005		User	External	group01	Visible	Enabled
user0006		User	External	group01	Visible	Enabled
user0007		User	External	group01	Visible	Enabled
user0008		User	External	group01	Visible	Enabled
user0009		User	External	group01	Visible	Enabled
user0010		User	External	group01	Visible	Enabled

2. Click the **Groups** sub-tab.

Groups

The Groups page appears.

Group Name	Group Source	Visibility
public	Internal	Visible
group01	Internal	Visible
group02	Internal	Visible
group03	Internal	Visible
group04	Internal	Visible
group05	Internal	Visible
group06	Internal	Visible
group07	Internal	Visible

3. Click **Add New Group**

Add New Group

The Group Create page appears.

The screenshot shows the Ranger UI interface. At the top, there's a navigation bar with tabs for Access Manager, Audit, and Settings, and a user icon labeled 'admin'. Below the navigation bar, there's a sub-navigation for 'Users/Groups' with 'Users' and 'Groups' tabs, where 'Groups' is selected. A search bar and a 'Set Visibility' button are also present. A red arrow points to the 'Set Visibility' dropdown menu, which has two options: 'Visible' and 'Visible'. To the right of the visibility dropdown, there's another Ranger UI window titled 'Group Create' with fields for 'Group Name' and 'Description', and 'Save' and 'Cancel' buttons.

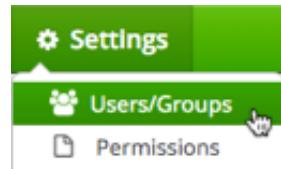
4. Enter a unique name for the group, and an optional description, then click **Save**.

Save

3.2.5.4. Edit a Group

To edit a group:

1. Click **Settings>Users/Groups**.



The Users/Groups page opens to the Users tab.

User Name	Email Address	Role	User Source	Groups	Visibility	Status
admin		Admin	Internal	--	Visible	Enabled
rangerusersync		Admin	Internal	--	Visible	Enabled
testuser		User	External	--	Visible	Enabled
user0001		User	External	group01	Visible	Enabled
user0002		User	External	group01	Visible	Enabled
user0003		User	External	group01	Visible	Enabled
user0004		User	External	group01	Visible	Enabled
user0005		User	External	group01	Visible	Enabled
user0006		User	External	group01	Visible	Enabled
user0007		User	External	group01	Visible	Enabled
user0008		User	External	group01	Visible	Enabled
user0009		User	External	group01	Visible	Enabled
user0010		User	External	group01	Visible	Enabled

2. Click the **Groups** sub-tab.

Groups

The Groups page appears.

Group Name	Group Source	Visibility
public	Internal	Visible
group01	Internal	Visible
group02	Internal	Visible
group03	Internal	Visible
group04	Internal	Visible
group05	Internal	Visible
group06	Internal	Visible
group07	Internal	Visible

3. Select the group name you wish to edit.

	Group Name	Group Source	Visibility
<input type="checkbox"/>	public	Internal	Visible
<input type="checkbox"/>	group01	Internal	Visible
<input type="checkbox"/>	group02	Internal	Visible
<input type="checkbox"/>	group03	Internal	Visible

4. The Group Edit page appears.

Group Detail

Group Name *

Description

Save **Cancel**

5. Edit the appropriate details, then click **Save**.



3.2.5.5. Add or Edit Permissions

To add or edit user or group:

1. Click **Settings>Permissions**.

Settings

Users/Groups

Permissions

The Users/Groups page opens to the Permissions page.

The screenshot shows a table titled 'Permissions' with columns: 'Permissions', 'Groups', 'Users', and 'Action'. The 'Permissions' column lists categories like 'Resource Based Policies', 'Users/Groups', 'Reports', 'Audit', and 'Key Manager'. The 'Groups' column lists groups such as 'rangerusersync', 'keyadmin', 'hive', 'knox', 'amb_ranger_admin', and 'ams'. The 'Users' column lists individual users: 'admin', 'rangerusersync', 'keyadmin', 'hive', 'amb_ranger_admin', and 'keyadmin'. The 'Action' column contains edit icons for each row.

2. Click



beside the permission you wish to edit.

The Edit Permission page appears.

The screenshot shows the 'Edit Permission' page for the 'Users/Groups' module. It has sections for 'Policy Details' (Module Name: Users/Groups) and 'User and Group Permissions'. In the 'User and Group Permissions' section, there are dropdown menus for 'Select Group' and 'Select User'. A dropdown menu for 'Select User' is open, showing options: 'x admin', 'x rangerusersync', 'x amb_ranger_admin', 'keyadmin', 'hive', 'knox', 'ambari-qa', 'hdfs', 'knox', 'ranger', and a separator line. Buttons for 'Save' and 'Cancel' are at the bottom.

3. Edit the necessary fields and click **Save**.



You can select multiple users and groups from the dropdown menus.

3.2.6. Reports Administration

The Reports module is used to manage the policies more efficiently as the number of policies grow. The page lists all HDFS, HBase, Hive, YARN, Knox, Storm, Solr, and Kafka policies.

You can edit policies from the Reports module by selecting the Policy ID.

The screenshot shows the Ranger Access Manager interface for editing an HDFS policy. The top navigation bar includes 'Ranger', 'Access Manager', 'Audit', 'Settings', 'Service Manager', 'test-hdfs Policies', and 'Edit Policy'. The main area is titled 'Edit Policy' and contains 'Policy Details' and 'User and Group Permissions' sections. A red arrow points from the left margin to the 'Edit Policy' button in the breadcrumb trail.

Policy Details:

- Policy ID: test-hdfs-1-20160202213734
- Status: Enabled
- Audit Logging: Enabled
- Groups: admin
- Users: admin

User and Group Permissions:

Permissions	Select Group	Select User	Permissions	Delegate Admin
	Select Group	admin	Read Write Execute	

Buttons at the bottom: Save, Cancel, Delete.

3.2.6.1. View Reports

To view reports on one or more policies, click **Access Manager>Reports**.



The screenshot shows the Ranger User Access Report interface. At the top, there are navigation links: Access Manager, Audit, Settings, and a user icon labeled "admin". Below the header, a "User Access Report" button is highlighted. The main area is titled "Reports" and contains a "Search Criteria" section with fields for Policy Name, Resource, and Search By (Group dropdown with "Select Group"). A "Search" button is also present. Below this, there are two sections: "HDFS" and "HBASE", each displaying a table of policies.

Policy ID	Policy Name	Status	Audit Logging	Groups	Users
1	test-hdfs-1-20160202213734	Enabled	Enabled		admin
4	c6401_TEST_hadoop-1-201602022306...	Enabled	Enabled		

Policy ID	Policy Name	Status	Audit Logging	Groups	Users
2	hbase-test-1-20160202224128	Enabled	Enabled		admin
3	c6401_TEST_hbase-1-20160202225955	Enabled	Enabled		

3.2.6.2. Search Reports

You can search:

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

The screenshot shows the Apache Ranger web interface. At the top, there is a green header bar with the 'Ranger' logo, 'Access Manager', 'Audit', 'Settings', and a user icon labeled 'admin'. Below the header, a 'User Access Report' button is highlighted in green. The main content area is titled 'Reports'.

Search Criteria:

- Policy Name:
- Resource:
- Search By:
-

HDFS

Policy ID	Policy Name	Status	Audit Logging	Groups	Users
1	test-hdfs-1-20160202213734	Enabled	Enabled		admin
4	c6401_TEST_hadoop-1-201602022306...	Enabled	Enabled		

HBASE

Policy ID	Policy Name	Status	Audit Logging	Groups	Users
2	hbase-test-1-20160202224128	Enabled	Enabled		admin
3	c6401_TEST_hbase-1-20160202225955	Enabled	Enabled		

3.2.7. Special Requirements for High Availability Environments

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

To enable Ranger in the HDFS HA environment, the HDFS plugin must be set up in each NameNode, and then pointed to the same HDFS 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 must be set to the full hostname of the primary NameNode. If the primary NameNode fails during policy creation, you can then temporarily use the `fs.default.name` of the secondary NameNode in the service details to enable directory lookup for policy creation.

If, while the primary node is down, you wish to create new policies, there is a slight difference in user experience when specifying the resource path. If everything is normal, this is a drop-down menu with selectable paths; however, if your cluster is running from the 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.

3.2.8. Adding a New Component to Apache Ranger

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

Apache Ranger has three main components:

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

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

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

Adding Component Details to the Admin Tool

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

Required UI changes to support the new component:

1. Add a new component template to the 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 existig objects and populate objects

For Policy Creation/Update of any component (i.e HDFS, Hive, Hbase), Apache Ranger uses only one common object, `VXPolicy.` The same goes for the 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 VXPolicy/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.

3.2.9. 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()

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`. WebHDFS, hftp, and shortcircuitread are not supported when SSL is enabled.

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
HDFS	WebHDFS	Hadoop SSL Keystore Factory
MapReduce	Shuffle	Hadoop SSL Keystore Factory
	TaskTracker	Hadoop SSL Keystore Factory
Yarn	Resource Manager	Hadoop SSL Keystore Factory

Component	Service	SSL Management
	JobHistory	Hadoop SSL Keystore Factory
Oozie		Configured in oozie-site.xml
HBase	REST API	Configured in hbase-site.xml
Hive	HiveServer2	Configured in hive-site.xml
Kafka		JDK: User and default
Solr		JDK: User and default
Accumulo		JDK: User and default
Falcon	REST API	JDK: User and default
Knox	Hadoop cluster (REST client)	JDK: default only
	Knox Gateway server	JDK: User and default
HDP Security Administration	Server/Agent	JDK: User and default

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>
(RTURN 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 -keystore <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" \[221\]](#) 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" [220] ; Section 4.6, "Enable SSL for WebHDFS, MapReduce Shuffle, and YARN" [230]
Oozie	Section 4.8, "Enable SSL on Oozie" [234]
HBase	Section 4.9, "Enable SSL on WebHDFS and the HBase REST API" [235]
Hive (HiveServer2)	Section 4.10, "Enable SSL on HiveServer2" [236]
Kafka	Section 4.11, "Enable SSL for Kafka Clients" [237]
Ambari Server	Ambari Security Guide, Advanced Security Options, Optional: Set Up SSL for Ambari
Falcon	Enabled by default (see Installing the Falcon Package)
Sqoop	Clients of Hive and HBase, see Data Integration Services with HDP, Apache Sqoop Connectors
Knox Gateway	Knox Administrator Guide, Gateway Security, Configure Wire Encryption
Flume	Apache Flume User Guide, Flume Sources
Accumulo	Apache Foundation Blog, Apache Accumulo: Generating Keystores for configuring Accumulo with SSL
Phoenix	Non-Ambari Cluster Installation, Installing Apache Phoenix: Configuring Phoenix for Security and Apache Phoenix, Flume Plug-in
HUE	Non-Ambari Cluster Installation, Installing Hue, Configure Hue

4.6. Enable SSL for WebHDFS, MapReduce Shuffle, 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 [Apache Knox Gateway Administrator Guide](#), [Gateway Security](#), [Configure Wire Encryption](#).

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>
```

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. Copy the configuration files (`core-site.xml`, `hdfs-site.xml`, `mapred-site.xml`, `yarn-site.xml`, `ssl-server.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.

8. 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 [Apache Knox Gateway Administrator Guide, Gateway Security, Configure Wire Encryption](#).

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 "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 "oozie-setup.sh prepare-war"
```

4.8.1. 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 WebHBase and the HBase REST API

Perform the following task to enable SSL on WebHBase and 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 [Apache Knox Gateway Administrator Guide](#), [Gateway Security](#), [Configure Wire Encryption](#).

1. Verify that the HBase REST API is running, on the HBase Master run:

```
curl http://localhost:60080/
```

If the rest daemon is not running on that port, run the following command to start it:

```
sudo /usr/lib/hbase/bin/hbase-daemon.sh start rest -p 60080
```

2. 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"
```

b. Export the certificate:

```
su -l hbase -c "keytool -exportcert -alias hbase -file certificate.cert -keystore hbase.jks"
```

c. 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
```

3. 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>
```

4. Restart all HBase nodes in the cluster.



Note

When using a self-signed certificate, manually add the certificate to the JVM truststore on all HBase clients.

4.10. 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. Run the following command to create a keystore for hiveserver2::

```
keytool -genkey -alias hbase -keyalg RSA -keysize 1024 -keystore hbase.jks
```

2. 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>
```

3. 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. 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.

4.11.1. 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)
ssl.client.auth	<p>Specify whether client authentication is required, requested, or not required.</p> <p>none: no client authentication.</p> <p>required: client authentication is required.</p> <p>requested: client authentication is requested, but a client without certs can still connect.</p> <p>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.</p>	none
ssl.cipher.suites	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.	
ssl.enabled.protocols	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
ssl.keystore.type	Specify the SSL keystore type.	JKS
ssl.truststore.type	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 should be listed under `ssl.enabled.protocols`)

In the `openssl` output you should see the server certificate; for example:

```
Server certificate
-----BEGIN CERTIFICATE-----
MIID+DCCAUACCQCx2Rz1tXx3NTANBqkqhkiG9w0BAQsFADB6MQswCQYDVQQGEwJV
UzELMAkGA1UECAwCQ0ExFDASBgNVBAcMC1NhbnRhIENsYXJhMQwwCgYDVQQKDANv
cmcxDDAKBgNVBAsMA29yZzEOMAwGA1UEAwFa2FmYWsxHDAaBqkqhkiG9w0BCQEW
DXRlc3RADGVzdC5jb20wHhcNMjUwNzMwMDQyOTMwWhcNMjTywNzI5MDQyOTMwWjBt
MQswCQYDVQQGEwJVUzELMAkGA1UECBMCQ0ExFDASBgNVBAcTC1NhbnRhIENsYXJh
MQwwCgYDVQQKEwNvcmcxDDAKBgNVBAsTA29yZzEfMB0GA1UEAxMWU3JpaGFyc2hh
IENoaW50YWxhcGFuaTCCAbcwggEsBgcqhkjOOAQBMIIBHwKBgQD9f1OBHXUSKVLf
Spwu7OTn9hG3UjzvRADDhj+At1EmaUVdQCJR+1k9jvj6v8X1ujd2y5tvbNeBO4Ad
NG/yZmC3a5lQpaSfn+gEexAiwk+7qdf+t8Yb+DtX58aophUPBPuD9tPFHsMCNVQT
WhaRMvZ1864rYdcq7/IiAxmd0UgBxwIVAJdgUI8VIwvMspK5ggLrhAvvWBz1AoGB
APfhIXWmz3ey7yrXDa4V7151K+7+jrqgv1XTAs9B4JnUV1XjrrUWU/mcQcQgYCo
SRZxI+hMKBYTt88JMoziPusE8FnqLVHyNKOCjrh4rs6z1kw6jfWv6ITV18ftiegEk
O8yk8b6oUZCJqIPf4VrlnwaSi2ZegHtVJWQBTDv+z0kqA4GEAAKBgB+Pdz0306bq
TpUAdb2FERMPLFsx06H0x+TULivcp7HbS5yrvkV9bXZmv/FD98x76QxXrOq1WpQhY
YDeGDjH+XQkJ6ZxBVBZNJDIpCnfQpfzXAvryQ+cm8oXUsKidtHf4pLMyViXX6BWX
Oc2hX4rG+1C8/NXW+1zVvCr9To9fngzjMA0GCSqGS1b3DQEBCwUAA4IBAQBFyVse
RJ+ugiNlWg5trZscqH0tlocbnek4UuV/xis2eAu914EFOM5kRt5GmkGZRcM/zHF8
BRJwXbf0fytmQKSPFk8R4/NGD01zoK+F7uXeJ0S2u/T29xk0u2i4tjvleq6OCphE
i9vdjM0E0Whf9SHRHoxirOYFX3cL775XwKdzKKRkk+AszFR+mRu90rdoaepQtgGh
9Kfwr4+6AU/dPtdGuomtBQqMxCz1rLd8EYhVVQ97wHIZ3sPv1M5PihoJ/YHSBJIC
75eo/4acDxZ+j3sR5kcFulzYwFlgDYBaKH/w3mYCgTALeB1zUkX53NVizIvhUd69
XJO41DSDtG0lfot
-----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.

4.11.2. 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)
ssl.provider	The name of the security provider used for SSL connections. Default value is the default security provider of the JVM.	
ssl.cipher.suites	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.	
ssl.enabled.protocols	List at least one of the protocols configured on the broker side.	TLSv1.2, TLSv1.1, TLSv1
ssl.keystore.type	Specify the SSL keystore type.	JKS
ssl.truststore.type	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
```

4.12. 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.

4.12.1. 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.

4.12.2. 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.

4.12.3. 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.

4.12.4. 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.

4.13. SPNEGO setup for WebHCat

To set up secure WebHCat, set the following properties in the `/etc/hcatalog/conf/webhcatt-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.

4.14. Configure SSL for Hue

To configure SSL for Hue, use the instructions on [this page](#).

4.15. 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.

4.15.1. 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
```

4.15.2. 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`.

4.15.3. 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.

4.16. Securing Phoenix

To configure Phoenix to run in a secure Hadoop cluster, use the instructions on [this page](#).

4.17. 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](#).

4.17.1. Set Up Truststore for Ambari Server

If you plan to [set up SSL for Ambari](#) 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
```

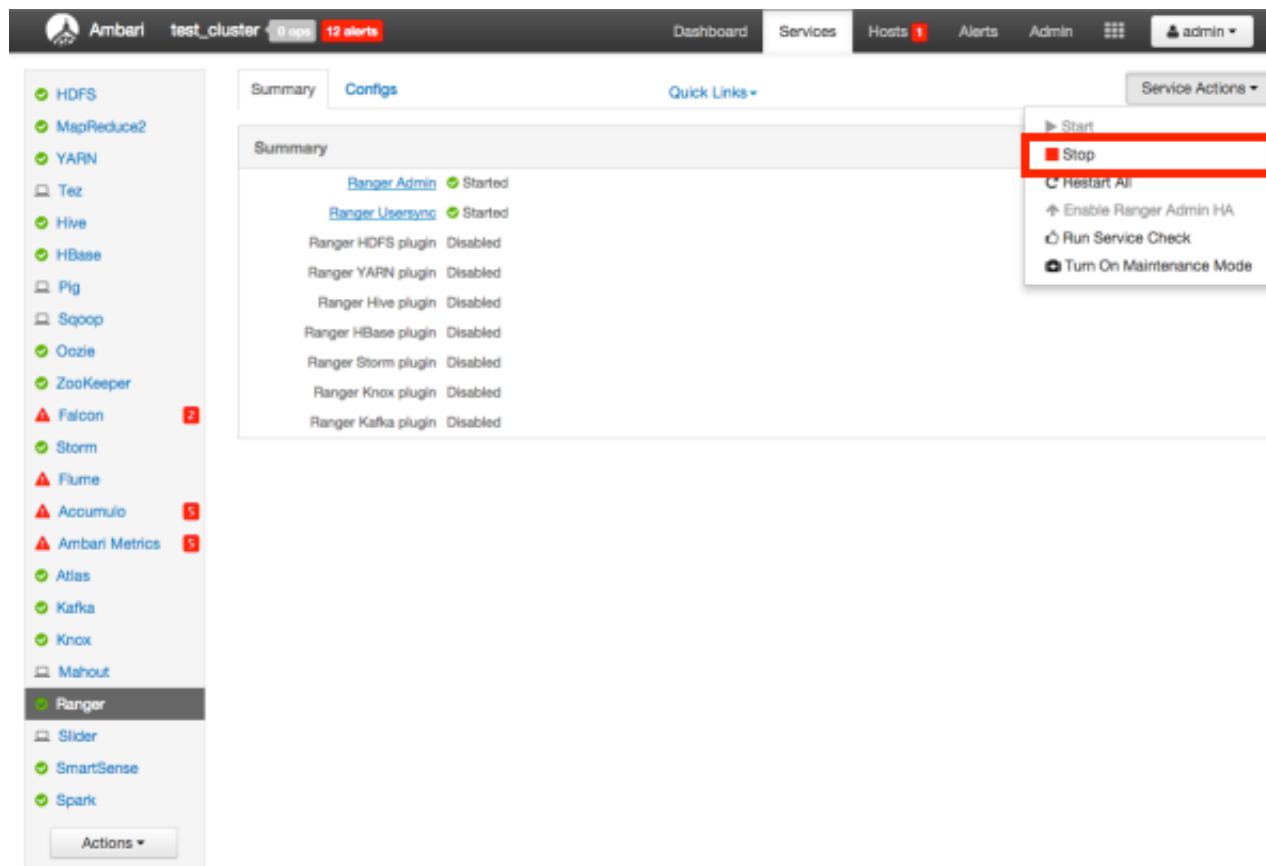
4.18. Configure Ambari Ranger SSL

4.18.1. 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.

4.18.1.1. Configuring Ranger Admin

1. Stop Ranger by selecting **Service Actions > Stop**.



The screenshot shows the Ambari interface for a cluster named 'test_cluster'. On the left, a sidebar lists various 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 currently selected. The main panel displays the 'Summary' tab for the Ranger service, which shows the status of Ranger Admin as 'Started' and other Ranger plugins as 'Disabled'. To the right, a 'Service Actions' dropdown menu is open, with the 'Stop' option highlighted by a red box. Other options in the menu include 'Start', '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 interface for managing a test cluster. The left sidebar lists various services: HDFS, MapReduce2, YARN, Tez, Hive, HBase, Pig, Sqoop, Oozie, ZooKeeper, Falcon, Storm, Flume, Accumulo, Ambari Metrics, Atlas, Kafka, Knox, Mahout, and Ranger. The 'Ranger' service is currently selected. The main panel displays the 'Manage Config Groups' screen for Ranger, showing four configuration versions (V4, V3, V2, V1) with their respective authors (admin) and dates (15 days ago or 2 months ago). A message indicates there is 1 configuration change in 1 service. Below this, tabs for Ranger Admin, Ranger User Info, Ranger Plugin, Ranger Audit, and Advanced are visible. The 'Advanced' tab is selected, revealing sections for Admin Settings and Ranger Settings. In the Ranger Settings section, the 'External URL' field contains the value 'https://c6403.ambari.apache.org:6182'. The 'Authentication method' dropdown is set to 'LDAP'. The 'HTTP enabled' section 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>`.

This is a detailed view of the 'Ranger Settings' section from the previous screenshot. The 'External URL' field is explicitly highlighted with a red box. Below it, the 'HTTP enabled' section is also highlighted with a red box. The 'Authentication method' dropdown is set to 'LDAP'.

- 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	solr			
ranger.credential.provider.path	/etc/ranger/admin/rangeradmin.jceks			
ranger.https.attrib.keystore.file	/etc/ranger/admin/conf/ranger-admin-keystore.jks			
ranger.jpa.audit.jdbc.credential.alias	rangeraudit			
ranger.jpa.audit.jdbc.dialect	{jdbc_dialect}}			
ranger.jpa.audit.jdbc.driver	{ranger_jdbc_driver}}			
ranger.jpa.audit.jdbc.password		
ranger.jpa.audit.jdbc.url	{audit_jdbc_url}			
ranger.jpa.audit.jdbc.user	{ranger_audit_db_user}			
ranger.jpa.jdbc.credential.alias	rangeradmin			
ranger.jpa.jdbc.dialect	{jdbc_dialect}}			
ranger.jpa.jdbc.password		
ranger.jpa.jdbc.user	{ranger_db_user}			
Group Search Base	{ranger_ug_ldap_group_searchbase}			
Group Search Filter	{ranger_ug_ldap_group_searchfilter}			
ranger.service.host	{ranger_host}			
ranger.service.http.port	6080			
ranger.service.https.attrib.clientAuth	want			
ranger.service.https.attrib.keystore.keyalias	rangeradmin			
ranger.service.https.attrib.keystore.pass		
ranger.service.https.attrib.ssl.enabled	true			
ranger.service.https.port	6182			

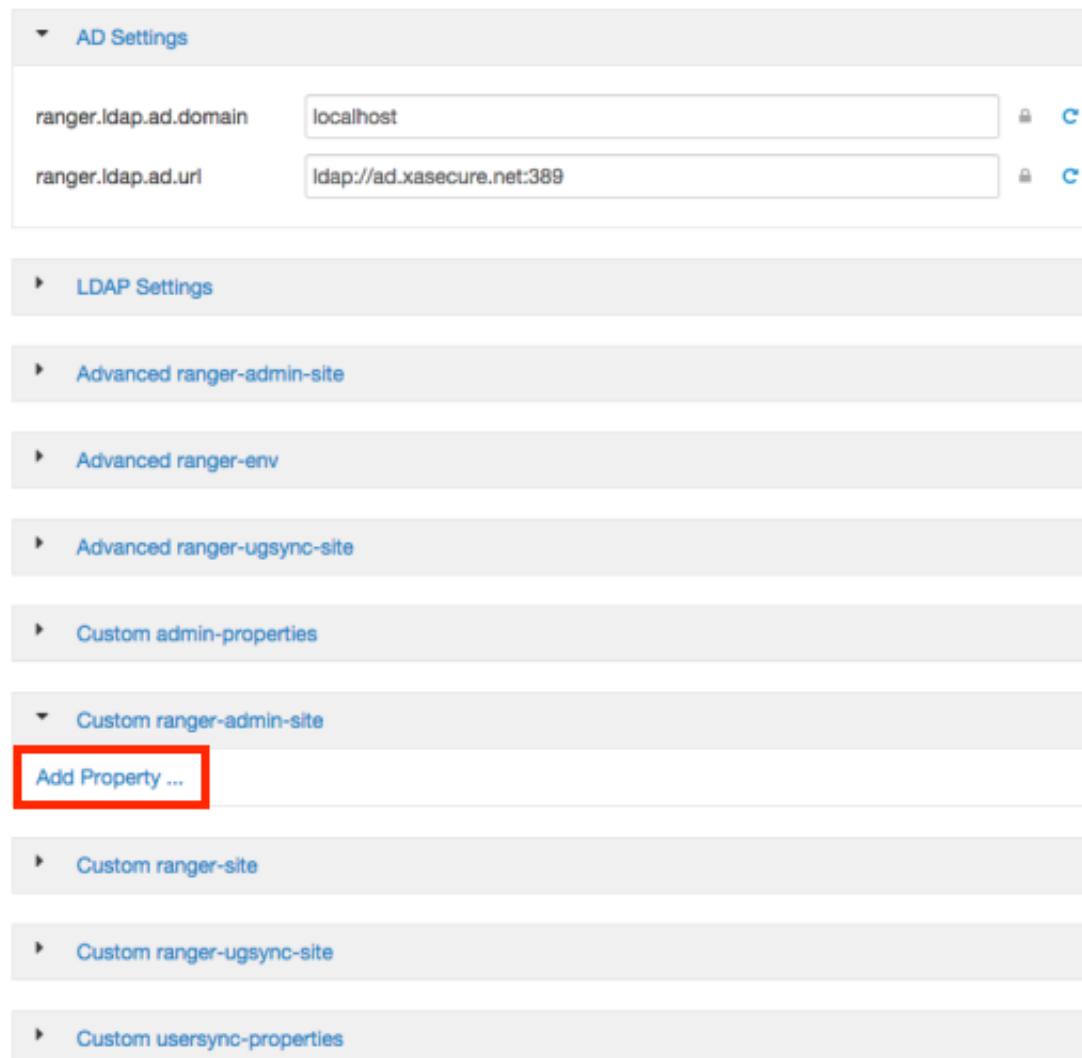
3. Under Custom ranger-admin-site, add the following properties:

- **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**.



- 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	

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.

4.18.1.2. Configuring Ranger Usersync

1. Stop Ranger Usersync by selecting the **Ranger Usersync** link, then select **Started > Stop** next to Ranger Usersync.

Ambari test_cluster 0 ops 16 alerts

Dashboard Services Hosts 1 Alerts Admin admin

c6403.ambari.apache.org

Back

Summary Configs Alerts 16 Versions Host Actions

Components	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	Restart
✓ SNameNode / HDFS	Stop
✓ Spark History Server / Spark	Started
✓ Storm UI Server / Storm	Started

Host Metrics Last 1 hour

No Data Available No Data Available

CPU Usage Disk Usage

No Data Available No Data Available

Load Memory Usage

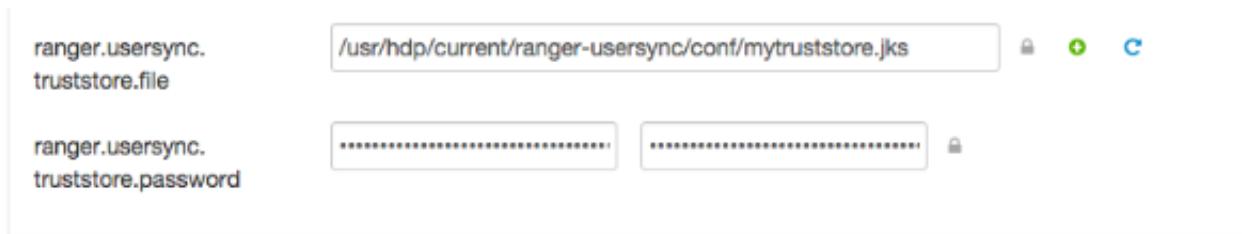
No Data Available No Data Available

Network Usage Processes

The screenshot shows the Ambari interface for a 'test_cluster'. At the top, there are navigation links for Dashboard, Services, Hosts (with 1 item), Alerts, Admin, and a user dropdown for 'admin'. Below this, the host name 'c6403.ambari.apache.org' is displayed with a warning icon, and a 'Back' link. The main area has tabs for Summary, Configs, Alerts (16), and Versions, with 'Host Actions' on the far right. The 'Summary' tab is active, showing a table of components. Each row contains the component name (e.g., 'Accumulo GC / Accumulo'), its status (e.g., 'Stopped' or 'Started'), and a dropdown menu. The 'ResourceManager / YARN' row's dropdown menu is open, showing options 'Restart' and 'Stop', with 'Stop' highlighted by a red box. Other components listed include App Timeline Server, Atlas Metadata Server, DRPC Server, Falcon Server, HBase Master, History Server, Hive Metastore, HiveServer2, SmartSense HST Server, Kafka Broker, Knox Gateway, Metrics Collector, MySQL Server, NameNode, Nimbus, Oozie Server, Ranger Admin, Ranger Usersync, SNameNode, Spark History Server, and Storm UI Server.

2. 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.



3. Start Ranger Usersync by selecting the **Ranger Usersync** link on the Summary tab, then select **Stopped > Start** next to Ranger Usersync.

4.18.1.3. 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.

4.18.1.3.1. 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 'test_cluster'. On the left, a sidebar lists various services: HDFS (green), MapReduce2 (green), YARN (green), Tez (grey), Hive (green), HBase (green), Pig (grey), Sqoop (grey), Oozie (green), ZooKeeper (green), Falcon (red), Storm (green), Flume (green), Accumulo (red), Ambari Metrics (red), Atlas (green), Kafka (green), Knox (green), Mahout (grey), Ranger (green), Slider (grey), SmartSense (red), and Spark (green). Some services have red notification counts (e.g., 2, 5, 1) and an 'Actions' dropdown.

The main area has tabs for 'Summary', 'Heatmaps', 'Configs', and 'Quick Links'. The 'Summary' tab is active, displaying service status (e.g., NameNode Started, DataNodes 1/1 Started), disk usage, and metrics like NameNode Uptime and Heap usage. The 'Metrics' tab shows counters for NameNode GC, NN Connection Load, NameNode Heap, and NameNode Host Load, all reporting 'No Data Available'. It also shows metrics for NameNode RPC, Failed disk volumes (n/a), Corrupted Blocks (0), Under Replicated Blocks (481), and HDFS Space Utilization (n/a).

A context menu is open over the 'NameNode' section, listing actions: Start, Stop (which is highlighted with a red box), Restart All, Restart DataNodes, Move NameNode, Move SNameNode, Enable NameNode HA, Run Service Check, Turn On Maintenance Mode, Rebalance HDFS, and Download Client Configs.

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. clientssl.keystore	/usr/hdp/current/hadoop-client/conf/ranger-plugin-keystore.jks	⊕ C
xasecure.policymgr. clientssl.keystore. credential.file	jceks://file{{credential_file}}	⊕ C
xasecure.policymgr. clientssl.keystore. password
xasecure.policymgr. clientssl.truststore	/usr/hdp/current/hadoop-client/conf/ranger-plugin-truststore.jks	⊕ C
xasecure.policymgr. clientssl.truststore. credential.file	jceks://file{{credential_file}}	⊕ C
xasecure.policymgr. clientssl.truststore. password

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
Ranger repository config user	hadoop	⊕ + C
common.name.for. certificate		⊕ +
hadoop.rpc.protection		⊕ +
Policy user for HDFS	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"
```

4.18.1.3.2. 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 UI for creating a new service. The top navigation bar has tabs for 'Ranger' (selected), 'Access Manager', and 'Encryption'. Below the navigation is a breadcrumb trail: 'Service Manager > Edit Service > Create Service'. The main form is titled '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 *: (redacted)

Add New Configurations:

Name	Value
commonNameForCertificate	ip-172-31-26-219.ec2.internal

Buttons:

- Test Connection
- Save (green)
- Cancel (black)
- Delete (red)

4.18.1.3.3. 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

Property	Value	Status
ranger.https.attrib.keystore.file	/etc/ranger/kms/conf/ranger-kms-keystore.jks	Green
ranger.service.https.attrib.client.auth	want	Green
ranger.service.https.attrib.clientAuth	false	Green
ranger.service.https.attrib.keystore.file	/etc/ranger/kms/conf/ranger-kms-keystore.jks	Green
ranger.service.https.attrib.keystore.keyalias	rangerkms	Green
ranger.service.https.attrib.keystore.pass	rangerkms	Green
ranger.service.https.attrib.ssl.enabled	true	Green
ranger.service.https.port	9393	Green

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.

4.18.2. 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.

4.18.2.1. Configuring Ranger Admin

1. Stop Ranger by selecting **Service Actions > Stop**.

The screenshot shows the Ambari interface for a cluster named 'test_cluster'. On the left, a sidebar lists various services: HDFS, MapReduce2, YARN, Tez, Hive, HBase, Pig, Sqoop, Oozie, ZooKeeper, Falcon, Storm, Flume, Accumulo, Ambari Metrics, Atlas, Kafka, Knox, Mahout, and Ranger. The 'Ranger' service is selected and highlighted in dark grey. The main panel displays the 'Summary' tab for the Ranger service, which shows it is 'Started'. Below this, a list of Ranger plugins is shown, all marked as 'Disabled'. To the right, a 'Service Actions' dropdown menu is open, with the 'Stop' option highlighted by a red box. Other options in the menu include 'Start', 'Restart All', 'Enable Ranger Admin HA', 'Run Service Check', and 'Turn On Maintenance Mode'.

2. 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.

3. 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 interface for a 'test_cluster'. On the left sidebar, under the 'Ranger' section, the 'Actions' dropdown is open, and the 'HTTP enabled' option is highlighted with a red box.

Ranger Settings:

- External URL:** https://c6403.ambari.apache.org:6182
- Authentication method:** LDAP (selected)
- HTTP enabled:** (checkbox checked)

- 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 the 'Ranger Settings' section of the Ambari configuration interface. The 'External URL' field is highlighted with a red box.

Ranger Settings:

- External URL:** https://c6403.ambari.apache.org:6182
- Authentication method:** LDAP (selected)
- HTTP enabled:** (checkbox checked)

- 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	solr		
ranger.credential.provider.path	/etc/ranger/admin/rangeradmin.jceks		
ranger.https.attrib.keystore.file	/etc/ranger/admin/conf/ranger-admin-keystore.jks		
ranger.jpa.audit.jdbc.credential.alias	rangeraudit		
ranger.jpa.audit.jdbc.dialect	{jdbc_dialect}}		
ranger.jpa.audit.jdbc.driver	{ranger_jdbc_driver}}		
ranger.jpa.audit.jdbc.password	
ranger.jpa.audit.jdbc.url	{audit_jdbc_url}		
ranger.jpa.audit.jdbc.user	{ranger_audit_db_user}		
ranger.jpa.jdbc.credential.alias	rangeradmin		
ranger.jpa.jdbc.dialect	{jdbc_dialect}}		
ranger.jpa.jdbc.password	
ranger.jpa.jdbc.user	{ranger_db_user}		
Group Search Base	{ranger_ug_ldap_group_searchbase}}		
Group Search Filter	{ranger_ug_ldap_group_searchfilter}}		
ranger.service.host	{ranger_host}		
ranger.service.http.port	6080		
ranger.service.https.attrib.clientAuth	want		
ranger.service.https.attrib.keystore.keyalias	rangeradmin		
ranger.service.https.attrib.keystore.pass	
ranger.service.https.attrib.ssl.enabled	true		
ranger.service.https.port	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 shows a navigation tree on the left side of a web-based configuration interface. At the top, under 'AD Settings', there are two fields: 'ranger.ldap.ad.domain' set to 'localhost' and 'ranger.ldap.ad.url' set to 'ldap://ad.xasecure.net:389'. Below this, several sections are listed: 'LDAP Settings', 'Advanced ranger-admin-site' (which is expanded), 'Advanced ranger-env', 'Advanced ranger-ugsync-site', 'Custom admin-properties', and 'Custom ranger-admin-site'. The 'Custom ranger-admin-site' section is also expanded, showing an 'Add Property ...' button which is highlighted with a red rectangular border. Underneath it are three more collapsed sections: 'Custom ranger-site', 'Custom ranger-ugsync-site', and 'Custom usersync-properties'.

- 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**.

The screenshot shows a 'Add Property' dialog box. It has three input fields: 'Type' (set to 'ranger-admin-site.xml'), 'Key' (set to 'ranger.service.https.attrib.keystore.file'), and 'Value' (set to '/etc/ranger/admin/conf/ranger-admin-keystore.jks'). At the bottom right of the dialog, there are two buttons: 'Cancel' and a green 'Add' button, which is highlighted with a red border.

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.

4.18.2.2. Configuring Ranger Usersync

1. Stop Ranger Usersync by selecting the **Ranger Usersync** link, then select **Started > Stop** next to Ranger Usersync.

Ambari test_cluster 0 ops 16 alerts

Dashboard Services Hosts 1 Alerts Admin admin

c6403.ambari.apache.org

Back

Summary Configs Alerts 16 Versions Host Actions

Components	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	Restart
✓ SNameNode / HDFS	Stop
✓ Spark History Server / Spark	Started
✓ Storm UI Server / Storm	Started

Host Metrics Last 1 hour

No Data Available No Data Available

CPU Usage Disk Usage

No Data Available No Data Available

Load Memory Usage

No Data Available No Data Available

Network Usage Processes

The screenshot shows the Ambari interface for a 'test_cluster'. At the top, there are navigation links for Dashboard, Services, Hosts (with 1 item), Alerts, Admin, and a user dropdown for 'admin'. Below this, the host name 'c6403.ambari.apache.org' is displayed with a warning icon, and a 'Back' link. The main area has tabs for Summary, Configs, Alerts (16), and Versions, with 'Host Actions' on the far right. The 'Summary' tab is active, showing a table of components. Each row contains the component name (e.g., 'Accumulo GC / Accumulo'), its status (e.g., 'Stopped' or 'Started'), and a dropdown menu with options like 'Restart' and 'Stop'. A red box highlights the 'Stop' option for the 'ResourceManager / YARN' component. To the right of the table, there's a 'Host Metrics' section showing CPU Usage, Disk Usage, Load, Memory Usage, Network Usage, and Processes, all reporting 'No Data Available' over the last hour.

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.cer
chown -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.



5. Start Ranger Usersync by selecting the **Ranger Usersync** link on the Summary tab, then select **Stopped** > **Start** next to Ranger Usersync.

4.18.2.3. 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
```

4.18.2.3.1. 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 'test_cluster'. On the left, a sidebar lists various services: HDFS (green checkmark), MapReduce2 (green checkmark), YARN (green checkmark), Tez (grey), Hive (green checkmark), HBase (green checkmark), Pig (grey), Sqoop (grey), Oozie (green checkmark), ZooKeeper (green checkmark), Falcon (red warning), Storm (green checkmark), Flume (green checkmark), Accumulo (red warning), Ambari Metrics (red warning), Atlas (green checkmark), Kafka (green checkmark), Knox (green checkmark), Mahout (grey), Ranger (green checkmark), Slider (grey), SmartSense (red warning), and Spark (green checkmark). The 'Actions' button is highlighted with a red box.

The main area displays the 'Summary' tab for the HDFS service. It shows the following status details:

- NameNode: Started
- SNameNode: Started
- DataNodes: 1/1 Started
- DataNodes Status: 1 live / 0 dead / 0 decommissioning
- NFSGateways: 0/0 Started
- 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%)

A 'Metrics' section follows, showing various counters like NameNode GC count, NN Connection Load, and Under Replicated Blocks (value 481).

On the right, a 'Service Actions' dropdown menu is open, listing options such as Start, Stop (which is highlighted with a red box), Restart All, and others.

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	/etc/hadoop/conf/ranger-plugin-keystore.jks	
xasecure.policymgr. clientssl.keystore. credential.file	jceks://file{{credential_file}}	
xasecure.policymgr. clientssl.keystore. password
xasecure.policymgr. clientssl.truststore	/etc/hadoop/conf/ranger-plugin-truststore.jks	
xasecure.policymgr. clientssl.truststore. credential.file	jceks://file{{credential_file}}	
xasecure.policymgr. clientssl.truststore. 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
Ranger repository config user	hadoop
common.name.for. certificate	
hadoop.rpc.protection	
Policy user for HDFS	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.

4.18.2.3.2. 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 UI with the 'Create Service' page open. The top navigation bar includes 'Ranger', 'Access Manager', and 'Encryption'. Below the navigation is a breadcrumb trail: 'Service Manager > Edit Service'. The main section is titled '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 *: (redacted)

Add New Configurations

Name	Value
commonNameForCertificate	ip-172-31-26-219.ec2.internal

Buttons:

- Test Connection
- Save
- Cancel
- Delete

4.18.2.3.3. 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

The screenshot shows a configuration interface for the 'Custom ranger-kms-site' section. It lists several properties with their current values and status indicators (green or red).

Property	Value	Status
ranger.https.attrib.keystore.file	/etc/ranger/kms/conf/ranger-kms-keystore.jks	Green
ranger.service.https.attrib.client.auth	want	Green
ranger.service.https.attrib.clientAuth	false	Green
ranger.service.https.attrib.keystore.file	/etc/ranger/kms/conf/ranger-kms-keystore.jks	Green
ranger.service.https.attrib.keystore.keyalias	rangerkms	Green
ranger.service.https.attrib.keystore.pass	rangerkms	Green
ranger.service.https.attrib.ssl.enabled	true	Green
ranger.service.https.port	9393	Green

[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.

4.19. Configure Non-Ambari Ranger SSL

4.19.1. 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.

4.19.1.1. 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.

4.19.1.2. 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
```

4.19.1.3. 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 1.7).

```
export JAVA_HOME=<path for java 1.7>
```

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.

4.19.2. 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.

4.19.2.1. 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.

4.19.2.2. 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
```

4.19.2.3. 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 1.7).

```
export JAVA_HOME=<path for java 1.7>
```

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.

4.20. 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 [Apache Knox Gateway Administrator Guide](#), [Gateway Security](#), [Configure Wire Encryption](#).

4.20.1. 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.

4.20.2. Connect to SSL Enabled Oozie Server

On every Oozie client system, follow the instructions for the type of certificate used in your environment.

4.20.2.1. 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 `$JRE_cacerts` is the path to the JRE's certs file. It's location may differ depending on the Operating System, but it's typically called `cacerts` and located at `$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.

4.20.2.2. 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.

4.20.2.3. 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.



Important

Solr must be installed and configured before installing RangerAdmin or any of the Ranger component plugins.

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.

Configuration Options

- Solr Standalone – Solr Standalone is only recommended for testing and evaluation. Solr Standalone is a single instance of Solr that does not require ZooKeeper.
- SolrCloud – This is the recommended configuration for Ranger. **SolrCloud** is a scalable architecture that can run as single node or as a multi-node cluster. It includes features such as replication and sharding, which are useful for high availability (HA) and scalability. With SolrCloud, you need to plan the deployment based on the cluster size.

The following sections describe how to install and configure Apache Solr for Ranger Audits:

- [Prerequisites](#)
- [Installing Solr](#)
- [Configuring Solr Standalone](#)
- [Configuring SolrCloud](#)

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.

5.1.2. Installing Solr

Use the following command to install Solr:

```
yum install lucidworks-hdpsearch
```

The HDP Search installer installs Solr in the /opt/lucidworks-hdpsearch/solr directory.

5.1.3. Configuring Solr Standalone

Use the following procedure to configure Solr Standalone.

1. Download the solr_for_audit_setup_v3 file to the /usr/local/ directory:

```
wget https://issues.apache.org/jira/secure/attachment/12761323/
solr_for_audit_setup_v3.tgz -O /usr/local/solr_for_audit_setup_v3.tgz
```

2. Use the following commands to switch to the /usr/local/ directory and extract the solr_for_audit_setup_v3 file.

```
cd /usr/local
tar xvf solr_for_audit_setup_v3.tgz
```

The contents of the .tgz file will be extracted into a /usr/local/solr_for_audit_setup_v3 directory.

3. Use the following command to switch to the /usr/local/solr_for_audit_setup_v3 directory.

```
cd /usr/local/solr_for_audit_setup
```

4. 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 5.1. Solr install.properties Values

Property Name	Value	Description
JAVA_HOME	<path_to_jdk>, for example: /usr/jdk64/jdk1.8.0_60	Provide the path to the JDK install folder. For Hadoop, you can check /etc/hadoop/conf/

Property Name	Value	Description
		hadoop-env.sh for the value of JAVA_HOME. As noted previously, Solr only supports JDK 1.7 and higher.
SOLR_USER	solr	The Linux user used to run Solr.
SOLR_INSTALL_FOLDER	/opt/lucidworks-hdpsearch/solr	The Solr installation directory.
SOLR_RANGER_HOME	/opt/lucidworks-hdpsearch/solr/ranger_audit_server	The location where the Ranger-related configuration and schema files will be copied.
SOLR_RANGER_PORT	6083	The Solr port for Ranger.
SOLR_DEPLOYMENT	standalone	The deployment type.
SOLR_RANGER_DATA_FOLDER	/opt/lucidworks-hdpsearch/solr/ranger_audit_server/data	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	/var/log/solr/ranger_audits	The folder for the Solr log files.
SOLR_MAX_MEM	2g	The memory allocation for Solr.

5. Use the following command to run the Solr for Ranger setup script.

```
./setup.sh
```

6. To start Solr, log in as the `solr` or `root` user and run the following command.

```
/opt/lucidworks-hdpsearch/solr/ranger_audit_server/scripts/start_solr.sh
```

When Solr starts, a confirmation message appears.

```
Started Solr server on port 6083 (pid=). Happy searching!
```

7. You can use a web browser to open the Solr Admin Console at the following address:

```
http:<solr_host>:6083/solr
```



Note

You can use the following command to stop Solr:

```
/opt/lucidworks-hdpsearch/solr/ranger_audit_server/scripts/stop_solr.sh
```

5.1.4. Configuring SolrCloud

Use the following procedure to configure SolrCloud.

1. Download the `solr_for_audit_setup_v3` file to the `/usr/local/` directory:

```
wget https://issues.apache.org/jira/secure/attachment/12761323/
solr_for_audit_setup_v3.tgz -O /usr/local/solr_for_audit_setup_v3.tgz
```

2. Use the following commands to switch to the `/usr/local/` directory and extract the `solr_for_audit_setup_v3` file.

```
cd /usr/local
```

```
tar xvf solr_for_audit_setup_v3.tgz
```

The contents of the .tgz file will be extracted into a /usr/local/solr_for_audit_setup_v3 directory.

3. Use the following command to switch to the /usr/local/solr_for_audit_setup_v3 directory.

```
cd /usr/local/solr_for_audit_setup
```

4. 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
JAVA_HOME	<path_to_jdk>, for example: /usr/jdk64/jdk1.8.0_40	Provide the path to the JDK install folder. For Hadoop, you can check /etc/hadoop/conf/hadoop-env.sh for the value of JAVA_HOME. As noted previously, Solr only supports JDK 1.7 and higher.
SOLR_USER	solr	The Linux user used to run Solr.
SOLR_INSTALL_FOLDER	/opt/lucidworks-hdpsearch/solr	The Solr installation directory.
SOLR_RANGER_HOME	/opt/lucidworks-hdpsearch/solr/ranger_audit_server	The location where the Ranger-related configuration and schema files will be copied.
SOLR_RANGER_PORT	6083	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.

5. Use the following command to run the set up script.

```
./setup.sh
```

6. Run the following command **only once** from any node. This command adds the Ranger Audit configuration (including schema.xml) to ZooKeeper.

```
/opt/lucidworks-hdpsearch/solr/ranger_audit_server/scripts/  
add_ranger_audits_conf_to_zk.sh
```

7. Log in as the `solr` or `root` user and run the following command to start Solr on each node.

```
/opt/lucidworks-hdpsearch/solr/ranger_audit_server/scripts/start_solr.sh
```

When Solr starts, a confirmation message appears.

```
Started Solr server on port 6083 (pid=). Happy searching!
```

8. Run the following command **only once** from any node. This command creates the Ranger Audit collection.

```
/opt/lucidworks-hdpsearch/solr/ranger_audit_server/scripts/  
create_ranger_audits_collection.sh
```

9. You can use a web browser to open the Solr Admin Console at the following address:

```
http:<solr_host>:6083/solr
```



Note

You can use the following command to stop Solr:

```
/opt/lucidworks-hdpsearch/solr/ranger_audit_server/scripts/  
stop_solr.sh
```

5.2. 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.

5.2.1. 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 [installed Solr](#) and [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`

`http://{SOLR_HOST}:6083/solr/ranger_audits`

5.2.2. Manually Updating Ambari HDFS Audit Settings



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 should 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 HFDS component logs on the NameNode:

```
hdfs://NAMENODE_FQDN:8020/ranger/audit
```



Note

For a secure cluster, use the following steps to test audit to HDFS for STORM/KAFKA/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).
- Link to `/etc/hadoop/conf/core-site.xml` under `/etc/<component_name>/conf`.
- Verify the service user principal.
- Make sure that the component user has permissions on HDFS.

5.3. 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`
- `audit_solr_urls=http://solr_host:6083/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`
- `XAAUDIT.SOLR.URL=http://solr_host:6083/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.

5.4. 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 four tabs on the Audit page:

- [Access](#)
- [Admin](#)
- [Login Sessions](#)
- [Plugins](#)



Note

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

5.4.1. 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

The screenshot displays three main components of the Hortonworks Data Platform interface:

- Operation : create**: A dialog box for creating a new policy. It shows policy details like Policy ID (13), Policy Name (New-Service-1-20160211205602), and Repository Type (Default). The Policy Details table includes fields such as Policy Name, Policy Description (Default Policy for Service: New-Service), and Policy Status (enabled).
- Ranger Audit Log**: A table of audit logs. The logs show various operations like User profile password changed, User updated, Policy defined, etc., with details like User (u7 or admin), Date (e.g., 10/29/2014 04:54:12 PM), Action (e.g., password change, update, delete), and Session ID.
- Operation : update**: A dialog box for updating a policy. It shows the Policy Name (s1), Repository Type (Storm), Date (11/03/2014 01:32:36 PM IST), and Updated By (admin). The User Permissions table shows changes for user 'hive' from 'Admin , Submit Topology , Kill Topology' to 'Submit Topology , File Download , Admin'. A legend indicates 'Added' (green) and 'Deleted' (red) values.

5.4.2. 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 5.3. Search Criteria

Search Criteria	Description
Access Enforcer	Access enforcer indicates who made the decision to allow or deny. In case of HDFS, the enforcer would XA (Ranger) or Hadoop.

Search Criteria	Description
Access Type	Type of access user attempted (E.G., REVOKE, GRANT, OPEN, USE).
Client IP	IP address of the user system that tried to access the resource.
Result	Shows whether the operation was successfull 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.

Policy ID	Event Time	User	Service Name / Type	Resource Name	Access Type	Result	Access Enforcer	Client IP	Event Count
4	07/30/2015 05:43:27 PM	hbase	hbasedev hbase	testtable2/*/*	revoke	Allowed	ranger-acl	172.22.115.226	1
4	07/30/2015 05:14:59 PM	hbase	hbasedev hbase	testtable2/*/*	grant	Allowed	ranger-acl	172.22.115.226	1
4	07/30/2015 05:14:03 PM	hbase	hbasedev hbase	testtable2	open	Allowed	ranger-acl		1
4	07/30/2015 04:56:30 PM	hbase	hbasedev hbase	icwgr/*/*	grant	Allowed	ranger-acl	172.22.115.207	1
4	07/30/2015 04:51:24 PM	hbase	hbasedev hbase	icwgr	open	Allowed	ranger-acl		1
5	07/30/2015 03:24:26 PM	hive	hive dev hive		USE	Allowed	ranger-acl	172.22.115.226	1
5	07/30/2015 03:23:51 PM	hive	hive dev hive		USE	Allowed	ranger-acl	172.22.115.226	1
--	07/30/2015 11:41:26 AM	hbase	cl1_hadoop hdfs	/apps/hbase/data/oldWALs	READ_EXECUTE	Allowed	hadoop-acl	172.22.115.226	1

5.4.3. Admin

The Admin tab contains all events for the 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-1-20160211265602	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-1-20160211010740	Ranger Policy	admin	02/10/2016 05:07:40 PM	create	52391

You can filter the data based on the following criteria:

Table 5.4. 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.

5.4.4. 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 5.5. 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.
Login Type	The mode through which the user tries to login (by entering username and password).
IP	The IP address of the system through which the user logged in.
User Agent	The login time and date for each session.
Login time	Logs whether or not the login was successful. Possible results can be Success, Wrong Password, Account Disabled, Locked, Password Expired or User Not Found.

Last Updated Time : 02/09/2016 12:54:22 PM

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

5.4.5. 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 5.6. 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.

6. Data Protection: HDFS Encryption

6.1. Ranger KMS Administration Guide

The Ranger Key Management Service (Ranger KMS) is a 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.
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](#).

6.1.1. 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 the *Ambari User's Guide*.)

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 the [Ambari Security Guide](#). Make sure that the Java location is specified in the \$PATH environment variable.



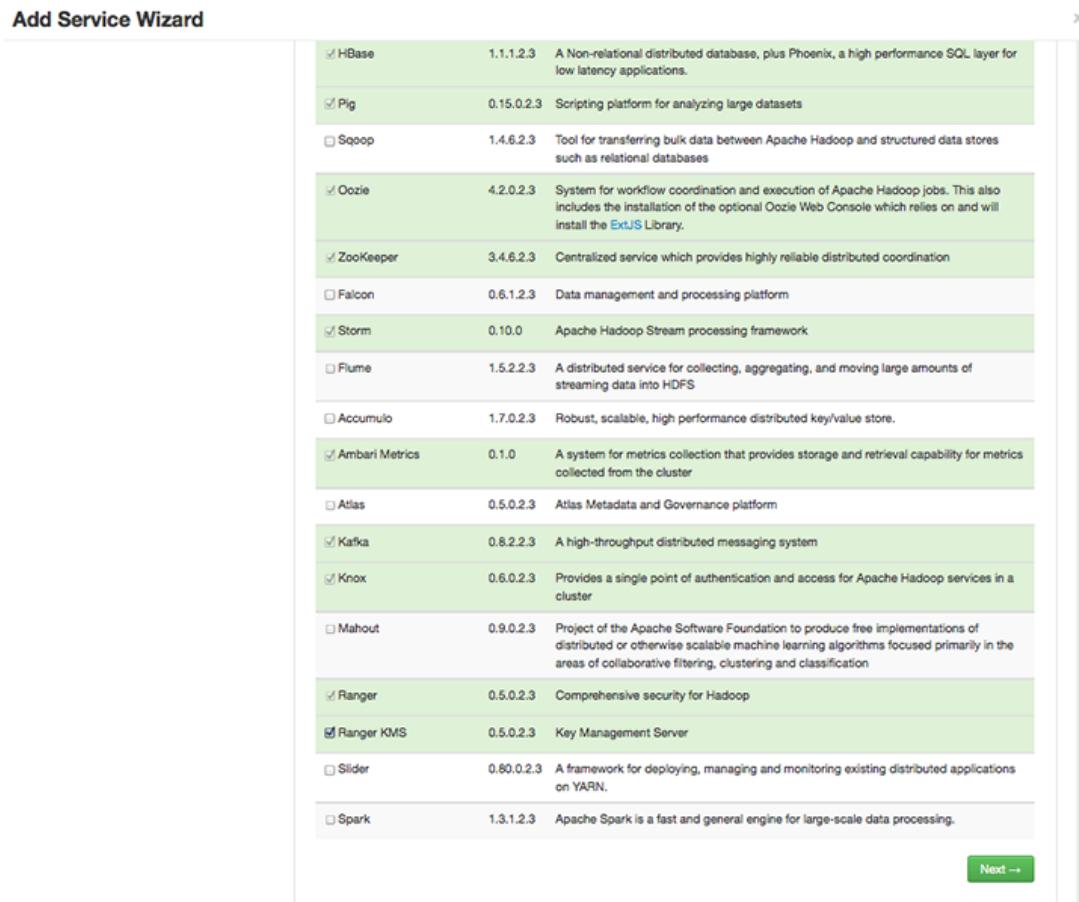
Note

If you use the OpenJDK package, the JCE file is already built into the package.

6.1.1.1. 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**:



The screenshot shows the 'Add Service Wizard' interface. A list of services is displayed in a table format. The 'Ranger KMS' service is highlighted with a green background and has a checked checkbox next to its name. Other services listed include HBase, Pig, Sqoop, Oozie, ZooKeeper, Falcon, Storm, Flume, Accumulo, Ambari Metrics, Atlas, Kafka, Knox, Mahout, Ranger, Slider, and Spark. Each service entry includes its version number and a brief description. A 'Next →' button is visible at the bottom right of the wizard window.

Add Service Wizard		
<input checked="" type="checkbox"/> HBase	1.1.1.2.3	A Non-relational distributed database, plus Phoenix, a high performance SQL layer for low latency applications.
<input checked="" type="checkbox"/> Pig	0.15.0.2.3	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 data stores such as relational databases
<input checked="" type="checkbox"/> Oozie	4.2.0.2.3	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 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 amounts of 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 capability for metrics 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 services in a 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 focused primarily in the 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

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 cores)
vp-os-rh6-my-sec-nokms-150727-1736-3.openstacklocal (15.6 GB, 2 cores)
vp-os-rh6-my-sec-nokms-150727-1736-4.openstacklocal (15.6 GB, 2 cores)
vp-os-rh6-my-sec-nokms-150727-1736-5.openstacklocal (15.6 GB, 2 cores)

Ranger Admin: vp-os-rh6-my-sec-nokms-1507

Ranger UserSync: vp-os-rh6-my-sec-nokms-1507

[Back](#) [Next →](#)

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](#).

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).

- 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	X
Properties key=value (one per line)	<pre>hadoop.kms.proxyuser.keyadmin.groups=* hadoop.kms.proxyuser.keyadmin.hosts=* hadoop.kms.proxyuser.keyadmin.users=*</pre>	 
		<input type="button" value="Cancel"/> <input type="button" value="Add"/>

- 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.

6.1.1.1. Setting up Database Users Without Sharing DBA Credentials

If you 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-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.

6.1.1.1.2. 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.

6.1.1.1.3. 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 [Ranger User 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 principle 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:

```
kadmin.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.

Property	Value	Status
REPOSITORY_CONFIG_USERNAME	keyadmin@EXAMPLE.COM	Green
SQL_CONNECTOR_JAR	/usr/share/java/mysql-connector-java.jar	Green
db_host	mp-secranger-2406-3.openstacklocal	Green
db_user	rangerkms	Green
db_password	Yellow
DB_FLAVOR	MYSQL	Green
db_root_user	root	Green
db_root_password	Green
db_name	rangerkms	Green
KMS_MASTER_KEY_PASWD	Green
REPOSITORY_CONFIG_PASSWORD	Green

6.1.2. 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' popup 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.

6.1.2.1. 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 `http://solr_host:6083/solr/ranger_audits`.

6.1.2.2. 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
`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.)

6.1.3. 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. Change the policy manager URL to point to HTTPS, and specify the keystore & truststore in ews/webapp/WEB-INF/classes/conf/ranger-policymgr-ssl.xml.
 - a. In xasecure.policymgr.clientssl.keystore, provide the location for the keystore that you created in the previous step.
 - b. In xasecure.policymgr.clientssl.keystore.password, provide the password for the keystore (myKeyFilePassword).
 - c. In xasecure.policymgr.clientssl.truststore, provide the location for the truststore that you created in the previous step.
 - d. In 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 interface. The 'Service Details' section includes fields for 'Service Name' (cl1_kms), 'Description' (kms repo), and 'Active Status' (Enabled). The 'Config Properties' section includes fields for 'KMS URL' (kms://https://ip-172-31-26-219.ec2), 'Username' (keyadmin), and 'Password'. A table for 'Add New Configurations' shows one entry: commonNameForCertificate (ip-172-31-26-219.ec2.internal). Buttons at the bottom include 'Test Connection', 'Save' (green), 'Cancel' (black), and 'Delete' (red).

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. Add the following properties and values to the Custom ranger-kms-site list:

Custom ranger-kms-site

ranger.https.attrib.keystore.file	/etc/ranger/kms/conf/ranger-kms-keystore.jks	● ●
ranger.service.https.attrib.client.auth	want	● ●
ranger.service.https.attrib.clientAuth	false	● ●
ranger.service.https.attrib.keystore.file	/etc/ranger/kms/conf/ranger-kms-keystore.jks	● ●
ranger.service.https.attrib.keystore.keyalias	rangerkms	● ●
ranger.service.https.attrib.keystore.pass	rangerkms	● ●
ranger.service.https.attrib.ssl.enabled	true	● ●
ranger.service.https.port	9393	● ●

[Add Property ...](#)

3. Update the value of `kms_port` (in Advanced `kms_env`) to the `ranger.service.https.port` value.
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)

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. Now in the Policy Manager UI, Audit → Plugin tab, you should see an entry for your service name with HTTP Response Code = 200.

6.1.4. 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.

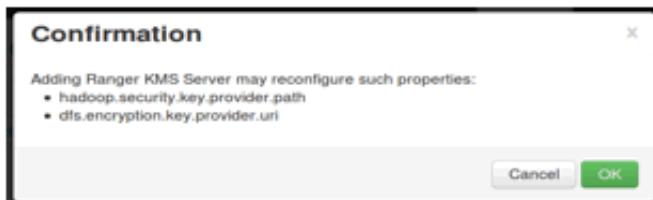
In the Ambari Web UI for the additional node, go to Ranger KMS service # Summary # Service Actions # Add Ranger KMS server.



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  
ip of first node}:2181,{internal ip of second node}:2181, ...
```

```
hadoop.kms.authentication.signer.secret.provider.zookeeper.auth.type=none
```

7. 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@<internal host name1>;<internal  
host name2>...: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.

6.1.5. 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.

6.1.5.1. 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:

The screenshot shows the Ranger Service Manager interface. At the top, there is a navigation bar with tabs for 'Ranger', 'Access Manager', and 'Encryption'. On the far right of the bar, it shows the user is logged in as 'keyadmin'. Below the bar, there is a breadcrumb trail labeled 'Service Manager'. The main content area is titled 'Service Manager' and contains a list of repositories. One repository, 'cl1_kms', is listed under the 'KMS' category. To the right of each repository entry, there are two small icons: a pencil for editing and a trash can for deleting. The entire interface has a clean, modern design with a light gray background and white text.

You will see a list of service details and config properties for the repository:

The screenshot shows the 'Create Service' page in the Ranger Web UI. At the top, there are tabs for 'Ranger', 'Access Manager', and 'Encryption'. The 'Encryption' tab is selected. Below the tabs, a breadcrumb navigation shows 'Service Manager > Edit Service'. The main section is titled 'Create Service'.

Service Details:

- Service Name *: d1_kms
- Description: kms repo
- Active Status: Enabled (radio button selected)

Config Properties:

- KMS URL *: kms://http@vp-os-rh6-my-sec-150
- Username *: keyadmin@EXAMPLE.COM
- Password *: ****

Add New Configurations:

Name	Value	X
		X

Buttons:

- + (Add new configuration)
- Test Connection (button)
- Save (green button)
- Cancel (black button)
- Delete (red button)

6.1.5.2. 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 Key Management interface under the KMS tab. A search bar at the top has 'cl1_kms' typed into it. Below the search bar is a table listing six existing keys:

Key Name	Cipher	Version	Attributes	Length	Created Date	Action
sensitivefolder	AES/CTR/NoPadding	1	key.acl.name → SensitiveFolder	128	08/06/2015 01:30:44 PM	<input type="checkbox"/> <input type="button"/>
test	AES/CTR/NoPadding	1	key.acl.name → test	128	08/13/2015 01:49:35 PM	<input type="checkbox"/> <input type="button"/>
testkeyfromcli	AES/CTR/NoPadding	1	key.acl.name → testkeyfromcli	128	07/24/2015 06:04:36 PM	<input type="checkbox"/> <input type="button"/>
testkeyfromui	AES/CTR/NoPadding	1	key.acl.name → testkeyfromui	128	07/24/2015 06:04:16 PM	<input type="checkbox"/> <input type="button"/>
testkeygmi	AES/CTR/NoPadding	1	key.acl.name → testkeyGMI	128	08/06/2015 02:02:40 PM	<input type="checkbox"/> <input type="button"/>
tk1	AES/CTR/NoPadding	1	key.acl.name → tk1	128	08/25/2015 12:22:23 PM	<input type="checkbox"/> <input type="button"/>

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.

The screenshot shows the Ranger Key Create interface under the KMS tab. The breadcrumb navigation shows 'KMS > cl1_kms > Key Create'. The form fields are as follows:

- Key Name *: An input field.
- Cipher: A dropdown menu set to 'AES/CTR/NoPadding'.
- Length: A dropdown menu set to '128'.
- Description: A text area.
- Attributes: A table with two columns, 'Name' and 'Value'. It contains one row with a 'Name' input field and a 'Value' input field. There is a red 'x' button to remove this row, and a '+' button below the table to add more rows.

At the bottom of the form are 'Save' and 'Cancel' buttons.

6.1.5.3. 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.

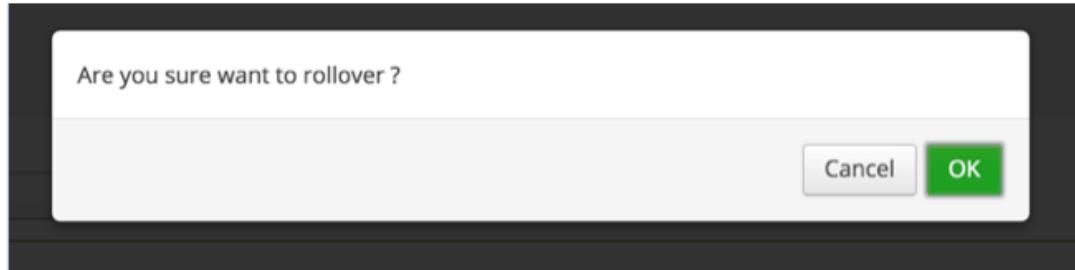
After a key is rotated in Ranger KMS, new files will have the file key encrypted by the new master key for the encryption zone.

To rotate a key, click the edit button next to the key name in the list of keys, as shown in the following screenshot:

Policy ID	Policy Name	Status	Audit Logging	Groups	Users	Action
1	cl1_kms-1-20150724233747	Enabled	Enabled	public	keyadmin	

Edit the key information, and then press Save.

When asked to confirm the rollover, click "OK":



6.1.5.4. 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 popup window; confirm or cancel.

6.1.6. Ranger KMS Properties

This chapter describes configuration properties for the Ranger Key Management Service (KMS).

Table 6.1. 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 6.2. 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 6.3. 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.

Property Name	Default Value	Description
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 6.4. 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 is set to true when multiple KMSs are used, or when the key operations are from different modes (Ranger UI, CURL, or hadoop command), it might cause inconsistency.
hadoop.kms.authentication.type	simple	Authentication type for the Ranger KMS. Can be either "simple" or "kerberos".

Property Name	Default Value	Description
hadoop.kms.authentication.signer.secret.provider.zookeeper.path	/hadoop-kms/hadoop-auth-signature-secret	The Zookeeper ZNode path where the Ranger KMS instances will store and retrieve the secret from.
hadoop.kms.authentication.signer.secret.provider.zookeeper.kerberos.principal	kms/#HOSTNAME#	The Kerberos service principal used to connect to Zookeeper
hadoop.kms.authentication.signer.secret.provider.zookeeper.kerberos.keytab	/etc/hadoop/conf/kms.keytab	The absolute path for the Kerberos keytab with the credentials to connect to Zookeeper.
hadoop.kms.authentication.signer.secret.provider.zookeeper.connection.string	#HOSTNAME#:#PORT#,...	The Zookeeper connection string, a list of hostnames and port comma separated. For example: <FQDN for first instance>:2181,<FQDN for second instance>:2181
hadoop.kms.authentication.signer.secret.provider.zookeeper.auth.type	kerberos	Zookeeper authentication type: 'none' or 'sasl' (Kerberos)
hadoop.kms.authentication.signer.secret.provider	random	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'.
hadoop.kms.authentication.kerberos.principal	HTTP/localhost	The Kerberos principal to use for the HTTP endpoint. The principal must start with 'HTTP/' as per the Kerberos HTTP SPNEGO specification.
hadoop.kms.authentication.kerberos.name.rules	DEFAULT	Rules used to resolve Kerberos principal names.
hadoop.kms.authentication.kerberos.keytab	\${user.home}/kms.keytab	Path to the keytab with credentials for the configured Kerberos principal.
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 6.5. 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.

Property Name	Default Value	Description
		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 6.6. 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 6.7. 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.

Property Name	Default Value	Description
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.

6.1.7. Troubleshooting Ranger KMS

Table 6.8. 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.

6.2. 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.

6.2.1. 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:

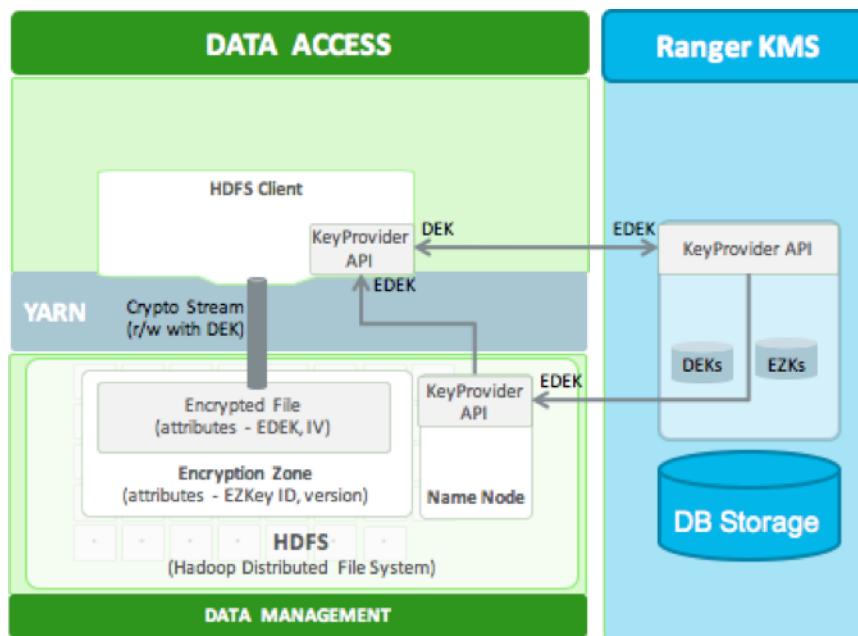
- **Encryption key:** A new level of permission-based access protection, in addition to standard HDFS permissions.
- **HDFS encryption zone:** A special HDFS directory within which all data is encrypted upon write, and decrypted upon read.
- Each encryption zone is associated with an encryption key that 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 6.1. 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](#).

6.2.2. 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 Guide](#).

6.2.3. 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.

6.2.3.1. 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.

6.2.3.1.1. 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'.

6.2.3.1.2. 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.1.0.0.1e /usr/lib64/libcrypto.so
```

On MapReduce client hosts, install the `openssl-devel` package:

```
$ sudo yum install openssl-devel
```

6.2.3.1.3. 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.

6.2.3.2. 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 `enrcr` 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.

The screenshot shows the Ranger interface with the 'Encryption' tab selected. Under the 'Key Management' section, there is a 'Select Service' dropdown menu. The menu is open, displaying two entries: 'cl1_kms' (which is highlighted) and 'dt1_kms'. Below the dropdown is a search bar labeled 'Search for your key' and a green 'Add New Key' button.

Key Name	Cipher	Version	Attributes	Length	Created Date	Action
sensitivefolder	AES/CTR/NoPadding	1	key.acl.name → SensitiveFolder	128	08/06/2015 01:30:44 PM	<input type="checkbox"/> <input type="button" value="Edit"/>
test	AES/CTR/NoPadding	1	key.acl.name → test	128	08/13/2015 01:49:35 PM	<input type="checkbox"/> <input type="button" value="Edit"/>
testkeyfromcli	AES/CTR/NoPadding	1	key.acl.name → testkeyfromcli	128	07/24/2015 06:04:36 PM	<input type="checkbox"/> <input type="button" value="Edit"/>
testkeyfromui	AES/CTR/NoPadding	1	key.acl.name → testkeyfromui	128	07/24/2015 06:04:16 PM	<input type="checkbox"/> <input type="button" value="Edit"/>
testkeygmi	AES/CTR/NoPadding	1	key.acl.name → testkeyGMI	128	08/06/2015 02:02:40 PM	<input type="checkbox"/> <input type="button" value="Edit"/>
tk1	AES/CTR/NoPadding	1	key.acl.name → tk1	128	08/25/2015 12:22:23 PM	<input type="checkbox"/> <input type="button" value="Edit"/>

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.

The screenshot shows the 'Key Create' interface under the 'KMS > cl1_kms' path. The 'Key Detail' form contains the following fields:

- Key Name *: A text input field.
- Cipher: A dropdown menu set to 'AES/CTR/NoPadding'.
- Length: A dropdown menu set to '128'.
- Description: A text area for additional notes.
- Attributes: A table with columns 'Name' and 'Value'. It currently has one row with a red 'x' button to delete it.
- Buttons: 'Save' and 'Cancel' at the bottom.

For information about rolling over and deleting keys, see [Using the Ranger Key Management Service](#) in the *Ranger KMS Administration Guide*.



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 the [Ambari Security Guide](#).

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](#) in the *Ranger KMS Administration Guide*.



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.

6.2.3.3. 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
```

6.2.3.4. 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 [Creating an HDFS Admin User](#) 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.

6.2.3.5. 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---
```

6.2.3.6. 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/OriginalPathToDeleteFile`).

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/OriginalPathToDeleteFile`. 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](#).

6.2.4. Configuring HDP Services for HDFS Encryption

The following HDP components support HDFS data at rest encryption:

- Hive
- HBase
- Sqoop
- YARN
- MapReduce
- Oozie
- WebHDFS

The following components do not currently support HDFS data at rest encryption:

- Hive on Tez
- Spark
- HDP Search
- Storm
- Accumulo
- Falcon

The remainder of this section describes scenarios and access considerations for accessing HDFS-encrypted files from supporting HDP components.

6.2.4.1. Hive

Recommendation: Store Hive data in an HDFS path called `/apps/hive`.

6.2.4.1.1. 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.")

6.2.4.1.2. 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.

6.2.4.1.3. 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`.

6.2.4.1.4. 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 datas 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.
- Previously, an `INSERT OVERWRITE` on a partitioned table inherited permissions for new data from the existing partition directory. With encryption enabled, permissions are inherited from the table.
- 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](#).

6.2.4.2. 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.

6.2.4.2.1. 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.

6.2.4.2.2. 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.

6.2.4.2.3. 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.

6.2.4.3. Sqoop

Following are considerations for using Sqoop to import or export HDFS-encrypted data.

6.2.4.3.1. 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.

6.2.4.4. 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.

6.2.4.4.1. 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

6.2.4.5. Oozie

6.2.4.5.1. 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
```

6.2.4.6. WebHDFS

6.2.4.6.1. Recommendations

WebHDFS is supported for writing and reading files to and from encryption zones.

6.2.4.6.1.1. 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_eek` 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:

V4	admin	about a month ago	HDP-2.3
V3	admin	about a month ago	HDP-2.3
V2	admin	about a month ago	HDP-2.3
V1	admin	about a month ago	HDP-2.3

hadoop.kms.blacklist.
DECRYPT_EEK

- 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.

The screenshot shows the 'Create Policy' interface in the Ranger KMS. The policy details are as follows:

- Policy Name ***: hdfs-svc
- Key Name ***:
- Description**:
- Audit Logging**: YES

In the 'User and Group Permissions' section, a row is selected for 'hdfs'. The 'Permissions' column contains two checked checkboxes: 'Decrypt EEK' and 'Generate EEK'. The 'Delegate Admin' column has an unchecked checkbox and a delete icon.

At the bottom, there are 'Add' and 'Cancel' buttons.

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](#).

6.2.5. 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 `enctr`.
- **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, add `operator` to `dfs.permissions.superusergroup`.
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.