cloudera®

Cloudera Security

Important Notice
© 2010-2021 Cloudera, Inc. All rights reserved.

Cloudera, the Cloudera logo, and any other product or
service names or slogans contained in this document are trademarks of Cloudera and its suppliers or licensors, and may not be copied, imitated or used, in whole or in part, without the prior written permission of Cloudera or the applicable trademark holder. If this documentation includes code, including but not limited to, code examples, Cloudera makes this available to you under the terms of the Apache License, Version 2.0, including any required notices. A copy of the Apache License Version 2.0, including any notices, is included herein. A copy of the Apache License Version 2.0 can also be found here: https://opensource.org/licenses/Apache-2.0

Hadoop and the Hadoop elephant logo are trademarks of the Apache Software Foundation. All other trademarks, registered trademarks, product names and company names or logos mentioned in this document are the property of their respective owners. Reference to any products, services, processes or other information, by trade name, trademark, manufacturer, supplier or otherwise does not constitute or imply endorsement, sponsorship or recommendation thereof by us.

Complying with all applicable copyright laws is the responsibility of the user. Without limiting the rights under copyright, no part of this document may be reproduced, stored in or introduced into a retrieval system, or transmitted in any form or by any means (electronic, mechanical, photocopying, recording, or otherwise), or for any purpose, without the express written permission of Cloudera.

Cloudera may have patents, patent applications, trademarks, copyrights, or other intellectual property rights covering subject matter in this document. Except as expressly provided in any written license agreement from Cloudera, the furnishing of this document does not give you any license to these patents, trademarks copyrights, or other intellectual property. For information about patents covering Cloudera products, see http://tiny.cloudera.com/patents.

The information in this document is subject to change without notice. Cloudera shall not be liable for any damages resulting from technical errors or omissions which may be present in this document, or from use of this document.

Cloudera, Inc.
395 Page Mill Road
Palo Alto, CA 94306
info@cloudera.com
US: 1-888-789-1488
Intl: 1-650-362-0488
www.cloudera.com

Release Information

Version: Cloudera Enterprise 5.3.x
Date: February 3, 2021

Table of Contents

About this Guide.....8

Authentication.....9

Configuring Authentication in Cloudera Manager.....9

Cloudera Manager User Accounts.....10

Configuring External Authentication for Cloudera Manager.....11

Kerberos Principals and Keytabs.....17

Enabling Kerberos Authentication Using the Wizard.....17

Enabling Kerberos Authentication for Single User Mode or Non-Default Users.....26

Viewing and Regenerating Kerberos Principals.....27

Mapping Kerberos Principals to Short Names.....28

Configuring YARN for Long-running Applications.....29

Enabling Kerberos Authentication Without the Wizard.....30

Configuring Authentication in Cloudera Navigator.....41

Configuring External Authentication for Cloudera Navigator.....41

Managing Users and Groups for Cloudera Navigator.....43

Configuring Authentication in CDH Using the Command Line.....45

Enabling Kerberos Authentication for Hadoop Using the Command Line.....45

Flume Authentication.....67

HBase Authentication.....70

HCatalog Authentication.....74

Hive Authentication.....76

HttpFS Authentication.....83

Hue Authentication.....85

Impala Authentication.....96

Llama Authentication.....101

Oozie Authentication.....102

Search Authentication.....104

ZooKeeper Authentication.....108

FUSE Kerberos Configuration.....110

Using kadmin to Create Kerberos Keytab Files.....111

Configuring the Mapping from Kerberos Principals to Short Names.....112

Enabling Debugging Output for the Sun Kerberos Classes.....115

Configuring a Cluster-dedicated MIT KDC with Cross-Realm Trust.....115

When to use kadmin.local and kadmin.....115

Setting up a Cluster-Dedicated KDC and Default Realm for the Hadoop Cluster.....116

Integrating Hadoop Security with Active Directory.....120

Configuring a Local MIT Kerberos Realm to Trust Active Directory....121
Integrating Hadoop Security with Alternate Authentication....122
Configuring the AuthenticationFilter to use Kerberos....123
Creating an AltKerberosAuthenticationHandler Subclass....123
Enabling Your AltKerberosAuthenticationHandler Subclass....123
Example Implementation for Oozie....124
Configuring LDAP Group Mappings....125
Using Cloudera Manager....127
Using the Command Line....128
Hadoop Users in Cloudera Manager and CDH....129
Authenticating Kerberos Principals in Java Code....134
Using a Web Browser to Access an URL Protected by Kerberos HTTP SPNEGO....134
Troubleshooting Authentication Issues....138
Sample Kerberos Configuration files: krb5.conf, kdc.conf, kadm5.acl....138
Potential Security Problems and Their Solutions....140

Encryption....149
SSL Certificates Overview....149
Creating Certificates....150
Creating Java Keystores and Truststores....151
Private Key and Certificate Reuse Across Java Keystores and OpenSSL....154
Configuring TLS Security for Cloudera Manager....155
Configuring TLS Encryption Only for Cloudera Manager....156
Level 1: Configuring TLS Encryption for Cloudera Manager Agents....160
Level 2: Configuring TLS Verification of Cloudera Manager Server by the Agents....161
Level 3: Configuring TLS Authentication of Agents to the Cloudera Manager Server....163
HTTPS Communication in Cloudera Manager....167
Troubleshooting SSL/TLS Connectivity....169
Configuring SSL for Cloudera Navigator....170
Configuring SSL/TLS Encryption for CDH Services....170
Prerequisites....170
Hadoop Services as SSL Servers and Clients....170
Configuring SSL for HDFS, YARN and MapReduce....171
Configuring SSL for HBase....173
Configuring SSL for Oozie....174
Configuring Encrypted Communication Between Hive and Client Drivers....175
Configuring SSL for Hue....177
Configuring SSL for Impala....179
Configuring HttpFS to use SSL....180
Encrypted Shuffle and Encrypted Web UIs....181
HDFS Data At Rest Encryption....187
Use Cases....187
Architecture....188

`crypto` *Command Line Interface*....188
Enabling HDFS Encryption on a Cluster....189
DistCp Considerations....193
Attack Vectors....194
Configuring the Key Management Server (KMS)....195
Securing the Key Management Server (KMS)....198
Configuring CDH Services for HDFS Encryption....205
HDFS Encryption Troubleshooting....209
Configuring Encrypted HDFS Data Transport....210
Using Cloudera Manager....210
Using the Command Line....211

Authorization....212

Cloudera Manager User Roles....212
User Roles....212
Determining the Role of the Currently Logged in User....214
Removing the Full Administrator User Role....214
Cloudera Navigator User Roles....214
User Roles....214
Determining the Roles of the Currently Logged in User....215
Enabling HDFS Extended ACLs....215
Enabling ACLs....215
Commands....216
The Sentry Service....217
Prerequisites....217
Terminologies....217
Privilege Model....218
User to Group Mapping....218
Appendix: Authorization Privilege Model for Hive and Impala....220
Installing and Upgrading the Sentry Service....223
Migrating from Sentry Policy Files to the Sentry Service....226
Configuring the Sentry Service....226
Sentry Debugging and Failure Scenarios....231
Hive SQL Syntax for Use with Sentry....232
Synchronizing HDFS ACLs and Sentry Permissions....235
Reporting Metrics for the Sentry Service....240
Sentry Policy File Authorization....240
Prerequisites....240
Terminologies....240
Privilege Model....241
User to Group Mapping....241
Policy File....242
Sample Sentry Configuration Files....244

Accessing Sentry-Secured Data Outside Hive/Impala....246
Debugging Failed Sentry Authorization Requests....247
Authorization Privilege Model for Hive and Impala....247
Installing and Upgrading Sentry for Policy File Authorization....250
Configuring Sentry Policy File Authorization Using Cloudera Manager....251
Configuring Sentry Policy File Authorization Using the Command Line....255
Enabling Sentry Authorization for Impala....257
The Sentry Privilege Model....257
Starting the impalad Daemon with Sentry Authorization Enabled....258
Using Impala with the Sentry Service (CDH 5.1 or higher only)....259
Using Impala with the Sentry Policy File....259
Setting Up Schema Objects for a Secure Impala Deployment....264
Privilege Model and Object Hierarchy....265
Debugging Failed Sentry Authorization Requests....268
Managing Sentry for Impala through Cloudera Manager....268
The DEFAULT Database in a Secure Deployment....268
Enabling Sentry Authorization for Search....268
Roles and Collection-Level Privileges....269
Users and Groups....269
Setup and Configuration....270
Policy File....270
Sample Configuration....270
Enabling Sentry in Cloudera Search for CDH 5....271
Providing Document-Level Security Using Sentry....272
Enabling Secure Impersonation....274
Debugging Failed Sentry Authorization Requests....274
Appendix: Authorization Privilege Model for Search....275
Configuring HBase Authorization....277
Understanding HBase Access Levels....277
Enable HBase Authorization....279
Configure Access Control Lists for Authorization....279

Overview of Impala Security....281
Security Guidelines for Impala....281
Securing Impala Data and Log Files....282
Installation Considerations for Impala Security....283
Securing the Hive Metastore Database....283
Securing the Impala Web User Interface....283

Miscellaneous Topics....285
Jsvc, Task Controller and Container Executor Programs....285
MRv1 and YARN: The jsvc Program....285

MRv1 Only: The Linux TaskController Program.....285
YARN Only: The Linux Container Executor Program.....285
Task-controller and Container-executor Error Codes.....286
MRv1 ONLY: Task-controller Error Codes.....286
YARN ONLY: Container-executor Error Codes.....288
Sqoop, Pig, and Whirr Security Support Status.....289
Setting Up a Gateway Node to Restrict Cluster Access.....289
Installing and Configuring the Firewall and Gateway.....290
Accessing HDFS.....290
Submitting and Monitoring Jobs.....290
Logging a Security Support Case.....291
Kerberos Issues.....291
SSL/TLS Issues.....291
LDAP Issues.....291
Using Antivirus Software on CDH Hosts.....292

Appendix: Apache License, Version 2.0.....293

About this Guide

This guide is intended for system administrators who want to secure a cluster using data encryption, user authentication, and authorization techniques. This topic also provides information about Hadoop security programs and shows you how to set up a gateway to restrict access.

Authentication

The purpose of authentication in Hadoop, as in other systems, is simply to prove that a user or service is who he or she claims to be.

Typically, authentication in enterprises is managed through a single distributed system, such as a Lightweight Directory Access Protocol (LDAP) directory. LDAP authentication consists of straightforward username/password services backed by a variety of storage systems, ranging from file to database.

A common enterprise-grade authentication system is Kerberos. Kerberos provides strong security benefits including capabilities that render intercepted authentication packets unusable by an attacker. It virtually eliminates the threat of impersonation by never sending a user's credentials in cleartext over the network.

Several components of the Hadoop ecosystem are converging to use Kerberos authentication with the option to manage and store credentials in LDAP or AD. For example, Microsoft's Active Directory (AD) is an LDAP directory that also provides Kerberos authentication for added security.

Before you use this guide to configure Kerberos on your cluster, ensure you have a working KDC (MIT KDC or Active Directory), set up. You can then use Cloudera Manager's Kerberos wizard to automate several aspects of Kerberos configuration on your cluster.

Important:

- You can use either Cloudera Manager or the following command-line instructions to complete this configuration.
- This information applies specifically to CDH 5.3.x . If you use an earlier version of CDH, see the documentation for that version located at Cloudera Documentation.

Configuring Authentication in Cloudera Manager

Why Use Cloudera Manager to Implement Kerberos Authentication?

If you do not use Cloudera Manager to implement Hadoop security, you must manually create and deploy the Kerberos principals and keytabs on *every* host in your cluster. If you have a large number of hosts, this can be a time-consuming and error-prone process. After creating and deploying the keytabs, you must also manually configure properties in the `core-site.xml`, `hdfs-site.xml`, `mapred-site.xml`, and `taskcontroller.cfg` files on *every* host in the cluster to enable and configure Hadoop security in HDFS and MapReduce. You must also manually configure properties in the `oozie-site.xml` and `hue.ini` files on certain cluster hosts in order to enable and configure Hadoop security in Oozie and Hue.

Cloudera Manager enables you to automate all of those manual tasks. Cloudera Manager can automatically create and deploy a keytab file for the `hdfs` user and a keytab file for the `mapred` user on every host in your cluster, as well as keytab files for the `oozie` and hue users on select hosts. The `hdfs` keytab file contains entries for the `hdfs` principal and a `host` principal, and the `mapred` keytab file contains entries for the `mapred` principal and a `host` principal. The `host` principal will be the same in both keytab files. The `oozie` keytab file contains entries for the `oozie` principal and a `HTTP` principal. The hue keytab file contains an entry for the hue principal. Cloudera Manager can also automatically configure the appropriate properties in the `core-site.xml`, `hdfs-site.xml`, `mapred-site.xml`, and `taskcontroller.cfg` files on every host in the cluster, and the appropriate properties in `oozie-site.xml` and `hue.ini` for select hosts. Lastly, Cloudera Manager can automatically start up the NameNode, DataNode, Secondary NameNode, JobTracker, TaskTracker, Oozie Server, and Hue roles once all the appropriate configuration changes have been made.

Ways to Configure Kerberos Authentication Using Cloudera Manager

You can use *one* of the following ways to set up Kerberos authentication on your cluster using Cloudera Manager:

- Cloudera Manager 5.1 introduced a new wizard to automate the procedure to set up Kerberos on a cluster. Using the KDC information you enter, the wizard will create new principals and keytab files for your CDH services. The wizard can be used to deploy the `krb5.conf` file cluster-wide, and automate other manual tasks such as stopping all services, deploying client configuration and restarting all services on the cluster.

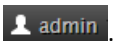
  If you want to use the Kerberos wizard, follow the instructions at Enabling Kerberos Authentication Using the Wizard on page 17.

- If you do not want to use the Kerberos wizard, follow the instructions at Enabling Kerberos Authentication Without the Wizard on page 30.

Cloudera Manager User Accounts

Minimum Required Role: User Administrator (also provided by **Full Administrator**)

Access to Cloudera Manager features is controlled by user accounts. A user account identifies how a user is authenticated and determines what privileges are granted to the user.

When you are logged in to the Cloudera Manager Admin Console, the username you are logged in as is at the far right of the top navigation bar—for example, if you are logged in as *admin* you will see 

.

A user with the User Administrator or Full Administrator role manages user accounts through the **Administration** > **Users** page.

User Authentication

Cloudera Manager provides several mechanisms for authenticating users. You can configure Cloudera Manager to authenticate users against the Cloudera Manager database or against an external authentication service. The external authentication service can be an LDAP server (Active Directory or an OpenLDAP compatible directory), or you can specify another external service. Cloudera Manager also supports using the Security Assertion Markup Language (SAML) to enable single sign-on.

If you are using LDAP or another external service you can configure Cloudera Manager so that it can use both methods of authentication (internal database and external service), and you can determine the order in which it performs these searches. If you select an external authentication mechanism, Full Administrator users can always authenticate against the Cloudera Manager database. This is to prevent locking everyone out if the authentication settings are misconfigured—such as with a bad LDAP URL.

With external authentication, you can restrict login access to members of specific groups, and can specify groups whose members are automatically given Full Administrator access to Cloudera Manager.

Users accounts in the Cloudera Manager database page show **Cloudera Manager** in the User Type column. User accounts in an LDAP directory or other external authentication mechanism show **External** in the User Type column.

User Roles

User accounts include the user's role, which determines the Cloudera Manager features visible to the user and the actions the user can perform. All the tasks in the Cloudera Manager documentation indicate which role is required to perform the task. For more information about user roles, see Cloudera Manager User Roles on page 212.

Determining the Role of the Currently Logged in User

1. Click the logged-in username at the far right of the top navigation bar. The role displays right under the username. For example:

Full Administrator

Changing the Logged-In Internal User Password

1. Right-click the logged-in username at the far right of the top navigation bar and select **Change Password**.
2. Enter the current password and a new password twice, and then click **Update**.

Adding an Internal User Account

1. Select **Administration** > **Users**.
2. Click the **Add User** button.
3. Enter a username and password.
4. In the Role drop-down, select a role for the new user.
5. Click **Add**.

Assigning User Roles

1. Select **Administration** > **Users**.
2. Check the checkbox next to one or more usernames.
3. Select **Actions for Selected** > **Assign User Roles**.
4. In the drop-down, select the role.
5. Click the **Assign Role** button.

Changing an Internal User Account Password

1. Select **Administration** > **Users**.
2. Click the **Change Password** button next to a username with User Type **Cloudera Manager**.
3. Type the new password and repeat it to confirm.
4. Click the **Update** button to make the change.

Deleting Internal User Accounts

1. Select **Administration** > **Users**.
2. Check the checkbox next to one or more usernames with User Type **Cloudera Manager**.
3. Select **Actions for Selected** > **Delete**.
4. Click the **OK** button. (There is no confirmation of the action.)

Configuring External Authentication for Cloudera Manager

Minimum Required Role: User Administrator (also provided by **Full Administrator**)

> **Important: This feature is available only with a Cloudera Enterprise license.**
>
> For other licenses, the following applies:
>
> - Cloudera Express- The feature is not available.
> - Cloudera Enterprise Data Hub Edition Trial - The feature is available until you end the trial or the trial license expires.
>
> To obtain a license for Cloudera Enterprise, fill in this form or call 866-843-7207. After you install a Cloudera Enterprise license, the feature will be available.

Cloudera Manager supports user authentication against an internal database and against an external service. The following sections describe how to configure the supported external services.

Configuring Authentication Using Active Directory

1. Select **Administration** > **Settings**.
2. In the left-hand column, select the **External Authentication** category.
3. In the **Authentication Backend Order** field, select the order in which Cloudera Manager should attempt its authentication. You can choose to authenticate users using just one of the methods (using Cloudera Manager's

own database is the default), or you can set it so that if the user cannot be authenticated by the first method, it will attempt using the second method.

4. For **External Authentication Type**, select **Active Directory**.
5. In the **LDAP URL** property, provide the URL of the Active Directory server.
6. In the **Active Directory NT Domain** property, provide the NT domain to authenticate against.
7. In the **LDAP User Groups** property, optionally provide a comma-separated list of case-sensitive LDAP group names. If this list is provided, only users who are members of one or more of the groups in the list will be allowed to log into Cloudera Manager. If this property is left empty, all authenticated LDAP users will be able to log into Cloudera Manager. For example, if there is a group called `CN=ClouderaManagerUsers,OU=Groups,DC=corp,DC=com`, add the group name `ClouderaManagerUsers` to the **LDAP User Groups** list to allow members of that group to log in to Cloudera Manager.
8. To automatically assign a role to users when they log in, provide a comma-separated list of LDAP group names in the following properties:

   - LDAP Full Administrator Groups
   - LDAP User Administrator Groups
   - LDAP Cluster Administrator Groups
   - LDAP BDR Administrator Groups
   - LDAP Configurator Groups
   - LDAP Navigator Administrator Groups
   - LDAP Operator Groups
   - LDAP Limited Operator Groups
   - LDAP Auditor Groups

   If you specify groups in these properties, users must also be a member of at least one of the groups specified in the **LDAP User Groups** property or they will not be allowed to log in. If these properties are left empty, users will be assigned to the Read-Only role and any other role assignment must be performed manually by an Administrator.

   > **Note:** A user that is added to an LDAP group will not automatically be assigned the corresponding role in the internal Cloudera Manager database. Hence, the Users page in Cloudera Manager will display such users' roles as Read-Only, as this page only queries the Cloudera Manager database, and not LDAP.

Configuring Authentication Using an OpenLDAP-compatible Server

For an OpenLDAP-compatible directory, you have several options for searching for users and groups:

- You can specify a single base Distinguished Name (DN) and then provide a "Distinguished Name Pattern" to use to match a specific user in the LDAP directory.
- Search filter options let you search for a particular user based on somewhat broader search criteria – for example Cloudera Manager users could be members of different groups or organizational units (OUs), so a single pattern won't find all those users. Search filter options also let you find all the groups to which a user belongs, to help determine if that user should have login or admin access.

1. Select **Administration** > **Settings**.
2. In the left-hand column, select the **External Authentication** category.
3. In the **Authentication Backend Order** field, select the order in which Cloudera Manager should attempt its authentication. You can choose to authenticate users using just one of the methods (using Cloudera Manager's own database is the default), or you can set it so that if the user cannot be authenticated by the first method, it will attempt using the second method.
4. For **External Authentication Type**, select **LDAP**.
5. In the **LDAP URL** property, provide the URL of the LDAP server and (optionally) the base Distinguished Name (DN) (the search base) as part of the URL — for example `ldap://ldap-server.corp.com/dc=corp,dc=com`.

6. If your server does not allow anonymous binding, provide the user DN and password to be used to bind to the directory. These are the **LDAP Bind User Distinguished Name** and **LDAP Bind Password** properties. By default, Cloudera Manager assumes anonymous binding.
7. Use one of the following methods to search for users and groups:
   - You can search using User or Group search filters, using the **LDAP User Search Base**, **LDAP User Search Filter**, **LDAP Group Search Base** and **LDAP Group Search Filter** settings. These allow you to combine a base DN with a search filter to allow a greater range of search targets.

     For example, if you want to authenticate users who may be in one of multiple OUs, the search filter mechanism will allow this. You can specify the User Search Base DN as `dc=corp,dc=com` and the user search filter as `uid={0}`. Then Cloudera Manager will search for the user anywhere in the tree starting from the Base DN. Suppose you have two OUs—`ou=Engineering` and `ou=Operations`—Cloudera Manager will find User "foo" if it exists in either of these OUs, that is, `uid=foo,ou=Engineering,dc=corp,dc=com` or `uid=foo,ou=Operations,dc=corp,dc=com`.

     You can use a user search filter along with a DN pattern, so that the search filter provides a fallback if the DN pattern search fails.

     The Groups filters let you search to determine if a DN or username is a member of a target group. In this case, the filter you provide can be something like `member={0}` where `{0}` will be replaced with the **DN** of the user you are authenticating. For a filter requiring the username, `{1}` may be used, as `memberUid={1}`. This will return a list of groups the user belongs to, which will be compared to the list in the group properties discussed in step 8 of Configuring Authentication Using Active Directory on page 11.

     OR

   - Alternatively, specify a single base Distinguished Name (DN) and then provide a "Distinguished Name Pattern" in the **LDAP Distinguished Name Pattern** property.

     Use `{0}` in the pattern to indicate where the username should go. For example, to search for a distinguished name where the `uid` attribute is the username, you might provide a pattern similar to `uid={0},ou=People,dc=corp,dc=com`. Cloudera Manager substitutes the name provided at login into this pattern and performs a search for that specific user. So if a user provides the username "foo" at the Cloudera Manager login page, Cloudera Manager will search for the DN `uid=foo,ou=People,dc=corp,dc=com`.

     If you provided a base DN along with the URL, the pattern only needs to specify the rest of the DN pattern. For example, if the URL you provide is `ldap://ldap-server.corp.com/dc=corp,dc=com`, and the pattern is `uid={0},ou=People`, then the search DN will be `uid=foo,ou=People,dc=corp,dc=com`.

8. Restart the Cloudera Manager Server.

Configuring Cloudera Manager to Use LDAPS

If the LDAP server certificate has been signed by a trusted Certificate Authority (that is, VeriSign, GeoTrust, and so on), steps 1 and 2 below may not be necessary.

1. Copy the CA certificate file to the Cloudera Manager Server host.
2. Import the CA certificate(s) from the CA certificate file to the local truststore. The default truststore is located in the `$JAVA_HOME/jre/lib/security/cacerts` file. This contains the default CA information shipped with the JDK. Create an alternate default file called `jssecacerts` in the same location as the `cacerts` file. You can now safely append CA certificates for any private or public CAs not present in the default `cacerts` file, while keeping the original file intact.

For our example, we will follow this recommendation by copying the default `cacerts` file into the new `jssecacerts` file, and then importing the CA certificate to this alternate truststore.

```
$ cp $JAVA_HOME/jre/lib/security/cacerts \
  $JAVA_HOME/jre/lib/jssecacerts
```

```
$ /usr/java/latest/bin/keytool -import -alias nt_domain_name \
-keystore /usr/java/latest/jre/lib/security/jssecacerts -file path_to_cert
```

Note:

- The default password for the cacerts store is **changeit**.
- The alias can be any name (not just the domain name).

3. Configure the **LDAP URL** property to use `ldaps://ldap_server` instead of `ldap://ldap_server`.
4. Restart the Cloudera Manager Server.

Configuring Authentication Using an External Program

You can configure Cloudera Manager to use an external authentication program of your own choosing. Typically, this may be a custom script that interacts with a custom authentication service. Cloudera Manager will call the external program with the username as the first command line argument. The password is passed over `stdin`. Cloudera Manager assumes the program will return the following exit codes identifying the user role for a successful authentication:

- 0 - Read-Only
- 1 - Full Administrator
- 2 - Limited Operator
- 3 - Operator
- 4 - Configurator
- 5 - Cluster Administrator
- 6 - BDR Administrator
- 7 - Navigator Administrator
- 8 - User Administrator
- 9 - Auditor

and a negative value is returned for a failure to authenticate.

To configure authentication using an external program:

1. Select **Administration** > **Settings**.
2. In the left-hand column, select the **External Authentication** category.
3. In the **Authentication Backend Order** field, select the order in which Cloudera Manager should attempt its authentication. You can choose to authenticate users using just one of the methods (using Cloudera Manager's own database is the default), or you can set it so that if the user cannot be authenticated by the first method, it will attempt using the second method.
4. For **External Authentication Type**, select **External Program**.
5. Provide a path to the external program in the **External Authentication Program Path** property.

Configuring Authentication Using SAML

Cloudera Manager supports the Security Assertion Markup Language (SAML), an XML-based open standard data format for exchanging authentication and authorization data between parties, in particular, between an identity provider (IDP) and a service provider (SP). The SAML specification defines three roles: the principal (typically a user), the IDP, and the SP. In the use case addressed by SAML, the principal (user agent) requests a service from the service provider. The service provider requests and obtains an identity assertion from the IDP. On the basis of this assertion, the SP can make an access control decision—in other words it can decide whether to perform some service for the connected principal.

The primary SAML use case is called web browser single sign-on (SSO). A user wielding a user agent (usually a web browser) requests a web resource protected by a SAML SP. The SP, wishing to know the identity of the requesting user, issues an authentication request to a SAML IDP through the user agent. In the context of this terminology, Cloudera Manager operates as a SP. This topic discusses the Cloudera Manager part of the configuration process; it assumes that you are familiar with SAML and SAML configuration in a general sense, and that you have a functioning IDP already deployed.

Note:

- Cloudera Manager supports both SP- and IDP-initiated SSO.
- The logout action in Cloudera Manager will send a single-logout request to the IDP.
- SAML authentication has been tested with specific configurations of SiteMinder and Shibboleth. While SAML is a standard, there is a great deal of variability in configuration between different IDP products, so it is possible that other IDP implementations, or other configurations of SiteMinder and Shibboleth, may not interoperate with Cloudera Manager.
- To bypass SSO if SAML configuration is incorrect or not working, you can login using a Cloudera Manager local account using the URL: `http://cm_host:7180/cmf/localLogin`

Setting up Cloudera Manager to use SAML requires the following steps.

Preparing Files

You will need to prepare the following files and information, and provide these to Cloudera Manager:

- A Java keystore containing a private key for Cloudera Manager to use to sign/encrypt SAML messages.
- The SAML metadata XML file from your IDP. This file must contain the public certificates needed to verify the sign/encrypt key used by your IDP per the SAML Metadata Interoperability Profile
- The entity ID that should be used to identify the Cloudera Manager instance
- How the user ID is passed in the SAML authentication response:
  - As an attribute. If so, what identifier is used.
  - As the NameID.
- The method by which the Cloudera Manager role will be established:
  - From an attribute in the authentication response:
    - What identifier will be used for the attribute
    - What values will be passed to indicate each role
  - From an external script that will be called for each use:
    - The script takes user ID as $1
    - The script sets an exit code to reflect successful authentication of the assigned role:
      - 0 - Full Administrator
      - 1 - Read-Only
      - 2 - Limited Operator
      - 3 - Operator
      - 4 - Configurator
      - 5 - Cluster Administrator
      - 6 - BDR Administrator
      - 7 - Navigator Administrator
      - 8 - User Administrator
      - 9 - Auditor

      and a negative value is returned for a failure to authenticate.

Configuring Cloudera Manager

1. Select **Administration** > **Settings**.
2. In the left-hand column, select the **External Authentication** category.
3. Set the **External Authentication Type** property to **SAML** (the Authentication Backend Order property is ignored for SAML).
4. Set the **Path to SAML IDP Metadata File** property to point to the IDP metadata file.
5. Set the **Path to SAML Keystore File** property to point to the Java keystore prepared earlier.
6. In the **SAML Keystore Password** property, set the keystore password.
7. In the **Alias of SAML Sign/Encrypt Private Key** property, set the alias used to identify the private key for Cloudera Manager to use.
8. In the **SAML Sign/Encrypt Private Key Password** property, set the private key password.
9. Set the **SAML Entity ID** property if:
   - There is more than one Cloudera Manager instance being used with the same IDP (each instance needs a different entity ID).
   - Entity IDs are assigned by organizational policy.
10. In the **Source of User ID in SAML Response** property, set whether the user ID will be obtained from an attribute or the NameID.
11. If an attribute will be used, set the attribute name in the **SAML attribute identifier for user ID** property. The default value is the normal OID used for user IDs and so may not need to be changed.
12. In the **SAML Role assignment mechanism** property, set whether the role assignment will be done from an attribute or an external script.
    - If an attribute will be used:
      - In the **SAML attribute identifier for user role** property, set the attribute name if necessary. The default value is the normal OID used for OrganizationalUnits and so may not need to be changed.
      - In the **SAML Attribute Values for Roles** property, set which attribute values will be used to indicate the user role.
    - If an external script will be used, set the path to that script in the **Path to SAML Role Assignment Script** property. Make sure that the script is executable (an executable binary is fine - it doesn't need to be a shell script).
13. Save the changes. Cloudera Manager will run a set of validations that ensure it can find the metadata XML and the keystore, and that the passwords are correct. If you see a validation error, correct the problem before proceeding.
14. Restart the Cloudera Manager Server.

Configuring the IDP

After the Cloudera Manager Server is restarted, it will attempt to redirect to the IDP login page instead of showing the normal CM page. This may or may not succeed, depending on how the IDP is configured. In either case, the IDP will need to be configured to recognize CM before authentication will actually succeed. The details of this process are specific to each IDP implementation - refer to your IDP documentation for details.

1. Download the Cloudera Manager's SAML metadata XML file from `http://hostname:7180/saml/metadata`.
2. Inspect the metadata file and ensure that any URLs contained in the file can be resolved by users' web browsers. The IDP will redirect web browsers to these URLs at various points in the process. If the browser cannot resolve them, authentication will fail. If the URLs are incorrect, you can manually fix the XML file or set the Entity Base URL in the CM configuration to the right value, and then re-download the file.
3. Provide this metadata file to your IDP using whatever mechanism your IDP provides.
4. Ensure that the IDP has access to whatever public certificates are necessary to validate the private key that was provided to Cloudera Manager earlier.
5. Ensure that the IDP is configured to provide the User ID and Role using the attribute names that Cloudera Manager was configured to expect, if relevant.

6. Ensure the changes to the IDP configuration have taken effect (a restart may be necessary).

Verifying Authentication and Authorization

1. Return to the Cloudera Manager Admin Console and refresh the login page.
2. Attempt to log in with credentials for a user that is entitled. The authentication should complete and you should see the Home page.
3. If authentication fails, you will see an IDP provided error message. Cloudera Manager is not involved in this part of the process, and you must ensure the IDP is working correctly to complete the authentication.
4. If authentication succeeds but the user is not authorized to use Cloudera Manager, they will be taken to an error page by Cloudera Manager that explains the situation. If an user who should be authorized sees this error, then you will need to verify their role configuration, and ensure that it is being properly communicated to Cloudera Manager, whether by attribute or external script. The Cloudera Manager log will provide details on failures to establish a user's role. If any errors occur during role mapping, Cloudera Manager will assume the user is unauthorized.

Kerberos Principals and Keytabs

Hadoop security uses Kerberos principals and keytabs to perform user authentication on all remote procedure calls.

Kerberos Principals

A user in Kerberos is called a ***principal***, which is made up of three distinct components: the primary, instance, and realm. A ***Kerberos principal*** is used in a Kerberos-secured system to represent a unique identity. The first component of the principal is called the ***primary***, or sometimes the user component. The primary component is an arbitrary string and may be the operating system username of the user or the name of a service. The primary component is followed by an optional section called the ***instance***, which is used to create principals that are used by users in special roles or to define the host on which a service runs, for example. An instance, if it exists, is separated from the primary by a slash and then the content is used to disambiguate multiple principals for a single user or service. The final component of the principal is the realm. The ***realm*** is similar to a domain in DNS in that it logically defines a related group of objects, although rather than hostnames as in DNS, the Kerberos realm defines a group of principals . Each realm can have its own settings including the location of the KDC on the network and supported encryption algorithms. Large organizations commonly create distinct realms to delegate administration of a realm to a group within the enterprise. Realms, by convention, are written in uppercase characters.

Kerberos assigns tickets to Kerberos principals to enable them to access Kerberos-secured Hadoop services. For the Hadoop daemon principals, the principal names should be of the format `*username*/fully.qualified.domain.name@YOUR-REALM.COM`. In this guide, *username* in the `*username*/fully.qualified.domain.name@YOUR-REALM.COM` principal refers to the username of an existing Unix account that is used by Hadoop daemons, such as `hdfs` or `mapred`. Human users who want to access the Hadoop cluster also need to have Kerberos principals; in this case, `*username*` refers to the username of the user's Unix account, such as `joe` or `jane`. Single-component principal names (such as `joe@YOUR-REALM.COM`) are acceptable for client user accounts. Hadoop does not support more than two-component principal names.

Kerberos Keytabs

A ***keytab*** is a file containing pairs of Kerberos principals and an encrypted copy of that principal's key. A keytab file for a Hadoop daemon is unique to each host since the principal names include the hostname. This file is used to authenticate a principal on a host to Kerberos without human interaction or storing a password in a plain text file. Because having access to the keytab file for a principal allows one to act as that principal, access to the keytab files should be tightly secured. They should be readable by a minimal set of users, should be stored on local disk, and should not be included in host backups, unless access to those backups is as secure as access to the local host.

Enabling Kerberos Authentication Using the Wizard

Minimum Required Role: **Cluster Administrator** (also provided by **Full Administrator**)

> **Important:** Ensure you have secured communication between the Cloudera Manager Server and Agents before you enable Kerberos on your cluster. Kerberos keytabs are sent from the Cloudera Manager Server to the Agents, and must be encrypted to prevent potential misuse of leaked keytabs. For secure communication, you should have *at least* Level 1 TLS enabled as described in Configuring TLS Security for Cloudera Manager (Level 1).

This guide describes how to use Cloudera Manager and the Kerberos wizard (introduced in Cloudera Manager 5.1.0) to automate many of the manual tasks of implementing Kerberos security on your CDH cluster.

- **Prerequisites** - These instructions assume you know how to install and configure Kerberos, you already have a working Kerberos key distribution center (KDC) and realm setup, and that you've installed the following Kerberos client packages on all cluster hosts and hosts that will be used to access the cluster, depending on the OS in use.

OS	Packages Required
RHEL 6 Compatible, **RHEL 5 Compatible**	• `openldap-clients` on the Cloudera Manager Server host • `krb5-workstation`, `krb5-libs` on ALL hosts
SLES	• `openldap2-client` on the Cloudera Manager Server host • `krb5-client` on ALL hosts
Ubuntu or Debian	• `ldap-utils` on the Cloudera Manager Server host • `krb5-user` on ALL hosts
Windows	• `krb5-workstation`, `krb5-libs` on ALL hosts

  Furthermore, Oozie and Hue require that the realm support renewable tickets. Cloudera Manager supports setting up kerberized clusters with MIT KDC and Active Directory.

  > **Important:** If you want to integrate Kerberos directly with Active Directory, ensure you have support from your AD administration team to do so. This includes any future support required to troubleshoot issues such as Kerberos TGT/TGS ticket renewal, access to KDC logs for debugging and so on.

  For more information about using Active Directory, refer the section below on **Considerations when using an Active Directory KDC** and the Microsoft AD documentation.

  For more information about installing and configuring MIT KDC, see:

  - MIT Kerberos Home
  - MIT Kerberos Documentation

- **Support**
  - Kerberos security in Cloudera Manager has been tested on the following version of MIT Kerberos 5:
    - krb5-1.6.1 on Red Hat Enterprise Linux 5 and CentOS 5
  - Kerberos security in Cloudera Manager is supported on the following versions of MIT Kerberos 5:
    - krb5-1.6.3 on SLES 11 Service Pack 1
    - krb5-1.8.1 on Ubuntu
    - krb5-1.8.2 on Red Hat Enterprise Linux 6 and CentOS 6
    - krb5-1.9 on Red Hat Enterprise Linux 6.1

Considerations when using an Active Directory KDC

Performance:

As your cluster grows, so will the volume of Authentication Service (AS) and Ticket Granting Service (TGS) interaction between the services on each cluster server. Consider evaluating the volume of this interaction against the Active Directory domain controllers you have configured for the cluster before rolling this feature out to a production environment. If cluster performance suffers, over time it might become necessary to dedicate a set of AD domain controllers to larger deployments.

Network Proximity:

By default, Kerberos uses UDP for client/server communication. Often, AD services are in a different network than project application services such as Hadoop. If the domain controllers supporting a cluster for Kerberos are not in the same subnet, or they're separated by a firewall, consider using the `udp_preference_limit = 1` setting in the `[libdefaults]` section of the `krb5.conf` used by cluster services. Cloudera strongly recommends *against* using AD domain controller (KDC) servers that are separated from the cluster by a WAN connection, as latency in this service will significantly impact cluster performance.

Process:

Troubleshooting the cluster's operations, especially for Kerberos-enabled services, will need to include AD administration resources. Evaluate your organizational processes for engaging the AD administration team, and how to escalate in case a cluster outage occurs due to issues with Kerberos authentication against AD services. In some situations it might be necessary to enable Kerberos event logging to address desktop and KDC issues within windows environments.

Step 1: Install Cloudera Manager and CDH

If you have not already done so, Cloudera strongly recommends that you install and configure the Cloudera Manager Server and Cloudera Manager Agents and CDH to set up a fully-functional CDH cluster *before* you begin doing the following steps to implement Hadoop security features.

Overview of the User Accounts and Groups in CDH and Cloudera Manager to Support Security

When you install the CDH packages and the Cloudera Manager Agents on your cluster hosts, Cloudera Manager takes some steps to provide system security such as creating the following Unix accounts and setting directory permissions as shown in the following table. These Unix accounts and directory permissions work with the Hadoop Kerberos security requirements.

This User	Runs These Roles
`hdfs`	NameNode, DataNodes, and Secondary Node
`mapred`	JobTracker and TaskTrackers (MR1) and Job History Server (YARN)
`yarn`	ResourceManager and NodeManagers (YARN)
`oozie`	Oozie Server
`hue`	Hue Server, Beeswax Server, Authorization Manager, and Job Designer

The `hdfs` user also acts as the HDFS superuser.

When you install the Cloudera Manager Server on the server host, a new Unix user account called `cloudera-scm` is created automatically to support security. The Cloudera Manager Server uses this account to create host principals and deploy the keytabs on your cluster.

If you installed CDH and Cloudera Manager at the Same Time

If you have a new installation and you installed CDH and Cloudera Manager at the same time, when you started the Cloudera Manager Agents on your cluster hosts, the Cloudera Manager Agent on each host automatically configured the directory owners shown in the following table to support security. Assuming the owners are configured as shown, the Hadoop daemons can then automatically set the permissions for each of the directories specified by the properties shown below to make sure they are properly restricted. It's critical that the owners are configured exactly as shown below, so do not change them:

Directory Specified in this Property	Owner
`dfs.name.dir`	`hdfs:hadoop`
`dfs.data.dir`	`hdfs:hadoop`
`mapred.local.dir`	`mapred:hadoop`
`mapred.system.dir` in HDFS	`mapred:hadoop`
`yarn.nodemanager.local-dirs`	`yarn:yarn`
`yarn.nodemanager.log-dirs`	`yarn:yarn`
`oozie.service.StoreService.jdbc.url` (if using Derby)	`oozie:oozie`
`[[database]] name`	`hue:hue`
`javax.jdo.option.ConnectionURL`	`hue:hue`

If you Installed and Used CDH Before Installing Cloudera Manager

If you have been using HDFS and running MapReduce jobs in an existing installation of CDH before you installed Cloudera Manager, you must manually configure the owners of the directories shown in the table above. Doing so enables the Hadoop daemons to automatically set the permissions for each of the directories. It's critical that you manually configure the owners exactly as shown above.

Step 2: If You are Using AES-256 Encryption, Install the JCE Policy File

If you are using CentOS or Red Hat Enterprise Linux 5.5 or higher, which use AES-256 encryption by default for tickets, you must install the Java Cryptography Extension (JCE) Unlimited Strength Jurisdiction Policy File on all cluster and Hadoop user hosts. There are 2 ways to do this:

- In the Cloudera Manager Admin Console, navigate to the **Hosts** page. Both, the **Add New Hosts to Cluster** wizard and the **Re-run Upgrade Wizard** will give you the option to have Cloudera Manager install the JCE Policy file for you.
- You can follow the JCE Policy File installation instructions in the `README.txt` file included in the `jce_policy-x.zip` file.

Alternatively, you can configure Kerberos to not use AES-256 by removing `aes256-cts:normal` from the `supported_enctypes` field of the `kdc.conf` or `krb5.conf` file. Note that after changing the `kdc.conf` file, you'll need to restart both the KDC and the kadmin server for those changes to take affect. You may also need to recreate or change the password of the relevant principals, including potentially the Ticket Granting Ticket principal (krbtgt/REALM@REALM). If AES-256 is still used after all of those steps, it's because the `aes256-cts:normal` setting existed when the Kerberos database was created. To fix this, create a new Kerberos database and then restart both the KDC and the kadmin server.

To verify the type of encryption used in your cluster:

1. **For MIT KDC:** On the local KDC host, type this command in the kadmin.local or kadmin shell to create a test principal:

```
kadmin:  addprinc test
```

   For Active Directory: Create a new AD account with the name, `test`.

2. On a cluster host, type this command to start a Kerberos session as test:

```
$ kinit test
```

3. On a cluster host, type this command to view the encryption type in use:

```
$ klist -e
```

If AES is being used, output like the following is displayed after you type the `klist` command (note that `AES-256` is included in the output):

```
Ticket cache: FILE:/tmp/krb5cc_0
Default principal: test@Cloudera Manager
Valid starting     Expires            Service principal
05/19/11 13:25:04  05/20/11 13:25:04  krbtgt/Cloudera Manager@Cloudera Manager
    Etype (skey, tkt): AES-256 CTS mode with 96-bit SHA-1 HMAC, AES-256 CTS mode with
96-bit SHA-1 HMAC
```

Step 3: Get or Create a Kerberos Principal for the Cloudera Manager Server

In order to create and deploy the host principals and keytabs on your cluster, the Cloudera Manager Server must have the correct Kerberos principal. Specifically, the Cloudera Manager Server must have a Kerberos principal that has privileges to create other accounts.

To get or create the Kerberos principal for the Cloudera Manager Server, you can do either of the following:

- Ask your Kerberos administrator to create a Kerberos administrator principal for the Cloudera Manager Server.
- Create the Kerberos principal for the Cloudera Manager Server yourself by using the following instructions in this step.

Creating the Cloudera Manager Principal

The following instructions illustrate an example of creating the Cloudera Manager Server principal for MIT KDC and Active Directory KDC. (If you are using another version of Kerberos, refer to your Kerberos documentation for instructions.)

If you are using Active Directory:

1. Create an Organizational Unit (OU) in your AD setup where all the principals used by your CDH cluster will reside.
2. Add a new user account to Active Directory, for example, `<username>@YOUR-REALM.COM`. The password for this user should be set to never expire.
3. Use AD's Delegate Control wizard to allow this new user to **Create, Delete and Manage User Accounts**.

If you are using MIT KDC:

Typically, principals with the second component of `admin` in the principal name (for example, `username/admin@YOUR-LOCAL-REALM.com`) have administrator privileges. This is why `admin` is shown in the following example.

> **Note:** If you are running `kadmin` and the Kerberos Key Distribution Center (KDC) on the same host, use `kadmin.local` in the following steps. If the Kerberos KDC is running on a remote host, you must use `kadmin` instead of `kadmin.local`.

In the `kadmin.local` or `kadmin` shell, type the following command to create the Cloudera Manager Server principal, replacing `YOUR-LOCAL-REALM.COM` with the name of your realm:

```
kadmin:  addprinc -pw <Password> cloudera-scm/admin@YOUR-LOCAL-REALM.COM
```

Step 4: Enabling Kerberos Using the Wizard

Minimum Required Role: Full Administrator

To start the Kerberos wizard:

1. Navigate to the Cloudera Manager Admin Console and click to the right of the cluster for which you want to enable Kerberos authentication.
2. Select **Enable Kerberos**.

Before you Begin Using the Wizard

The Welcome page lists the following action items that you should complete before you begin to secure the cluster using this wizard:

- Set up a working KDC. Cloudera Manager supports authentication with MIT KDC and Active Directory.
- Configure the KDC to allow renewable tickets with non-zero ticket lifetimes.

  Active Directory allows renewable tickets with non-zero lifetimes by default. You can verify this by checking **Domain Security Settings** > **Account Policies** > **Kerberos Policy** in Active Directory.

  For MIT KDC, make sure you have the following lines in the `kdc.conf`.

```
max_life = 1d
max_renewable_life = 7d
```

- If you are using Active Directory, make sure LDAP over SSL (LDAPS) is enabled for the Domain Controllers.
- Install the following packages on your cluster depending on the OS in use.

OS	Packages Required
RHEL 6 Compatible, RHEL 5 Compatible	• `openldap-clients` on the Cloudera Manager Server host • `krb5-workstation`, `krb5-libs` on ALL hosts
SLES	• `openldap2-client` on the Cloudera Manager Server host • `krb5-client` on ALL hosts
Ubuntu or Debian	• `ldap-utils` on the Cloudera Manager Server host • `krb5-user` on ALL hosts
Windows	• `krb5-workstation`, `krb5-libs` on ALL hosts

- Create an account for Cloudera Manager that has the permissions to create other accounts in the KDC. This should have been completed as part of Step 3: Get or Create a Kerberos Principal for the Cloudera Manager Server on page 21.

> **Important:**
>
> If you have enabled YARN Resource Manager HA in your non-secure cluster, you should clear the StateStore znode in ZooKeeper before enabling Kerberos. To do this:
>
> 1. Go to the Cloudera Manager Admin Console home page, click to the right of the YARN service and select **Stop**.
> 2. When you see a **Finished** status, the service has stopped.
> 3. Go to the YARN service and select **Actions** > **Format State Store**.
> 4. When the command completes, click **Close**.

Once you are able to check all the items on this list, click **Continue**.

KDC Information

On this page, select the KDC type you are using, MIT KDC or Active Directory, and complete the fields as applicable to enable Cloudera Manager to generate principals/accounts for the CDH services running on the cluster.

Note:

- If you are using AD and have multiple Domain Controllers behind a Load Balancer, enter the name of the Load Balancer in the **KDC Server Host** field and any *one* of the Domain Controllers in **Active Directory Domain Controller Override**. Hadoop daemons will use the Load Balancer for authentication, but Cloudera Manager will use the override for creating accounts.
- If you have multiple Domain Controllers (in case of AD) or MIT KDC servers, only enter the name of any *one* of them in the **KDC Server Host** field. Cloudera Manager will use that server only for creating accounts. If you choose to use Cloudera Manager to manage `krb5.conf`, you can specify the rest of the Domain Controllers using Safety Valve as explained below.
- Make sure the entries for the **Kerberos Encryption Types** field matches what your KDC supports.

Click **Continue** to proceed.

KRB5 Configuration

Manage krb5.conf through Cloudera Manager allows you to choose whether Cloudera Manager should deploy the `krb5.conf` on your cluster or not. If left unchecked, you must ensure that the `krb5.conf` is deployed on all hosts in the cluster, including the Cloudera Manager Server's host.

If you check **Manage krb5.conf through Cloudera Manager**, this page will let you configure the properties that will be emitted in it. In particular, the safety valves on this page can be used to configure cross-realm authentication. More information can be found at Configuring a Cluster-dedicated MIT KDC with Cross-Realm Trust on page 115.

Note: Cloudera Manager is unable to use a non-default realm. You must specify the default realm.

Click **Continue** to proceed.

Import KDC Account Manager Credentials

Enter the username and password for the user that can create principals for CDH cluster in the KDC. This is the user/principal you created in Step 3: Get or Create a Kerberos Principal for the Cloudera Manager Server on page 21. Cloudera Manager encrypts the username and password into a keytab and uses it as needed to create new principals.

Note: The username entered should have the realm portion in upper-case only as shown in the example in the UI.

Click **Continue** to proceed.

Configure HDFS DataNode Ports

On this page, specify the privileged ports needed by the DataNode's Transceiver Protocol and the HTTP Web UI in a secure cluster.

Use the checkbox to confirm you are ready to restart the cluster. Click **Continue**.

Enabling Kerberos

This page lets you track the progress made by the wizard as it first stops all services on your cluster, deploys the `krb5.conf`, generates keytabs for other CDH services, deploys client configuration and finally restarts all services. Click **Continue**.

Congratulations

The final page lists the cluster(s) for which Kerberos has been successfully enabled. Click **Finish** to return to the Cloudera Manager Admin Console home page.

Step 5: Create the HDFS Superuser

To be able to create home directories for users, you will need access to the HDFS superuser account. (CDH automatically created the HDFS superuser account on each cluster host during CDH installation.) When you enabled Kerberos for the HDFS service, you lost access to the HDFS superuser account using `sudo -u hdfs` commands. To enable your access to the HDFS superuser account now that Kerberos is enabled, you must create a Kerberos principal or an AD user whose first component is `hdfs`:

If you are using Active Directory

Add a new user account to Active Directory, `hdfs@YOUR-REALM.COM`.

If you are using MIT KDC

1. In the `kadmin.local` or `kadmin` shell, type the following command to create a Kerberos principal called `hdfs`:

```
kadmin:  addprinc hdfs@YOUR-LOCAL-REALM.COM
```

> **Note:** This command prompts you to create a password for the `hdfs` principal. You should use a strong password because having access to this principal provides superuser access to all of the files in HDFS.

2. To run commands as the HDFS superuser, you must obtain Kerberos credentials for the `hdfs` principal. To do so, run the following command and provide the appropriate password when prompted.

```
$ kinit hdfs@YOUR-LOCAL-REALM.COM
```

Step 6: Get or Create a Kerberos Principal for Each User Account

Now that Kerberos is configured and enabled on your cluster, you and every other Hadoop user must have a Kerberos principal or keytab to obtain Kerberos credentials to be allowed to access the cluster and use the Hadoop services. In the next step of this procedure, you will need to create your own Kerberos principals in order to verify that Kerberos security is working on your cluster. If you and the other Hadoop users already have a Kerberos principal or keytab, or if your Kerberos administrator can provide them, you can skip ahead to the next step.

The following instructions explain how to create a Kerberos principal for a user account.

If you are using Active Directory

Add a new AD user account, `<username>@YOUR-REALM.COM` for each Cloudera Manager service that should use Kerberos authentication.

If you are using MIT KDC

1. In the kadmin.local or kadmin shell, use the following command to create a principal for your account by replacing `YOUR-LOCAL-REALM.COM` with the name of your realm, and replacing `USERNAME` with a username:

```
kadmin:  addprinc USERNAME@YOUR-LOCAL-REALM.COM
```

2. When prompted, enter the password twice.

Step 7: Prepare the Cluster for Each User

Before you and other users can access the cluster, there are a few tasks you must do to prepare the hosts for each user.

1. Make sure all hosts in the cluster have a Linux user account with the same name as the first component of that user's principal name. For example, the Linux account `joe` should exist on every box if the user's principal name is `joe@YOUR-REALM.COM`. You can use LDAP for this step if it is available in your organization.

> **Note:** Each account must have a user ID that is greater than or equal to 1000. In the `/etc/hadoop/conf/taskcontroller.cfg` file, the default setting for the `banned.users` property is `mapred`, `hdfs`, and `bin` to prevent jobs from being submitted using those user accounts. The default setting for the `min.user.id` property is `1000` to prevent jobs from being submitted with a user ID less than 1000, which are conventionally Unix super users.

2. Create a subdirectory under `/user` on HDFS for each user account (for example, `/user/joe`). Change the owner and group of that directory to be the user.

```
$ hadoop fs -mkdir /user/joe
$ hadoop fs -chown joe /user/joe
```

> **Note:** `sudo -u hdfs` is not included in the commands above. This is because it is not required if Kerberos is enabled on your cluster. You will, however, need to have Kerberos credentials for the HDFS super user in order to successfully run these commands. For information on gaining access to the HDFS super user account, see Step 14: Create the HDFS Superuser Principal on page 38

Step 8: Verify that Kerberos Security is Working

After you have Kerberos credentials, you can verify that Kerberos security is working on your cluster by trying to run MapReduce jobs. To confirm, try launching a sleep or a pi job from the provided Hadoop examples (`/usr/lib/hadoop/hadoop-examples.jar`).

> **Note:**
>
> This section assumes you have a fully-functional CDH cluster and you have been able to access HDFS and run MapReduce jobs before you followed these instructions to configure and enable Kerberos on your cluster. If you have not already done so, you should at a minimum use the Cloudera Manager Admin Console to generate a client configuration file to enable you to access the cluster. For instructions, see Deploying Client Configuration Files.

To verify that Kerberos security is working:

1. Acquire Kerberos credentials for your user account.

```
$ kinit USERNAME@YOUR-LOCAL-REALM.COM
```

2. Enter a password when prompted.
3. Submit a sample pi calculation as a test MapReduce job. Use the following command if you use a package-based setup for Cloudera Manager:

```
$ hadoop jar /usr/lib/hadoop-0.20-mapreduce/hadoop-examples.jar pi 10 10000
Number of Maps = 10
Samples per Map = 10000
...
Job Finished in 38.572 seconds
Estimated value of Pi is 3.14120000000000000000
```

If you have a parcel-based setup, use the following command instead:

```
$ hadoop jar /opt/cloudera/parcels/CDH/lib/hadoop-0.20-mapreduce/hadoop-examples.jar pi
 10 10000
Number of Maps = 10
Samples per Map = 10000
...
Job Finished in 30.958 seconds
Estimated value of Pi is 3.14120000000000000000
```

You have now verified that Kerberos security is working on your cluster.

Important:

Running a MapReduce job will fail if you do not have a valid Kerberos ticket in your credentials cache. You can examine the Kerberos tickets currently in your credentials cache by running the `klist` command. You can obtain a ticket by running the `kinit` command and either specifying a keytab file containing credentials, or entering the password for your principal. If you do not have a valid ticket, you will receive an error such as:

```
11/01/04 12:08:12 WARN ipc.Client:
Exception encountered while connecting to the server :
javax.security.sasl.SaslException:GSS initiate failed
[Caused by GSSException: No valid credentials provided (Mechanism level:
 Failed to find any
Kerberos tgt)]
Bad connection to FS. command aborted. exception: Call to
nn-host/10.0.0.2:8020 failed on local exception:
java.io.IOException:javax.security.sasl.SaslException: GSS initiate
failed
[Caused by GSSException: No valid credentials provided
(Mechanism level: Failed to find any Kerberos tgt)]
```

Step 9: (Optional) Enable Authentication for HTTP Web Consoles for Hadoop Roles

Minimum Required Role: Configurator (also provided by **Cluster Administrator, Full Administrator**)

Authentication for access to the HDFS, MapReduce, and YARN roles' web consoles can be enabled using a configuration option for the appropriate service. To enable this authentication:

1. From the **Clusters** tab, select the service (HDFS, MapReduce, or YARN) for which you want to enable authentication.
2. Click the **Configuration** tab.
3. Expand **Service-Wide** > **Security**, check the **Enable Authentication for HTTP Web-Consoles** property, and save your changes.

   A command is triggered to generate the new required credentials.

4. Once the command finishes, restart all roles of that service.

Enabling SPNEGO as an Authentication Backend for Hue

To enable SPNEGO authentication:

1. From the **Clusters** tab, select the Hue service.
2. Click the **Configuration** tab.
3. Expand **Service-Wide** > **Security**, and for the **Authentication Backend** property, select `desktop.auth.backend.SpnegoDjangoBackend`.
4. Click **Save Changes**.
5. Restart the Hue service.

Enabling Kerberos Authentication for Single User Mode *or* Non-Default Users

The steps described in this topic are only applicable in the following cases:

- You are running the Cloudera Manager in the single user mode. In this case, configure all the services described in the table below.

  OR

- You are running one or more CDH services with non-default users. This means if you have modified the default value for the **System User** property for any service in Cloudera Manager, you must only perform the command (as described below) corresponding to that service, to be able to successfully run jobs with the non-default user.

MapReduce	Configure the `mapred.system.dir` directory to be owned by the `mapred` user. `sudo -u hdfs hadoop fs -chown mapred:hadoop ${mapred.system.dir}` By default, `mapred.system.dir` is `/tmp/mapred/system`.
HBase	Give the `hbase` user ownership of the HBase root directory: `sudo -u hdfs hadoop fs -chown -R hbase ${hbase.rootdir}` By default, `hbase.rootdir` is `/hbase`.
Hive	Give the `hive` user ownership of the `/user/hive` directory. `sudo -u hdfs hadoop fs -chown hive /user/hive`
YARN	For every NodeManager host, for each path in `yarn.nodemanager.local-dirs`, run: `rm -rf ${yarn.nodemanager.local-dirs}/usercache/*` This removes the `/usercache` directory that contains intermediate data stored for previous jobs.

Viewing and Regenerating Kerberos Principals

Minimum Required Role: Full Administrator

As soon as you enable Hadoop secure authentication for HDFS and MapReduce service instances, Cloudera Manager starts creating the Kerberos principals for each of the role instances. The amount of time this process will take depends on the number of hosts and HDFS and MapReduce role instances on your cluster. The process can take from a few seconds for a small cluster to several minutes for a larger cluster. After the process is completed, you can use the Cloudera Manager Admin Console to view the list of Kerberos principals that Cloudera Manager has created for the cluster. Make sure there are principals for each of the hosts and HDFS and MapReduce role instances on your cluster. If there are no principals after 10 minutes, then there is most likely a problem with the principal creation. See the Troubleshooting Authentication Issues on page 138 section below for more information. If necessary, you can use Cloudera Manager to regenerate the principals.

If you make a global configuration change in your cluster, such as changing the encryption type, you must use the following instructions to regenerate the principals for your cluster.

Important:

- Regenerate principals using the following steps in the Cloudera Manager Admin Console and not directly using `kadmin` shell.
- Do not regenerate the principals for your cluster unless you have made a global configuration change. Before regenerating, be sure to read Configuring a Cluster-dedicated MIT KDC with Cross-Realm Trust on page 115 to avoid making your existing host keytabs invalid.
- If you are using Active Directory, delete the AD accounts with the userPrincipalName (or login names) that you want to manually regenerate before continuing with the steps below.

To view and regenerate the Kerberos principals for your cluster:

1. Select **Administration** > **Kerberos**.
2. The currently configured Kerberos principals are displayed. If you are running HDFS, the `hdfs/hostname` and `host/hostname` principals are listed. If you are running MapReduce, the `mapred/hostname` and `host/hostname` principals are listed. The principals for other running services are also listed.
3. Only if necessary, select the principals you want to regenerate.

4. Click **Regenerate**.

The Security Inspector

The Security Inspector uses the Host Inspector to run a security-related set of commands on the hosts in your cluster. It reports on things such as how Java is configured for encryption and on the default realms configured on each host:

1. Select **Administration** > **Kerberos**.
2. Click **Security Inspector**. Cloudera Manager begins several tasks to inspect the managed hosts.
3. After the inspection completes, click **Download Result Data** or **Show Inspector Results** to review the results.

Mapping Kerberos Principals to Short Names

Kerberos user principals typically have the format `username@REALM`, whereas Hadoop usernames are typically just `username`. To translate Kerberos principals to Hadoop usernames, Hadoop uses rules defined in the `hadoop.security.auth_to_local` property. The default setting strips the `@REALM` portion from the Kerberos principal, where `REALM` is the Kerberos realm defined by the `default_realm` setting in the NameNode `krb5.conf` file.

If you configure your cluster's Kerberos realm to trust other realms, such as a trust between your cluster's realm and a central Active Directory or MIT Kerberos realm, you must identify the trusted realms in Cloudera Manager so it can automatically generate the appropriate rules. If you do not do so, user accounts in those realms cannot access the cluster.

To specify trusted realms using Cloudera Manager:

1. Navigate to the **HDFS Service** > **Configuration** tab.
2. In the Search field, type **Kerberos Realms** to find the **Trusted Kerberos Realms** and **Additional Rules to Map Kerberos Principals to Short Names** settings (in the **Service-Wide** > **Security** category).
3. Add realms that are trusted by the cluster's Kerberos realm. Realm names, including Active Directory realms, must be specified in uppercase letters (for example, `CORP.EXAMPLE.COM`). To add multiple realms, use the ➕ button.
4. Click **Save Changes**.

The auto-generated mapping rules strip the Kerberos realm (for example, `@CORP.EXAMPLE.COM`) for each realm specified in the **Trusted Kerberos Realms** setting. To customize the mapping rules, specify additional rules in the **Additional Rules to Map Kerberos Principals to Short Names** setting, one rule per line. Only enter rules in this field; Cloudera Manager automatically surrounds the rules with the appropriate XML tags for the generated `core-site.xml` file. For more information on creating custom rules, including how to translate mixed-case Kerberos principals to lowercase Hadoop usernames, see Mapping Rule Syntax on page 113.

If you specify custom mapping rules for a Kerberos realm using the **Additional Rules to Map Kerberos Principals to Short Names** setting, ensure that the same realm is not specified in the **Trusted Kerberos Realms** setting. If it is, the auto-generated rule (which only strips the realm from the principal and does no additional transformations) takes precedent, and the custom rule is ignored.

For these changes to take effect, you must restart the cluster and re-deploy the client configuration. On the Cloudera Manager **Home** page, click the cluster-wide button and select **Deploy Client Configuration**.

Using Auth-to-Local Rules to Isolate Cluster Users

By default, the Hadoop auth-to-local rules map a principal of the form `<username>/<hostname>@<REALM>` to `<username>`. This means if there are multiple clusters in the same realm, then principals associated with hosts of one cluster would map to the same user in all other clusters.

For example, if you have two clusters, `cluster1-host-[1..4].example.com` and `cluster2-host-[1..4].example.com`, that are part of the same Kerberos realm, `EXAMPLE.COM`, then the `cluster2` principal, `hdfs/cluster2-host1.example.com@EXAMPLE.COM`, will map to the `hdfs` user even on `cluster1` hosts.

To prevent this, use auth-to-local rules as follows to ensure only principals containing hostnames of `cluster1` are mapped to legitimate users.

1. Navigate to the **HDFS Service** > **Configuration** tab.

2. In the Search field, type **Additional Rules** to find the **Additional Rules to Map Kerberos Principals to Short Names** settings (in the **Service-Wide** > **Security** category).
3. Additional mapping rules can be added to the **Additional Rules to Map Kerberos Principals to Short Names** property. These rules will be inserted before the rules generated from the list of trusted realms (configured above) and before the default rule.

```
RULE:[2:$1/$2@$0](hdfs/cluster1-host1.example.com@EXAMPLE.COM)s/(.*)@EXAMPLE.COM/hdfs/
RULE:[2:$1/$2@$0](hdfs/cluster1-host2.example.com@EXAMPLE.COM)s/(.*)@EXAMPLE.COM/hdfs/
RULE:[2:$1/$2@$0](hdfs/cluster1-host3.example.com@EXAMPLE.COM)s/(.*)@EXAMPLE.COM/hdfs/
RULE:[2:$1/$2@$0](hdfs/cluster1-host4.example.com@EXAMPLE.COM)s/(.*)@EXAMPLE.COM/hdfs/
RULE:[2:$1/$2@$0](hdfs.*@EXAMPLE.COM)s/(.*)@EXAMPLE.COM/nobody/
```

In the example, the principal `hdfs/<hostname>@REALM` is mapped to the `hdfs` user if `<hostname>` is one of the cluster hosts. Otherwise it gets mapped to `nobody`, thus ensuring that principals from other clusters do not have access to `cluster1`.

If the cluster hosts can be represented with a regular expression, that expression can be used to make the configuration easier and more conducive to scaling. For example:

```
RULE:[2:$1/$2@$0](hdfs/cluster1-host[1-4].example.com@EXAMPLE.COM)s/(.*)@EXAMPLE.COM/hdfs/
RULE:[2:$1/$2@$0](hdfs.*@EXAMPLE.COM)s/(.*)@EXAMPLE.COM/nobody/
```

4. Click **Save Changes**.
5. Restart the HDFS service and any dependent services.

Configuring YARN for Long-running Applications

Long-running applications such as Spark Streaming jobs will need additional configuration since the default settings only allow the `hdfs` user's delegation tokens a maximum lifetime of 7 days which is not always sufficient.

You can work around this by configuring the ResourceManager as a proxy user for the corresponding HDFS NameNode so that the ResourceManager can request new tokens when the existing ones are past their maximum lifetime. YARN will then be able to continue performing localization and log-aggregation on behalf of the `hdfs` user.

Configure the proxy user in Cloudera Manager as follows:

1. Go to the Cloudera Manager Admin Console.
2. Using the **Clusters** tab, navigate to the **YARN** service.
3. Click **Configuration**.
4. Under the **ResourceManager Default Group** > **Advanced** category, add the following string to the **ResourceManager Advanced Configuration Snippet (Safety Valve) for yarn-site.xml** property.

```
<property>
<name>yarn.resourcemanager.proxy-user-privileges.enabled</name>
<value>true</value>
</property>
```

5. Click **Save Changes**.
6. Using the **Clusters** tab, navigate to the **HDFS** service.
7. Click **Configuration**.
8. Under the **Service-Wide** > **Advanced** category, add the following string to the **Cluster-wide Advanced Configuration Snippet (Safety Valve) for core-site.xml** property.

```
<property>
<name>hadoop.proxyuser.yarn.hosts</name>
<value>*</value>
</property>

<property>
<name>hadoop.proxyuser.yarn.groups</name>
<value>*</value>
</property>
```

9. Click **Save Changes**.
10. Restart the YARN and HDFS services.

Enabling Kerberos Authentication Without the Wizard

Minimum Required Role: Configurator (also provided by **Cluster Administrator, Full Administrator**)

Note that certain steps in the following procedure to configure Kerberos security may not be completed without `Full Administrator` role privileges.

> **Important:** Ensure you have secured communication between the Cloudera Manager Server and Agents before you enable Kerberos on your cluster. Kerberos keytabs are sent from the Cloudera Manager Server to the Agents, and must be encrypted to prevent potential misuse of leaked keytabs. For secure communication, you should have *at least* Level 1 TLS enabled as described in Configuring TLS Security for Cloudera Manager (Level 1).

- **Prerequisites** - These instructions assume you know how to install and configure Kerberos, you already have a working Kerberos key distribution center (KDC) and realm setup, and that you've installed the following Kerberos client packages on all cluster hosts and hosts that will be used to access the cluster, depending on the OS in use.

OS	Packages Required
RHEL 6 Compatible, **RHEL 5 Compatible**	• `openldap-clients` on the Cloudera Manager Server host • `krb5-workstation`, `krb5-libs` on ALL hosts
SLES	• `openldap2-client` on the Cloudera Manager Server host • `krb5-client` on ALL hosts
Ubuntu or Debian	• `ldap-utils` on the Cloudera Manager Server host • `krb5-user` on ALL hosts
Windows	• `krb5-workstation`, `krb5-libs` on ALL hosts

  Furthermore, Oozie and Hue require that the realm support renewable tickets. Cloudera Manager supports setting up kerberized clusters with MIT KDC and Active Directory.

  > **Important:** If you want to integrate Kerberos directly with Active Directory, ensure you have support from your AD administration team to do so. This includes any future support required to troubleshoot issues such as Kerberos TGT/TGS ticket renewal, access to KDC logs for debugging and so on.

  For more information about using Active Directory, refer the section below on **Considerations when using an Active Directory KDC** and the Microsoft AD documentation.

  For more information about installing and configuring MIT KDC, see:

  - MIT Kerberos Home
  - MIT Kerberos Documentation

- **Support**
  - Kerberos security in Cloudera Manager has been tested on the following version of MIT Kerberos 5:
    - krb5-1.6.1 on Red Hat Enterprise Linux 5 and CentOS 5
  - Kerberos security in Cloudera Manager is supported on the following versions of MIT Kerberos 5:
    - krb5-1.6.3 on SLES 11 Service Pack 1
    - krb5-1.8.1 on Ubuntu

- krb5-1.8.2 on Red Hat Enterprise Linux 6 and CentOS 6
- krb5-1.9 on Red Hat Enterprise Linux 6.1

Here are the general steps to using Cloudera Manager to configure Hadoop security on your cluster, each of which is described in more detail in the following sections:

Step 1: Install Cloudera Manager and CDH

If you have not already done so, Cloudera strongly recommends that you install and configure the Cloudera Manager Server and Cloudera Manager Agents and CDH to set up a fully-functional CDH cluster *before* you begin doing the following steps to implement Hadoop security features.

Overview of the User Accounts and Groups in CDH and Cloudera Manager to Support Security

When you install the CDH packages and the Cloudera Manager Agents on your cluster hosts, Cloudera Manager takes some steps to provide system security such as creating the following Unix accounts and setting directory permissions as shown in the following table. These Unix accounts and directory permissions work with the Hadoop Kerberos security requirements.

> **Note:** Cloudera Manager 5.3 introduces a new ***single user mode***. In single user mode, the Cloudera Manager Agent and *all the processes run by services managed by Cloudera Manager* are started as a single configured user and group. See Single User Mode Requirements for more information.

This User	Runs These Roles
`hdfs`	NameNode, DataNodes, and Secondary Node
`mapred`	JobTracker and TaskTrackers (MR1) and Job History Server (YARN)
`yarn`	ResourceManager and NodeManagers (YARN)
`oozie`	Oozie Server
`hue`	Hue Server, Beeswax Server, Authorization Manager, and Job Designer

The `hdfs` user also acts as the HDFS superuser.

When you install the Cloudera Manager Server on the server host, a new Unix user account called `cloudera-scm` is created automatically to support security. The Cloudera Manager Server uses this account to create and deploy the host principals and keytabs on your cluster.

If you installed CDH and Cloudera Manager at the Same Time

If you have a new installation and you installed CDH and Cloudera Manager at the same time, when you started the Cloudera Manager Agents on your cluster hosts, the Cloudera Manager Agent on each host automatically configured the directory owners shown in the following table to support security. Assuming the owners are configured as shown, the Hadoop daemons can then automatically set the permissions for each of the directories specified by the properties shown below to make sure they are properly restricted. It's critical that the owners are configured exactly as shown below, so don't change them:

Directory Specified in this Property	Owner
`dfs.name.dir`	`hdfs:hadoop`
`dfs.data.dir`	`hdfs:hadoop`
`mapred.local.dir`	`mapred:hadoop`
`mapred.system.dir` in HDFS	`mapred:hadoop`
`yarn.nodemanager.local-dirs`	`yarn:yarn`

Directory Specified in this Property	Owner
`yarn.nodemanager.log-dirs`	`yarn:yarn`
`oozie.service.StoreService.jdbc.url` (if using Derby)	`oozie:oozie`
`[[database]] name`	`hue:hue`
`javax.jdo.option.ConnectionURL`	`hue:hue`

If you Installed and Used CDH Before Installing Cloudera Manager

If you have been using HDFS and running MapReduce jobs in an existing installation of CDH before you installed Cloudera Manager, you must manually configure the owners of the directories shown in the table above. Doing so enables the Hadoop daemons to automatically set the permissions for each of the directories. It's critical that you manually configure the owners exactly as shown above.

Step 2: If You are Using AES-256 Encryption, Install the JCE Policy File

If you are using CentOS or RHEL 5.5 or higher, which use AES-256 encryption by default for tickets, you must install the Java Cryptography Extension (JCE) Unlimited Strength Jurisdiction Policy File on all cluster and Hadoop user hosts. There are 2 ways to do this:

- In the Cloudera Manager Admin Console, navigate to the **Hosts** page. Both, the **Add New Hosts to Cluster** wizard and the **Re-run Upgrade Wizard** will give you the option to have Cloudera Manager install the JCE Policy file for you.
- You can follow the JCE Policy File installation instructions in the `README.txt` file included in the `jce_policy-x.zip` file.

Alternatively, you can configure Kerberos to not use AES-256 by removing `aes256-cts:normal` from the `supported_enctypes` field of the `kdc.conf` or `krb5.conf` file. Note that after changing the `kdc.conf` file, you'll need to restart both the KDC and the kadmin server for those changes to take affect. You may also need to recreate or change the password of the relevant principals, including potentially the Ticket Granting Ticket principal (for example, `krbtgt/EXAMPLE.COM@EXAMPLE.COM`). If AES-256 is still used after all of those steps, it's because the `aes256-cts:normal` setting existed when the Kerberos database was created. To fix this, create a new Kerberos database and then restart both the KDC and the kadmin server.

To verify the type of encryption used in your cluster:

1. On the local KDC host, type this command in the kadmin.local or kadmin shell to create a test principal:

```
kadmin:  addprinc test
```

2. On a cluster host, type this command to start a Kerberos session as the test principal:

```
$ kinit test
```

3. After successfully running the previous command, type this command to view the encryption type in use:

```
$ klist -e
```

   If AES is being used, output like the following is displayed after you type the `klist` command (note that `AES-256` is included in the output):

```
Ticket cache: FILE:/tmp/krb5cc_0
Default principal: test@EXAMPLE.COM
Valid starting     Expires            Service principal
05/19/11 13:25:04  05/20/11 13:25:04  krbtgt/EXAMPLE.COM@EXAMPLE.COM
    Etype (skey, tkt): AES-256 CTS mode with 96-bit SHA-1 HMAC, AES-256 CTS mode with
96-bit SHA-1 HMAC
```

Step 3: Get or Create a Kerberos Principal for the Cloudera Manager Server

In order to create and deploy the host principals and keytabs on your cluster, the Cloudera Manager Server must have the correct Kerberos principal. Specifically, the Cloudera Manager Server must have a Kerberos principal that has administrator privileges. Typically, principals with the second component of `admin` in the principal name (for example, `username/admin@EXAMPLE.COM`) have administrator privileges. This is why `admin` is shown in the following instructions and examples.

To get or create the Kerberos principal for the Cloudera Manager Server, you can do either of the following:

- Ask your Kerberos administrator to create a Kerberos administrator principal for the Cloudera Manager Server.
- Create the Kerberos principal for the Cloudera Manager Server yourself by using the following instructions in this step.

Creating the Cloudera Manager Principal

If you are using Active Directory

1. Create an Organizational Unit (OU) in your AD where all the principals used by your CDH cluster will reside.
2. Add a new AD user, for example, `<username>@EXAMPLE.COM`. The password for this user should be set to never expire.
3. Use AD's Delegate Control wizard to allow this new user to **Create, Delete and Manage User Accounts**.

If you are using MIT KDC

The instructions in this section illustrate an example of creating the Cloudera Manager Server principal for MIT Kerberos. (If you are using another version of Kerberos, refer to your Kerberos documentation for instructions.)

> **Note:** If you are running `kadmin` and the Kerberos Key Distribution Center (KDC) on the same host, use `kadmin.local` in the following steps. If the Kerberos KDC is running on a remote host, you must use `kadmin` instead of `kadmin.local`.

In the `kadmin.local` or `kadmin` shell, type the following command to create the Cloudera Manager Server principal, replacing `EXAMPLE.COM` with the name of your realm:

```
kadmin:  addprinc -pw <Password> cloudera-scm/admin@EXAMPLE.COM
```

Step 4: Import KDC Account Manager Credentials

1. In the Cloudera Manager Admin Console, select **Administration** > **Kerberos**.
2. Navigate to the **Credentials** tab and click **Import Kerberos Account Manager Credentials**.
3. In the **Import Kerberos Account Manager Credentials** dialog box, enter the username and password for the user that can create principals for CDH cluster in the KDC. This is the user/principal you created in Step 3: Get or Create a Kerberos Principal for the Cloudera Manager Server on page 33. Cloudera Manager encrypts the username and password into a keytab and uses it as needed to create new principals.

> **Note:** The username entered should have the realm portion in upper-case only as shown in the example in the UI.

Click **Close** when complete.

Step 5: Configure the Kerberos Default Realm in the Cloudera Manager Admin Console

Minimum Required Role: Full Administrator

> **Important:** Hadoop is unable to use a non-default realm. The Kerberos default realm is configured in the `libdefaults` property in the `/etc/krb5.conf` file on every host in the cluster:
>
> ```
> [libdefaults]
>  default_realm = EXAMPLE.COM
> ```

1. In the Cloudera Manager Admin Console, select **Administration** > **Settings**.
2. Click the **Security** category, and enter the Kerberos realm for the cluster in the **Kerberos Security Realm** field (for example, EXAMPLE.COM or HADOOP.EXAMPLE.COM) that you configured in the `krb5.conf` file.
3. Click **Save Changes**.

Step 6: Stop All Services

Minimum Required Role: Operator (also provided by **Configurator**, **Cluster Administrator**, **Full Administrator**)

Before you enable security in CDH, you must stop all Hadoop daemons in your cluster and then change some configuration properties. You must stop all daemons in the cluster because after one Hadoop daemon has been restarted with the configuration properties set to enable security. Daemons running without security enabled will be unable to communicate with that daemon. This requirement to stop all daemons makes it impossible to do a rolling upgrade to enable security on a Hadoop cluster.

Stop all running services, and the Cloudera Management service, as follows:

Stopping All Services

1. On the Home page, click

   to the right of the cluster name and select **Stop**.
2. Click **Stop** in the confirmation screen. The **Command Details** window shows the progress of stopping services.

   When **All services successfully stopped** appears, the task is complete and you can close the **Command Details** window.

Stopping the Cloudera Management Service

1. On the Home page, click

   to the right of **Cloudera Management Service** and select **Stop**.
2. Click **Stop** to confirm. The **Command Details** window shows the progress of stopping the roles.
3. When **Command completed with *n*/*n* successful subcommands** appears, the task is complete. Click **Close**.

Step 7: Enable Hadoop Security

Minimum Required Role: Configurator (also provided by **Cluster Administrator, Full Administrator**)

To enable Hadoop security for the cluster, you enable it on an HDFS service. After you do so, the Cloudera Manager Server automatically enables Hadoop security on the MapReduce and YARN services associated with that HDFS service.

1. Navigate to the **HDFS Service > Configuration** tab.
2. In the Search field, type **Hadoop Secure** to show the Hadoop security properties (found under the **Service-Wide** > **Security** category).
3. Click the value for the **Hadoop Secure Authentication** property and select the `kerberos` option to enable Hadoop security on the selected HDFS service.
4. Click the value for the **Hadoop Secure Authorization** property and select the checkbox to enable service-level authorization on the selected HDFS service. You can specify comma-separated lists of users and groups authorized

to use Hadoop services or perform admin operations using the following properties under the **Service-Wide > Security** section:

- **Authorized Users**: Comma-separated list of users authorized to use Hadoop services.
- **Authorized Groups**: Comma-separated list of groups authorized to use Hadoop services.
- **Authorized Admin Users**: Comma-separated list of users authorized to perform admin operations on Hadoop.
- **Authorized Admin Groups**: Comma-separated list of groups authorized to perform admin operations on Hadoop.

> **Important:** For Cloudera Manager's Monitoring services to work, the hue user should always be added as an authorized user.

5. In the Search field, type **DataNode Transceiver** to find the **DataNode Transceiver Port** property.
6. Click the value for the **DataNode Transceiver Port** property and specify a privileged port number (below 1024). Cloudera recommends 1004.

> **Note:** If there is more than one DataNode Role Group, you must specify a privileged port number for each DataNode Transceiver Port property.

7. In the Search field, type **DataNode HTTP** to find the **DataNode HTTP Web UI Port** property and specify a privileged port number (below 1024). Cloudera recommends 1006.

> **Note:** These port numbers for the two DataNode properties must be below 1024 in order to provide part of the security mechanism to make it impossible for a user to run a MapReduce task that impersonates a DataNode. The port numbers for the NameNode and Secondary NameNode can be anything you want, but the default port numbers are good ones to use.

8. In the Search field type **Data Directory Permissions** to find the **DataNode Data Directory Permissions** property.
9. Reset the value for the **DataNode Data Directory Permissions** property to the default value of `700` if not already set to that.
10. Make sure you have changed the **DataNode Transceiver Port**, **DataNode Data Directory Permissions** and **DataNode HTTP Web UI Port** properties for every DataNode role group.
11. Click **Save Changes** to save the configuration settings.

To enable ZooKeeper security:

1. Navigate to the **ZooKeeper Service > Configuration** tab and click **View and Edit**.
2. Click the value for **Enable Kerberos Authentication** property.
3. Click **Save Changes** to save the configuration settings.

To enable HBase security:

1. Navigate to the **HBase Service > Configuration** tab and click **View and Edit**.
2. In the Search field, type **HBase Secure** to show the Hadoop security properties (found under the **Service-Wide** > **Security** category).
3. Click the value for the **HBase Secure Authorization** property and select the checkbox to enable authorization on the selected HBase service.
4. Click the value for the **HBase Secure Authentication** property and select `kerberos` to enable authorization on the selected HBase service.
5. Click **Save Changes** to save the configuration settings.

(CDH 4.3 or later) To enable Solr security:

1. Navigate to the **Solr Service > Configuration** tab and click **View and Edit**.
2. In the Search field, type **Solr Secure** to show the Solr security properties (found under the **Service-Wide** > **Security** category).

3. Click the value for the **Solr Secure Authentication** property and select `kerberos` to enable authorization on the selected Solr service.
4. Click **Save Changes** to save the configuration settings.

> **Note:** If you use the Cloudera Manager Admin Console to generate a client configuration file after you enable Hadoop security on your cluster, the generated configuration file will not contain the Kerberos principal and keytab file that end users need to authenticate. Users must obtain Kerberos principal and keytab file from your Kerberos administrator and then run the `kinit` command themselves.

Step 8: Wait for the Generate Credentials Command to Finish

Minimum Required Role: Configurator (also provided by **Cluster Administrator, Full Administrator**)

After you enable security for any of the services in Cloudera Manager, a command called Generate Credentials will be triggered automatically. You can watch the progress of the command on the top right corner of the screen that shows the running commands. Wait for this command to finish (indicated by a grey box containing "0" in it).

Step 9: Enable Hue to Work with Hadoop Security using Cloudera Manager

Minimum Required Role: Cluster Administrator (also provided by **Full Administrator**)

If you are using a Hue service, you must add a role instance of Kerberos Ticket Renewer to the Hue service to enable Hue to work properly with the secure Hadoop cluster using Cloudera Manager.

The Hue Kerberos Ticket Renewer service will only renew tickets for the Hue service, for the principal `hue/<hostname>@<YOUR-REALM.COM>`. The Hue principal is then used to impersonate other users for applications within Hue such as the Job Browser, File Browser and so on.

Other services, such as HDFS and MapReduce, do not use the Hue Kerberos Ticket Renewer. They obtain tickets at startup and use those tickets to obtain Delegation Tokens for various access privileges. Each service handles its own ticket renewal as needed.

1. Go to the Hue service.
2. Click the **Instances** tab.
3. Click the **Add Role Instances** button.
4. Assign the Kerberos Ticket Renewer role instance to the same host as the Hue server.
5. When the wizard is finished, the status will display **Finished** and the Kerberos Ticket Renewer role instance is configured. The Hue service will now work with the secure Hadoop cluster.

Troubleshooting the Kerberos Ticket Renewer:

If the Hue Kerberos Ticket Renewer does not start, check your KDC configuration and the ticket renewal property, `maxrenewlife`, for the `hue/<hostname>` and `krbtgt` principals to ensure they are renewable. If not, running the following commands on the KDC will enable renewable tickets for these principals.

```
kadmin.local: modprinc -maxrenewlife 90day krbtgt/YOUR_REALM.COM
kadmin.local: modprinc -maxrenewlife 90day +allow_renewable hue/<hostname>@YOUR-REALM.COM
```

Step 10: (Flume Only) Use Substitution Variables for the Kerberos Principal and Keytab

Minimum Required Role: Configurator (also provided by **Cluster Administrator, Full Administrator**)

As described in Flume Security Configuration in the CDH 4 Security Guide, if you are using Flume on a secure cluster you must configure the HDFS sink with the following configuration options in the `flume.conf` file:

- `hdfs.kerberosPrincipal` - fully-qualified principal.
- `hdfs.kerberosKeytab` - location on the local host of the keytab containing the user and host keys for the above principal

Here is an example of an HDFS sink configuration in the `flume.conf` file (the majority of the HDFS sink configuration options have been omitted):

```
agent.sinks.sink-1.type = HDFS
agent.sinks.sink-1.hdfs.kerberosPrincipal = flume/_HOST@YOUR-REALM.COM
agent.sinks.sink-1.hdfs.kerberosKeytab = /etc/flume-ng/conf/flume.keytab
agent.sinks.sink-1.hdfs.proxyUser = weblogs
```

Since Cloudera Manager generates the Flume keytab files for you, and the locations of the keytab files cannot be known beforehand, substitution variables are required for Flume. Cloudera Manager provides two Flume substitution variables called `$KERBEROS_PRINCIPAL` and `$KERBEROS_KEYTAB` to configure the principal name and the keytab file path respectively on each host.

Here is an example of using the substitution variables to configure the options shown in the previous example:

```
agent.sinks.sink-1.type = hdfs
agent.sinks.sink-1.hdfs.kerberosPrincipal = $KERBEROS_PRINCIPAL
agent.sinks.sink-1.hdfs.kerberosKeytab = $KERBEROS_KEYTAB
agent.sinks.sink-1.hdfs.proxyUser = weblogs
```

Use the following instructions to have Cloudera Manager add these variables to the `flume.conf` file on every host that Cloudera Manager manages.

To use the Flume substitution variables for the Kerberos principal and keytab:

1. Go to the **Flume service > Configuration** page in Cloudera Manager.
2. Click **Agent**.
3. In the **Configuration File** property, add the configuration options with the substitution variables. For example:

```
agent.sinks.sink-1.type = hdfs
agent.sinks.sink-1.hdfs.kerberosPrincipal = $KERBEROS_PRINCIPAL
agent.sinks.sink-1.hdfs.kerberosKeytab = $KERBEROS_KEYTAB
agent.sinks.sink-1.hdfs.proxyUser = weblogs
```

4. Click **Save**.

Step 11: (CDH 4.0 and 4.1 only) Configure Hue to Use a Local Hive Metastore

If using Hue and the **Bypass Hive Metastore Server** option is not selected (metastore bypass is disabled by default), then Hue will not be able to communicate with Hive with CDH 4.0 or CDH 4.1. This is not a problem with CDH 4.2 or higher.

If you are using CDH 4.0 or 4.1, you can workaround this issue following the instructions in the Known Issues section of the Cloudera Manager 4 Release Notes.

Step 12: Start All Services

Minimum Required Role: Operator (also provided by **Configurator**, **Cluster Administrator**, **Full Administrator**)

Start all services on your cluster:

Starting All Services

1. On the Home page, click

   to the right of the cluster name and select **Start**.
2. Click **Start** that appears in the next screen to confirm. The **Command Details** window shows the progress of starting services.

   When **All services successfully started** appears, the task is complete and you can close the **Command Details** window.

Starting the Cloudera Management Service

1. On the Home page, click

   to the right of **Cloudera Management Service** and select **Start**.
2. Click **Start** to confirm. The **Command Details** window shows the progress of starting the roles.
3. When **Command completed with *n*/*n* successful subcommands** appears, the task is complete. Click **Close**.

Step 13: Deploy Client Configurations

Minimum Required Role: Configurator (also provided by **Cluster Administrator, Full Administrator**)

1. On the Home page, click

   to the right of the cluster name and select **Deploy Client Configuration**.
2. Click **Deploy Client Configuration**.

Step 14: Create the HDFS Superuser Principal

To create home directories for users you require access to a superuser account. In HDFS, the user account running the NameNode process (`hdfs` by default) is a superuser. CDH automatically creates the `hdfs` superuser account on each cluster host during CDH installation. When you enable Kerberos for the HDFS service, you lose access to the `hdfs` superuser account via `sudo -u hdfs` commands. To enable access to the `hdfs` superuser account when Kerberos is enabled, you must create a Kerberos principal or an AD user whose first or only component is `hdfs`. Alternatively, you can designate a superuser group, whose members are superusers.

To create the `hdfs` superuser principal:

If you are using Active Directory

Add a new user account to Active Directory, `hdfs@EXAMPLE.COM`.

If you are using MIT KDC

In the `kadmin.local` or `kadmin` shell, type the following command to create a Kerberos principal named `hdfs`:

```
kadmin:  addprinc hdfs@EXAMPLE.COM
```

This command prompts you to create a password for the `hdfs` principal. Use a strong password because this principal provides superuser access to all of the files in HDFS.

To run commands as the `hdfs` superuser, you must obtain Kerberos credentials for the `hdfs` principal. To do so, run the following command and provide the password:

```
$ kinit hdfs@EXAMPLE.COM
```

Designating a Superuser Group

To designate a group of superusers instead of using the default `hdfs` account, follow these steps:

1. Navigate to the **HDFS Service** > **Configuration** tab.
2. In the Search field, type **Superuser** to display the **Superuser Group** property.
3. Change the value from the default `supergroup` to the appropriate group name for your environment.
4. Click **Save Changes**.

For this change to take effect, you must restart the cluster.

Step 15: Get or Create a Kerberos Principal for Each User Account

Now that Kerberos is configured and enabled on your cluster, you and every other Hadoop user must have a Kerberos principal or keytab to obtain Kerberos credentials to be allowed to access the cluster and use the Hadoop services. In the next step of this procedure, you will need to create your own Kerberos principals in order to verify that Kerberos security is working on your cluster. If you and the other Hadoop users already have a Kerberos principal or keytab, or if your Kerberos administrator can provide them, you can skip ahead to the next step.

The following instructions explain how to create a Kerberos principal for a user account.

If you are using Active Directory

Add a new AD user account, `<username>@EXAMPLE.COM` for each Cloudera Manager service that should use Kerberos authentication.

If you are using MIT KDC

1. In the kadmin.local or kadmin shell, use the following command to create a principal for your account by replacing `EXAMPLE.COM` with the name of your realm, and replacing `username` with a username:

```
kadmin:  addprinc username@EXAMPLE.COM
```

2. When prompted, enter the password twice.

Step 16: Prepare the Cluster for Each User

Before you and other users can access the cluster, there are a few tasks you must do to prepare the hosts for each user.

1. Make sure all hosts in the cluster have a Unix user account with the same name as the first component of that user's principal name. For example, the Unix account `joe` should exist on every box if the user's principal name is `joe@YOUR-REALM.COM`. You can use LDAP for this step if it is available in your organization.

   Note: Each account must have a user ID that is greater than or equal to 1000. In the `/etc/hadoop/conf/taskcontroller.cfg` file, the default setting for the `banned.users` property is `mapred`, `hdfs`, and `bin` to prevent jobs from being submitted from those user accounts. The default setting for the `min.user.id` property is `1000` to prevent jobs from being submitted with a user ID less than 1000, which are conventionally Unix super users.

2. Create a subdirectory under `/user` on HDFS for each user account (for example, `/user/joe`). Change the owner and group of that directory to be the user.

```
$ hadoop fs -mkdir /user/joe
$ hadoop fs -chown joe /user/joe
```

Note: `sudo -u hdfs` is not included in the commands above. This is because it is not required if Kerberos is enabled on your cluster. You will, however, need to have Kerberos credentials for the HDFS super user in order to successfully run these commands. For information on gaining access to the HDFS super user account, see Step 14: Create the HDFS Superuser Principal on page 38

Step 17: Verify that Kerberos Security is Working

After you have Kerberos credentials, you can verify that Kerberos security is working on your cluster by trying to run MapReduce jobs. To confirm, try launching a sleep or a pi job from the provided Hadoop examples (`/usr/lib/hadoop/hadoop-examples.jar`).

> **Note:**
>
> This section assumes you have a fully-functional CDH cluster and you have been able to access HDFS and run MapReduce jobs before you followed these instructions to configure and enable Kerberos on your cluster. If you have not already done so, you should at a minimum use the Cloudera Manager Admin Console to generate a client configuration file to enable you to access the cluster. For instructions, see Deploying Client Configuration Files.

To verify that Kerberos security is working:

1. Acquire Kerberos credentials for your user account.

```
$ kinit USERNAME@YOUR-LOCAL-REALM.COM
```

2. Enter a password when prompted.
3. Submit a sample pi calculation as a test MapReduce job. Use the following command if you use a package-based setup for Cloudera Manager:

```
$ hadoop jar /usr/lib/hadoop-0.20/hadoop-0.20.2*examples.jar pi 10 10000
Number of Maps = 10
Samples per Map = 10000
...
Job Finished in 38.572 seconds
Estimated value of Pi is 3.14120000000000000000
```

If you have a parcel-based setup, use the following command instead:

```
$ hadoop jar /opt/cloudera/parcels/CDH/lib/hadoop-0.20-mapreduce/hadoop-examples.jar pi
 10 10000
Number of Maps = 10
Samples per Map = 10000
...
Job Finished in 30.958 seconds
Estimated value of Pi is 3.14120000000000000000
```

You have now verified that Kerberos security is working on your cluster.

> **Important:**
>
> Running a MapReduce job will fail if you do not have a valid Kerberos ticket in your credentials cache. You can examine the Kerberos tickets currently in your credentials cache by running the `klist` command. You can obtain a ticket by running the `kinit` command and either specifying a keytab file containing credentials, or entering the password for your principal. If you do not have a valid ticket, you will receive an error such as:
>
> ```
> 11/01/04 12:08:12 WARN ipc.Client:
> Exception encountered while connecting to the server :
> javax.security.sasl.SaslException:GSS initiate failed
> [Caused by GSSException: No valid credentials provided (Mechanism level:
> Failed to find any
> Kerberos tgt)]
> Bad connection to FS. command aborted. exception: Call to
> nn-host/10.0.0.2:8020 failed on local exception:
> java.io.IOException:javax.security.sasl.SaslException: GSS initiate
> failed
> [Caused by GSSException: No valid credentials provided
> (Mechanism level: Failed to find any Kerberos tgt)]
> ```

Step 18: (Optional) Enable Authentication for HTTP Web Consoles for Hadoop Roles

Minimum Required Role: Configurator (also provided by **Cluster Administrator, Full Administrator**)

Authentication for access to the HDFS, MapReduce, and YARN roles' web consoles can be enabled using a configuration option for the appropriate service. To enable this authentication:

1. From the **Clusters** tab, select the service (HDFS, MapReduce, or YARN) for which you want to enable authentication.
2. Click the **Configuration** tab.
3. Expand **Service-Wide** > **Security**, check the **Enable Authentication for HTTP Web-Consoles** property, and save your changes.

   A command is triggered to generate the new required credentials.

4. Once the command finishes, restart all roles of that service.

Configuring Authentication in Cloudera Navigator

Cloudera Navigator supports user authentication against Cloudera Manager user accounts and against an external LDAP or Active Directory service. External authentication enables you to assign Cloudera Navigator user roles to LDAP or Active Directory groups containing the appropriate users for each user role. Authentication with a Cloudera Manager user account requires either the Full Administrator or Navigator Administrator user role, and enables the user to use Cloudera Navigator features or to configure the external authentication service.

Configuring External Authentication for Cloudera Navigator

Minimum Required Role: Navigator Administrator (also provided by **Full Administrator**)

> **Important: This feature is available only with a Cloudera Enterprise license.**
>
> For other licenses, the following applies:
>
> - Cloudera Express- The feature is not available.
> - Cloudera Enterprise Data Hub Edition Trial - The feature is available until you end the trial or the trial license expires.
>
> To obtain a license for Cloudera Enterprise, fill in this form or call 866-843-7207. After you install a Cloudera Enterprise license, the feature will be available.

Cloudera Navigator supports user authentication against Cloudera Manager user accounts and against an external service. The external service can be either LDAP or Active Directory. User authentication against Cloudera Manager user accounts requires users to have one of two Cloudera Manager user roles, either Full Administrator or Navigator Administrator. External authentication enables you to assign Cloudera Navigator user roles to LDAP or Active Directory groups to which the appropriate users belong.

For more information about Cloudera Manager user accounts, see Cloudera Manager User Accounts on page 10. For more information about Cloudera Navigator user roles, see Cloudera Navigator User Roles on page 214.

The following sections describe how to configure the supported external directory services.

Configuring Cloudera Navigator Authentication Using Active Directory

1. Click **Cloudera Management Service** on the **Home** page or the **Clusters** menu.
2. Click **Configuration** > **Navigator Metadata Server Default Group** > **External Authentication**.
3. In the **Authentication Backend Order** field, select the order in which Cloudera Navigator should attempt its authentication. You can choose to authenticate users using just one of the methods (using Cloudera Manager user accounts is the default), or you can set it so that if the user cannot be authenticated by the first method, it will attempt using the second method.
4. For **External Authentication Type**, select **Active Directory**.
5. In the **LDAP URL** property, provide the URL of the Active Directory server.
6. For **Bind Distinguished Name**, enter the distinguished name of the user to bind as. This is used to connect to Active Directory for searching groups and to get other user information.

7. For **LDAP Bind Password**, enter the password for the bind user entered above.
8. In the **Active Directory NT Domain** property, provide the NT domain to authenticate against.
9. After changing the configuration settings, restart the **Navigator Metadata Service**: click the **Instances** tab on the **Cloudera Management Service** page, check **Navigator Metadata Service**, and click **Actions for Selected** > **Restart**.

> **Important:** Cloudera recommends that the **Authentication Backend Order** be set initially to **Cloudera Manager then External**. A Cloudera Manager user with Full Administrator or Navigator Administrator privileges can then log in to Cloudera Navigator and set up groups and users. Otherwise, the external authentication system will be checked first, and if the same user credentials also exist in the specified LDAP or Active Directory, the user will be authenticated there, and will not be authenticated as a Cloudera Manager administrator. Since no user roles will have been set up yet for the users in the external authentication system, the user's attempt to log in will fail. Once the groups and user roles for Cloudera Navigator are set up, the **Authentication Backend Order** can be changed to **External then Cloudera Manager** or **External Only**, if desired.

Configuring Cloudera Navigator Authentication Using an OpenLDAP-compatible Server

For an OpenLDAP-compatible directory, you have several options for searching for users and groups:

- You can specify a single base Distinguished Name (DN) and then provide a "Distinguished Name Pattern" to use to match a specific user in the LDAP directory.
- Search filter options let you search for a particular user based on somewhat broader search criteria – for example Cloudera Navigator users could be members of different groups or organizational units (OUs), so a single pattern won't find all those users. Search filter options also let you find all the groups to which a user belongs, to help determine if that user should be allowed to log in.

1. Click **Cloudera Management Service** on the **Home** page or the **Clusters** menu.
2. Click **Configuration** > **Navigator Metadata Server Default Group** > **External Authentication**.
3. In the **Authentication Backend Order** field, select the order in which Cloudera Navigator should attempt its authentication. You can choose to authenticate users using just one of the methods (using Cloudera Manager user accounts is the default), or you can set it so that if the user cannot be authenticated by the first method, it will attempt using the second method.
4. For **External Authentication Type**, select **LDAP**.
5. In the **LDAP URL** property, provide the URL of the LDAP server and (optionally) the base Distinguished Name (DN) (the search base) as part of the URL — for example `ldap://ldap-server.corp.com/dc=corp,dc=com`.
6. For **Bind Distinguished Name**, enter the distinguished name of the user to bind as. This is used to connect to the LDAP server for searching groups and to get other user information.
7. For **LDAP Bind Password**, enter the password for the bind user entered above.
8. To use a single "Distinguished Name Pattern", provide a pattern in the **LDAP Distinguished Name Pattern** property.

   Use `{0}` in the pattern to indicate where the username should go. For example, to search for a distinguished name where the `uid` attribute is the username, you might provide a pattern similar to `uid={0},ou=People,dc=corp,dc=com`. Cloudera Navigator substitutes the name provided at login into this pattern and performs a search for that specific user. So if a user provides the username "foo" at the Cloudera Navigator login page, Cloudera Navigator will search for the DN `uid=foo,ou=People,dc=corp,dc=com`.

   If you provided a base DN along with the URL, the pattern only needs to specify the rest of the DN pattern. For example, if the URL you provide is `ldap://ldap-server.corp.com/dc=corp,dc=com`, and the pattern is `uid={0},ou=People`, then the search DN will be `uid=foo,ou=People,dc=corp,dc=com`.

9. You can also search using User or Group search filters, using the **LDAP User Search Base**, **LDAP User Search Filter**, **LDAP Group Search Base** and **LDAP Group Search Filter** settings. These allow you to combine a base DN with a search filter to allow a greater range of search targets.

   For example, if you want to authenticate users who may be in one of multiple OUs, the search filter mechanism will allow this. You can specify the User Search Base DN as `dc=corp,dc=com` and the user search filter as `uid={0}`. Then Cloudera Navigator will search for the user anywhere in the tree starting from the Base DN. Suppose you

have two OUs—`ou=Engineering` and `ou=Operations`—Cloudera Navigator will find User "foo" if it exists in either of these OUs, that is, `uid=foo,ou=Engineering,dc=corp,dc=com` or `uid=foo,ou=Operations,dc=corp,dc=com`.

You can use a user search filter along with a DN pattern, so that the search filter provides a fallback if the DN pattern search fails.

The Groups filters let you search to determine if a DN or username is a member of a target group. In this case, the filter you provide can be something like `member={0}` where `{0}` will be replaced with the **DN** of the user you are authenticating. For a filter requiring the username, `{1}` may be used, as `memberUid={1}`. This will return a list of groups to which the user belongs.

10. Click **Save Changes** to commit the changes.
11. After changing the configuration settings, restart the **Navigator Metadata Service**: click the **Instances** tab on the **Cloudera Management Service** page, check **Navigator Metadata Service**, and click **Actions for Selected** > **Restart**.

Configuring Cloudera Navigator to Use LDAPS

If the LDAP server certificate has been signed by a trusted Certificate Authority (that is, VeriSign, GeoTrust, and so on), steps 1 and 2 below may not be necessary.

1. Copy the CA certificate file to the Cloudera Navigator Server host.
2. Import the CA certificate(s) from the CA certificate file to the local truststore. The default truststore is located in the `$JAVA_HOME/jre/lib/security/cacerts` file. This contains the default CA information shipped with the JDK. Create an alternate default file called `jssecacerts` in the same location as the `cacerts` file. You can now safely append CA certificates for any private or public CAs not present in the default `cacerts` file, while keeping the original file intact.

   For our example, we will follow this recommendation by copying the default `cacerts` file into the new `jssecacerts` file, and then importing the CA certificate to this alternate truststore.

```
$ cp $JAVA_HOME/jre/lib/security/cacerts \
  $JAVA_HOME/jre/lib/jssecacerts
```

```
$ /usr/java/latest/bin/keytool -import -alias nt_domain_name \
-keystore /usr/java/latest/jre/lib/security/jssecacerts -file path_to_cert
```

> **Note:**
> - The default password for the cacerts store is **changeit**.
> - The alias can be any name (not just the domain name).

3. Configure the **LDAP URL** property to use `ldaps://ldap_server` instead of `ldap://ldap_server`.

Managing Users and Groups for Cloudera Navigator

Required Role:

User Administrator

Full Administrator

> **Note:** The above are Cloudera Navigator user roles. Users with the Cloudera Manager user roles Navigator Administrator or Full Administrator who log into the Cloudera Navigator Web UI with their Cloudera Manager credentials will be logged in with the Full Administrator Cloudera Navigator user role.

Cloudera Navigator supports user authentication against Cloudera Manager user accounts and against an external LDAP or Active Directory service. External authentication enables you to assign Cloudera Navigator user roles to LDAP or Active Directory groups containing the appropriate users for each user role.

Assigning Cloudera Navigator User Roles to LDAP or Active Directory Groups

This section assumes that values for your LDAP or Active Directory directory service have been configured in Cloudera Manager as described in Configuring External Authentication for Cloudera Navigator. This section also assumes that your LDAP or Active Directory service contains user groups that correspond to Cloudera Navigator user roles having the permissions you want each group of users to have. If not, you should assign your users to such groups now. The Cloudera Navigator user roles are as follows:

- Full Administrator
- User Administrator
- Auditing Viewer
- Lineage Viewer
- Metadata Administrator
- Policy Viewer
- Policy Administrator

Each of these roles and the permissions associated with it are described in Cloudera Navigator User Roles.

To add or remove Cloudera Navigator user roles to LDAP or Active Directory user groups, you should know the names of the directory groups you want to configure, and then perform the following steps:

1. Open the Cloudera Navigator Web UI in one of the following ways:
   - On the **Clusters** menu of Cloudera Manager, click **Cloudera Navigator** in the **Cloudera Management Service** section for the desired cluster.
   - Click the **Instances** tab on the **Cloudera Management Service** page, and click **Navigator Metadata Server**. In the **Summary** section's **Quick Links**, click **Cloudera Navigator**.
2. Log in to Cloudera Navigator with the credentials of a user having one or more of the following user roles:
   - Cloudera Manager Full Administrator
   - Cloudera Manager Navigator Administrator
   - Cloudera Navigator Full Administrator
   - Cloudera Navigator User Administrator
3. Click the **Administration** tab in the upper right.
4. Search for an LDAP or Active Directory group by entering its name (or the first portion of the name) in the search field.
   - Select **All Groups** to search among all groups in the external directory.
   - Select **Groups with Navigator Roles** to display only external directory groups that have already been assigned one or more Cloudera Navigator user roles.
5. From the LDAP or Active Directory groups displayed, select the group to which you want to assign a Cloudera Navigator user role or roles. If roles have already been assigned to the group, they are listed beneath the name of the group in the main panel.
6. Click **Manage Role Assignment** in the upper right.
7. Click the checkbox for each Cloudera Navigator user role you want assigned to that Active Directory or LDAP group. Uncheck any already-assigned roles that you want to remove from the group.
8. Click **Save**.

If a user's role assignments are changed, the changes take effect with the user's next new session, that is, the next time the user logs in to Cloudera Navigator.

Configuring Authentication in CDH Using the Command Line

The security features in CDH 5 enable Hadoop to prevent malicious user impersonation. The Hadoop daemons leverage Kerberos to perform user authentication on all remote procedure calls (RPCs). Group resolution is performed on the Hadoop master nodes, NameNode, JobTracker and ResourceManager to guarantee that group membership cannot be manipulated by users. Map tasks are run under the user account of the user who submitted the job, ensuring isolation there. In addition to these features, new authorization mechanisms have been introduced to HDFS and MapReduce to enable more control over user access to data.

The security features in CDH 5 meet the needs of most Hadoop customers because typically the cluster is accessible only to trusted personnel. In particular, Hadoop's current threat model assumes that users cannot:

1. Have `root` access to cluster machines.
2. Have `root` access to shared client machines.
3. Read or modify packets on the network of the cluster.

> **Note:**
>
> CDH 5 supports encryption of all user data sent over the network. For configuration instructions, see Configuring Encrypted Shuffle, Encrypted Web UIs, and Encrypted HDFS Transport.
>
> Note also that there is no built-in support for on-disk encryption.

Enabling Kerberos Authentication for Hadoop Using the Command Line

> **Important:**
>
> These instructions assume you know how to install and configure Kerberos, you already have a working Kerberos Key Distribution Center (KDC) and realm setup, and that you've installed the Kerberos user packages on all cluster machines and machines which will be used to access the cluster. Furthermore, Oozie and Hue require that the realm support renewable tickets. For more information about installing and configuring Kerberos, see:
>
> - MIT Kerberos Home
> - MIT Kerberos Documentation
> - Kerberos Explained
> - Microsoft Kerberos Overview
> - Microsoft Kerberos in Windows Server 2008
> - Microsoft Kerberos in Windows Server 2003

Here are the general steps to configuring secure Hadoop, each of which is described in more detail in the following sections:

1. Install CDH 5.
2. Verify User Accounts and Groups in CDH 5 Due to Security.
3. If you are Using AES-256 Encryption, install the JCE Policy File.
4. Create and Deploy the Kerberos Principals and Keytab Files.
5. Shut Down the Cluster.
6. Enable Hadoop security.
7. Configure secure HDFS.
8. Optional: Configuring Security for HDFS High Availability.
9. Optional: Configuring secure WebHDFS.
10. Optional: Configuring secure NFS
11. Set Variables for Secure DataNodes.

12. Start up the NameNode.
13. Start up a DataNode.
14. Set the Sticky Bit on HDFS Directories.
15. Start up the Secondary NameNode (if used).
16. Configure Either MRv1 Security or YARN Security.

> **Note:**
>
> Kerberos security in CDH 5 has been tested with the following version of MIT Kerberos 5:
>
> - krb5-1.6.1 on Red Hat Enterprise Linux 5 and CentOS 5
>
> Kerberos security in CDH 5 is supported with the following versions of MIT Kerberos 5:
>
> - krb5-1.6.3 on SUSE Linux Enterprise Server (SLES) 11 Service Pack 1
> - krb5-1.8.1 on Ubuntu
> - krb5-1.8.2 on Red Hat Enterprise Linux 6 and CentOS 6
> - krb5-1.9 on Red Hat Enterprise Linux 6.1

> **Note:**
>
> If you want to enable Kerberos SPNEGO-based authentication for the Hadoop web interfaces, see the Hadoop Auth, Java HTTP SPNEGO Documentation.

Step 1: Install CDH 5

Cloudera strongly recommends that you set up a fully-functional CDH 5 cluster before you begin configuring it to use Hadoop's security features. When a secure Hadoop cluster is not configured correctly, the resulting error messages are in a preliminary state, so it's best to start implementing security after you are sure your Hadoop cluster is working properly without security.

For information about installing and configuring Hadoop and CDH 5 components, and deploying them on a cluster, see Cloudera Installation Guide.

Step 2: Verify User Accounts and Groups in CDH 5 Due to Security

> **Note:** CDH 5 introduces a new version of MapReduce: MapReduce 2.0 (MRv2) built on the YARN framework. In this document, we refer to this new version as YARN. CDH 5 also provides an implementation of the previous version of MapReduce, referred to as MRv1 in this document.

- If you are using MRv1, see Step 2a (MRv1 only): Verify User Accounts and Groups in MRv1 on page 46 for configuration information.
- If you are using YARN, see Step 2b (YARN only): Verify User Accounts and Groups in YARN on page 48 for configuration information.

Step 2a (MRv1 only): Verify User Accounts and Groups in MRv1

> **Note:** If you are using YARN, skip this step and proceed to Step 2b (YARN only): Verify User Accounts and Groups in YARN.

During CDH 5 package installation of MRv1, the following Unix user accounts are automatically created to support security:

This User	Runs These Hadoop Programs
`hdfs`	HDFS: NameNode, DataNodes, Secondary NameNode (or Standby NameNode if you are using HA)
`mapred`	MRv1: JobTracker and TaskTrackers

The `hdfs` user also acts as the HDFS superuser.

The hadoop user no longer exists in CDH 5. If you currently use the hadoop user to run applications as an HDFS super-user, you should instead use the new `hdfs` user, or create a separate Unix account for your application such as `myhadoopapp`.

MRv1: Directory Ownership in the Local File System

Because the HDFS and MapReduce services run as different users, you must be sure to configure the correct directory ownership of the following files on the local file system of each host:

File System	Directory	Owner	Permissions
Local	`dfs.namenode.name.dir` (`dfs.name.dir` is deprecated but will also work)	`hdfs:hdfs`	drwx------
Local	`dfs.datanode.data.dir` (`dfs.data.dir` is deprecated but will also work)	`hdfs:hdfs`	drwx------
Local	`mapred.local.dir`	`mapred:mapred`	drwxr-xr-x

See also Deploying MapReduce v1 (MRv1) on a Cluster.

You must also configure the following permissions for the HDFS and MapReduce log directories (the default locations in `/var/log/hadoop-hdfs` and `/var/log/hadoop-0.20-mapreduce`), and the `$MAPRED_LOG_DIR/userlogs/` directory:

File System	Directory	Owner	Permissions
Local	`HDFS_LOG_DIR`	`hdfs:hdfs`	drwxrwxr-x
Local	`MAPRED_LOG_DIR`	`mapred:mapred`	drwxrwxr-x
Local	`userlogs` directory in `MAPRED_LOG_DIR`	`mapred:anygroup`	*permissions will be set automatically at daemon start time*

MRv1: Directory Ownership on HDFS

The following directories on HDFS must also be configured as follows:

[1] **In CDH 5, package installation and the Hadoop daemons will automatically configure the correct permissions for you if you configure the directory ownership correctly as shown in the table above.**

File System	Directory	Owner	Permissions
HDFS	`mapreduce.jobtracker.system.dir` (`mapred.system.dir` is deprecated but will also work)	`mapred:hadoop`	drwx------
HDFS	/ (root directory)	`hdfs:hadoop`	drwxr-xr-x

MRv1: Changing the Directory Ownership on HDFS

- If Hadoop security is enabled, use `kinit hdfs` to obtain Kerberos credentials for the `hdfs` user by running the following commands before changing the directory ownership on HDFS:

```
$ sudo -u hdfs kinit -k -t hdfs.keytab hdfs/fully.qualified.domain.name@YOUR-REALM.COM
```

If `kinit hdfs` does not work initially, run `kinit -R` after running `kinit` to obtain credentials. (For more information, see Troubleshooting Authentication Issues on page 138). To change the directory ownership on HDFS, run the following commands. Replace the example `/mapred/system` directory in the commands below with the HDFS directory specified by the `mapreduce.jobtracker.system.dir` (or `mapred.system.dir`) property in the `conf/mapred-site.xml` file:

```
$ sudo -u hdfs hadoop fs -chown mapred:hadoop /mapred/system
$ sudo -u hdfs hadoop fs -chown hdfs:hadoop /
$ sudo -u hdfs hadoop fs -chmod -R 700 /mapred/system
$ sudo -u hdfs hadoop fs -chmod 755 /
```

- In addition (whether or not Hadoop security is enabled) create the `/tmp` directory. For instructions on creating `/tmp` and setting its permissions, see these instructions.

Step 2b (YARN only): Verify User Accounts and Groups in YARN

> **Note:** If you are using MRv1, skip this step and proceed to Step 3: If you are Using AES-256 Encryption, Install the JCE Policy File on page 50.

During CDH 5 package installation of MapReduce 2.0 (YARN), the following Unix user accounts are automatically created to support security:

This User	Runs These Hadoop Programs
`hdfs`	HDFS: NameNode, DataNodes, Standby NameNode (if you are using HA)
`yarn`	YARN: ResourceManager, NodeManager
`mapred`	YARN: MapReduce Job History Server

> **Important:** The HDFS and YARN daemons must run as different Unix users; for example, `hdfs` and `yarn`. The MapReduce Job History server must run as user `mapred`. Having all of these users share a common Unix group is recommended; for example, `hadoop`.

YARN: Directory Ownership in the Local File System

Because the HDFS and MapReduce services run as different users, you must be sure to configure the correct directory ownership of the following files on the local file system of each host:

[2] **When starting up, MapReduce sets the permissions for the `mapreduce.jobtracker.system.dir` (or `mapred.system.dir`) directory in HDFS, assuming the user `mapred` owns that directory.**

File System	Directory	Owner	Permissions (see Footnote 1)
Local	`dfs.namenode.name.dir` (`dfs.name.dir` is deprecated but will also work)	`hdfs:hdfs`	drwx------
Local	`dfs.datanode.data.dir` (`dfs.data.dir` is deprecated but will also work)	`hdfs:hdfs`	drwx------
Local	`yarn.nodemanager.local-dirs`	`yarn:yarn`	drwxr-xr-x
Local	`yarn.nodemanager.log-dirs`	`yarn:yarn`	drwxr-xr-x
Local	`container-executor`	`root:yarn`	--Sr-s---
Local	`conf/container-executor.cfg`	`root:yarn`	r--------

Important: Configuration changes to the Linux container executor could result in local NodeManager directories (such as `usercache`) being left with incorrect permissions. To avoid this, when making changes using either Cloudera Manager or the command line, first manually remove the existing NodeManager local directories from all configured local directories (`yarn.nodemanager.local-dirs`), and let the NodeManager recreate the directory structure.

You must also configure the following permissions for the HDFS, YARN and MapReduce log directories (the default locations in `/var/log/hadoop-hdfs`, `/var/log/hadoop-yarn` and `/var/log/hadoop-mapreduce`):

File System	Directory	Owner	Permissions
Local	`HDFS_LOG_DIR`	`hdfs:hdfs`	drwxrwxr-x
Local	`$YARN_LOG_DIR`	`yarn:yarn`	drwxrwxr-x
Local	`MAPRED_LOG_DIR`	`mapred:mapred`	drwxrwxr-x

YARN: Directory Ownership on HDFS

The following directories on HDFS must also be configured as follows:

File System	Directory	Owner	Permissions
HDFS	/ (root directory)	`hdfs:hadoop`	drwxr-xr-x
HDFS	`yarn.nodemanager.remote-app-log-dir`	`yarn:hadoop`	drwxrwxrwxt
HDFS	`mapreduce.jobhistory.intermediate-done-dir`	`mapred:hadoop`	drwxrwxrwxt
HDFS	`mapreduce.jobhistory.done-dir`	`mapred:hadoop`	drwxr-x---

[3] **In CDH 5, package installation and the Hadoop daemons will automatically configure the correct permissions for you if you configure the directory ownership correctly as shown in the two tables above. See also Deploying MapReduce v2 (YARN) on a Cluster.**

YARN: Changing the Directory Ownership on HDFS

If Hadoop security is enabled, use `kinit hdfs` to obtain Kerberos credentials for the `hdfs` user by running the following commands:

```
$ sudo -u hdfs kinit -k -t hdfs.keytab hdfs/fully.qualified.domain.name@YOUR-REALM.COM
$ hadoop fs -chown hdfs:hadoop /
$ hadoop fs -chmod 755 /
```

If `kinit hdfs` does not work initially, run `kinit -R` after running `kinit` to obtain credentials. See Troubleshooting Authentication Issues on page 138. To change the directory ownership on HDFS, run the following commands:

```
$ sudo -u hdfs hadoop fs -chown hdfs:hadoop /
$ sudo -u hdfs hadoop fs -chmod 755 /
$ sudo -u hdfs hadoop fs -chown yarn:hadoop [yarn.nodemanager.remote-app-log-dir]
$ sudo -u hdfs hadoop fs -chmod 1777 [yarn.nodemanager.remote-app-log-dir]
$ sudo -u hdfs hadoop fs -chown mapred:hadoop [mapreduce.jobhistory.intermediate-done-dir]
$ sudo -u hdfs hadoop fs -chmod 1777 [mapreduce.jobhistory.intermediate-done-dir]
$ sudo -u hdfs hadoop fs -chown mapred:hadoop [mapreduce.jobhistory.done-dir]
$ sudo -u hdfs hadoop fs -chmod 750 [mapreduce.jobhistory.done-dir]
```

- In addition (whether or not Hadoop security is enabled) create the `/tmp` directory. For instructions on creating `/tmp` and setting its permissions, see Step 7: If Necessary, Create the HDFS /tmp Directory.
- In addition (whether or not Hadoop security is enabled), change permissions on the `/user/history` Directory. See Step 8: Create the history Directory and Set Permissions and Owner.

Step 3: If you are Using AES-256 Encryption, Install the JCE Policy File

If you are using CentOS/Red Hat Enterprise Linux 5.6 or higher, or Ubuntu, which use AES-256 encryption by default for tickets, you must install the Java Cryptography Extension (JCE) Unlimited Strength Jurisdiction Policy File on all cluster and Hadoop user machines. For JCE Policy File installation instructions, see the `README.txt` file included in the `jce_policy-x.zip` file.

Alternatively, you can configure Kerberos to not use AES-256 by removing `aes256-cts:normal` from the `supported_enctypes` field of the `kdc.conf` or `krb5.conf` file. After changing the `kdc.conf` file, you must restart both the KDC and the kadmin server for those changes to take affect. You may also need to re-create or change the password of the relevant principals, including potentially the Ticket Granting Ticket principal (krbtgt/REALM@REALM). If AES-256 is still used after completing steps, the `aes256-cts:normal` setting existed when the Kerberos database was created. To fix this, create a new Kerberos database and then restart both the KDC and the kadmin server.

To verify the type of encryption used in your cluster:

1. On the local KDC host, type this command to create a test principal:

```
$ kadmin -q "addprinc test"
```

2. On a cluster host, type this command to start a Kerberos session as test:

```
$ kinit test
```

3. On a cluster host, type this command to view the encryption type in use:

```
$ klist -e
```

   If AES is being used, output like the following is displayed after you type the `klist` command; note that `AES-256` is included in the output:

```
Ticket cache: FILE:/tmp/krb5cc_0
Default principal: test@SCM
Valid starting     Expires            Service principal
05/19/11 13:25:04  05/20/11 13:25:04  krbtgt/SCM@SCM
    Etype (skey, tkt): AES-256 CTS mode with 96-bit SHA-1 HMAC, AES-256 CTS mode with
96-bit SHA-1 HMAC
```

Step 4: Create and Deploy the Kerberos Principals and Keytab Files

A Kerberos principal is used in a Kerberos-secured system to represent a unique identity. Kerberos assigns tickets to Kerberos principals to enable them to access Kerberos-secured Hadoop services. For Hadoop, the principals should be of the format `username/fully.qualified.domain.name@YOUR-REALM.COM`. In this guide, the term `username` in the `username/fully.qualified.domain.name@YOUR-REALM.COM` principal refers to the username of an existing Unix account, such as `hdfs` or `mapred`.

A keytab is a file containing pairs of Kerberos principals and an encrypted copy of that principal's key. The keytab files are unique to each host since their keys include the hostname. This file is used to authenticate a principal on a host to Kerberos without human interaction or storing a password in a plain text file. Because having access to the keytab file for a principal allows one to act as that principal, access to the keytab files should be tightly secured. They should be readable by a minimal set of users, should be stored on local disk, and should not be included in machine backups, unless access to those backups is as secure as access to the local machine.

Important:

For both MRv1 and YARN deployments: *On every machine in your cluster*, there must be a keytab file for the `hdfs` user and a keytab file for the `mapred` user. The `hdfs` keytab file must contain entries for the `hdfs` principal and a `HTTP` principal, and the `mapred` keytab file must contain entries for the `mapred` principal and a `HTTP` principal. On each respective machine, the `HTTP` principal will be the same in both keytab files.

In addition, for YARN deployments only: *On every machine in your cluster*, there must be a keytab file for the `yarn` user. The `yarn` keytab file must contain entries for the `yarn` principal and a `HTTP` principal. On each respective machine, the `HTTP` principal in the `yarn` keytab file will be the same as the `HTTP` principal in the `hdfs` and `mapred` keytab files.

Note:

The following instructions illustrate an example of creating keytab files for MIT Kerberos. If you are using another version of Kerberos, refer to your Kerberos documentation for instructions. You may use either `kadmin` or `kadmin.local` to run these commands.

When to Use kadmin.local and kadmin

When creating the Kerberos principals and keytabs, you can use kadmin.local or kadmin depending on your access and account:

- If you have root access to the KDC machine, but you do not have a Kerberos admin account, use kadmin.local.
- If you do not have root access to the KDC machine, but you do have a Kerberos admin account, use kadmin.
- If you have both root access to the KDC machine and a Kerberos admin account, you can use either one.

To start kadmin.local (on the KDC machine) or kadmin from any machine, run this command:

```
$ sudo kadmin.local
```

OR:

```
$ kadmin
```

Note:

In this guide, `kadmin` is shown as the prompt for commands in the kadmin shell, but you can type the same commands at the `kadmin.local` prompt in the kadmin.local shell.

> **Note:**
>
> Running kadmin.local may prompt you for a password because it is being run via sudo. You should provide your Unix password. Running kadmin may prompt you for a password because you need Kerberos admin privileges. You should provide your Kerberos admin password.

To create the Kerberos principals

> **Important:**
>
> If you plan to use Oozie, Impala, or the Hue Kerberos ticket renewer in your cluster, you must configure your KDC to allow tickets to be renewed, and you must configure `krb5.conf` to request renewable tickets. Typically, you can do this by adding the `max_renewable_life` setting to your realm in `kdc.conf`, and by adding the `renew_lifetime` parameter to the `libdefaults` section of `krb5.conf`. For more information about renewable tickets, see the Kerberos documentation.

Do the following steps for every host in your cluster. Run the commands in the kadmin.local or kadmin shell, replacing the `fully.qualified.domain.name` in the commands with the fully qualified domain name of each host. Replace `YOUR-REALM.COM` with the name of the Kerberos realm your Hadoop cluster is in.

1. In the kadmin.local or kadmin shell, create the `hdfs` principal. This principal is used for the NameNode, Secondary NameNode, and DataNodes.

```
kadmin:  addprinc -randkey hdfs/fully.qualified.domain.name@YOUR-REALM.COM
```

> **Note:**
>
> If your Kerberos administrator or company has a policy about principal names that does not allow you to use the format shown above, you can work around that issue by configuring the `<kerberos principal>` to `<short name>` mapping that is built into Hadoop. For more information, see Appendix C - Configuring the Mapping from Kerberos Principals to Short Names.

2. Create the `mapred` principal. If you are using MRv1, the `mapred` principal is used for the JobTracker and TaskTrackers. If you are using YARN, the `mapred` principal is used for the MapReduce Job History Server.

```
kadmin:  addprinc -randkey mapred/fully.qualified.domain.name@YOUR-REALM.COM
```

3. **YARN only:** Create the `yarn` principal. This principal is used for the ResourceManager and NodeManager.

```
kadmin:  addprinc -randkey yarn/fully.qualified.domain.name@YOUR-REALM.COM
```

4. Create the `HTTP` principal.

```
kadmin:  addprinc -randkey HTTP/fully.qualified.domain.name@YOUR-REALM.COM
```

> **Important:**
>
> The HTTP principal must be in the format `HTTP/fully.qualified.domain.name@YOUR-REALM.COM`. The first component of the principal must be the literal string "HTTP". This format is standard for HTTP principals in SPNEGO and is hard-coded in Hadoop. It cannot be deviated from.

To create the Kerberos keytab files

> **Important:**
>
> The instructions in this section for creating keytab files require using the Kerberos `norandkey` option in the `xst` command. If your version of Kerberos does not support the `norandkey` option, or if you cannot use `kadmin.local`, then use these alternate instructions in Appendix F to create appropriate Kerberos keytab files. After using those alternate instructions to create the keytab files, continue with the next section To deploy the Kerberos keytab files.

Do the following steps for every host in your cluster. Run the commands in the kadmin.local or kadmin shell, replacing the `fully.qualified.domain.name` in the commands with the fully qualified domain name of each host:

1. Create the `hdfs` keytab file that will contain the `hdfs` principal and `HTTP` principal. This keytab file is used for the NameNode, Secondary NameNode, and DataNodes.

```
kadmin:  xst -norandkey -k hdfs.keytab hdfs/fully.qualified.domain.name
HTTP/fully.qualified.domain.name
```

2. Create the `mapred` keytab file that will contain the `mapred` principal and `HTTP` principal. If you are using MRv1, the `mapred` keytab file is used for the JobTracker and TaskTrackers. If you are using YARN, the `mapred` keytab file is used for the MapReduce Job History Server.

```
kadmin:  xst -norandkey -k mapred.keytab mapred/fully.qualified.domain.name
HTTP/fully.qualified.domain.name
```

3. **YARN only:** Create the `yarn` keytab file that will contain the `yarn` principal and `HTTP` principal. This keytab file is used for the ResourceManager and NodeManager.

```
kadmin:  xst -norandkey -k yarn.keytab yarn/fully.qualified.domain.name
HTTP/fully.qualified.domain.name
```

4. Use `klist` to display the keytab file entries; a correctly-created hdfs keytab file should look something like this:

```
$ klist -e -k -t hdfs.keytab
Keytab name: WRFILE:hdfs.keytab
slot KVNO Principal
---- ---- ---------------------------------------------------------------------
   1    7    HTTP/fully.qualified.domain.name@YOUR-REALM.COM (DES cbc mode with CRC-32)

   2    7    HTTP/fully.qualified.domain.name@YOUR-REALM.COM (Triple DES cbc mode with
 HMAC/sha1)
   3    7    hdfs/fully.qualified.domain.name@YOUR-REALM.COM (DES cbc mode with CRC-32)

   4    7    hdfs/fully.qualified.domain.name@YOUR-REALM.COM (Triple DES cbc mode with
 HMAC/sha1)
```

5. Continue with the next section To deploy the Kerberos keytab files.

To deploy the Kerberos keytab files

On every node in the cluster, repeat the following steps to deploy the `hdfs.keytab` and `mapred.keytab` files. If you are using YARN, you will also deploy the `yarn.keytab` file.

1. On the host machine, copy or move the keytab files to a directory that Hadoop can access, such as `/etc/hadoop/conf`.

   a. **If you are using MRv1:**

```
$ sudo mv hdfs.keytab mapred.keytab /etc/hadoop/conf/
```

If you are using YARN:

```
$ sudo mv hdfs.keytab mapred.keytab yarn.keytab /etc/hadoop/conf/
```

b. Make sure that the `hdfs.keytab` file is only readable by the `hdfs` user, and that the `mapred.keytab` file is only readable by the `mapred` user.

```
$ sudo chown hdfs:hadoop /etc/hadoop/conf/hdfs.keytab
$ sudo chown mapred:hadoop /etc/hadoop/conf/mapred.keytab
$ sudo chmod 400 /etc/hadoop/conf/*.keytab
```

> **Note:**
>
> To enable you to use the same configuration files on every host, Cloudera recommends that you use the same name for the keytab files on every host.

c. YARN only: Make sure that the `yarn.keytab` file is only readable by the `yarn` user.

```
$ sudo chown yarn:hadoop /etc/hadoop/conf/yarn.keytab
$ sudo chmod 400 /etc/hadoop/conf/yarn.keytab
```

> **Important:**
>
> If the NameNode, Secondary NameNode, DataNode, JobTracker, TaskTrackers, HttpFS, or Oozie services are configured to use Kerberos HTTP SPNEGO authentication, and two or more of these services are running on the same host, then all of the running services must use the same HTTP principal and keytab file used for their HTTP endpoints.

Step 5: Shut Down the Cluster

To enable security in CDH, you must stop all Hadoop daemons in your cluster and then change some configuration properties. You must stop all daemons in the cluster because after one Hadoop daemon has been restarted with the configuration properties set to enable security, daemons running without security enabled will be unable to communicate with that daemon. This requirement to shut down all daemons makes it impossible to do a rolling upgrade to enable security on a Hadoop cluster.

To shut down the cluster, run the following command on every node in your cluster (as root):

```
$ for x in `cd /etc/init.d ; ls hadoop-*` ; do sudo service $x stop ; done
```

Step 6: Enable Hadoop Security

Cloudera recommends that all of the Hadoop configuration files throughout the cluster have the same contents.

To enable Hadoop security, add the following properties to the `core-site.xml` file *on every machine* in the cluster:

```
<property>
  <name>hadoop.security.authentication</name>
  <value>kerberos</value> <!-- A value of "simple" would disable security. -->
</property>

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

Enabling Service-Level Authorization for Hadoop Services

The `hadoop-policy.xml` file maintains access control lists (ACL) for Hadoop services. Each ACL consists of comma-separated lists of users and groups separated by a space. For example:

```
user_a,user_b group_a,group_b
```

If you only want to specify a set of users, add a comma-separated list of users followed by a blank space. Similarly, to specify only authorized groups, use a blank space at the beginning. A * can be used to give access to all users.

For example, to give users, ann, bob, and groups, `group_a`, `group_b` access to Hadoop's DataNodeProtocol service, modify the `security.datanode.protocol.acl` property in `hadoop-policy.xml`. Similarly, to give all users access to the InterTrackerProtocol service, modify `security.inter.tracker.protocol.acl` as follows:

```
<property>
    <name>security.datanode.protocol.acl</name>
    <value>ann,bob group_a,group_b</value>
    <description>ACL for DatanodeProtocol, which is used by datanodes to
    communicate with the namenode.</description>
</property>

<property>
    <name>security.inter.tracker.protocol.acl</name>
    <value>*</value>
    <description>ACL for InterTrackerProtocol, which is used by tasktrackers to
    communicate with the jobtracker.</description>
</property>
```

For more details, see Service-Level Authorization in Hadoop.

Step 7: Configure Secure HDFS

When following the instructions in this section to configure the properties in the `hdfs-site.xml` file, keep the following important guidelines in mind:

- The properties for each daemon (NameNode, Secondary NameNode, and DataNode) must specify both the HDFS and HTTP principals, as well as the path to the HDFS keytab file.
- The Kerberos principals for the NameNode, Secondary NameNode, and DataNode are configured in the `hdfs-site.xml` file. The same `hdfs-site.xml` file with *all three* of these principals must be installed on every host machine in the cluster. That is, it is not sufficient to have the NameNode principal configured on the NameNode host machine only. This is because, for example, the DataNode must know the principal name of the NameNode in order to send heartbeats to it. Kerberos authentication is bi-directional.
- The special string `_HOST` in the properties is replaced at run-time by the fully-qualified domain name of the host machine where the daemon is running. This requires that reverse DNS is properly working on all the hosts configured this way. You may use `_HOST` only as the entirety of the second component of a principal name. For example, hdfs/_HOST@YOUR-REALM.COM is valid, but hdfs._HOST@YOUR-REALM.COM and hdfs/_HOST.example.com@YOUR-REALM.COM are not.
- When performing the `_HOST` substitution for the Kerberos principal names, the NameNode determines its own hostname based on the configured value of `fs.default.name`, whereas the DataNodes determine their hostnames based on the result of reverse DNS resolution on the DataNode hosts. Likewise, the JobTracker uses the configured value of `mapred.job.tracker` to determine its hostname whereas the TaskTrackers, like the DataNodes, use reverse DNS.
- The `dfs.datanode.address` and `dfs.datanode.http.address` port numbers for the DataNode *must* be below 1024, because this provides part of the security mechanism to make it impossible for a user to run a map task which impersonates a DataNode. The port numbers for the NameNode and Secondary NameNode can be anything you want, but the default port numbers are good ones to use.

To configure secure HDFS

Add the following properties to the `hdfs-site.xml` file *on every machine* in the cluster. Replace these example values shown below with the correct settings for your site: *path to the HDFS keytab*, *YOUR-REALM.COM*, *fully qualified domain name of NN*, and *fully qualified domain name of 2NN*

```
<!-- General HDFS security config -->
<property>
  <name>dfs.block.access.token.enable</name>
  <value>true</value>
</property>

<!-- NameNode security config -->
<property>
  <name>dfs.namenode.keytab.file</name>
  <value>/etc/hadoop/conf/hdfs.keytab</value> <!-- path to the HDFS keytab -->
</property>
<property>
  <name>dfs.namenode.kerberos.principal</name>
  <value>hdfs/_HOST@YOUR-REALM.COM</value>
</property>
<property>
  <name>dfs.namenode.kerberos.internal.spnego.principal</name>
  <value>HTTP/_HOST@YOUR-REALM.COM</value>
</property>

<!-- Secondary NameNode security config -->
<property>
  <name>dfs.secondary.namenode.keytab.file</name>
  <value>/etc/hadoop/conf/hdfs.keytab</value> <!-- path to the HDFS keytab -->
</property>
<property>
  <name>dfs.secondary.namenode.kerberos.principal</name>
  <value>hdfs/_HOST@YOUR-REALM.COM</value>
</property>
<property>
  <name>dfs.secondary.namenode.kerberos.internal.spnego.principal</name>
  <value>HTTP/_HOST@YOUR-REALM.COM</value>
</property>

<!-- DataNode security config -->
<property>
  <name>dfs.datanode.data.dir.perm</name>
  <value>700</value>
</property>
<property>
  <name>dfs.datanode.address</name>
  <value>0.0.0.0:1004</value>
</property>
<property>
  <name>dfs.datanode.http.address</name>
  <value>0.0.0.0:1006</value>
</property>
<property>
  <name>dfs.datanode.keytab.file</name>
  <value>/etc/hadoop/conf/hdfs.keytab</value> <!-- path to the HDFS keytab -->
</property>
<property>
  <name>dfs.datanode.kerberos.principal</name>
  <value>hdfs/_HOST@YOUR-REALM.COM</value>
</property>

<!-- Web Authentication config -->
<property>
  <name>dfs.web.authentication.kerberos.principal</name>
  <value>HTTP/_HOST@YOUR_REALM</value>
 </property>
```

To enable SSL for HDFS

Add the following property to `hdfs-site.xml` on *every machine* in your cluster.

```
<property>
<name>dfs.http.policy</name>
<value>HTTPS_ONLY</value>
</property>
```

Optional Step 8: Configuring Security for HDFS High Availability

CDH 5 supports the HDFS High Availability (HA) feature with Kerberos security enabled. There are two use cases that affect security for HA:

- If you are not using Quorum-based Storage (see Software Configuration for Quorum-based Storage), then no extra configuration for HA is necessary if automatic failover is not enabled. If automatic failover is enabled then access to ZooKeeper should be secured. See the Software Configuration for Shared Storage Using NFS documentation for details.
- If you are using Quorum-based Storage, then you must configure security for Quorum-based Storage by following the instructions in this section.

To configure security for Quorum-based Storage:

Add the following Quorum-based Storage configuration properties to the `hdfs-site.xml` file on all of the machines in the cluster:

```
<property>
  <name>dfs.journalnode.keytab.file</name>
  <value>/etc/hadoop/conf/hdfs.keytab</value> <!-- path to the HDFS keytab -->
</property>
<property>
  <name>dfs.journalnode.kerberos.principal</name>
  <value>hdfs/_HOST@YOUR-REALM.COM</value>
</property>
<property>
  <name>dfs.journalnode.kerberos.internal.spnego.principal</name>
  <value>HTTP/_HOST@YOUR-REALM.COM</value>
</property>
```

> **Note:**
>
> If you already have principals and keytabs created for the machines where the JournalNodes are running, then you should reuse those principals and keytabs in the configuration properties above. You will likely have these principals and keytabs already created if you are collocating a JournalNode on a machine with another HDFS daemon.

Optional Step 9: Configure secure WebHDFS

> **Note:**
>
> If you are not using WebHDFS, you can skip this step.

Security for WebHDFS is disabled by default. If you want use WebHDFS with a secure cluster, this is the time to enable and configure it.

To configure secure WebHDFS:

1. If you have not already done so, enable WebHDFS by adding the following property to the `hdfs-site.xml` file *on every machine* in the cluster.

```
<property>
  <name>dfs.webhdfs.enabled</name>
  <value>true</value>
</property>
```

2. Add the following properties to the `hdfs-site.xml` file *on every machine* in the cluster. Replace the example values shown below with the correct settings for your site.

```
<property>
  <name>dfs.web.authentication.kerberos.principal</name>
  <value>HTTP/_HOST@YOUR-REALM.COM</value>
</property>

<property>
  <name>dfs.web.authentication.kerberos.keytab</name>
  <value>/etc/hadoop/conf/HTTP.keytab</value> <!-- path to the HTTP keytab -->
</property>
```

Optional Step 10: Configuring a secure HDFS NFS Gateway

To deploy a Kerberized HDFS NFS gateway, add the following configuration properties to `hdfs-site.xml` on the NFS server.

```
<property>
<name>dfs.nfs.keytab.file</name>
<value>/etc/hadoop/conf/hdfs.keytab</value> <!-- path to the HDFS or NFS gateway keytab
 -->
</property>

<property>
<name>dfs.nfs.kerberos.principal</name>
<value>hdfs/_HOST@YOUR-REALM.COM</value>
</property>
```

Step 11: Set Variables for Secure DataNodes

In order to allow DataNodes to start on a secure Hadoop cluster, you must set the following variables on all DataNodes in `/etc/default/hadoop-hdfs-datanode`.

```
export HADOOP_SECURE_DN_USER=hdfs
export HADOOP_SECURE_DN_PID_DIR=/var/lib/hadoop-hdfs
export HADOOP_SECURE_DN_LOG_DIR=/var/log/hadoop-hdfs
export JSVC_HOME=/usr/lib/bigtop-utils/
```

Note:

Depending on the version of Linux you are using, you may not have the `/usr/lib/bigtop-utils` directory on your system. If that is the case, set the `JSVC_HOME` variable to the `/usr/libexec/bigtop-utils` directory by using this command:

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

Step 12: Start up the NameNode

You are now ready to start the NameNode. Use the `service` command to run the `/etc/init.d` script.

```
$ sudo service hadoop-hdfs-namenode start
```

You'll see some extra information in the logs such as:

```
10/10/25 17:01:46 INFO security.UserGroupInformation:
Login successful for user hdfs/fully.qualified.domain.name@YOUR-REALM.COM using keytab
 file /etc/hadoop/conf/hdfs.keytab
```

and:

```
12/05/23 18:18:31 INFO http.HttpServer: Adding Kerberos (SPNEGO) filter to
getDelegationToken
12/05/23 18:18:31 INFO http.HttpServer: Adding Kerberos (SPNEGO) filter to
renewDelegationToken
12/05/23 18:18:31 INFO http.HttpServer: Adding Kerberos (SPNEGO) filter to
cancelDelegationToken
12/05/23 18:18:31 INFO http.HttpServer: Adding Kerberos (SPNEGO) filter to fsck
12/05/23 18:18:31 INFO http.HttpServer: Adding Kerberos (SPNEGO) filter to getimage
12/05/23 18:18:31 INFO http.HttpServer: Jetty bound to port 50070
12/05/23 18:18:31 INFO mortbay.log: jetty-6.1.26
12/05/23 18:18:31 INFO server.KerberosAuthenticationHandler: Login using keytab
/etc/hadoop/conf/hdfs.keytab, for principal
HTTP/fully.qualified.domain.name@YOUR-REALM.COM
12/05/23 18:18:31 INFO server.KerberosAuthenticationHandler: Initialized, principal
[HTTP/fully.qualified.domain.name@YOUR-REALM.COM] from keytab
[/etc/hadoop/conf/hdfs.keytab]
```

You can verify that the NameNode is working properly by opening a web browser to `http://machine:50070/` where *machine* is the name of the machine where the NameNode is running.

Cloudera also recommends testing that the NameNode is working properly by performing a metadata-only HDFS operation, which will now require correct Kerberos credentials. For example:

```
$ hadoop fs -ls
```

Information about the kinit Command

Important:

Running the `hadoop fs -ls` command will fail if you do not have a valid Kerberos ticket in your credentials cache. You can examine the Kerberos tickets currently in your credentials cache by running the `klist` command. You can obtain a ticket by running the `kinit` command and either specifying a keytab file containing credentials, or entering the password for your principal. If you do not have a valid ticket, you will receive an error such as:

```
11/01/04 12:08:12 WARN ipc.Client: Exception encountered while connecting
 to the server : javax.security.sasl.SaslException:
 GSS initiate failed [Caused by GSSException: No valid credentials
provided (Mechanism level: Failed to find any Kerberos tgt)]
Bad connection to FS. command aborted. exception: Call to
nn-host/10.0.0.2:8020 failed on local exception: java.io.IOException:
javax.security.sasl.SaslException: GSS initiate failed [Caused by
GSSException: No valid credentials provided (Mechanism level: Failed to
 find any Kerberos tgt)]
```

Note:

The `kinit` command must either be on the path for user accounts running the Hadoop client, or else the `hadoop.kerberos.kinit.command` parameter in `core-site.xml` must be manually configured to the absolute path to the `kinit` command.

> **Note:**
>
> If you are running MIT Kerberos 1.8.1 or higher, a bug in versions of the Oracle JDK 6 Update 26 and higher causes Java to be unable to read the Kerberos credentials cache even after you have successfully obtained a Kerberos ticket using `kinit`. To workaround this bug, run `kinit -R` after running `kinit` initially to obtain credentials. Doing so will cause the ticket to be renewed, and the credentials cache rewritten in a format which Java can read. For more information about this problem, see Troubleshooting.

Step 12: Start up a DataNode

Begin by starting one DataNode only to make sure it can properly connect to the NameNode. Use the `service` command to run the `/etc/init.d` script.

```
$ sudo service hadoop-hdfs-datanode start
```

You'll see some extra information in the logs such as:

```
10/10/25 17:21:41 INFO security.UserGroupInformation:
Login successful for user hdfs/fully.qualified.domain.name@YOUR-REALM.COM using keytab
 file /etc/hadoop/conf/hdfs.keytab
```

If you can get a single DataNode running and you can see it registering with the NameNode in the logs, then start up all the DataNodes. You should now be able to do all HDFS operations.

Step 14: Set the Sticky Bit on HDFS Directories

This step is optional but strongly recommended for security. In CDH 5, HDFS file permissions have support for the sticky bit. The sticky bit can be set on directories, preventing anyone except the superuser, directory owner, or file owner from deleting or moving the files within the directory. Setting the sticky bit for a file has no effect. This is useful for directories such as `/tmp` which previously had to be set to be world-writable. To set the sticky bit on the `/tmp` directory, run the following command:

```
$ sudo -u hdfs kinit -k -t hdfs.keytab hdfs/fully.qualified.domain.name@YOUR-REALM.COM
$ sudo -u hdfs hadoop fs -chmod 1777 /tmp
```

After running this command, the permissions on `/tmp` will appear as shown below. (Note the "`t`" instead of the final "`x`".)

```
$ hadoop fs -ls /
Found 2 items
drwxrwxrwt - hdfs supergroup 0 2011-02-14 15:55 /tmp
drwxr-xr-x - hdfs supergroup 0 2011-02-14 14:01 /user
```

Step 15: Start up the Secondary NameNode (if used)

At this point, you should be able to start the Secondary NameNode if you are using one:

```
$ sudo service hadoop-hdfs-secondarynamenode start
```

> **Note:**
>
> If you are using HDFS HA, do not use the Secondary NameNode. See Configuring HDFS High Availability for instructions on configuring and deploying the Standby NameNode.

You'll see some extra information in the logs such as:

```
10/10/26 12:03:18 INFO security.UserGroupInformation:
Login successful for user hdfs/fully.qualified.domain.name@YOUR-REALM using keytab file
 /etc/hadoop/conf/hdfs.keytab
```

and:

```
12/05/23 18:33:06 INFO http.HttpServer: Adding Kerberos (SPNEGO) filter to getimage
12/05/23 18:33:06 INFO http.HttpServer: Jetty bound to port 50090
12/05/23 18:33:06 INFO mortbay.log: jetty-6.1.26
12/05/23 18:33:06 INFO server.KerberosAuthenticationHandler: Login using keytab
/etc/hadoop/conf/hdfs.keytab, for principal
HTTP/fully.qualified.domain.name@YOUR-REALM.COM
12/05/23 18:33:06 INFO server.KerberosAuthenticationHandler: Initialized, principal
[HTTP/fully.qualified.domain.name@YOUR-REALM.COM] from keytab
[/etc/hadoop/conf/hdfs.keytab]
```

You should make sure that the Secondary NameNode not only starts, but that it is successfully checkpointing.

If you're using the `service` command to start the Secondary NameNode from the `/etc/init.d scripts`, Cloudera recommends setting the property `fs.checkpoint.period` in the `hdfs-site.xml` file to a very low value (such as 5), and then monitoring the Secondary NameNode logs for a successful startup and checkpoint. Once you are satisfied that the Secondary NameNode is checkpointing properly, you should reset the `fs.checkpoint.period` to a reasonable value, or return it to the default, and then restart the Secondary NameNode.

You can make the Secondary NameNode perform a checkpoint by doing the following:

```
$ sudo -u hdfs hdfs secondarynamenode -checkpoint force
```

Note that this will not cause a running Secondary NameNode to checkpoint, but rather will start up a Secondary NameNode that will immediately perform a checkpoint and then shut down. This can be useful for debugging.

> **Note:**
>
> If you encounter errors during Secondary NameNode checkpointing, it may be helpful to enable Kerberos debugging output. For instructions, see Appendix D - Enabling Debugging Output for the Sun Kerberos Classes.

Step 16: Configure Either MRv1 Security or YARN Security

At this point, you are ready to configure either MRv1 Security or YARN Security.

- If you are using MRv1, do the steps in Configuring MRv1 Security to configure, start, and test secure MRv1.
- If you are using YARN, do the steps in Configuring YARN Security to configure, start, and test secure YARN.

Configuring MRv1 Security

If you are using YARN, skip this section and see Configuring YARN Security.

If you are using MRv1, do the following steps to configure, start, and test secure MRv1.

1. Step 1: Configure Secure MRv1 on page 61
2. Step 2: Start up the JobTracker on page 63
3. Step 3: Start up a TaskTracker on page 63
4. Step 4: Try Running a Map/Reduce Job on page 63

Step 1: Configure Secure MRv1

Keep the following important information in mind when configuring secure MapReduce:

- The properties for Job Tracker and Task Tracker must specify the mapred principal, as well as the path to the `mapred` keytab file.

- The Kerberos principals for the Job Tracker and Task Tracker are configured in the `mapred-site.xml` file. The same `mapred-site.xml` file with *both* of these principals must be installed on every host machine in the cluster. That is, it is not sufficient to have the Job Tracker principal configured on the Job Tracker host machine only. This is because, for example, the TaskTracker must know the principal name of the JobTracker in order to securely register with the JobTracker. Kerberos authentication is bi-directional.
- Do not use `${user.name}` in the value of the `mapred.local.dir` or `hadoop.log.dir` properties in `mapred-site.xml`. Doing so can prevent tasks from launching on a secure cluster.
- Make sure that each user who will be running MRv1 jobs exists on all cluster nodes (that is, on every node that hosts any MRv1 daemon).
- Make sure the value specified for `mapred.local.dir` is identical in `mapred-site.xml` and `taskcontroller.cfg`. If the values are different, this error message is returned.
- Make sure the value specified in `taskcontroller.cfg` for `hadoop.log.dir` is the same as what the Hadoop daemons are using, which is `/var/log/hadoop-0.20-mapreduce` by default and can be configured in `mapred-site.xml`. If the values are different, this error message is returned.

To configure secure MapReduce:

1. Add the following properties to the `mapred-site.xml` file *on every machine* in the cluster:

```
<!-- JobTracker security configs -->
<property>
  <name>mapreduce.jobtracker.kerberos.principal</name>
  <value>mapred/_HOST@YOUR-REALM.COM</value>
</property>
<property>
  <name>mapreduce.jobtracker.keytab.file</name>
  <value>/etc/hadoop/conf/mapred.keytab</value> <!-- path to the MapReduce keytab -->
</property>

<!-- TaskTracker security configs -->
<property>
  <name>mapreduce.tasktracker.kerberos.principal</name>
  <value>mapred/_HOST@YOUR-REALM.COM</value>
</property>
<property>
  <name>mapreduce.tasktracker.keytab.file</name>
  <value>/etc/hadoop/conf/mapred.keytab</value> <!-- path to the MapReduce keytab -->
</property>

<!-- TaskController settings -->
<property>
  <name>mapred.task.tracker.task-controller</name>
  <value>org.apache.hadoop.mapred.LinuxTaskController</value>
</property>
<property>
  <name>mapreduce.tasktracker.group</name>
  <value>mapred</value>
</property>
```

2. Create a file called `taskcontroller.cfg` that contains the following information:

```
hadoop.log.dir=<Path to Hadoop log directory. Should be same value used to start the
TaskTracker. This is required to set proper permissions on the log files so that they
can be written to by the user's tasks and read by the TaskTracker for serving on the
web UI.>
mapreduce.tasktracker.group=mapred
banned.users=mapred,hdfs,bin
min.user.id=1000
```

> **Note:**
>
> In the `taskcontroller.cfg` file, the default setting for the `banned.users` property is `mapred`, `hdfs`, and `bin` to prevent jobs from being submitted using those user accounts. The default setting for the `min.user.id` property is `1000` to prevent jobs from being submitted with a user ID less than 1000, which are conventionally Unix super users. Note that some operating systems such as CentOS 5 use a default value of 500 and above for user IDs, not 1000. If this is the case on your system, change the default setting for the `min.user.id` property to `500`. If there are user accounts on your cluster that have a user ID less than the value specified for the `min.user.id` property, the TaskTracker returns an error code of 255.

3. The path to the `taskcontroller.cfg` file is determined relative to the location of the `task-controller` binary. Specifically, the path is `<path of task-controller binary>/../../conf/taskcontroller.cfg`. If you installed the CDH 5 package, this path will always correspond to `/etc/hadoop/conf/taskcontroller.cfg`.

> **Note:**
>
> For more information about the `task-controller` program, see Information about Other Hadoop Security Programs.

> **Important:**
>
> The same `mapred-site.xml` file and the same `hdfs-site.xml` file must both be installed on every host machine in the cluster so that the NameNode, Secondary NameNode, DataNode, Job Tracker and Task Tracker can all connect securely with each other.

Step 2: Start up the JobTracker

You are now ready to start the JobTracker.

If you're using the `/etc/init.d/hadoop-0.20-mapreduce-jobtracker` script, then you can use the `service` command to run it now:

```
$ sudo service hadoop-0.20-mapreduce-jobtracker start
```

You can verify that the JobTracker is working properly by opening a web browser to `http://machine:50030/` where *machine* is the name of the machine where the JobTracker is running.

Step 3: Start up a TaskTracker

You are now ready to start a TaskTracker.

If you're using the `/etc/init.d/hadoop-0.20-mapreduce-tasktracker` script, then you can use the `service` command to run it now:

```
$ sudo service hadoop-0.20-mapreduce-tasktracker start
```

Step 4: Try Running a Map/Reduce Job

You should now be able to run Map/Reduce jobs. To confirm, try launching a sleep or a pi job from the provided Hadoop examples (`/usr/lib/hadoop-0.20-mapreduce/hadoop-examples.jar`). Note that you will need Kerberos credentials to do so.

> **Important:**
>
> Remember that the user who launches the job must exist on every node.

Configuring YARN Security

If you are using MRv1, skip this section and see Configuring MRv1 Security.

If you are using YARN, do the following steps to configure, start, and test secure YARN.

1. Configure Secure YARN.
2. Start up the ResourceManager.
3. Start up the NodeManager.
4. Start up the MapReduce Job History Server.
5. Try Running a Map/Reduce YARN Job.
6. (Optional) Step 6: Configuring YARN for long-running applications on page 66

Step 1: Configure Secure YARN

Before you start:

- The Kerberos principals for the ResourceManager and NodeManager are configured in the `yarn-site.xml` file. The same `yarn-site.xml` file must be installed on every host machine in the cluster.
- Make sure that each user who will be running YARN jobs exists on all cluster nodes (that is, on every node that hosts any YARN daemon).

To configure secure YARN:

1. Add the following properties to the `yarn-site.xml` file *on every machine* in the cluster:

```
<!-- ResourceManager security configs -->
<property>
  <name>yarn.resourcemanager.keytab</name>
  <value>/etc/hadoop/conf/yarn.keytab</value> <!-- path to the YARN keytab -->
</property>
<property>
  <name>yarn.resourcemanager.principal</name>
  <value>yarn/_HOST@YOUR-REALM.COM</value>
</property>

<!-- NodeManager security configs -->
<property>
  <name>yarn.nodemanager.keytab</name>
  <value>/etc/hadoop/conf/yarn.keytab</value> <!-- path to the YARN keytab -->
</property>
<property>
  <name>yarn.nodemanager.principal</name>
  <value>yarn/_HOST@YOUR-REALM.COM</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.group</name>
  <value>yarn</value>
</property>

<!-- To enable SSL -->
<property>
  <name>yarn.http.policy</name>
  <value>HTTPS_ONLY</value>
</property>
```

2. Add the following properties to the `mapred-site.xml` file *on every machine* in the cluster:

```
<!-- MapReduce Job History Server security configs -->
<property>
  <name>mapreduce.jobhistory.address</name>
  <value>host:port</value> <!-- Host and port of the MapReduce Job History Server;
default port is 10020  -->
</property>
<property>
```

```
  <name>mapreduce.jobhistory.keytab</name>
  <value>/etc/hadoop/conf/mapred.keytab</value> <!-- path to the MAPRED keytab for the
 Job History Server -->
</property>
<property>
  <name>mapreduce.jobhistory.principal</name>
  <value>mapred/_HOST@YOUR-REALM.COM</value>
</property>

<!-- To enable SSL -->
<property>
  <name>mapreduce.jobhistory.http.policy</name>
  <value>HTTPS_ONLY</value>
</property>
```

3. Create a file called `container-executor.cfg` for the Linux Container Executor program that contains the following information:

```
yarn.nodemanager.local-dirs=<comma-separated list of paths to local NodeManager
directories. Should be same values specified in yarn-site.xml. Required to validate
paths passed to container-executor in order.>
yarn.nodemanager.linux-container-executor.group=yarn
yarn.nodemanager.log-dirs=<comma-separated list of paths to local NodeManager log
directories. Should be same values specified in yarn-site.xml. Required to set proper
permissions on the log files so that they can be written to by the user's containers
and read by the NodeManager for log aggregation.
banned.users=hdfs,yarn,mapred,bin
min.user.id=1000
```

> **Note:**
>
> In the `container-executor.cfg` file, the default setting for the `banned.users` property is `hdfs`, `yarn`, `mapred`, and `bin` to prevent jobs from being submitted via those user accounts. The default setting for the `min.user.id` property is `1000` to prevent jobs from being submitted with a user ID less than 1000, which are conventionally Unix super users. Note that some operating systems such as CentOS 5 use a default value of 500 and above for user IDs, not 1000. If this is the case on your system, change the default setting for the `min.user.id` property to `500`. If there are user accounts on your cluster that have a user ID less than the value specified for the `min.user.id` property, the NodeManager returns an error code of 255.

4. The path to the `container-executor.cfg` file is determined relative to the location of the container-executor binary. Specifically, the path is `<dirname of container-executor binary>/../etc/hadoop/container-executor.cfg`. If you installed the CDH 5 package, this path will always correspond to `/etc/hadoop/conf/container-executor.cfg`.

> **Note:**
>
> The `container-executor` program requires that the paths including and leading up to the directories specified in `yarn.nodemanager.local-dirs` and `yarn.nodemanager.log-dirs` to be set to 755 permissions as shown in this table on permissions on directories.

5. Verify that the ownership and permissions of the `container-executor` program corresponds to:

```
---Sr-s--- 1 root yarn 36264 May 20 15:30 container-executor
```

> **Note:**
>
> For more information about the Linux Container Executor program, see Appendix B - Information about Other Hadoop Security Programs.

Step 2: Start up the ResourceManager

You are now ready to start the ResourceManager.

> **Note:**
>
> Make sure you always start ResourceManager before starting NodeManager.

If you're using the `/etc/init.d/hadoop-yarn-resourcemanager` script, then you can use the `service` command to run it now:

```
$ sudo service hadoop-yarn-resourcemanager start
```

You can verify that the ResourceManager is working properly by opening a web browser to http://`host`:8088/ where `host` is the name of the machine where the ResourceManager is running.

Step 3: Start up the NodeManager

You are now ready to start the NodeManager.

If you're using the `/etc/init.d/hadoop-yarn-nodemanager` script, then you can use the `service` command to run it now:

```
$ sudo service hadoop-yarn-nodemanager start
```

You can verify that the NodeManager is working properly by opening a web browser to http://`host`:8042/ where `host` is the name of the machine where the NodeManager is running.

Step 4: Start up the MapReduce Job History Server

You are now ready to start the MapReduce Job History Server.

If you're using the `/etc/init.d/hadoop-mapreduce-historyserver` script, then you can use the `service` command to run it now:

```
$ sudo service hadoop-mapreduce-historyserver start
```

You can verify that the MapReduce JobHistory Server is working properly by opening a web browser to http://`host`:19888/ where `host` is the name of the machine where the MapReduce JobHistory Server is running.

Step 5: Try Running a Map/Reduce YARN Job

You should now be able to run Map/Reduce jobs. To confirm, try launching a sleep or a pi job from the provided Hadoop examples (`/usr/lib/hadoop-mapreduce/hadoop-mapreduce-examples.jar`). Note that you will need Kerberos credentials to do so.

> **Important:**
>
> Remember that the user who launches the job must exist on every node.

To try running a MapReduce job using YARN, set the `HADOOP_MAPRED_HOME` environment variable and then submit the job. For example:

```
$ export HADOOP_MAPRED_HOME=/usr/lib/hadoop-mapreduce
$ /usr/bin/hadoop jar /usr/lib/hadoop-mapreduce/hadoop-mapreduce-examples.jar pi 10
10000
```

(Optional) Step 6: Configuring YARN for long-running applications

Long-running applications such as Spark Streaming jobs will need additional configuration since the default settings only allow the `hdfs` user's delegation tokens a maximum lifetime of 7 days which is not always sufficient.

You can work around this by configuring the ResourceManager as a proxy user for the corresponding HDFS NameNode so that the ResourceManager can request new tokens when the existing ones are past their maximum lifetime. YARN will then be able to continue performing localization and log-aggregation on behalf of the `hdfs` user.

Set the following property in `yarn-site.xml` to `true`:

```
<property>
<name>yarn.resourcemanager.proxy-user-privileges.enabled</name>
<value>true</value>
</property>
```

Configure the following properties in `core-site.xml` on the HDFS NameNode. You can use a more restrictive configuration by specifying hosts/groups instead of `*` as in the example below.

```
<property>
<name>hadoop.proxyuser.yarn.hosts</name>
<value>*</value>
</property>

<property>
<name>hadoop.proxyuser.yarn.groups</name>
<value>*</value>
</property>
```

Flume Authentication

Flume agents have the ability to store data on an HDFS filesystem configured with Hadoop security. The Kerberos system and protocols authenticate communications between clients and services. Hadoop clients include users and MapReduce jobs on behalf of users, and the services include HDFS and MapReduce. Flume acts as a Kerberos principal (user) and needs Kerberos credentials to interact with the Kerberos security-enabled service. Authenticating a user or a service can be done using a Kerberos keytab file. This file contains a key that is used to obtain a ticket-granting ticket (TGT). The TGT is used to mutually authenticate the client and the service using the Kerberos KDC.

The following sections describe how to use Flume 1.3.x and CDH 5 with Kerberos security on your Hadoop cluster:

- Configuring Flume's Security Properties on page 67
- Flume Account Requirements on page 69
- Testing the Flume HDFS Sink Configuration on page 69
- Writing to a Secure HBase cluster on page 70

> **Important:**
>
> To enable Flume to work with Kerberos security on your Hadoop cluster, make sure you perform the installation and configuration steps in Configuring Hadoop Security in CDH 5.

> **Note:**
>
> These instructions have been tested with CDH 5 and MIT Kerberos 5 only. The following instructions describe an example of how to configure a Flume agent to be a client as the user `flume` to a secure HDFS service. This section does not describe how to secure the communications between Flume agents, which is not currently implemented.

Configuring Flume's Security Properties

Contents:

Writing as a single user for all HDFS sinks in a given Flume agent

The Hadoop services require a three-part principal that has the form of `username/fully.qualified.domain.name@YOUR-REALM.COM`. Cloudera recommends using `flume` as the first

component and the fully qualified domain name of the host machine as the second. Assuming that Kerberos and security-enabled Hadoop have been properly configured on the Hadoop cluster itself, you must add the following parameters to the Flume agent's `flume.conf` configuration file, which is typically located at `/etc/flume-ng/conf/flume.conf`:

```
agentName.sinks.sinkName.hdfs.kerberosPrincipal =
flume/fully.qualified.domain.name@YOUR-REALM.COM
agentName.sinks.sinkName.hdfs.kerberosKeytab = /etc/flume-ng/conf/flume.keytab
```

where:

`agentName` is the name of the Flume agent being configured, which in this release defaults to the value "agent". `sinkName` is the name of the HDFS sink that is being configured. The respective sink's `type` must be `HDFS`.

In the previous example, `flume` is the first component of the principal name, `fully.qualified.domain.name` is the second, and `YOUR-REALM.COM` is the name of the Kerberos realm your Hadoop cluster is in. The `/etc/flume-ng/conf/flume.keytab` file contains the keys necessary for `flume/fully.qualified.domain.name@YOUR-REALM.COM` to authenticate with other services.

Flume and Hadoop also provide a simple keyword, `_HOST`, that gets expanded to be the fully qualified domain name of the host machine where the service is running. This allows you to have one `flume.conf` file with the same `hdfs.kerberosPrincipal` value on all of your agent host machines.

```
agentName.sinks.sinkName.hdfs.kerberosPrincipal = flume/_HOST@YOUR-REALM.COM
```

Writing as different users across multiple HDFS sinks in a single Flume agent

In this release, support has been added for secure impersonation of Hadoop users (similar to "sudo" in UNIX). This is implemented in a way similar to how Oozie implements secure user impersonation.

The following steps to set up secure impersonation from Flume to HDFS assume your cluster is configured using Kerberos. (However, impersonation also works on non-Kerberos secured clusters, and Kerberos-specific aspects should be omitted in that case.)

1. Configure Hadoop to allow impersonation. Add the following configuration properties to your `core-site.xml`.

```
<property>
     <name>hadoop.proxyuser.flume.groups</name>
     <value>group1,group2</value>
     <description>Allow the flume user to impersonate any members of group1 and
group2</description>
</property>
<property>
     <name>hadoop.proxyuser.flume.hosts</name>
     <value>host1,host2</value>
    <description>Allow the flume user to connect only from host1 and host2 to impersonate
 a user</description>
</property>
```

   You can use the wildcard character `*` to enable impersonation of any user from any host. For more information, see Secure Impersonation.

2. Set up a Kerberos keytab for the Kerberos principal and host Flume is connecting to HDFS from. This user must match the Hadoop configuration in the preceding step. For instructions, see Configuring Hadoop Security in CDH 5.
3. Configure the HDFS sink with the following configuration options:
4. `hdfs.kerberosPrincipal` - fully-qualified principal. Note: `_HOST` will be replaced by the hostname of the local machine (only in-between the `/` and `@` characters)
5. `hdfs.kerberosKeytab` - location on the local machine of the keytab containing the user and host keys for the above principal
6. `hdfs.proxyUser` - the proxy user to impersonate

Example snippet (the majority of the HDFS sink configuration options have been omitted):

```
agent.sinks.sink-1.type = HDFS
agent.sinks.sink-1.hdfs.kerberosPrincipal = flume/_HOST@YOUR-REALM.COM
agent.sinks.sink-1.hdfs.kerberosKeytab = /etc/flume-ng/conf/flume.keytab
agent.sinks.sink-1.hdfs.proxyUser = weblogs

agent.sinks.sink-2.type = HDFS
agent.sinks.sink-2.hdfs.kerberosPrincipal = flume/_HOST@YOUR-REALM.COM
agent.sinks.sink-2.hdfs.kerberosKeytab = /etc/flume-ng/conf/flume.keytab
agent.sinks.sink-2.hdfs.proxyUser = applogs
```

In the above example, the flume Kerberos principal impersonates the user `weblogs` in `sink-1` and the user `applogs` in `sink-2`. This will only be allowed if the Kerberos KDC authenticates the specified principal (`flume` in this case), and the if NameNode authorizes impersonation of the specified proxy user by the specified principal.

Limitations

At this time, Flume does not support using multiple Kerberos principals or keytabs in the same agent. Therefore, if you want to create files as multiple users on HDFS, then impersonation must be configured, and exactly one principal must be configured in Hadoop to allow impersonation of all desired accounts. In addition, the same keytab path must be used across all HDFS sinks in the same agent. If you attempt to configure multiple principals or keytabs in the same agent, Flume will emit the following error message:

```
Cannot use multiple kerberos principals in the same agent. Must restart agent to use
new principal or keytab.
```

Flume Account Requirements

This section provides an overview of the account and credential requirements for Flume to write to a Kerberized HDFS. Note the distinctions between the Flume agent machine, DataNode machine, and NameNode machine, as well as the `flume` Unix user account versus the `flume` Hadoop/Kerberos user account.

- Each Flume agent machine that writes to HDFS (using a configured HDFS sink) needs a Kerberos principal of the form:

```
flume/fully.qualified.domain.name@YOUR-REALM.COM
```

  where `fully.qualified.domain.name` is the fully qualified domain name of the given Flume agent host machine, and `YOUR-REALM.COM` is the Kerberos realm.

- Each Flume agent machine that writes to HDFS does *not* need to have a `flume` Unix user account to write files owned by the `flume` Hadoop/Kerberos user. Only the keytab for the `flume` Hadoop/Kerberos user is required on the Flume agent machine.
- DataNode machines do *not* need Flume Kerberos keytabs and also do *not* need the `flume` Unix user account.
- TaskTracker (MRv1) or NodeManager (YARN) machines need a `flume` Unix user account *if and only if* MapReduce jobs are being run as the `flume` Hadoop/Kerberos user.
- The NameNode machine needs to be able to resolve the groups of the `flume` user. The groups of the `flume` user on the NameNode machine are mapped to the Hadoop groups used for authorizing access.
- The NameNode machine does *not* need a Flume Kerberos keytab.

Testing the Flume HDFS Sink Configuration

To test whether your Flume HDFS sink is properly configured to connect to your secure HDFS cluster, you must run data through Flume. An easy way to do this is to configure a Netcat source, a Memory channel, and an HDFS sink. Start Flume with that configuration, and use the nc command (available freely online and with many UNIX distributions) to send events to the Netcat source port. The resulting events should appear on HDFS in the configured location. If the events do not appear, check the Flume log at `/var/log/flume-ng/flume.log` for any error messages related to Kerberos.

Writing to a Secure HBase cluster

If you want to write to a secure HBase cluster, be aware of the following:

- Flume must be configured to use Kerberos security as documented above, and HBase must be configured to use Kerberos security as documented in HBase Security Configuration.
- The `hbase-site.xml` file, which must be configured to use Kerberos security, must be in Flume's classpath or `HBASE_HOME/conf`.
- HBaseSink `org.apache.flume.sink.hbase.HBaseSink` supports secure HBase, but AsyncHBaseSink `org.apache.flume.sink.hbase.AsyncHBaseSink` does not.
- The Flume HBase Sink takes these two parameters:
- `kerberosPrincipal` – specifies the Kerberos principal to be used
- `kerberosKeytab` – specifies the path to the Kerberos keytab These are defined as:

```
agent.sinks.hbaseSink.kerberosPrincipal = flume/fully.qualified.domain.name@YOUR-REALM.COM
agent.sinks.hbaseSink.kerberosKeytab = /etc/flume-ng/conf/flume.keytab
```

- If HBase is running with the AccessController coprocessor, the `flume` user (or whichever user the agent is running as) must have permissions to write to the same table and the column family that the sink is configured to write to. You can grant permissions using the `grant` command from HBase shell as explained in HBase Security Configuration.
- The Flume HBase Sink does not currently support impersonation; it will write to HBase as the user the agent is being run as.
- If you want to use HDFS Sink and HBase Sink to write to HDFS and HBase from the same agent respectively, both sinks have to use the same principal and keytab. If you want to use different credentials, the sinks have to be on different agents.
- Each Flume agent machine that writes to HBase (using a configured HBase sink) needs a Kerberos principal of the form:

```
flume/fully.qualified.domain.name@YOUR-REALM.COM
```

  where `fully.qualified.domain.name` is the fully qualified domain name of the given Flume agent host machine, and YOUR-REALM.COM is the Kerberos realm.

HBase Authentication

There are two major parts in the process of configuring HBase security:

1. **Configure HBase Authentication**: You must establish a mechanism for HBase servers and clients to securely identify themselves with HDFS, ZooKeeper, and each other (called *authentication*). This ensures that, for example, a host claiming to be an HBase Region Server or a particular HBase client are in fact who they claim to be.
2. **Configure HBase Authorization**: You must establish rules for the resources that clients are allowed to access (called *authorization*). For more information, see Configuring HBase Authorization on page 277.

For more background information, see this blog post.

The following sections describe how to use Apache HBase and CDH 5 with Kerberos security on your Hadoop cluster:

- Configuring Kerberos Authentication for HBase on page 71
- Configuring Secure HBase Replication on page 73
- Configuring the HBase Client TGT Renewal Period on page 74

> **Important:** To enable HBase to work with Kerberos security on your Hadoop cluster, make sure you perform the installation and configuration steps in Configuring Hadoop Security in CDH 5 and ZooKeeper Security Configuration.

Note: These instructions have been tested with CDH and MIT Kerberos 5 only.

Important: Although an HBase Thrift server can connect to a secured Hadoop cluster, access is not secured from clients to the HBase Thrift server.

Configuring Kerberos Authentication for HBase

Here are the two high-level steps for configuring HBase authentication:

Step 1: Configure HBase Servers to Authenticate with a Secure HDFS Cluster on page 71

Step 2: Configure HBase Servers and Clients to Authenticate with a Secure ZooKeeper on page 72.

Step 1: Configure HBase Servers to Authenticate with a Secure HDFS Cluster

To configure HBase servers to authenticate with a secure HDFS cluster, you must do the following tasks:

- Enable HBase Authentication
- Configure HBase's Kerberos Principals

Enabling HBase Authentication

To enable HBase Authentication, set the `hbase.security.authentication` property to `kerberos` in `hbase-site.xml` on every host acting as an HBase master, region server, or client. In CDH 5, `hbase.rpc.engine` is automatically detected and does not need to be set.

```
<property>
    <name>hbase.security.authentication</name>
    <value>kerberos</value>
</property>
```

Configuring HBase's Kerberos Principals

In order to run HBase on a secure HDFS cluster, HBase must authenticate itself to the HDFS services. HBase acts as a Kerberos principal and needs Kerberos credentials to interact with the Kerberos-enabled HDFS daemons. Authenticating a service can be done using a keytab file. This file contains a key which allows the service to authenticate to the Kerberos Key Distribution Center (KDC).

To configure HBase's Kerberos principals:

1. Create a service principal for the HBase server using the syntax: `hbase/<fully.qualified.domain.name>@<YOUR-REALM>`. This principal is used to authenticate the HBase server with the HDFS services. Cloudera recommends using `hbase` as the username portion of this principal.

```
$ kadmin
kadmin: addprinc -randkey hbase/fully.qualified.domain.name@YOUR-REALM.COM
```

   where: `fully.qualified.domain.name` is the host where the HBase server is running `YOUR-REALM` is the name of your Kerberos realm

2. Create a keytab file for the HBase server.

```
$ kadmin
kadmin: xst -k hbase.keytab hbase/fully.qualified.domain.name
```

3. Copy the `hbase.keytab` file to the `/etc/hbase/conf` directory on the HBase server host. The owner of the `hbase.keytab` file should be the `hbase` user and the file should have owner-only read permissions. That is, assign the file `0400` permissions and make it owned by `hbase:hbase`.

```
-r--------  1 hbase   hbase     1343 2012-01-09 10:39  hbase.keytab
```

4. To test that the keytab file was created properly, try to obtain Kerberos credentials as the HBase principal using only the keytab file. Substitute your `fully.qualified.domain.name` and realm in the following command:

```
$ kinit -k -t /etc/hbase/conf/hbase.keytab
hbase/fully.qualified.domain.name@YOUR-REALM.COM
```

5. In the `/etc/hbase/conf/hbase-site.xml` configuration file on all of your cluster hosts running the HBase daemon, add the following lines:

```
<property>
  <name>hbase.regionserver.kerberos.principal</name>
  <value>hbase/_HOST@YOUR-REALM.COM</value>
</property>

<property>
  <name>hbase.regionserver.keytab.file</name>
  <value>/etc/hbase/conf/hbase.keytab</value>
</property>

<property>
  <name>hbase.master.kerberos.principal</name>
  <value>hbase/_HOST@YOUR-REALM.COM</value>
</property>

<property>
<name>hbase.master.keytab.file</name>
<value>/etc/hbase/conf/hbase.keytab</value>
</property>
```

> **Important:**
>
> Make sure you change the `/etc/hbase/conf/hbase-site.xml` configuration file on *all* of your cluster hosts that are running the HBase daemon.

Step 2: Configure HBase Servers and Clients to Authenticate with a Secure ZooKeeper

In order to run a secure HBase, you must also use a secure ZooKeeper. To use your secure ZooKeeper, each HBase host machine (Master, Region Server, and client) must have a principal that allows it to authenticate with your secure ZooKeeper ensemble. Note, this HBase section assumes that your secure ZooKeeper is already configured according to the instructions in the ZooKeeper Security Configuration section and **not** managed by HBase.

This HBase section also assumes that you have successfully completed the previous steps, and already have a principal and keytab file created and in place for every HBase server and client.

Configure HBase JVMs (all Masters, Region Servers, and clients) to use JAAS

1. On each host, set up a Java Authentication and Authorization Service (JAAS) by creating a `/etc/hbase/conf/zk-jaas.conf` file that contains the following:

```
   Client {
      com.sun.security.auth.module.Krb5LoginModule required
      useKeyTab=true
      useTicketCache=false
      keyTab="/etc/hbase/conf/hbase.keytab"
      principal="hbase/fully.qualified.domain.name@<YOUR-REALM>";
   };
```

2. Modify the `hbase-env.sh` file on HBase server and client hosts to include the following:

```
   export HBASE_OPTS="$HBASE_OPTS
-Djava.security.auth.login.config=/etc/hbase/conf/zk-jaas.conf"
   export HBASE_MANAGES_ZK=false
```

Configure the HBase Servers (Masters and Region Servers) to use Authentication to connect to ZooKeeper

1. Update your `hbase-site.xml` on each HBase server host with the following properties:

```
<configuration>
   <property>
      <name>hbase.zookeeper.quorum</name>
      <value>$ZK_NODES</value>
   </property>
   <property>
      <name>hbase.cluster.distributed</name>
      <value>true</value>
   </property>
</configuration>
```

   where `$ZK_NODES` is the comma-separated list of hostnames of the ZooKeeper Quorum hosts that you configured according to the instructions in ZooKeeper Security Configuration.

2. Add the following lines to the ZooKeeper configuration file `zoo.cfg`:

```
kerberos.removeHostFromPrincipal=true
kerberos.removeRealmFromPrincipal=true
```

Start HBase

If the configuration worked, you should see something similar to the following in the HBase Master and Region Server logs when you start the cluster:

```
INFO zookeeper.ZooKeeper: Initiating client connection,
connectString=ZK_QUORUM_SERVER:2181 sessionTimeout=180000 watcher=master:60000
INFO zookeeper.ClientCnxn: Opening socket connection to server /ZK_QUORUM_SERVER:2181
INFO zookeeper.RecoverableZooKeeper: The identifier of this process is
PID@ZK_QUORUM_SERVER
INFO zookeeper.Login: successfully logged in.
INFO client.ZooKeeperSaslClient: Client will use GSSAPI as SASL mechanism.
INFO zookeeper.Login: TGT refresh thread started.
INFO zookeeper.ClientCnxn: Socket connection established to ZK_QUORUM_SERVER:2181,
initiating session
INFO zookeeper.Login: TGT valid starting at:        Sun Apr 08 22:43:59 UTC 2012
INFO zookeeper.Login: TGT expires:                  Mon Apr 09 22:43:59 UTC 2012
INFO zookeeper.Login: TGT refresh sleeping until: Mon Apr 09 18:30:37 UTC 2012
INFO zookeeper.ClientCnxn: Session establishment complete on server ZK_QUORUM_SERVER:2181,
 sessionid = 0x134106594320000, negotiated timeout = 180000
```

Configuring Secure HBase Replication

If you are using HBase Replication and you want to make it secure, read this section for instructions. Before proceeding, you should already have configured HBase Replication by following the instructions in the HBase Replication section of the *CDH 5 Installation Guide*.

To configure secure HBase replication, you must configure cross realm support for Kerberos, ZooKeeper, and Hadoop.

To configure secure HBase replication:

1. Create krbtgt principals for the two realms. For example, if you have two realms called `ONE.COM` and `TWO.COM`, you need to add the following principals: `krbtgt/ONE.COM@TWO.COM` and `krbtgt/TWO.COM@ONE.COM`. Add

these two principals at both realms. Note that there must be at least one common encryption mode between these two realms.

```
kadmin: addprinc -e "<enc_type_list>" krbtgt/ONE.COM@TWO.COM
kadmin: addprinc -e "<enc_type_list>" krbtgt/TWO.COM@ONE.COM
```

2. Add rules for creating short names in Zookeeper. To do this, add a system level property in `java.env`, defined in the `conf` directory. Here is an example rule that illustrates how to add support for the realm called `ONE.COM`, and have two members in the principal (such as `service/instance@ONE.COM`):

```
-Dzookeeper.security.auth_to_local=RULE:[2:\$1@\$0](.*@\\QONE.COM\\E$)s/@\\QONE.COM\\E$//DEFAULT
```

The above code example adds support for the `ONE.COM` realm in a different realm. So, in the case of replication, you must add a rule for the master cluster realm in the slave cluster realm. `DEFAULT` is for defining the default rule.

3. Add rules for creating short names in the Hadoop processes. To do this, add the `hadoop.security.auth_to_local` property in the `core-site.xml` file in the slave cluster. For example, to add support for the ONE.COM realm:

```
<property>
  <name>hadoop.security.auth_to_local</name>
  <value>
    RULE:[2:$1@$0](.*@\QONE.COM\E$)s/@\QONE.COM\E$//
  DEFAULT
  </value>
</property>
```

For more information about adding rules, see Appendix C - Configuring the Mapping from Kerberos Principals to Short Names.

Configuring the HBase Client TGT Renewal Period

An HBase client user must also have a Kerberos principal which typically has a password that only the user knows. You should configure the `maxrenewlife` setting for the client's principal to a value that allows the user enough time to finish HBase client processes before the ticket granting ticket (TGT) expires. For example, if the HBase client processes require up to four days to complete, you should create the user's principal and configure the `maxrenewlife` setting by using this command:

```
kadmin: addprinc -maxrenewlife 4days
```

HCatalog Authentication

This section describes how to configure HCatalog in CDH 5 with Kerberos security in a Hadoop cluster:

- Before You Start on page 74
- Step 1: Create the HTTP keytab file on page 75
- Step 2: Configure WebHCat to Use Security on page 75
- Step 3: Create Proxy Users on page 75
- Step 4: Verify the Configuration on page 76

For more information about HCatalog see Installing and Using HCatalog.

Before You Start

Secure Web HCatalog requires a running remote Hive metastore service configured in secure mode. See Hive MetaStoreServer Security Configuration for instructions. Running secure WebHCat with an embedded repository is not supported.

Step 1: Create the HTTP keytab file

You need to create a `keytab` file for WebHCat. Follow these steps:

1. Create the file:

```
kadmin: addprinc -randkey HTTP/fully.qualified.domain.name@YOUR-REALM.COM
kadmin: xst -k HTTP.keytab HTTP/fully.qualified.domain.name
```

2. Move the file into the WebHCat configuration directory and restrict its access exclusively to the `hcatalog` user:

```
$ mv HTTP.keytab /etc/webhcat/conf/
$ chown hcatalog /etc/webhcat/conf/HTTP.keytab
$ chmod 400 /etc/webhcat/conf/HTTP.keytab
```

Step 2: Configure WebHCat to Use Security

Create or edit the WebHCat configuration file `webhcat-site.xml` in the configuration directory and set following properties:

Property	Value
templeton.kerberos.secret	Any random value
templeton.kerberos.keytab	`/etc/webhcat/conf/HTTP.keytab`
templeton.kerberos.principal	HTTP/fully.qualified.domain.name@YOUR-REALM.COM

Example configuration:

```
<property>
    <name>templeton.kerberos.secret</name>
    <value>SuPerS3c3tV@lue!</value>
  </property>

  <property>
    <name>templeton.kerberos.keytab</name>
    <value>/etc/webhcat/conf/HTTP.keytab</value>
  </property>

  <property>
    <name>templeton.kerberos.principal</name>
    <value>HTTP/fully.qualified.domain.name@YOUR-REALM.COM</value>
  </property>
```

Step 3: Create Proxy Users

WebHCat needs access to your NameNode in order to work properly, and so you must configure Hadoop to allow impersonation from the `hcatalog` user. To do this, edit your `core-site.xml` configuration file and set the `hadoop.proxyuser.HTTP.hosts` and `hadoop.proxyuser.HTTP.groups` properties to specify the hosts from which HCatalog can do the impersonation and what users can be impersonated. You can use the value `*` for "any".

Example configuration:

```
  <property>
    <name>hadoop.proxyuser.HTTP.hosts</name>
    <value>*</value>
  </property>
  <property>
    <name>hadoop.proxyuser.HTTP.groups</name>
    <value>*</value>
  </property>
```

Step 4: Verify the Configuration

After restarting WebHcat you can verify that it is working by using `curl` (you may need to run `kinit` first):

```
$ curl --negotiate -i -u :
'http://fully.qualified.domain.name:50111/templeton/v1/ddl/database'
```

Hive Authentication

Here is a summary of the status of Hive security in CDH 5:

- Sentry enables role-based, fine-grained authorization for HiveServer2. See Sentry Policy File Authorization on page 240.
- HiveServer2 supports authentication of the Thrift client using Kerberos or user/password validation backed by LDAP. For configuration instructions, see HiveServer2 Security Configuration.
- Earlier versions of HiveServer do not support Kerberos authentication for clients. However, the Hive MetaStoreServer does support Kerberos authentication for Thrift clients. For configuration instructions, see Hive MetaStoreServer Security Configuration.

See also: Using Hive to Run Queries on a Secure HBase Server on page 82

HiveServer2 Security Configuration

HiveServer2 supports authentication of the Thrift client using either of these methods:

- Kerberos authentication
- LDAP authentication

If Kerberos authentication is used, authentication is supported between the Thrift client and HiveServer2, and between HiveServer2 and secure HDFS. If LDAP authentication is used, authentication is supported only between the Thrift client and HiveServer2. To configure HiveServer2 to use one of these authentication modes, you configure the `hive.server2.authentication` configuration property.

If you want to enable encrypted communication between HiveServer2 and its Client Drivers, see Configuring Encrypted Communication Between Hive and Client Drivers on page 175.

Enabling Kerberos Authentication for HiveServer2

If you configure HiveServer2 to use Kerberos authentication, HiveServer2 acquires a Kerberos ticket during start-up. HiveServer2 requires a principal and keytab file specified in the configuration. The client applications (for example JDBC or Beeline) must get a valid Kerberos ticket before initiating a connection to HiveServer2.

Configuring HiveServer2 for Kerberos-Secured Clusters

To enable Kerberos Authentication for HiveServer2, add the following properties in the `/etc/hive/conf/hive-site.xml` file:

```
<property>
  <name>hive.server2.authentication</name>
  <value>KERBEROS</value>
</property>
<property>
  <name>hive.server2.authentication.kerberos.principal</name>
  <value>hive/_HOST@YOUR-REALM.COM</value>
</property>
<property>
  <name>hive.server2.authentication.kerberos.keytab</name>
  <value>/etc/hive/conf/hive.keytab</value>
</property>
```

where:

- `hive.server2.authentication` in particular, is a client-facing property that controls the type of authentication HiveServer2 uses for connections to clients. In this case, HiveServer2 uses Kerberos to authenticate incoming clients.
- The `_HOST@YOUR-REALM.COM` value in the example above is the Kerberos principal for the host where HiveServer2 is running. The special string _HOST in the properties is replaced at run-time by the fully-qualified domain name of the host machine where the daemon is running. This requires that reverse DNS is properly working on all the hosts configured this way. Replace `YOUR-REALM.COM` with the name of the Kerberos realm your Hadoop cluster is in.
- The `/etc/hive/conf/hive.keytab` value in the example above is a keytab file for that principal.

If you configure HiveServer2 to use both Kerberos authentication and secure impersonation, JDBC clients and Beeline can specify an alternate session user. If these clients have proxy user privileges, HiveServer2 will impersonate the alternate user instead of the one connecting. The alternate user can be specified by the JDBC connection string `proxyUser=`*`userName`*

Configuring JDBC Clients for Kerberos Authentication with HiveServer2

JDBC-based clients must include `principal=<hive.server2.authentication.principal>` in the JDBC connection string. For example:

```
String url =
"jdbc:hive2://node1:10000/default;principal=hive/HiveServer2Host@YOUR-REALM.COM"
Connection con = DriverManager.getConnection(url);
```

where `hive` is the principal configured in `hive-site.xml` and `HiveServer2Host` is the host where HiveServer2 is running.

For ODBC Clients, refer the Cloudera ODBC Driver for Apache Hive documentation.

Using Beeline to Connect to a Secure HiveServer2

Use the following command to start `beeline` and connect to a secure running HiveServer2 process. In this example, the HiveServer2 process is running on `localhost` at port `10000`:

```
$ /usr/lib/hive/bin/beeline
beeline> !connect
jdbc:hive2://localhost:10000/default;principal=hive/HiveServer2Host@YOUR-REALM.COM
0: jdbc:hive2://localhost:10000/default>
```

For more information about the Beeline CLI, see Using the Beeline CLI.

Using LDAP Username/Password Authentication with HiveServer2

As an alternative to Kerberos authentication, you can configure HiveServer2 to use user and password validation backed by LDAP. In this case, the client sends a user name and password during the connection initiation. HiveServer2 validates these credentials using an external LDAP service.

You can enable LDAP Authentication with HiveServer2 using Active Directory or OpenLDAP.

> **Important:** When using LDAP username/password authentication with HiveServer2, make sure you have enabled encrypted communication between HiveServer2 and its client drivers to avoid sending cleartext passwords. For instructions, see Configuring Encrypted Communication Between Hive and Client Drivers on page 175. Also see Configuring LDAPS Authentication with HiveServer2 on page 78.

Enabling LDAP Authentication with HiveServer2 using Active Directory

To enable the LDAP mode of authentication using Active Directory, include the following properties in the `hive-site.xml` file:

```
<property>
  <name>hive.server2.authentication</name>
  <value>LDAP</value>
</property>
<property>
  <name>hive.server2.authentication.ldap.url</name>
  <value>LDAP_URL</value>
</property>
<property>
  <name>hive.server2.authentication.ldap.Domain</name>
  <value>DOMAIN</value>
</property>
```

where:

- The `LDAP_URL` value is the access URL for your LDAP server. For example, `ldap://ldaphost@company.com`.

Enabling LDAP Authentication with HiveServer2 using OpenLDAP

To enable the LDAP mode of authentication using OpenLDAP, include the following properties in the `hive-site.xml` file:

```
<property>
  <name>hive.server2.authentication</name>
  <value>LDAP</value>
</property>
<property>
  <name>hive.server2.authentication.ldap.url</name>
  <value>LDAP_URL</value>
</property>
<property>
  <name>hive.server2.authentication.ldap.baseDN</name>
  <value>LDAP_BaseDN</value>
</property>
```

where:

- The `LDAP_URL` value is the access URL for your LDAP server.
- The `LDAP_BaseDN` value is the base LDAP DN for your LDAP server. For example, `ou=People,dc=example,dc=com`.

Configuring JDBC Clients for LDAP Authentication with HiveServer2

The JDBC client needs to use a connection URL as shown below. -

JDBC-based clients must include `user=LDAP_Userid;password=LDAP_Password` in the JDBC connection string. For example:

```
String url = "jdbc:hive2://node1:10000/default;user=LDAP_Userid;password=LDAP_Password"
Connection con = DriverManager.getConnection(url);
```

where the `LDAP_Userid` value is the user id and `LDAP_Password` is the password of the client user.

For ODBC Clients, refer the Cloudera ODBC Driver for Apache Hive documentation.

Configuring LDAPS Authentication with HiveServer2

HiveServer2 supports LDAP username/password authentication for clients. Clients send LDAP credentials to HiveServer2 which in turn verifies them with the configured LDAP provider such as OpenLDAP or Microsoft's Active Directory. Most vendors now support LDAPS (LDAP over SSL), an authentication protocol that uses SSL to encrypt communication between the LDAP service and its client (in this case, HiveServer2) to avoid sending LDAP credentials in cleartext.

Perform the following steps to configure the LDAPS service with HiveServer2:

- Import either the LDAP server issuing Certificate Authority's SSL certificate into a local truststore, or import the SSL server certificate for a specific trust. If you import the CA certificate, HiveServer2 will trust any server with a certificate issued by the LDAP server's CA. If you only import the SSL certificate for a specific trust, HiveServer2 will trust only that server. In both cases, the SSL certificate must be imported on to the same host as HiveServer2. Refer the keytool documentation for more details.
- Make sure the truststore file is readable by the `hive` user.
- Set the `hive.server2.authentication.ldap.url` configuration property in `hive-site.xml` to the LDAPS URL. For example, `ldaps://sample.myhost.com`.

> **Note:** The URL scheme should be `ldaps` and *not* `ldap`.

- Set the environment variable HADOOP_OPTS as follows. You can refer the Creating Java Keystores and Truststores on page 151 topic for details on adding CA certificates to existing truststores or creating new truststores:

```
HADOOP_OPTS="-Djavax.net.ssl.trustStore=<trustStore-file-path>
-Djavax.net.ssl.trustStorePassword=<trustStore-password>"
```

  For clusters managed by Cloudera Manager, go to the Hive service and select **Configuration > View and Edit**. Under the **HiveServer2** category, go to the **Advanced** section and set the **HiveServer2 Environment Safety Valve** property.
- Restart HiveServer2.

Pluggable Authentication

Pluggable authentication allows you to provide a custom authentication provider for HiveServer2.

To enable pluggable authentication:

1. Set the following properties in `/etc/hive/conf/hive-site.xml`:

```
<property>
  <name>hive.server2.authentication</name>
  <value>CUSTOM</value>
  <description>Client authentication types.
  NONE: no authentication check
  LDAP: LDAP/AD based authentication
  KERBEROS: Kerberos/GSSAPI authentication
  CUSTOM: Custom authentication provider
  (Use with property hive.server2.custom.authentication.class)
  </description>
</property>

<property>
  <name>hive.server2.custom.authentication.class</name>
  <value>pluggable-auth-class-name</value>
  <description>
  Custom authentication class. Used when property
  'hive.server2.authentication' is set to 'CUSTOM'. Provided class
  must be a proper implementation of the interface
  org.apache.hive.service.auth.PasswdAuthenticationProvider. HiveServer2
  will call its Authenticate(user, passed) method to authenticate requests.
  The implementation may optionally extend the Hadoop's
  org.apache.hadoop.conf.Configured class to grab Hive's Configuration object.
  </description>
</property>
```

2. Make the class available in the CLASSPATH of HiveServer2.

Trusted Delegation with HiveServer2

HiveServer2 determines the identity of the connecting user from the underlying authentication subsystem (Kerberos or LDAP). Any new session started for this connection runs on behalf of this connecting user. If the server is configured

to proxy the user at the Hadoop level, then all MapReduce jobs and HDFS accesses will be performed with the identity of the connecting user. If Apache Sentry is configured, then this connecting userid can also be used to verify access rights to underlying tables, views and so on.

In CDH 4.5, a connecting user (for example, hue) with Hadoop-level superuser privileges, can request an alternate user for the given session. HiveServer2 will check if the connecting user has Hadoop-level privileges to proxy the requested userid (for example, bob). If it does, then the new session will be run on behalf of the alternate user, bob, requested by connecting user, hue.

To specify an alternate user for new connections, the JDBC client needs to add the `hive.server2.proxy.user=<alternate_user_id>` property to the JDBC connection URL. Note that the connecting user needs to have Hadoop-level proxy privileges over the alternate user. For example, if user hue requests access to run a session as user bob, the JDBC connection string should be as follows:

```
# Login as super user Hue
kinit hue -k -t hue.keytab hue@MY-REALM.COM

# Connect using following JDBC connection string
#
jdbc:hive2://myHost.myOrg.com:10000/default;principal=hive/_HOST@MY-REALM.COM;hive.server2.proxy.user=bob
```

HiveServer2 Impersonation

> **Note:** This is not the recommended method to implement HiveServer2 impersonation. Cloudera recommends you use Sentry to implement this instead.

Impersonation support in HiveServer2 allows users to execute queries and access HDFS files as the connected user rather than the super user who started the HiveServer2 daemon. Impersonation allows admins to enforce an access policy at the file level using HDFS file and directory permissions.

To enable impersonation in HiveServer2:

1. Add the following property to the `/etc/hive/conf/hive-site.xml` file and set the value to `true`. (The default value is `false`.)

```
<property>
  <name>hive.server2.enable.impersonation</name>
  <description>Enable user impersonation for HiveServer2</description>
  <value>true</value>
</property>
```

2. In HDFS or MapReduce configurations, add the following property to the `core-site.xml` file:

```
<property>
  <name>hadoop.proxyuser.hive.hosts</name>
  <value>*</value>
</property>
<property>
  <name>hadoop.proxyuser.hive.groups</name>
  <value>*</value>
</property>
```

See also File System Permissions.

Securing the Hive Metastore

> **Note:** This is not the recommended method to protect the Hive Metastore. Cloudera recommends you use Sentry to implement this instead.

To prevent users from accessing the Hive metastore and the Hive metastore database using any method other than through HiveServer2, the following actions are recommended:

- Add a firewall rule on the metastore service host to allow access to the metastore port only from the HiveServer2 host. You can do this using iptables.
- Grant access to the metastore database only from the metastore service host. This is specified for MySQL as:

```
GRANT ALL PRIVILEGES ON metastore.* TO 'hive'@'metastorehost';
```

  where `metastorehost` is the host where the metastore service is running.

- Make sure users who are not admins cannot log on to the host on which HiveServer2 runs.

Disabling the Hive Client/Server Authentication

Hive's security related metadata is stored in the configuration file `hive-site.xml`. The following sections describe how to disable security for the Hive service.

Disable Client/Server Authentication

To disable client/server authentication, set `hive.server2.authentication` to `NONE`. For example,

```
<property>
  <name>hive.server2.authentication</name>
  <value>NONE</value>
  <description>
    Client authentication types.
       NONE: no authentication check
       LDAP: LDAP/AD based authentication
       KERBEROS: Kerberos/GSSAPI authentication
       CUSTOM: Custom authentication provider
               (Use with property hive.server2.custom.authentication.class)
  </description>
</property>
```

Disable Hive Metastore security

To disable Hive Metastore security, perform the following steps:

- Set the `hive.metastore.sasl.enabled` property to `false` in all configurations, the metastore service side as well as for all clients of the metastore. For example, these might include HiveServer2, Impala, Pig and so on.
- Remove or comment the following parameters in `hive-site.xml` for the metastore service. Note that this is a server-only change.
  - `hive.metastore.kerberos.keytab.file`
  - `hive.metastore.kerberos.principal`

Disable Underlying Hadoop Security

If you also want to disable the underlying Hadoop security, remove or comment out the following parameters in `hive-site.xml`.

- `hive.server2.authentication.kerberos.keytab`
- `hive.server2.authentication.kerberos.principal`

Hive Metastore Server Security Configuration

> **Important:**
>
> This section describes how to configure security for the Hive metastore server. If you are using HiveServer2, see HiveServer2 Security Configuration.

Here is a summary of Hive metastore server security in CDH 5:

- No additional configuration is required to run Hive on top of a security-enabled Hadoop cluster in standalone mode using a local or embedded metastore.

- HiveServer does not support Kerberos authentication for clients. While it is possible to run HiveServer with a secured Hadoop cluster, doing so creates a security hole since HiveServer does not authenticate the Thrift clients that connect to it. Instead, you can use HiveServer2 HiveServer2 Security Configuration.
- The Hive metastore server supports Kerberos authentication for Thrift clients. For example, you can configure a standalone Hive metastore server instance to force clients to authenticate with Kerberos by setting the following properties in the `hive-site.xml` configuration file used by the metastore server:

```
<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/hive/conf/hive.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@YOUR-REALM.COM</value>
  <description>The service principal for the metastore thrift server. The special string
 _HOST will be replaced automatically with the correct host name.</description>
</property>
```

> **Note:**
>
> The values shown above for the `hive.metastore.kerberos.keytab.file` and `hive.metastore.kerberos.principal` properties are examples which you will need to replace with the appropriate values for your cluster. Also note that the Hive keytab file should have its access permissions set to `600` and be owned by the same account that is used to run the Metastore server, which is the `hive` user by default.

- Requests to access the metadata are fulfilled by the Hive metastore impersonating the requesting user. This includes read access to the list of databases, tables, properties of each table such as their HDFS location, file type and so on. You can restrict access to the Hive metastore service by allowing it to impersonate only a subset of Kerberos users. This can be done by setting the `hadoop.proxyuser.hive.groups` property in `core-site.xml` on the Hive metastore host.

  For example, if you want to give the `hive` user permission to impersonate members of groups `hive` and `user1`:

```
<property>
<name>hadoop.proxyuser.hive.groups</name>
<value>hive,user1</value>
</property>
```

  In this example, the Hive metastore can impersonate users belonging to *only* the `hive` and `user1` groups. Connection requests from users not belonging to these groups will be rejected.

Using Hive to Run Queries on a Secure HBase Server

To use Hive to run queries on a secure HBase Server, you must set the following `HIVE_OPTS` environment variable:

```
env HIVE_OPTS="-hiveconf hbase.security.authentication=kerberos -hiveconf
hbase.master.kerberos.principal=hbase/_HOST@YOUR-REALM.COM -hiveconf
hbase.regionserver.kerberos.principal=hbase/_HOST@YOUR-REALM.COM -hiveconf
hbase.zookeeper.quorum=zookeeper1,zookeeper2,zookeeper3" hive
```

where:

- You replace `YOUR-REALM` with the name of your Kerberos realm
- You replace `zookeeper1`, `zookeeper2`, `zookeeper3` with the names of your ZooKeeper servers. The `hbase.zookeeper.quorum` property is configured in the `hbase-site.xml` file.
- The special string `_HOST` is replaced at run-time by the fully-qualified domain name of the host machine where the HBase Master or Region Server is running. This requires that reverse DNS is properly working on all the hosts configured this way.

In the following, `_HOST` is the name of the host where the HBase Master is running:

```
-hiveconf hbase.master.kerberos.principal=hbase/_HOST@YOUR-REALM.COM
```

In the following, `_HOST` is the hostname of the HBase RegionServer that the application is connecting to:

```
-hiveconf hbase.regionserver.kerberos.principal=hbase/_HOST@YOUR-REALM.COM
```

Note:

You can also set the `HIVE_OPTS` environment variable in your shell profile.

HttpFS Authentication

This section describes how to configure HttpFS CDH 5 with Kerberos security on a Hadoop cluster:

- Configuring the HttpFS Server to Support Kerberos Security on page 83
- Using curl to access an URL Protected by Kerberos HTTP SPNEGO on page 85

For more information about HttpFS, see https://archive.cloudera.com/cdh5/cdh/5/hadoop/hadoop-hdfs-httpfs/index.html.

Important:

To enable HttpFS to work with Kerberos security on your Hadoop cluster, make sure you perform the installation and configuration steps in Configuring Hadoop Security in CDH 5.

Important:

If the NameNode, Secondary NameNode, DataNode, JobTracker, TaskTrackers, ResourceManager, NodeManagers, HttpFS, or Oozie services are configured to use Kerberos HTTP SPNEGO authentication, and two or more of these services are running on the same host, then all of the running services must use the same HTTP principal and keytab file used for their HTTP endpoints.

Configuring the HttpFS Server to Support Kerberos Security

1. Create an HttpFS service user principal that is used to authenticate with the Hadoop cluster. The syntax of the principal is: `httpfs/<fully.qualified.domain.name>@<YOUR-REALM>` where: `fully.qualified.domain.name` is the host where the HttpFS server is running `YOUR-REALM` is the name of your Kerberos realm

```
kadmin: addprinc -randkey httpfs/fully.qualified.domain.name@YOUR-REALM.COM
```

2. Create a HTTP service user principal that is used to authenticate user requests coming to the HttpFS HTTP web-services. The syntax of the principal is: `HTTP/<fully.qualified.domain.name>@<YOUR-REALM>` where:

`'fully.qualified.domain.name'` is the host where the HttpFS server is running `YOUR-REALM` is the name of your Kerberos realm

```
kadmin: addprinc -randkey HTTP/fully.qualified.domain.name@YOUR-REALM.COM
```

> **Important:**
>
> The HTTP/ component of the HTTP service user principal must be upper case as shown in the syntax and example above.

3. Create keytab files with both principals.

```
$ kadmin
kadmin: xst -k httpfs.keytab httpfs/fully.qualified.domain.name
kadmin: xst -k http.keytab HTTP/fully.qualified.domain.name
```

4. Merge the two keytab files into a single keytab file:

```
$ ktutil
ktutil: rkt httpfs.keytab
ktutil: rkt http.keytab
ktutil: wkt httpfs-http.keytab
```

5. Test that credentials in the merged keytab file work. For example:

```
$ klist -e -k -t httpfs-http.keytab
```

6. Copy the `httpfs-http.keytab` file to the HttpFS configuration directory. The owner of the `httpfs-http.keytab` file should be the `httpfs` user and the file should have owner-only read permissions.
7. Edit the HttpFS server `httpfs-site.xml` configuration file in the HttpFS configuration directory by setting the following properties:

Property	Value
httpfs.authentication.type	kerberos
httpfs.hadoop.authentication.type	kerberos
httpfs.authentication.kerberos.principal	HTTP/<HTTPFS-HOSTNAME>@<YOUR-REALM.COM>
httpfs.authentication.kerberos.keytab	/etc/hadoop-httpfs/conf/httpfs-http.keytab
httpfs.hadoop.authentication.kerberos.principal	httpfs/<HTTPFS-HOSTNAME>@<YOUR-REALM.COM>
httpfs.hadoop.authentication.kerberos.keytab	/etc/hadoop-httpfs/conf/httpfs-http.keytab
httpfs.authentication.kerberos.name.rules	Use the value configured for 'hadoop.security.auth_to_local' in 'core-site.xml'

> **Important:**
>
> You must restart the HttpFS server to have the configuration changes take effect.

Using curl to access an URL Protected by Kerberos HTTP SPNEGO

> **Important:**
>
> Your version of `curl` must support GSS and be capable of running `curl -V`.

To configure curl to access an URL protected by Kerberos HTTP SPNEGO:

1. Run `curl -V`:

```
$ curl -V
curl 7.19.7 (universal-apple-darwin10.0) libcurl/7.19.7 OpenSSL/0.9.8l
zlib/1.2.3
Protocols: tftp ftp telnet dict ldap http file https ftps
Features: GSS-Negotiate IPv6 Largefile NTLM SSL libz
```

2. Login to the KDC using `kinit`.

```
$ kinit
Please enter the password for tucu@LOCALHOST:
```

3. Use `curl` to fetch the protected URL:

```
$ curl --negotiate -u : -b ~/cookiejar.txt -c ~/cookiejar.txt
http://localhost:14000/webhdfs/v1/?op=liststatus
```

where: The `--negotiate` option enables SPNEGO in `curl`. The `-u :` option is required but the user name is ignored (the principal that has been specified for `kinit` is used). The `-b` and `-c` options are used to store and send HTTP cookies.

Hue Authentication

The following sections describe how to configure Hue CDH 5 with Kerberos security, enabling single sign-on with SAML and encrypting communication between Hue and other services among other available configuration settings.

- Hue Security Enhancements on page 85
- Configuring Kerberos Authentication for Hue on page 87
- Integrating Hue with LDAP on page 89
- Configuring Hue for SAML on page 93

> **Important:**
>
> To enable Hue to work with Kerberos security on your Hadoop cluster, make sure you perform the installation and configuration steps in Configuring Hadoop Security in CDH 5.

Hue Security Enhancements

Enabling SSL Communication with HiveServer2

By providing a CA certificate, private key, and public certificate, Hue can communicate with HiveServer2 over SSL. You can now configure the following properties in the `[beeswax]` section under `[[ssl]]` in the Hue configuration file, `hue.ini`.

`enabled`	Choose to enable/disable SSL communication for this server. Default: `false`
`cacerts`	Path to Certificate Authority certificates.

	Default: `/etc/hue/cacerts.pem`
`key`	Path to the private key file. Default: `/etc/hue/key.pem`
`cert`	Path to the public certificate file. Default: `/etc/hue/cert.pem`
`validate`	Choose whether Hue should validate certificates received from the server. Default: `true`

Enabling LDAP Authentication with HiveServer2 and Impala

LDAP authentication with HiveServer2 and Impala can be enabled by setting the following properties under the `[desktop]` section in `hue.ini`.

`ldap_username`	LDAP username of the Hue user to be authenticated.
`ldap_password`	LDAP password for the Hue user to be authenticated.

Secure Database Connection

Connections vary depending on the database. Hue uses different clients to communicate with each database internally. They all specify a common interface known as the DBAPI version 2 interface. Client specific options, such as secure connectivity, can be passed through the interface. For example, for MySQL you can enable SSL communication by specifying the `options` configuration property under the `desktop>[[database]]` section in `hue.ini`.

```
[desktop]
  [[databases]]
    …
    options={"ssl":{"ca":"/tmp/ca-cert.pem"}}
```

Session Timeout

Session timeouts can be set by specifying the `ttl` configuration property under the `[desktop]>[[session]]` section in `hue.ini`.

`ttl`	The cookie containing the users' session ID will expire after this amount of time in seconds. Default: `60*60*24*14`

Secure Cookies

Secure session cookies can be enable by specifying the `secure` configuration property under the `[desktop]>[[session]]` section in `hue.ini`. Additionally, you can set the `http-only` flag for cookies containing users' session IDs.

`secure`	The cookie containing the users' session ID will be secure. Should only be enabled with HTTPS. Default: `false`
`http-only`	The cookie containing the users' session ID will use the HTTP only flag. Default: `false`

Allowed HTTP Methods

You can specify the HTTP request methods that the server should respond to using the `http_allowed_methods` property under the `[desktop]` section in `hue.ini`.

`http_allowed_methods`	Default: `options,get,head,post,put,delete,connect`

Restricting the Cipher List

Cipher list support with HTTPS can be restricted by specifying the `ssl_cipher_list` configuration property under the `[desktop]` section in `hue.ini`.

`ssl_cipher_list`	Default: `!aNULL:!eNULL:!LOW:!EXPORT:!SSLv2`

URL Redirect Whitelist

Restrict the domains or pages to which Hue can redirect users. The `redirect_whitelist` property can be found under the `[desktop]` section in `hue.ini`.

`redirect_whitelist`	For example, to restrict users to your local domain and FQDN, the following value can be used: `^\/.*$,^http:\/\/www.mydomain.com\/.*$`

Configuring Kerberos Authentication for Hue

You can configure Hue in CDH 5 to support Hadoop security on a cluster using Kerberos.

To configure the Hue server to support Hadoop security using Kerberos:

1. Create a Hue user principal in the same realm as the Hadoop cluster of the form:

```
kadmin: addprinc -randkey hue/hue.server.fully.qualified.domain.name@YOUR-REALM.COM
```

   where: hue is the principal the Hue server is running as, `hue.server.fully.qualified.domain.name` is the fully-qualified domain name (FQDN) of your Hue server, `YOUR-REALM.COM` is the name of the Kerberos realm your Hadoop cluster is in
2. Create a keytab file for the Hue principal using the same procedure that you used to create the keytab for the `hdfs` or `mapred` principal for a specific host. You should name this file `hue.keytab` and put this keytab file in the directory `/etc/hue` on the machine running the Hue server. Like all keytab files, this file should have the most limited set of permissions possible. It should be owned by the user running the hue server (usually hue) and should have the permission `400`.
3. To test that the keytab file was created properly, try to obtain Kerberos credentials as the Hue principal using only the keytab file. Substitute your FQDN and realm in the following command:

```
$ kinit -k -t /etc/hue/hue.keytab
hue/hue.server.fully.qualified.domain.name@YOUR-REALM.COM
```

4. In the `/etc/hue/hue.ini` configuration file, add the following lines in the sections shown. Replace the `kinit_path` value, `/usr/kerberos/bin/kinit`, shown below with the correct path on the user's system.

```
[desktop]

 [[kerberos]]
 # Path to Hue's Kerberos keytab file
 hue_keytab=/etc/hue/hue.keytab
 # Kerberos principal name for Hue
 hue_principal=hue/FQDN@REALM
 # add kinit path for non root users
 kinit_path=/usr/kerberos/bin/kinit
```

```
[beeswax]
 # If Kerberos security is enabled, use fully-qualified domain name (FQDN)
   ## hive_server_host=<FQDN of Hive Server>
 # Hive configuration directory, where hive-site.xml is located
   ## hive_conf_dir=/etc/hive/conf

[impala]
   ## server_host=localhost
 # The following property is required when impalad and Hue
 # are not running on the same host
   ## impala_principal=impala/impalad.hostname.domainname.com

[search]
 # URL of the Solr Server
   ## solr_url=http://localhost:8983/solr/
 # Requires FQDN in solr_url if enabled
   ## security_enabled=false

[hadoop]

 [[hdfs_clusters]]

 [[[default]]]
 # Enter the host and port on which you are running the Hadoop NameNode
 namenode_host=FQDN
 hdfs_port=8020
 http_port=50070
 security_enabled=true

 # Thrift plugin port for the name node
   ## thrift_port=10090

 # Configuration for YARN (MR2)
 # ------------------------------------------------------------------------
 [[yarn_clusters]]

 [[[default]]]
 # Enter the host on which you are running the ResourceManager
   ## resourcemanager_host=localhost
 # Change this if your YARN cluster is Kerberos-secured
   ## security_enabled=false

 # Thrift plug-in port for the JobTracker
   ## thrift_port=9290

[liboozie]
 # The URL where the Oozie service runs on. This is required in order for users to submit
 jobs.
   ## oozie_url=http://localhost:11000/oozie
 # Requires FQDN in oozie_url if enabled
   ## security_enabled=false
```

Important:

In the `/etc/hue/hue.ini` file, verify the following:

— Make sure the `jobtracker_host` property is set to the fully-qualified domain name of the host running the JobTracker. The JobTracker hostname must be fully-qualified in a secured environment.

— Make sure the `fs.defaultfs` property under each [[hdfs_clusters]] section contains the fully-qualified domain name of the file system access point, which is typically the NameNode.

— Make sure the `hive_conf_dir` property under the `[beeswax]` section points to a directory containing a valid `hive-site.xml` (either the original or a synced copy).

— Make sure the FQDN specified for HiveServer2 is the same as the FQDN specified for the `hue_principal` configuration property. Without this, HiveServer2 will not work with security enabled.

5. In the `/etc/hadoop/conf/core-site.xml` configuration file on all of your cluster nodes, add the following lines:

```
<!-- Hue security configuration -->
<property>
  <name>hue.kerberos.principal.shortname</name>
  <value>hue</value>
</property>
<property>
  <name>hadoop.proxyuser.hue.groups</name>
  <value>*</value> <!-- A group which all users of Hue belong to, or the wildcard value
 "*" -->
</property>
<property>
  <name>hadoop.proxyuser.hue.hosts</name>
  <value>hue.server.fully.qualified.domain.name</value>
</property>
```

> **Important:**
>
> Make sure you change the `/etc/hadoop/conf/core-site.xml` configuration file on *all* of your cluster nodes.

6. If Hue is configured to communicate to Hadoop using HttpFS, then you must add the following properties to `httpfs-site.xml`:

```
<property>
  <name>httpfs.proxyuser.hue.hosts</name>
  <value>fully.qualified.domain.name</value>
</property>
<property>
  <name>httpfs.proxyuser.hue.groups</name>
  <value>*</value>
</property>
```

7. Add the following properties to the Oozie server `oozie-site.xml` configuration file in the Oozie configuration directory:

```
<property>
   <name>oozie.service.ProxyUserService.proxyuser.hue.hosts</name>
   <value>*</value>
</property>
<property>
   <name>oozie.service.ProxyUserService.proxyuser.hue.groups</name>
   <value>*</value>
</property>
```

8. Restart the JobTracker to load the changes from the `core-site.xml` file.

```
$ sudo service hadoop-0.20-mapreduce-jobtracker restart
```

9. Restart Oozie to load the changes from the `oozie-site.xml` file.

```
$ sudo service oozie restart
```

10. Restart the NameNode, JobTracker, and all DataNodes to load the changes from the `core-site.xml` file.

```
$ sudo service hadoop-0.20-(namenode|jobtracker|datanode) restart
```

Integrating Hue with LDAP

When Hue is integrated with LDAP users can use their existing credentials to authenticate and inherit their existing groups transparently. There is no need to save or duplicate any employee password in Hue. There are several other

ways to authenticate with Hue such as PAM, SPNEGO, OpenID, OAuth, SAML2 and so on. This topic details how you can configure Hue to authenticate against an LDAP directory server.

When authenticating using LDAP, Hue validates login credentials against an LDAP directory service if configured with the LDAP authentication backend:

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

The LDAP authentication backend will automatically create users that don't exist in Hue by default. Hue needs to import users in order to properly perform the authentication. Passwords are never imported when importing users. If you want to disable automatic import set the `create_users_on_login` property under the `[desktop] > [[ldap]]` section of `hue.ini` to `false`.

```
[desktop]
[[ldap]]
create_users_on_login=false
```

The purpose of disabling the automatic import is to allow only a predefined list of manually imported users to login.

There are two ways to authenticate with a directory service through Hue:

- Search Bind
- Direct Bind

You can specify the authentication mechanism using the `search_bind_authentication` property under the `[desktop] > [[ldap]]` section of `hue.ini`.

`search_bind_authentication`	Uses search bind authentication by default. Set this property to `false` to use direct bind authentication. Default: `true`

Search Bind

The search bind mechanism for authenticating will perform an `ldapsearch` against the directory service and bind using the found distinguished name (DN) and password provided. This is the default method of authentication used by Hue with LDAP.

The following configuration properties under the `[desktop] > [[ldap]] > [[[users]]]` section in `hue.ini` can be set to restrict the search process.

`user_filter`	General LDAP filter to restrict the search. Default: `"objectclass=*"`
`user_name_attr`	The attribute that will be considered the username to be searched against. Typical attributes to search for include: `uid`, `sAMAccountName`. Default: `sAMAccountName`

With the above configuration, the LDAP search filter will take on the form:

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

> **Important:** Setting `search_bind_authentication=true` in `hue.ini` tells Hue to perform an LDAP search using the bind credentials specified for the `bind_dn` and `bind_password` configuration properties. Hue will start searching the subtree starting from the base DN specified for the `base_dn` property. It will then search the base DN for an entry whose attribute, specified in `user_name_attr`, has the same value as the short name provided on login. The search filter, defined in `user_filter` will also be used to limit the search.

Direct Bind

The direct bind mechanism for authenticating will bind to the LDAP server using the username and password provided at login.

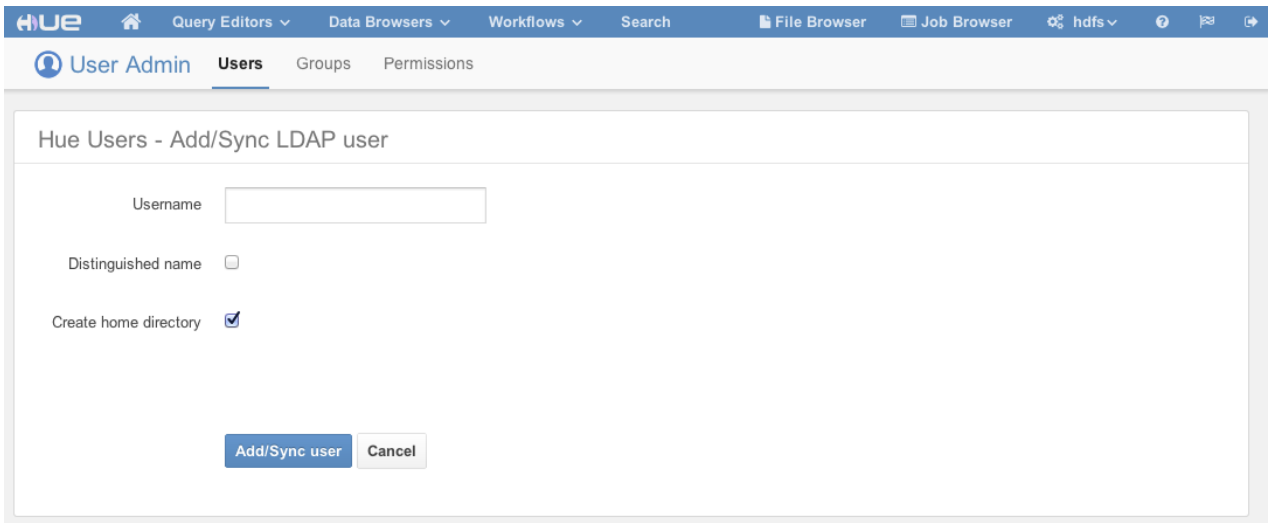
The following configuration properties can be used to determine how Hue binds to the LDAP server. These can be set under the `[desktop] > [[ldap]]` section of `hue.ini`.

`nt_domain`	The NT domain to connect to (only for use with Active Directory). This AD-specific property allows Hue to authenticate with AD without having to follow LDAP references to other partitions. This typically maps to the email address of the user or the user's ID in conjunction with the domain. If provided, Hue will use User Principal Names (UPNs) to bind to the LDAP service. Default: `mycompany.com`
`ldap_username_pattern`	Provides a template for the DN that will ultimately be sent to the directory service when authenticating. The `<username>` parameter will be replaced with the username provided at login. Default: `"uid=<username>,ou=People,dc=mycompany,dc=com"`

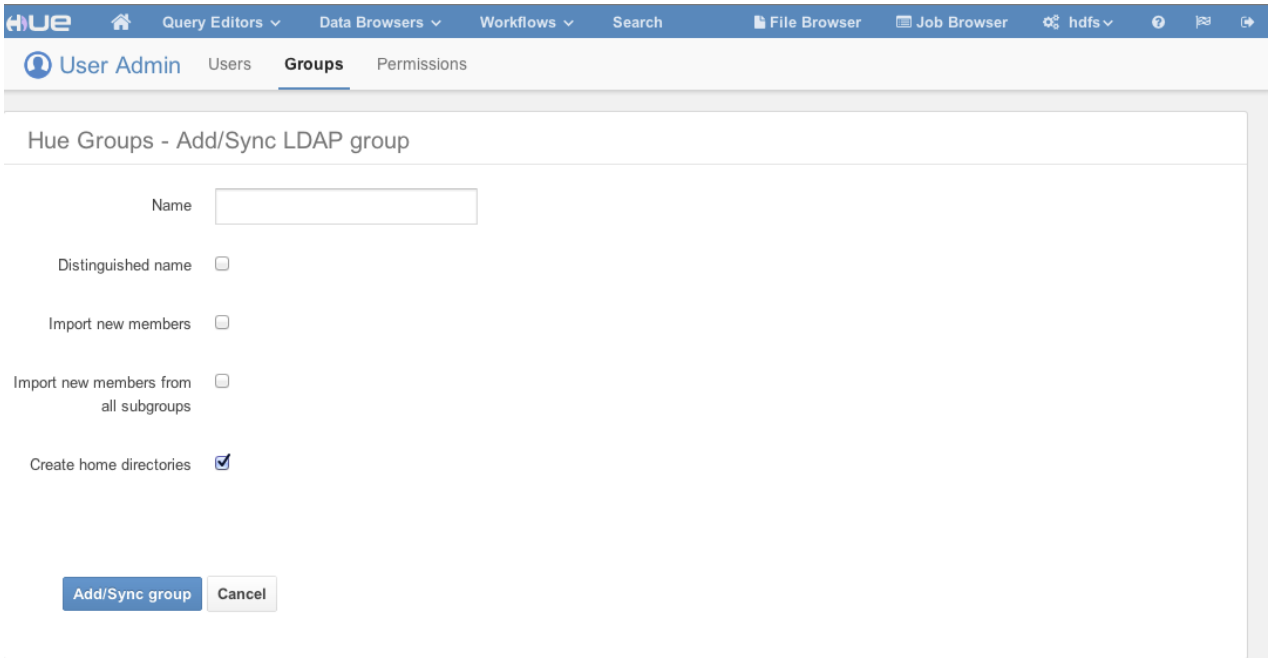
> **Important:** Setting `search_bind_authentication=false` in `hue.ini` tells Hue to perform a direct bind to LDAP using the credentials provided (*not* `bind_dn` and `bind_password` specified in `hue.ini`). There are two ways direct bind works depending on whether the `nt_domain` property is specified in `hue.ini`:
>
> - `nt_domain` is specified: This is used to connect to an Active Directory service. In this case, the User Principal Name (UPN) is used to perform a direct bind. Hue forms the UPN by concatenating the short name provided at login with the `nt_domain`. For example, `<short name>@<nt_domain>`. The `ldap_username_pattern` property is ignored.
> - `nt_domain` is *not* specified: This is used to connect to all other directory services (can handle Active Directory, but `nt_domain` is the preferred way for AD). In this case, `ldap_username_pattern` is used and it should take on the form `cn=<username>,dc=example,dc=com` where `<username>` will be replaced with the username provided at login.

Importing LDAP Users and Groups

If an LDAP user needs to be part of a certain group and be given a particular set of permissions, you can import this user with the User Admin interface in Hue.



Groups can also be imported using the User Admin interface, and users can be added to this group. As in the image below, not only can groups be discovered using DN and rDN search, but users that are members of the group or members of its subordinate groups can be imported as well.



You have the following options available when importing a user/group:

- **Distinguished name:** If checked, the username provided must be a full distinguished name (for example, `uid=hue,ou=People,dc=gethue,dc=com`). Otherwise, the **Username** provided should be a fragment of a Relative Distinguished Name (rDN) (for example, the username hue maps to the rDN `uid=hue`). Hue will perform an LDAP search using the same methods and configurations as described above. That is, Hue will take the provided username and create a search filter using the `user_filter` and `user_name_attr` configurations.
- **Create home directory:** If checked, when the user is imported, their home directory in HDFS will automatically be created if it doesn't already exist.

> **Important:** When managing LDAP entries, the User Admin app will always perform an LDAP search and will always use `bind_dn`, `bind_password`, `base_dn`, as defined in `hue.ini`.

Synchronizing LDAP Users and Groups

Users and groups can be synchronized with the directory service using the User Admin interface or using a command line utility. The image from the **Importing LDAP Users and Groups** section uses the words **Add/Sync** to indicate that when a user or group that already exists in Hue is being added, it will in fact be synchronized instead. In the case of importing users for a particular group, new users will be imported and existing users will be synchronized.

> **Note:** Users that have been deleted from the directory service will not be deleted from Hue. Those users can be manually deactivated from Hue using the User Admin interface.

Attributes Synchronized

Currently, only the first name, last name, and email address are synchronized. Hue looks for the LDAP attributes `givenName`, `sn`, and `mail` when synchronizing. The `user_name_attr` configuration property is used to appropriately choose the username in Hue. For instance, if `user_name_attr'` is set to `uid`", then the "uid" returned by the directory service will be used as the username of the user in Hue.

User Admin interface

The **Sync LDAP users/groups** button in the User Admin interface will automatically synchronize all users and groups.

Synchronize Using a Command-Line Interface

For example, to synchronize users and groups using a command-line interface:

```
<hue root>/build/env/bin/hue sync_ldap_users_and_groups
```

LDAPS/StartTLS support

Secure communication with LDAP is provided using the SSL/TLS and StartTLS protocols. They allow Hue to validate the directory service it is going to converse with. Hence, if a Certificate Authority certificate file is provided, Hue will communicate using LDAPS. You can specify the path to the CA certificate under :

```
[desktop]
    [[ldap]]
    ldap_cert=/etc/hue/ca.crt
```

The StartTLS protocol can be used as well:

```
[desktop]
    [[ldap]]
    use_start_tls=true
```

Configuring Hue for SAML

This section describes the configuration changes required to use Hue with SAML 2.0 (Security Assertion Markup Language) to enable single sign-on (SSO) authentication.

The SAML 2.0 Web Browser SSO profile has three components: a *Security Provider*, a *User Agent* and an *Identity Provider.* In this case, Hue is the Service Provider (SP), you can use an Identity Provider (IdP) of your choice, and you are the user acting through your browser (User Agent). When a user requests access to an application, Hue uses your browser to send an authentication request to the Identity Provider which then authenticates the user and redirects them back to Hue .

This blog post guides users through setting up SSO with Hue, using the SAML backend and Shibboleth as the Identity Provider.

> **Note:** The following instructions assume you already have an Identity Provider set up and running.

Step 1: Install swig and openssl packages

Install `swig` and `openssl`. For example, on RHEL systems, use the following commands:

```
yum install swig
```

```
yum install openssl
```

Step 2: Install libraries to support SAML in Hue

Install the `djangosaml2` and `pysaml2` libraries to support SAML in Hue. These libraries are dependent on the `xmlsec1` package to be installed and available on the machine for Hue to use. Follow these instructions to install the `xmlsec1` package on your system.

RHEL, CentOS and SLES:

For RHEL, CentOS and SLES systems, the `xmlsec1` package is available for download from the EPEL repository. In order to install packages from the EPEL repository, first download the appropriate the rpm package to your machine, substituting the version in the package URL with the one required for your system. For example, use the following commands for CentOS 5 or RHEL 5:

```
rpm -Uvh http://download.fedoraproject.org/pub/epel/5/i386/epel-release-5-4.noarch.rpm

yum install xmlsec1
```

Oracle Linux:

For Oracle Linux systems, download the `xmlsec1` package from http://www.aleksey.com/xmlsec/ and execute the following commands:

```
tar -xvzf xmlsec1-<version>.tar.gz
cd xmlsec1-<version>
./configure && make
sudo make install
```

> **Important:** The `xmlsec1` package must be executable by the user running Hue.

You should now be able to install `djangosaml` and `pysaml2` on your machines.

```
build/env/bin/pip install -e git+https://github.com/abec/pysaml2@HEAD#egg=pysaml2
build/env/bin/pip install -e git+https://github.com/abec/djangosaml2@HEAD#egg=djangosaml2
```

Step 3: Update the Hue configuration file

Several configuration parameters need to be updated in Hue's configuration file, `hue.ini` to enable support for SAML. The table given below describes the available parameters for SAML in `hue.ini` under the `[libsaml]` section.

Parameter	Description
`xmlsec_binary`	This is a path to the `xmlsec_binary`, an executable used to sign, verify, encrypt and decrypt SAML requests and assertions. This program should be executable by the user running Hue.
`create_users_on_login`	Create Hue users received in assertion response upon successful login. The value for this parameter can be either "true" or "false".
`required_attributes`	Attributes Hue asks for from the IdP. This is a comma-separated list of attributes. For example, `uid`, `email` and so on.
`optional_attributes`	Optional attributes Hue can ask for from the IdP. Also a comma-separated list of attributes.

Parameter	Description
`metadata_file`	This is a path to the IdP metadata copied to a local file. This file should be readable.
`key_file`	Path to the private key used to encrypt the metadata. File format .PEM
`cert_file`	Path to the X.509 certificate to be sent along with the encrypted metadata. File format .PEM
`user_attribute_mapping`	Mapping from attributes received from the IdP to the Hue's django user attributes. For example, {'uid':'username', 'email':'email'}.
`logout_requests_signed`	Have Hue initiated logout requests be signed and provide a certificate.

Step 3a: Update the SAML metadata file

Update the metadata file pointed to by your Hue configuration file, `hue.ini`. Check your IdP documentation for details on how to procure the XML metadata and paste it into the `<metadata_file_name>.xml` file at the location specified by the configuration parameter `metadata_file`.

For example, if you were using the Shibboleth IdP, you would visit `https://<IdPHOST>:8443/idp/shibboleth`, copy the metadata content available there and paste it into the Hue metadata file.

> **Note:**
>
> You may have to edit the content copied over from your IdP's metadata file in case of missing fields such as port numbers (8443), from URLs that point to the IdP.

Step 3b: Private key and certificate files

To enable Hue to communicate with the IdP, you will need to specify the location of a private key, for the, `key_file` property, that can be used to sign requests sent to the IdP. You will also need to specify the location of the certificate file, for the `cert_pem` property, which you will use to verify and decrypt messages from the IdP.

> **Note:** The key and certificate files specified by the `key_file` and `cert_file` parameters must be .PEM files.

Step 3c: Configure Hue to use SAML Backend

To enable SAML to allow user logins and create users, update the `backend` configuration property in `hue.ini` to use the SAML authentication backend. You will find the `backend` property in the `[[auth]]` sub-section under `[desktop]`.

```
backend=libsaml.backend.SAML2Backend
```

Here is an example configuration of the `[libsaml]` section from `hue.ini`.

```
xmlsec_binary=/usr/local/bin/xmlsec1
create_users_on_login=true
metadata_file=/etc/hue/saml/metadata.xml
key_file=/etc/hue/saml/key.pem
cert_file=/etc/hue/saml/cert.pem
logout_requests_signed=true
```

Step 4: Restart the Hue server

Use the following command to restart the Hue server.

```
sudo service hue restart
```

Impala Authentication

Authentication is the mechanism to ensure that only specified hosts and users can connect to Impala. It also verifies that when clients connect to Impala, they are connected to a legitimate server. This feature prevents spoofing such as ***impersonation*** (setting up a phony client system with the same account and group names as a legitimate user) and ***man-in-the-middle attacks*** (intercepting application requests before they reach Impala and eavesdropping on sensitive information in the requests or the results).

Impala supports authentication using either Kerberos or LDAP.

Impala currently does not support application data wire encryption.

> **Note:** Regardless of the authentication mechanism used, Impala always creates HDFS directories and data files owned by the same user (typically `impala`). To implement user-level access to different databases, tables, columns, partitions, and so on, use the Sentry authorization feature, as explained in Enabling Sentry Authorization for Impala on page 257.

Once you are finished setting up authentication, move on to authorization, which involves specifying what databases, tables, HDFS directories, and so on can be accessed by particular users when they connect through Impala. See Enabling Sentry Authorization for Impala on page 257 for details.

Enabling Kerberos Authentication for Impala

Impala supports Kerberos authentication. For more information on enabling Kerberos authentication, see the topic on Configuring Hadoop Security in the CDH 5 Security Guide.

Impala currently does not support application data wire encryption.

When using Impala in a managed environment, Cloudera Manager automatically completes Kerberos configuration. In an unmanaged environment, create a Kerberos principal for each host running `impalad` or `statestored`. Cloudera recommends using a consistent format, such as `impala/_HOST@Your-Realm`, but you can use any three-part Kerberos server principal.

In Impala 2.0 and later, `user()` returns the full Kerberos principal string, such as `user@example.com`, in a Kerberized environment.

> **Note:** Regardless of the authentication mechanism used, Impala always creates HDFS directories and data files owned by the same user (typically `impala`). To implement user-level access to different databases, tables, columns, partitions, and so on, use the Sentry authorization feature, as explained in Enabling Sentry Authorization for Impala on page 257.

An alternative form of authentication you can use is LDAP, described in Enabling LDAP Authentication for Impala on page 99.

Requirements for Using Impala with Kerberos

On version 5 of Red Hat Enterprise Linux and comparable distributions, some additional setup is needed for the `impala-shell` interpreter to connect to a Kerberos-enabled Impala cluster:

```
sudo yum install python-devel openssl-devel python-pip
sudo pip-python install ssl
```

Important:

- If you plan to use Impala in your cluster, you must configure your KDC to allow tickets to be renewed, and you must configure `krb5.conf` to request renewable tickets. Typically, you can do this by adding the `max_renewable_life` setting to your realm in `kdc.conf`, and by adding the `renew_lifetime` parameter to the `libdefaults` section of `krb5.conf`.

  For more information about renewable tickets, see the Kerberos documentation.

- The Impala Web UI does not support Kerberos authentication.
- You cannot use the Impala resource management feature on a cluster that has Kerberos authentication enabled.

Start all `impalad` and `statestored` daemons with the `--principal` and `--keytab-file` flags set to the principal and full path name of the `keytab` file containing the credentials for the principal.

Impala supports the Cloudera ODBC driver and the Kerberos interface provided. To use Kerberos through the ODBC driver, the host type must be set depending on the level of the ODBD driver:

- `SecImpala` for the ODBC 1.0 driver.
- `SecBeeswax` for the ODBC 1.2 driver.
- Blank for the ODBC 2.0 driver or higher, when connecting to a secure cluster.
- `HS2NoSasl` for the ODBC 2.0 driver or higher, when connecting to a non-secure cluster.

To enable Kerberos in the Impala shell, start the `impala-shell` command using the `-k` flag.

To enable Impala to work with Kerberos security on your Hadoop cluster, make sure you perform the installation and configuration steps in Authentication in the CDH 5 Security Guide.

Configuring Impala to Support Kerberos Security

Enabling Kerberos authentication for Impala involves steps that can be summarized as follows:

- Creating service principals for Impala and the HTTP service. Principal names take the form: *`serviceName/fully.qualified.domain.name@KERBEROS.REALM`*
- Creating, merging, and distributing key tab files for these principals.
- Editing `/etc/default/impala` (in cluster not managed by Cloudera Manager), or editing the **Security** settings in the Cloudera Manager interface, to accommodate Kerberos authentication.

Enabling Kerberos for Impala

1. Create an Impala service principal, specifying the name of the OS user that the Impala daemons run under, the fully qualified domain name of each node running `impalad`, and the realm name. For example:

```
$ kadmin
kadmin: addprinc -requires_preauth -randkey
impala/impala_host.example.com@TEST.EXAMPLE.COM
```

2. Create an HTTP service principal. For example:

```
kadmin: addprinc -randkey HTTP/impala_host.example.com@TEST.EXAMPLE.COM
```

Note: The `HTTP` component of the service principal must be uppercase as shown in the preceding example.

3. Create keytab files with both principals. For example:

```
kadmin: xst -k impala.keytab impala/impala_host.example.com
kadmin: xst -k http.keytab HTTP/impala_host.example.com
kadmin: quit
```

4. Use ktutil to read the contents of the two keytab files and then write those contents to a new file. For example:

```
$ ktutil
ktutil: rkt impala.keytab
ktutil: rkt http.keytab
ktutil: wkt impala-http.keytab
ktutil: quit
```

5. (Optional) Test that credentials in the merged keytab file are valid, and that the "renew until" date is in the future. For example:

```
$ klist -e -k -t impala-http.keytab
```

6. Copy the impala-http.keytab file to the Impala configuration directory. Change the permissions to be only read for the file owner and change the file owner to the impala user. By default, the Impala user and group are both named impala. For example:

```
$ cp impala-http.keytab /etc/impala/conf
$ cd /etc/impala/conf
$ chmod 400 impala-http.keytab
$ chown impala:impala impala-http.keytab
```

7. Add Kerberos options to the Impala defaults file, /etc/default/impala. Add the options for both the impalad and statestored daemons, using the IMPALA_SERVER_ARGS and IMPALA_STATE_STORE_ARGS variables. For example, you might add:

```
-kerberos_reinit_interval=60
-principal=impala_1/impala_host.example.com@TEST.EXAMPLE.COM
-keytab_file=/var/run/cloudera-scm-agent/process/3212-impala-IMPALAD/impala.keytab
```

For more information on changing the Impala defaults specified in /etc/default/impala, see Modifying Impala Startup Options.

Note: Restart impalad and statestored for these configuration changes to take effect.

Enabling Kerberos for Impala with a Proxy Server

A common configuration for Impala with High Availability is to use a proxy server to submit requests to the actual impalad daemons on different hosts in the cluster. This configuration avoids connection problems in case of machine failure, because the proxy server can route new requests through one of the remaining hosts in the cluster. This configuration also helps with load balancing, because the additional overhead of being the "coordinator node" for each query is spread across multiple hosts.

Although you can set up a proxy server with or without Kerberos authentication, typically users set up a secure Kerberized configuration. For information about setting up a proxy server for Impala, including Kerberos-specific steps, see Using Impala through a Proxy for High Availability.

Enabling Impala Delegation for Kerberos Users

See Configuring Impala Delegation for Hue and BI Tools on page 101 for details about the delegation feature that lets certain users submit queries using the credentials of other users.

Using TLS/SSL with Business Intelligence Tools

You can use Kerberos authentication, TLS/SSL encryption, or both to secure connections from JDBC and ODBC applications to Impala. See Configuring Impala to Work with JDBC and Configuring Impala to Work with ODBC for details.

Currently, the Hive JDBC driver does not support connections that use both Kerberos authentication and SSL encryption. To use both of these security features with Impala through a JDBC application, use the Cloudera JDBC Connector as the JDBC driver.

Enabling LDAP Authentication for Impala

Authentication is the process of allowing only specified named users to access the server (in this case, the Impala server). This feature is crucial for any production deployment, to prevent misuse, tampering, or excessive load on the server. Impala users LDAP for authentication, verifying the credentials of each user who connects through `impala-shell`, Hue, a Business Intelligence tool, JDBC or ODBC application, and so on.

> **Note:** Regardless of the authentication mechanism used, Impala always creates HDFS directories and data files owned by the same user (typically `impala`). To implement user-level access to different databases, tables, columns, partitions, and so on, use the Sentry authorization feature, as explained in Enabling Sentry Authorization for Impala on page 257.

An alternative form of authentication you can use is Kerberos, described in Enabling Kerberos Authentication for Impala on page 96.

Requirements for Using Impala with LDAP

Authentication against LDAP servers is available in Impala 1.2.2 and higher. Impala 1.4.0 adds support for secure LDAP authentication through SSL and TLS.

The Impala LDAP support lets you use Impala with systems such as Active Directory that use LDAP behind the scenes.

Kerberos Authentication for Connections Between Impala Components

Only client->Impala connections can be authenticated by LDAP.

You must use the Kerberos authentication mechanism for connections between internal Impala components, such as between the `impalad`, `statestored`, and `catalogd` daemons. See Enabling Kerberos Authentication for Impala on page 96 on how to set up Kerberos for Impala.

Server-Side LDAP Setup

These requirements apply on the server side when configuring and starting Impala:

To enable LDAP authentication, set the following startup options for `impalad`:

- `--enable_ldap_auth` enables LDAP-based authentication between the client and Impala.
- `--ldap_uri` sets the URI of the LDAP server to use. Typically, the URI is prefixed with `ldap://`. In Impala 1.4.0 and higher, you can specify secure SSL-based LDAP transport by using the prefix `ldaps://`. The URI can optionally specify the port, for example: `ldap://ldap_server.cloudera.com:389` or `ldaps://ldap_server.cloudera.com:636`. (389 and 636 are the default ports for non-SSL and SSL LDAP connections, respectively.)
- For `ldaps://` connections secured by SSL, `--ldap_ca_certificate="`*`/path/to/certificate/pem`*`"` specifies the location of the certificate in standard `.PEM` format. Store this certificate on the local filesystem, in a location that only the `impala` user and other trusted users can read.

Support for Custom Bind Strings

When Impala connects to LDAP it issues a bind call to the LDAP server to authenticate as the connected user. Impala clients, including the Impala shell, provide the short name of the user to Impala. This is necessary so that Impala can use Sentry for role-based access, which uses short names.

However, LDAP servers often require more complex, structured usernames for authentication. Impala supports three ways of transforming the short name (for example, `'henry'`) to a more complicated string. If necessary, specify one of the following configuration options when starting the `impalad` daemon on each data node:

- `--ldap_domain`: Replaces the username with a string *`username@ldap_domain`*.
- `--ldap_baseDN`: Replaces the username with a "distinguished name" (DN) of the form: `uid=`*`userid`*`, ldap_baseDN`. (This is equivalent to a Hive option).
- `--ldap_bind_pattern`: This is the most general option, and replaces the username with the string *ldap_bind_pattern* where all instances of the string `#UID` are replaced with *userid*. For example, an `ldap_bind_pattern` of `"user=#UID,OU=foo,CN=bar"` with a username of `henry` will construct a bind name of `"user=henry,OU=foo,CN=bar"`.

For clusters not managed by Cloudera Manager, specify the option on the `impalad` command line. For clusters managed by Cloudera Manager 5.4.0 and higher, search for the configuration field names `ldap_domain`, `ldap_basedn`, or `ldap_bind_pattern`, fill in and save the appropriate field values, and restart the Impala service. Prior to Cloudera Manager 5.4.0, these values were filled in using the **Impala Daemon Command Line Argument Advanced Configuration Snippet (Safety Valve)** field.

These options are mutually exclusive; Impala does not start if more than one of these options is specified.

Secure LDAP Connections

To avoid sending credentials over the wire in cleartext, you must configure a secure connection between both the client and Impala, and between Impala and the LDAP server. The secure connection could use SSL or TLS.

Secure LDAP connections through SSL:

For SSL-enabled LDAP connections, specify a prefix of `ldaps://` instead of `ldap://`. Also, the default port for SSL-enabled LDAP connections is 636 instead of 389.

Secure LDAP connections through TLS:

TLS, the successor to the SSL protocol, is supported by most modern LDAP servers. Unlike SSL connections, TLS connections can be made on the same server port as non-TLS connections. To secure all connections using TLS, specify the following flags as startup options to the `impalad` daemon:

- `--ldap_tls` tells Impala to start a TLS connection to the LDAP server, and to fail authentication if it cannot be done.
- `--ldap_ca_certificate="`*`/path/to/certificate/pem`*`"` specifies the location of the certificate in standard `.PEM` format. Store this certificate on the local filesystem, in a location that only the `impala` user and other trusted users can read.

LDAP Authentication for impala-shell Interpreter

To connect to Impala using LDAP authentication, you specify command-line options to the `impala-shell` command interpreter and enter the password when prompted:

- `-l` enables LDAP authentication.
- `-u` sets the user. Per Active Directory, the user is the short user name, not the full LDAP distinguished name. If your LDAP settings include a search base, use the `--ldap_bind_pattern` on the `impalad` daemon to translate the short user name from `impala-shell` automatically to the fully qualified name.
- `impala-shell` automatically prompts for the password.

For the full list of available `impala-shell` options, see impala-shell Configuration Options.

LDAP authentication for JDBC applications: See Configuring Impala to Work with JDBC for the format to use with the JDBC connection string for servers using LDAP authentication.

Enabling Impala Delegation for LDAP Users

See Configuring Impala Delegation for Hue and BI Tools on page 101 for details about the delegation feature that lets certain users submit queries using the credentials of other users.

LDAP Restrictions for Impala

The LDAP support is preliminary. It currently has only been tested against Active Directory.

Using Multiple Authentication Methods with Impala

Impala 2.0 and later automatically handles both Kerberos and LDAP authentication. Each `impalad` daemon can accept both Kerberos and LDAP requests through the same port. No special actions need to be taken if some users authenticate through Kerberos and some through LDAP.

Prior to Impala 2.0, you had to configure each `impalad` to listen on a specific port depending on the kind of authentication, then configure your network load balancer to forward each kind of request to a data node that was set up with the appropriate authentication type. Once the initial request was made using either Kerberos or LDAP authentication, Impala automatically handled the process of coordinating the work across multiple nodes and transmitting intermediate results back to the coordinator node.

Configuring Impala Delegation for Hue and BI Tools

When users submit Impala queries through a separate application, such as Hue or a business intelligence tool, typically all requests are treated as coming from the same user. In Impala 1.2 and higher,,Impala supports applications to pass along credentials for the users that connect to them, known as "delegation", and to issue Impala queries with the privileges for those users. Currently, the delegation feature is available only for Impala queries submitted through application interfaces such as Hue and BI tools; for example, Impala cannot issue queries using the privileges of the HDFS user.

The delegation feature is enabled by a startup option for `impalad`: `--authorized_proxy_user_config`. When you specify this option, users whose names you specify (such as hue) can delegate the execution of a query to another user. The query runs with the privileges of the delegated user, not the original user such as hue. The name of the delegated user is passed using the HiveServer2 configuration property `impala.doas.user`.

You can specify a list of users that the application user can delegate to, or `*` to allow a superuser to delegate to any other user. For example:

```
impalad --authorized_proxy_user_config 'hue=user1,user2;admin=*' ...
```

> **Note:** Make sure to use single quotes or escape characters to ensure that any `*` characters do not undergo wildcard expansion when specified in command-line arguments.

See Modifying Impala Startup Options for details about adding or changing `impalad` startup options. See this Cloudera blog post for background information about the delegation capability in HiveServer2.

To set up authentication for the delegated users:

- On the server side, configure either user/password authentication through LDAP, or Kerberos authentication, for all the delegated users. See Enabling LDAP Authentication for Impala on page 99 or Enabling Kerberos Authentication for Impala on page 96 for details.
- On the client side, follow the instructions in the "Using User Name and Password" section in the ODBC driver installation guide. Then search for "delegation" in that same installation guide to learn about the **Delegation UID** field and `DelegationUID` configuration keyword to enable the delegation feature for ODBC-based BI tools.

Enabling Delegation in Cloudera Manager

To enable delegation in Cloudera Manager:

1. Navigate to **Clusters** > **Impala** > **Configuration** > **Policy File-Based Sentry**.
2. In the **Proxy User Configuration** field, type the a semicolon-separated list of key=value pairs of authorized proxy users to the user(s) they can impersonate. The list of delegated users are delimited with a comma, e.g. **hue=user1, user2**.
3. Click **Save Changes** and then restart Impala service.

Llama Authentication

This section describes how to configure Llama in CDH 5 with Kerberos security in a Hadoop cluster.

> **Note:** Llama has been tested only in a Cloudera Manager deployment. For information on using Cloudera Manager to configure Llama and Impala, see Installing Impala.

Configuring Llama to Support Kerberos Security

1. Create a Llama service user principal using the syntax: `llama/fully.qualified.domain.name@YOUR-REALM`. This principal is used to authenticate with the Hadoop cluster, where *fully.qualified.domain.name* is the host where Llama is running and *YOUR-REALM* is the name of your Kerberos realm:

```
$ kadmin
kadmin: addprinc -randkey
llama/fully.qualified.domain.name@YOUR-REALM
```

2. Create a keytab file with the Llama principal:

```
$ kadmin
kadmin: xst -k llama.keytab llama/fully.qualified.domain.name
```

3. Test that the credentials in the keytab file work. For example:

```
$ klist -e -k -t llama.keytab
```

4. Copy the `llama.keytab` file to the Llama configuration directory. The owner of the `llama.keytab` file should be the `llama` user and the file should have owner-only read permissions.
5. Edit the Llama `llama-site.xml` configuration file in the Llama configuration directory by setting the following properties:

Property	Value
`llama.am.server.thrift.security`	`true`
`llama.am.server.thrift.kerberos.keytab.file`	`llama/conf.keytab`
`llama.am.server.thrift.kerberos.server.principal.name`	`llama/fully.qualified.domain.name`
`llama.am.server.thrift.kerberos.notification.principal.name`	`impala`

6. Restart Llama to make the configuration changes take effect.

Oozie Authentication

This section describes how to configure Oozie CDH 5 with Kerberos security on a Hadoop cluster:

- Configuring Kerberos Authentication for the Oozie Server on page 103
- Configuring Oozie HA with Kerberos on page 104

> **Important:**
>
> To enable Oozie to work with Kerberos security on your Hadoop cluster, make sure you perform the installation and configuration steps in Configuring Hadoop Security in CDH 5. Also note that when Kerberos security is enabled in Oozie, a web browser that supports Kerberos HTTP SPNEGO is required to access the Oozie web-console (for example, Firefox, Internet Explorer or Chrome).

> **Important:**
>
> If the NameNode, Secondary NameNode, DataNode, JobTracker, TaskTrackers, ResourceManager, NodeManagers, HttpFS, or Oozie services are configured to use Kerberos HTTP SPNEGO authentication, and two or more of these services are running on the same host, then all of the running services must use the same HTTP principal and keytab file used for their HTTP endpoints.

Configuring Kerberos Authentication for the Oozie Server

1. Create a Oozie service user principal using the syntax: `oozie/<fully.qualified.domain.name>@<YOUR-REALM>`. This principal is used to authenticate with the Hadoop cluster. where: `fully.qualified.domain.name` is the host where the Oozie server is running `YOUR-REALM` is the name of your Kerberos realm.

```
kadmin: addprinc -randkey oozie/fully.qualified.domain.name@YOUR-REALM.COM
```

2. Create a HTTP service user principal using the syntax: `HTTP/<fully.qualified.domain.name>@<YOUR-REALM>`. This principal is used to authenticate user requests coming to the Oozie web-services. where: `fully.qualified.domain.name` is the host where the Oozie server is running `YOUR-REALM` is the name of your Kerberos realm.

```
kadmin: addprinc -randkey HTTP/fully.qualified.domain.name@YOUR-REALM.COM
```

> **Important:**
>
> The `HTTP/` component of the HTTP service user principal must be upper case as shown in the syntax and example above.

3. Create keytab files with both principals.

```
$ kadmin
kadmin: xst -k oozie.keytab oozie/fully.qualified.domain.name
kadmin: xst -k http.keytab HTTP/fully.qualified.domain.name
```

4. Merge the two keytab files into a single keytab file:

```
$ ktutil
ktutil: rkt oozie.keytab
ktutil: rkt http.keytab
ktutil: wkt oozie-http.keytab
```

5. Test that credentials in the merged keytab file work. For example:

```
$ klist -e -k -t oozie-http.keytab
```

6. Copy the `oozie-http.keytab` file to the Oozie configuration directory. The owner of the `oozie-http.keytab` file should be the `oozie` user and the file should have owner-only read permissions.
7. Edit the Oozie server `oozie-site.xml` configuration file in the Oozie configuration directory by setting the following properties:

> **Important:** You must restart the Oozie server to have the configuration changes take effect.

Property	Value
`oozie.service.HadoopAccessorService.kerberos.enabled`	`true`

Property	Value
local.realm	<REALM>
oozie.service.HadoopAccessorService.keytab.file	/etc/oozie/conf/oozie-http.keytab for a package installation, or <EXPANDED_DIR>/conf/oozie-http.keytab for a tarball installation
oozie.service.HadoopAccessorService.kerberos.principal	oozie/<fully.qualified.domain.name>@<YOUR-REALM.COM>
oozie.authentication.type	kerberos
oozie.authentication.kerberos.principal	HTTP/<fully.qualified.domain.name>@<YOUR-REALM.COM>
oozie.authentication.kerberos.name.rules	Use the value configured for hadoop.security.auth_to_local in core-site.xml

Configuring Oozie HA with Kerberos

In CDH 5, you can configure multiple active Oozie servers against the same database, providing high availability for Oozie. For instructions on setting up Oozie HA, see Oozie High Availability

Let's assume you have three hosts running Oozie servers, host1.example.com, host2.example.com, host3.example.com and the Load Balancer running on oozie.example.com. The Load Balancer directs traffic to the Oozie servers: host1, host2 and host3. For such a configuration, assuming your Kerberos realm is EXAMPLE.COM, create the following Kerberos principals:

- oozie/host1.example.com@EXAMPLE.COM
- oozie/host2.example.com@EXAMPLE.COM
- oozie/host3.example.com@EXAMPLE.COM
- HTTP/host1.example.com@EXAMPLE.COM
- HTTP/host2.example.com@EXAMPLE.COM
- HTTP/host3.example.com@EXAMPLE.COM
- HTTP/oozie.example.com@EXAMPLE.COM

On each of the hosts, host1, host2 and host3, create a keytab file with its corresponding oozie and HTTP principals from the list above. All keytab files should also have the load balancer's HTTP principal. Hence, each keytab file should have 3 principals in all.

Edit the following property in the Oozie server configuration file, oozie-site.xml:

```
<property>
<name>oozie.authentication.kerberos.principal</name>
<value>*</value>
</property>
```

Search Authentication

This section describes how to configure Search in CDH 5 to enable Kerberos security and Sentry.

Configuring Search to Use Kerberos

Cloudera Search supports Kerberos authentication. All necessary packages are installed when you install Search. To enable Kerberos, create principals and keytabs and then modify default configurations.

The following instructions only apply to configuring Kerberos in an unmanaged environment. Kerberos configuration is automatically handled by Cloudera Manager if you are using Search in a Cloudera Manager environment.

To create principals and keytabs

Repeat this process on all Solr server hosts.

1. Create a Solr service user principal using the syntax: `solr/<fully.qualified.domain.name>@<YOUR-REALM>`. This principal is used to authenticate with the Hadoop cluster. where: `fully.qualified.domain.name` is the host where the Solr server is running `YOUR-REALM` is the name of your Kerberos realm.

   ```
   $ kadmin
   kadmin: addprinc -randkey solr/fully.qualified.domain.name@YOUR-REALM.COM
   ```

2. Create a HTTP service user principal using the syntax: `HTTP/<fully.qualified.domain.name>@<YOUR-REALM>`. This principal is used to authenticate user requests coming to the Solr web-services. where: `fully.qualified.domain.name` is the host where the Solr server is running `YOUR-REALM` is the name of your Kerberos realm.

   ```
   kadmin: addprinc -randkey HTTP/fully.qualified.domain.name@YOUR-REALM.COM
   ```

   > **Note:**
   >
   > The `HTTP/` component of the HTTP service user principal must be upper case as shown in the syntax and example above.

3. Create keytab files with both principals.

   ```
   kadmin: xst -norandkey -k solr.keytab solr/fully.qualified.domain.name \
   HTTP/fully.qualified.domain.name
   ```

4. Test that credentials in the merged keytab file work. For example:

   ```
   $ klist -e -k -t solr.keytab
   ```

5. Copy the `solr.keytab` file to the Solr configuration directory. The owner of the `solr.keytab` file should be the `solr` user and the file should have owner-only read permissions.

To modify default configurations

Repeat this process on all Solr server hosts.

1. Ensure that the following properties appear in `/etc/default/solr` and that they are uncommented. Modify these properties to match your environment. The relevant properties to be uncommented and modified are:

   ```
   SOLR_AUTHENTICATION_TYPE=kerberos
   SOLR_AUTHENTICATION_SIMPLE_ALLOW_ANON=true
   SOLR_AUTHENTICATION_KERBEROS_KEYTAB=/etc/solr/conf/solr.keytab
   SOLR_AUTHENTICATION_KERBEROS_PRINCIPAL=HTTP/localhost@LOCALHOST
   SOLR_AUTHENTICATION_KERBEROS_NAME_RULES=DEFAULT
   SOLR_AUTHENTICATION_JAAS_CONF=/etc/solr/conf/jaas.conf
   ```

   > **Note:** Modify the values for these properties to match your environment. For example, the `SOLR_AUTHENTICATION_KERBEROS_PRINCIPAL=HTTP/localhost@LOCALHOST` must include the principal instance and Kerberos realm for your environment. That is often different from `localhost@LOCALHOST`.

2. Set `hadoop.security.auth_to_local` to match the value specified by `SOLR_AUTHENTICATION_KERBEROS_NAME_RULES` in `/etc/default/solr`.

   > **Note:** For information on how to configure the rules, see Configuring the Mapping from Kerberos Principals to Short Names on page 112. For additional information on using Solr with HDFS, see Configuring Solr for Use with HDFS.

3. If using applications that use the `solrj` library, set up the Java Authentication and Authorization Service (JAAS).

a. Create a `jaas.conf` file in the Solr configuration directory containing the following settings. This file and its location must match the `SOLR_AUTHENTICATION_JAAS_CONF` value. Make sure that you substitute a value for `principal` that matches your particular environment.

```
Client {
  com.sun.security.auth.module.Krb5LoginModule required
  useKeyTab=true
  useTicketCache=false
  keyTab="/etc/solr/conf/solr.keytab"
  principal="solr/fully.qualified.domain.name@<YOUR-REALM>";
};
```

Using Kerberos

The process of enabling Solr clients to authenticate with a secure Solr is specific to the client. This section demonstrates:

- Using Kerberos and curl
- Using solrctl
- Configuring SolrJ Library Usage.

  This enables technologies including:

  - Command line solutions
  - Java applications
  - The MapReduceIndexerTool

- Configuring Flume Morphline Solr Sink Usage

Secure Solr requires that the CDH components that it interacts with are also secure. Secure Solr interacts with HDFS, ZooKeeper and optionally HBase, MapReduce, andFlume. See Cloudera Security or the CDH 4 Security Guide for more information.

Using Kerberos and curl

You can use Kerberos authentication with clients such as `curl`. To use `curl`, begin by acquiring valid Kerberos credentials and then execute the desired command. For example, you might use commands similar to the following:

```
$ kinit -kt username.keytab username
$ curl --negotiate -u foo:bar http://solrserver:8983/solr/
```

Note: Depending on the tool used to connect, additional arguments may be required. For example, with `curl`, `--negotiate` and `-u` are required. The username and password specified with `-u` is not actually checked because Kerberos is used. As a result, any value such as foo:bar or even just : is acceptable. While any value can be provided for `-u`, note that the option is required. Omitting `-u` results in a 401 Unauthorized error, even though the `-u` value is not actually used.

Using solrctl

If you are using `solrctl` to manage your deployment in an environment that requires Kerberos authentication, you must have valid Kerberos credentials, which you can get using `kinit`. For more information on `solrctl`, see Solrctl Reference

Configuring SolrJ Library Usage

If using applications that use the solrj library, begin by establishing a Java Authentication and Authorization Service (JAAS) configuration file.

Create a JAAS file:

- If you have already used `kinit` to get credentials, you can have the client use those credentials. In such a case, modify your `jaas-client.conf` file to appear as follows:

```
Client {
 com.sun.security.auth.module.Krb5LoginModule required
 useKeyTab=false
 useTicketCache=true
 principal="user/fully.qualified.domain.name@<YOUR-REALM>";
 };
```

  where `user/fully.qualified.domain.name@<YOUR-REALM>` is replaced with your credentials.

- You want the client application to authenticate using a keytab you specify:

```
Client {
 com.sun.security.auth.module.Krb5LoginModule required
 useKeyTab=true
 keyTab="/path/to/keytab/user.keytab"
 storeKey=true
 useTicketCache=false
 principal="user/fully.qualified.domain.name@<YOUR-REALM>";
};
```

  where `/path/to/keytab/user.keytab` is the keytab file you wish to use and `user/fully.qualified.domain.name@<YOUR-REALM>` is the principal in that keytab you wish to use.

Use the JAAS file to enable solutions:

- **Command line solutions**

  Set the property when invoking the program. For example, if you were using a jar, you might use:

```
java -Djava.security.auth.login.config=/home/user/jaas-client.conf -jar app.jar
```

- **Java applications**

  Set the Java system property `java.security.auth.login.config`. For example, if the JAAS configuration file is located on the filesystem as `/home/user/jaas-client.conf`. The Java system property `java.security.auth.login.config` must be set to point to this file. Setting a Java system property can be done programmatically, for example using a call such as:

```
System.setProperty("java.security.auth.login.config", "/home/user/jaas-client.conf");
```

- **The MapReduceIndexerTool**

  The MapReduceIndexerTool uses SolrJ to pass the JAAS configuration file. Using the MapReduceIndexerTool in a secure environment requires the use of the `HADOOP_OPTS` variable to specify the JAAS configuration file. For example, you might issue a command such as the following:

```
HADOOP_OPTS="-Djava.security.auth.login.config=/home/user/jaas.conf" \
hadoop jar MapReduceIndexerTool
```

- **Configuring the hbase-indexer CLI**

  Certain hbase-indexer CLI commands such as `replication-status` attempt to read ZooKeeper hosts owned by HBase. To successfully use these commands in Search for CDH 5 in a secure environment, specify a JAAS configuration file with the HBase principal in the `HBASE_INDEXER_OPTS` environment variable. For example, you might issue a command such as the following:

```
HBASE_INDEXER_OPTS="-Djava.security.auth.login.config=/home/user/hbase-jaas.conf" \
hbase-indexer replication-status
```

Configuring Flume Morphline Solr Sink Usage

Repeat this process on all Flume hosts:

1. If you have not created a keytab file, do so now at `/etc/flume-ng/conf/flume.keytab`. This file should contain the service principal `flume/<fully.qualified.domain.name>@<YOUR-REALM>`. See the CDH 5 Security Guide for more information.
2. Create a JAAS configuration file for flume at `/etc/flume-ng/conf/jaas-client.conf`. The file should appear as follows:

```
Client {
 com.sun.security.auth.module.Krb5LoginModule required
 useKeyTab=true
 useTicketCache=false
 keyTab="/etc/flume-ng/conf/flume.keytab"
 principal="flume/<fully.qualified.domain.name>@<YOUR-REALM>";
};
```

3. Add the flume JAAS configuration to the `JAVA_OPTS` in `/etc/flume-ng/conf/flume-env.sh`. For example, you might change:

```
JAVA_OPTS="-Xmx500m"
```

to:

```
JAVA_OPTS="-Xmx500m -Djava.security.auth.login.config=/etc/flume-ng/conf/jaas-client.conf"
```

ZooKeeper Authentication

This section describes how to configure ZooKeeper in CDH 5 to enable Kerberos security:

- Configuring ZooKeeper Server for Kerberos Authentication on page 108
- Configuring the ZooKeeper Client Shell to Support Kerberos Security on page 109
- Verifying the Configuration on page 110

> **Important:**
>
> Prior to enabling ZooKeeper to work with Kerberos security on your cluster, make sure you first review the requirements in Configuring Hadoop Security in CDH 5.

Configuring ZooKeeper Server for Kerberos Authentication

You can configure the ZooKeeper server for Kerberos authentication in Cloudera Manager or through the command line.

Using Cloudera Manager to Configure ZooKeeper Server for Kerberos Authentication

To set up the ZooKeeper server for Kerberos authentication in Cloudera Manager, complete the following steps:

1. In Cloudera Manager, open the ZooKeeper service.
2. Click the **Configuration** tab.
3. Enter **Kerberos** in the in the **Search** bar.
4. Find the **Enable Kerberos Authentication** property and select the check-box next to the ZooKeeper services that you want to configure for Kerberos authentication.

Using the Command Line to Configure ZooKeeper Server for Kerberos Authentication

Follow the steps below for each ZooKeeper server in the ensemble. To maintain consistency across ZooKeeper servers in the ensemble, use the same name for the keytab file you deploy to each server, for example, `zookeeper.keytab`. Each keytab file will contain its respective host's fully-qualified domain name (FQDN).

1. Create a service principal for the ZooKeeper server using the fully-qualified domain name (FQDN) of the host on which ZooKeeper server is running and the name of your Kerberos realm using the pattern `zookeeper/fqdn.example.com@ YOUR-REALM`. This principal will be used to authenticate the ZooKeeper server with the Hadoop cluster. Create this service principal as follows:

```
kadmin: addprinc -randkey zookeeper/fqdn.example.com@YOUR-REALM
```

2. Create a keytab file for the ZooKeeper server:

```
$ kadmin
kadmin: xst -k zookeeper.keytab zookeeper/fqdn.example.com@YOUR-REALM
```

> **Note:** For consistency across ZooKeeper Servers, use the same name for the keytab file you create for each subsequent ZooKeeper Server host system you configure using these steps, for example, `zookeeper.keytab`.

3. Copy the `zookeeper.keytab` file to the ZooKeeper configuration directory on the ZooKeeper server host, using the appropriate ZooKeeper configuration directory: `/etc/zookeeper/conf/`. The `zookeeper.keytab` file should be owned by the `zookeeper` user, with owner-only read permissions.
4. Add the following lines to the ZooKeeper configuration file `zoo.cfg`:

```
authProvider.1=org.apache.zookeeper.server.auth.SASLAuthenticationProvider
jaasLoginRenew=3600000
```

5. Set up the Java Authentication and Authorization Service (JAAS) by creating a `jaas.conf` file in the ZooKeeper configuration directory with the settings shown below, replacing *`fqdn.example.com`* with the ZooKeeper server's hostname.

```
Server {
  com.sun.security.auth.module.Krb5LoginModule required
  useKeyTab=true
  keyTab="/etc/zookeeper/conf/zookeeper.keytab"
  storeKey=true
  useTicketCache=false
  principal="zookeeper/fqdn.example.com
  @YOUR-REALM";
};
```

6. Add the following setting to the `java.env` file located in the ZooKeeper configuration directory, creating the file if necessary:

```
export JVMFLAGS="-Djava.security.auth.login.config=/etc/zookeeper/conf/jaas.conf"
```

7. Repeat these steps for each ZooKeeper server in the ensemble.
8. Restart the ZooKeeper server to have the configuration changes take effect. See ZooKeeper Installation for details.

Configuring the ZooKeeper Client Shell to Support Kerberos Security

1. If you want to use the ZooKeeper client shell `zookeeper-client` with Kerberos authentication, create a principal using the syntax: `zkcli@<YOUR-REALM>`. This principal is used to authenticate the ZooKeeper client shell to the ZooKeeper service. where: `YOUR-REALM` is the name of your Kerberos realm.

```
kadmin: addprinc -randkey zkcli@YOUR-REALM.COM
```

2. Create a keytab file for the ZooKeeper client shell.

```
$ kadmin
kadmin: xst -norandkey -k zkcli.keytab zkcli@YOUR-REALM.COM
```

> **Note:**
>
> Some versions of kadmin do not support the `-norandkey` option in the command above. If your version does not, you can omit it from the command. Note that doing so will result in a new password being generated every time you export a keytab, which will invalidate previously-exported keytabs.

3. Set up JAAS in the configuration directory on the host where the ZooKeeper client shell is running. For a package installation, the configuration directory is `/etc/zookeeper/conf/`. For a tar ball installation, the configuration directory is `<EXPANDED_DIR>/conf`. Create a `jaas.conf` file containing the following settings:

```
Client {
  com.sun.security.auth.module.Krb5LoginModule required
  useKeyTab=true
  keyTab="/path/to/zkcli.keytab"
  storeKey=true
  useTicketCache=false
  principal="zkcli@<YOUR-REALM>";
};
```

4. Add the following setting to the `java.env` file located in the configuration directory. (Create the file if it does not already exist.)

```
export JVMFLAGS="-Djava.security.auth.login.config=/etc/zookeeper/conf/jaas.conf"
```

Verifying the Configuration

1. Make sure that you have restarted the ZooKeeper cluster with Kerberos enabled, as described above.
2. Start the client (where the hostname is the name of a ZooKeeper server):

```
zookeeper-client -server hostname:port
```

3. Create a protected znode from within the ZooKeeper CLI. Make sure that you substitute `YOUR-REALM` as appropriate.

```
create /znode1 znode1data sasl:zkcli@{{YOUR-REALM}}:cdwra
```

4. Verify the znode is created and the ACL is set correctly:

```
getAcl /znode1
```

The results from `getAcl` should show that the proper scheme and permissions were applied to the znode.

FUSE Kerberos Configuration

This section describes how to use FUSE (Filesystem in Userspace) and CDH with Kerberos security on your Hadoop cluster. FUSE enables you to mount HDFS, which makes HDFS files accessible just as if they were UNIX files.

To use FUSE and CDH with Kerberos security, follow these guidelines:

- For each HDFS user, make sure that there is a UNIX user with the same name. If there isn't, some files in the FUSE mount point will appear to be owned by a non-existent user. Although this is harmless, it can cause confusion.

- When using Kerberos authentication, users must run `kinit` before accessing the FUSE mount point. Failure to do this will result in I/O errors when the user attempts to access the mount point. For security reasons, it is not possible to list the files in the mount point without first running `kinit`.
- When a user runs `kinit`, all processes that run as that user can use the Kerberos credentials. It is not necessary to run `kinit` in the same shell as the process accessing the FUSE mount point.

Using kadmin to Create Kerberos Keytab Files

If your version of Kerberos does not support the Kerberos `-norandkey` option in the `xst` command, or if you must use `kadmin` because you cannot use `kadmin.local`, then you can use the following procedure to create Kerberos keytab files. Using the `-norandkey` option when creating keytabs is optional and a convenience, but it is not required.

> **Important:**
>
> **For both MRv1 and YARN deployments:** *On every machine in your cluster*, there must be a keytab file for the `hdfs` user and a keytab file for the `mapred` user. The `hdfs` keytab file must contain entries for the `hdfs` principal and an `HTTP` principal, and the `mapred` keytab file must contain entries for the `mapred` principal and an `HTTP` principal. On each respective machine, the `HTTP` principal will be the same in both keytab files.
>
> **In addition, for YARN deployments only:** *On every machine in your cluster*, there must be a keytab file for the `yarn` user. The `yarn` keytab file must contain entries for the `yarn` principal and an `HTTP` principal. On each respective machine, the `HTTP` principal in the `yarn` keytab file will be the same as the `HTTP` principal in the `hdfs` and `mapred` keytab files.

For instructions, see To create the Kerberos keytab files on page 111.

> **Note:**
>
> These instructions illustrate an example of creating keytab files for MIT Kerberos. If you are using another version of Kerberos, refer to your Kerberos documentation for instructions. You can use either `kadmin` or `kadmin.local` to run these commands.

To create the Kerberos keytab files

Do the following steps for every host in your cluster, replacing the `fully.qualified.domain.name` in the commands with the fully qualified domain name of each host:

1. Create the `hdfs` keytab file, which contains an entry for the `hdfs` principal. This keytab file is used for the NameNode, Secondary NameNode, and DataNodes.

```
$ kadmin
kadmin:  xst -k hdfs-unmerged.keytab hdfs/fully.qualified.domain.name
```

2. Create the `mapred` keytab file, which contains an entry for the `mapred` principal. If you are using MRv1, the `mapred` keytab file is used for the JobTracker and TaskTrackers. If you are using YARN, the `mapred` keytab file is used for the MapReduce Job History Server.

```
kadmin:  xst -k mapred-unmerged.keytab mapred/fully.qualified.domain.name
```

3. **YARN only:** Create the `yarn` keytab file, which contains an entry for the `yarn` principal. This keytab file is used for the ResourceManager and NodeManager.

```
kadmin:  xst -k yarn-unmerged.keytab yarn/fully.qualified.domain.name
```

4. Create the `http` keytab file, which contains an entry for the HTTP principal.

```
kadmin:  xst -k http.keytab HTTP/fully.qualified.domain.name
```

5. Use the `ktutil` command to merge the previously-created keytabs:

```
$ ktutil
ktutil:  rkt hdfs-unmerged.keytab
ktutil:  rkt http.keytab
ktutil:  wkt hdfs.keytab
ktutil:  clear
ktutil:  rkt mapred-unmerged.keytab
ktutil:  rkt http.keytab
ktutil:  wkt mapred.keytab
ktutil:  clear
ktutil:  rkt yarn-unmerged.keytab
ktutil:  rkt http.keytab
ktutil:  wkt yarn.keytab
```

This procedure creates three new files: `hdfs.keytab`, `mapred.keytab` and `yarn.keytab`. These files contain entries for the `hdfs` and HTTP principals, the `mapred` and HTTP principals, and the `yarn` and HTTP principals respectively.

6. Use `klist` to display the keytab file entries. For example, a correctly-created `hdfs` keytab file should look something like this:

```
$ klist -e -k -t hdfs.keytab
Keytab name: WRFILE:hdfs.keytab
slot KVNO Principal
---- ---- ---------------------------------------------------------------------
   1    7    HTTP/fully.qualified.domain.name@YOUR-REALM.COM (DES cbc mode with CRC-32)

   2    7    HTTP/fully.qualified.domain.name@YOUR-REALM.COM (Triple DES cbc mode with
 HMAC/sha1)
   3    7    hdfs/fully.qualified.domain.name@YOUR-REALM.COM (DES cbc mode with CRC-32)

   4    7    hdfs/fully.qualified.domain.name@YOUR-REALM.COM (Triple DES cbc mode with
 HMAC/sha1)
```

7. To verify that you have performed the merge procedure correctly, make sure you can obtain credentials as both the `hdfs` and HTTP principals using the single merged keytab:

```
$ kinit -k -t hdfs.keytab hdfs/fully.qualified.domain.name@YOUR-REALM.COM
$ kinit -k -t hdfs.keytab HTTP/fully.qualified.domain.name@YOUR-REALM.COM
```

If either of these commands fails with an error message such as "`kinit: Key table entry not found while getting initial credentials`", then something has gone wrong during the merge procedure. Go back to step 1 of this document and verify that you performed all the steps correctly.

8. To continue the procedure of configuring Hadoop security in CDH 5, follow the instructions in the section To deploy the Kerberos keytab files.

Configuring the Mapping from Kerberos Principals to Short Names

You configure the mapping from Kerberos principals to short names in the `hadoop.security.auth_to_local` property setting in the `core-site.xml` file. Kerberos has this support natively, and Hadoop's implementation reuses Kerberos's configuration language to specify the mapping.

A mapping consists of a set of rules that are evaluated in the order listed in the `hadoop.security.auth_to_local` property. The first rule that matches a principal name is used to map that principal name to a short name. Any later rules in the list that match the same principal name are ignored.

You specify the mapping rules on separate lines in the `hadoop.security.auth_to_local` property as follows:

```
<property>
  <name>hadoop.security.auth_to_local</name>
  <value>
  RULE:[<principal translation>](<acceptance filter>)<short name substitution>
  RULE:[<principal translation>](<acceptance filter>)<short name substitution>
  DEFAULT
  </value>
</property>
```

Mapping Rule Syntax

To specify a mapping rule, use the prefix string `RULE:` followed by three sections—principal translation, acceptance filter, and short name substitution—described in more detail below. The syntax of a mapping rule is:

```
RULE:[<principal translation>](<acceptance filter>)<short name substitution>
```

Principal Translation

The first section of a rule, `<principal translation>`, performs the matching of the principal name to the rule. If there is a match, the principal translation also does the initial translation of the principal name to a short name. In the `<principal translation>` section, you specify the number of components in the principal name and the pattern you want to use to translate those principal component(s) and realm into a short name. In Kerberos terminology, a principal name is a set of components separated by slash ("/") characters.

The principal translation is composed of two parts that are both specified within "[ ]" using the following syntax:

```
[<number of components in principal name>:<initial specification of short name>]
```

where:

<number of components in principal name> – This first part specifies the number of components in the principal name (not including the realm) and must be `1` or `2`. A value of `1` specifies principal names that have a single component (for example, `hdfs`), and `2` specifies principal names that have two components (for example, `hdfs/fully.qualified.domain.name`). A principal name that has only one component will only match single-component rules, and a principal name that has two components will only match two-component rules.

<initial specification of short name> – This second part specifies a pattern for translating the principal component(s) and the realm into a short name. The variable $0 translates the realm, $1 translates the first component, and $2 translates the second component.

Here are some examples of principal translation sections. These examples use `atm@YOUR-REALM.COM` and `atm/fully.qualified.domain.name@YOUR-REALM.COM` as principal name inputs:

This Principal Translation	Translates atm@YOUR-REALM.COM into this short name	Translates atm/fully.qualified.domain.name@YOUR-REALM.COM into this short name
`[1:$1@$0]`	atm@YOUR-REALM.COM	**Rule does not match**[1]
`[1:$1]`	atm	**Rule does not match**[1]
`[1:$1.foo]`	atm.foo	**Rule does not match**[1]
`[2:$1/$2@$0]`	**Rule does not match**[2]	atm/fully.qualified.domain.name@YOUR-REALM.COM
`[2:$1/$2]`	**Rule does not match**[2]	atm/fully.qualified.domain.name
`[2:$1@$0]`	**Rule does not match**[2]	atm@YOUR-REALM.COM
`[2:$1]`	**Rule does not match**[2]	atm

Footnotes:

[1]Rule does not match because there are two components in principal name `atm/fully.qualified.domain.name@YOUR-REALM.COM`

[2]Rule does not match because there is one component in principal name `atm@YOUR-REALM.COM`

Acceptance Filter

The second section of a rule, (`<acceptance filter>`), matches the translated short name from the principal translation (that is, the output from the first section). The acceptance filter is specified in "( )" characters and is a standard regular expression. A rule matches only if the specified regular expression matches the entire translated short name from the principal translation. That is, there's an implied ^ at the beginning of the pattern and an implied $ at the end.

Short Name Substitution

The third and final section of a rule is the (`<short name substitution>`). If there is a match in the second section, the acceptance filter, the (`<short name substitution>`) section does a final translation of the short name from the first section. This translation is a sed replacement expression (`s/.../.../g`) that translates the short name from the first section into the final short name string. The short name substitution section is optional. In many cases, it is sufficient to use the first two sections only.

Converting Principal Names to Lowercase

In some organizations, naming conventions result in mixed-case usernames (for example, `John.Doe`) or even uppercase usernames (for example, `JDOE`) in Active Directory or LDAP. This can cause a conflict when the Linux username and HDFS home directory are lowercase.

To convert principal names to lowercase, append `/L` to the rule.

Example Rules

Suppose all of your service principals are either of the form `App.service-name/fully.qualified.domain.name@YOUR-REALM.COM` or `App.service-name@YOUR-REALM.COM`, and you want to map these to the short name string `service-name`. To do this, your rule set would be:

```
<property>
  <name>hadoop.security.auth_to_local</name>
  <value>
 RULE:[1:$1](App\..*)s/App\.(.*)/$1/g
  RULE:[2:$1](App\..*)s/App\.(.*)/$1/g
  DEFAULT
  </value>
</property>
```

The first `$1` in each rule is a reference to the first component of the full principal name, and the second `$1` is a regular expression back-reference to text that is matched by (`.*`).

In the following example, suppose your company's naming scheme for user accounts in Active Directory is FirstnameLastname (for example, `JohnDoe`), but user home directories in HDFS are `/user/firstnamelastname`. The following rule set converts user accounts in the `CORP.EXAMPLE.COM` domain to lowercase.

```
<property>
  <name>hadoop.security.auth_to_local</name>
  <value>
  RULE:[2:$1@$0](HTTP@\QCORP.EXAMPLE.COM\E$)s/@\QCORP.EXAMPLE.COM\E$//
  RULE:[1:$1@$0](.*@\QCORP.EXAMPLE.COM\E$)s/@\QCORP.EXAMPLE.COM\E$///L
  RULE:[2:$1@$0](.*@\QCORP.EXAMPLE.COM\E$)s/@\QCORP.EXAMPLE.COM\E$///L
  DEFAULT
  </value>
</property>
```

In this example, the `JohnDoe@CORP.EXAMPLE.COM` principal becomes the `johndoe` HDFS user.

Default Rule

You can specify an optional default rule called `DEFAULT` (see example above). The default rule reduces a principal name down to its first component only. For example, the default rule reduces the principal names `atm@YOUR-REALM.COM` or `atm/fully.qualified.domain.name@YOUR-REALM.COM` down to `atm`, assuming that the default domain is `YOUR-REALM.COM`.

The default rule applies only if the principal is in the default realm.

If a principal name does not match any of the specified rules, the mapping for that principal name will fail.

Testing Mapping Rules

You can test mapping rules for a long principal name by running:

```
$ hadoop org.apache.hadoop.security.HadoopKerberosName name1 name2 name3
```

Enabling Debugging Output for the Sun Kerberos Classes

Initially getting a secure Hadoop cluster configured properly can be tricky, especially for those who are not yet familiar with Kerberos. To help with this, it can be useful to enable debugging output for the Sun Kerberos classes. To do so, set the `HADOOP_OPTS` environment variable to the following:

```
HADOOP_OPTS="-Dsun.security.krb5.debug=true"
```

Configuring a Cluster-dedicated MIT KDC with Cross-Realm Trust

> **Important:** If you have existing Kerberos host keytabs at your site, it's important that you read this section to prevent your existing host keytabs from becoming invalid.

If you use Cloudera Manager to enable Hadoop security on your cluster, the Cloudera Manager Server will create several principals and then generate keytabs for those principals. Cloudera Manager will then deploy the keytab files on every host in the cluster. See Hadoop Users in Cloudera Manager and CDH on page 129 for a complete listing of the principals created by Cloudera Manager.

> **Note:** The following instructions illustrate an example of creating and deploying the principals and keytab files for MIT Kerberos. (If you are using another version of Kerberos, refer to the Kerberos documentation for the version of the operating system you are using, for instructions.)

When to use kadmin.local and kadmin

When performing the Kerberos commands in this document, you can use `kadmin.local` or `kadmin` depending on your access and account:

- If you can log on to the KDC host directly, and have root access or a Kerberos admin account, use the `kadmin.local` command.
- When accessing the KDC from a remote host, use the `kadmin` command.

To start `kadmin.local` on the KDC host:

```
$ sudo kadmin.local
```

To run `kadmin` from any host:

```
$ kadmin
```

Note:

- In this guide, `kadmin` is shown as the prompt for commands in the `kadmin` shell, but you can type the same commands at the `kadmin.local` prompt in the `kadmin.local` shell.
- Running `kadmin.local` may prompt you for a password because it is being run using `sudo`. You should provide your Unix password. Running `kadmin` may prompt you for a password because you need Kerberos admin privileges. You should provide your Kerberos admin password.

Setting up a Cluster-Dedicated KDC and Default Realm for the Hadoop Cluster

Cloudera has tested the following configuration approaches to Kerberos security for clusters managed by Cloudera Manager. For administration teams that are just getting started with Kerberos security, we recommend starting with these approaches to the configuration of KDC services for a number of reasons.

The number of Service Principal Names (SPNs) that are created and managed by the Cloudera Manager server for a CDH cluster can be significant, and it is important to realize the impact on cluster uptime and overall operations when keytabs must be managed manually. The Cloudera Manager server manages the creation of service keytabs on the proper hosts based on the current configuration of the database. Manual keytab management can be error prone and introduce delays when deploying or moving services within the cluster, especially under time-sensitive conditions.

Cloudera Manager creates SPNs within a KDC that it can access with the `kadmin` command and reach based on configuration of the `/etc/krb5.conf` on all systems participating in the cluster. SPNs must be created in the form of `service-name/host.fqdn.name@EXAMPLE.COM` where service name is the relevant CDH service name such as hue or `hbase` or `hdfs`.

If your site already has a working KDC and keytabs for any of the principals that Cloudera Manager creates, as described in the following sections, the Cloudera Manager Server will randomize the key stored in the keytab file and consequently cause your existing host keytabs to become invalid.

This is why Cloudera recommends you prevent your existing host keytabs from becoming invalid is by using a dedicated local MIT Kerberos KDC and default realm for the Hadoop cluster and create all Hadoop `hdfs`, `mapred`, `oozie`, `HTTP`, hue, and `host` service principals in that realm. You can also set up a one-way cross-realm trust from the cluster-dedicated KDC and realm to your existing central MIT Kerberos KDC, or to an existing Active Directory realm. Using this method, there is no need to create service principals in the central MIT Kerberos KDC or in Active Directory, but principals (users) in the central MIT KDC or in Active Directory can be authenticated to Hadoop. The steps to implement this approach are as follows:

1. Install and configure a cluster-dedicated MIT Kerberos KDC that will be managed by Cloudera Manager for creating and storing the service principals for your Hadoop cluster.
2. See the example `kdc.conf` and `krb5.conf` files in Sample Kerberos Configuration files: krb5.conf, kdc.conf, kadm5.acl on page 138 for configuration considerations for the KDC and Kerberos clients.
3. Configure a default Kerberos realm for the cluster you want Cloudera Manager to manage and set up one-way cross-realm trust between the cluster-dedicated KDC and either your central KDC or Active Directory. Follow the appropriate instructions below for your deployment: Using a Cluster-Dedicated KDC with a Central MIT KDC on page 117 or Using a Cluster-Dedicated MIT KDC with Active Directory on page 118.

Cloudera strongly recommends the method above because:

- It requires minimal configuration in Active Directory.
- It is comparatively easy to script the creation of many principals and keytabs. A principal and keytab must be created for every daemon in the cluster, and in a large cluster this can be extremely onerous to do directly in Active Directory.
- There is no need to involve central Active Directory administrators in order to get service principals created.
- It allows for incremental configuration. The Hadoop administrator can completely configure and verify the functionality the cluster independently of integrating with Active Directory.

Using a Cluster-Dedicated KDC with a Central MIT KDC

> **Important:** If you plan to use Oozie or the Hue Kerberos Ticket Renewer in your cluster, you must configure your KDC to allow tickets to be renewed, and you must configure `krb5.conf` to request renewable tickets. Typically, you can do this by adding the `max_renewable_life` setting to your realm in `kdc.conf`, and by adding the `renew_lifetime` parameter to the `libdefaults` section of `krb5.conf`. For more information about renewable tickets, see the Kerberos documentation. This is covered in our example `krb5.conf` and `kdc.conf` files.

1. In the `/var/kerberos/krb5kdc/kdc.conf` file on the local dedicated KDC server host, configure the default realm for the Hadoop cluster by substituting your Kerberos realm in the following `realms` property. Please refer to our example `kdc.conf` file for more information:

```
[realms]
 HADOOP.EXAMPLE.COM = {
```

2. In the `/etc/krb5.conf` file on all cluster hosts and all Hadoop client user hosts, configure the default realm for the Hadoop cluster by substituting your Kerberos realm in the following `realms` property. Also specify the local dedicated KDC server hostname in the `/etc/krb5.conf` file (for example, `kdc01.example.com`).

```
[libdefaults]
  default_realm = HADOOP.EXAMPLE.COM
[realms]
  HADOOP.EXAMPLE.COM = {
    kdc = kdc01.hadoop.example.com:88
    admin_server = kdc01.hadoop.example.com:749
    default_domain = hadoop.example.com
  }
  EXAMPLE.COM = {
    kdc = kdc01.example.com:88
    admin_server = kdc01.example.com:749
    default_domain = example.com
  }
[domain_realm]
  .hadoop.example.com = HADOOP.EXAMPLE.COM
  hadoop.example.com = HADOOP.EXAMPLE.COM
  .example.com = EXAMPLE.COM
  example.com = EXAMPLE.COM
```

3. To set up the cross-realm trust in the cluster-dedicated KDC, type the following command in the `kadmin.local` or `kadmin` shell on the cluster-dedicated KDC host to create a `krbtgt` principal. Substitute your cluster-dedicated KDC realm for `HADOOP.EXAMPLE.COM`, and substitute your central KDC realm for `EXAMPLE.COM`. Enter a password when prompted. Note the password because you will need to enter the same exact password in the central KDC in the next step.

```
kadmin:  addprinc krbtgt/HADOOP.EXAMPLE.COM@EXAMPLE.COM
```

4. Each of your Hadoop client users must also place this information in their local `core-site.xml` file. The easiest way to do so is by using the Cloudera Manager Admin Console to generate a client configuration file.
5. To set up the cross-realm trust in the central KDC, type the same command in the `kadmin.local` or `kadmin` shell on the central KDC host to create the exact same `krbtgt` principal and password.

```
kadmin:  addprinc krbtgt/HADOOP.EXAMPLE.COM@EXAMPLE.COM
```

> **Important:** In order for a cross-realm trust to operate properly, both KDCs must have the same `krbtgt` principal and password, and both KDCs must be configured to use the same encryption type.

6. To properly translate principal names from the central KDC realm into the cluster-dedicated KDC realm for the Hadoop cluster, configure the **Trusted Kerberos Realms** property of the HDFS service.

a. Open the Cloudera Manager Admin Console.
b. Go to the HDFS service.
c. Click the **Configuration** tab.
d. Expand the **Service-Wide** category and click **Security**.
e. Scroll down to the **Trusted Kerberos Realms** property, and click on the **Value** field to add the name of your central KDC realm. If you need to use more advanced mappings which do more than just allow principals from another domain, you may enter them in the **Additional Rules to Map Kerberos Principals to Short Names** property. For more information about name mapping rules, see Configuring the Mapping from Kerberos Principals to Short Names on page 112.

7. Each of your Hadoop client users must also place this information in their local `core-site.xml` file. The easiest way to do so is by using the Cloudera Manager Admin Console to generate a client configuration file.
8. Proceed to Step 2: If You are Using AES-256 Encryption, Install the JCE Policy File on page 32. Later in this procedure, you will restart the services to have the configuration changes in `core-site.xml` take effect.

Using a Cluster-Dedicated MIT KDC with Active Directory

> **Important:** If you are using Cloudera Manager, ensure you have installed the `openldap-clients` package on the Cloudera Manager Server host before you begin configuring Kerberos authentication.

On the Active Directory Server

1. On the Active Directory server host, type the following command to add the local realm trust to Active Directory:

```
netdom trust HADOOP.EXAMPLE.COM /Domain:EXAMPLE.COM /add /realm /passwordt:TrustPassword
```

2. On the Active Directory server host, type the following command to set the proper encryption type:

   Windows 2003 RC2

   Windows 2003 server installations do not support AES encryption for Kerberos. Therefore RC4 should be used. Please see the Microsoft reference documentation for more information.

```
ktpass /MITRealmName HADOOP.EXAMPLE.COM /TrustEncryp RC4
```

   Windows 2008

```
ksetup /SetEncTypeAttr HADOOP.EXAMPLE.COM <enc_type>
```

   Where the <enc_type> parameter can be replaced with parameter strings for AES, DES, or RC4 encryption modes. For example, for AES encryption, replace <enc_type> with `AES256-CTS-HMAC-SHA1-96` or `AES128-CTS-HMAC-SHA1-96` and for RC4 encryption, replace with `RC4-HMAC-MD5`. See the Microsoft reference documentation for more information.

   > **Important:** Make the encryption type you specify is supported on both your version of Windows Active Directory and your version of MIT Kerberos.

On the MIT KDC server

1. In the `/var/kerberos/krb5kdc/kdc.conf` file on the local dedicated KDC server host, configure the default realm for the Hadoop cluster by substituting your Kerberos realm in the following `realms` property:

```
[realms]
 HADOOP.EXAMPLE.COM = {
```

2. Each of your Hadoop client users must also place this information in their local `core-site.xml` file. The easiest way to do so is by using the Cloudera Manager Admin Console to generate a client configuration file.
3. On the local MIT KDC server host, type the following command in the kadmin.local or kadmin shell to add the cross-realm `krbtgt` principal:

```
kadmin:  addprinc -e "<enc_type_list>" krbtgt/HADOOP.EXAMPLE.COM@EXAMPLE.COM
```

where the <enc_type_list> parameter specifies the types of encryption this cross-realm `krbtgt` principal will support: either AES, DES, or RC4 encryption. You can specify multiple encryption types using the parameter in the command above, what's important is that at least one of the encryption types corresponds to the encryption type found in the tickets granted by the KDC in the remote realm.

Examples by Active Directory Domain or Forest "Functional level"

Active Directory will, based on the Domain or Forest functional level, use encryption types supported by that release of the Windows Server operating system. It is not possible to use AES encryption types with an AD 2003 functional level. If you notice that DES encryption types are being used when authenticating or requesting service tickets to Active Directory then it might be necessary to enable weak encryption types in the `/etc/krb5.conf`. Please see the example `krb5.conf` file for more information.

- **Windows 2003**

```
kadmin: addprinc -e "rc4-hmac:normal" krbtgt/HADOOP.EXAMPLE.COM@EXAMPLE.COM
```

- **Windows 2008**

```
kadmin: addprinc -e "aes256-cts:normal aes128-cts:normal rc4-hmac:normal"
krbtgt/HADOOP.EXAMPLE.COM@EXAMPLE.COM
```

Note: The cross-realm `krbtgt` principal that you add in this step must have *at least one entry* that uses the same encryption type as the tickets that are issued by the remote KDC. If no entries have the same encryption type, then the problem you will see is that authenticating as a principal in the local realm will allow you to successfully run Hadoop commands, but authenticating as a principal in the remote realm will not allow you to run Hadoop commands.

On all of the cluster hosts

1. In the `/etc/krb5.conf` file on all cluster hosts and all Hadoop client user hosts, configure both Kerberos realms. Note that the default realm and the domain realm should be configured as the local MIT Kerberos realm for the cluster. Your `krb5.conf` will contain more configuration properties than those provided below. This example has been provided to clarify REALM/KDC configuration. Please see our example `krb5.conf` file for more information.

```
[libdefaults]
  default_realm = HADOOP.EXAMPLE.COM
[realms]
  EXAMPLE.COM = {
    kdc = dc01.example.com:88
    admin_server = dc01.example.com:749
    default_domain = example.com
  }
  HADOOP.EXAMPLE.COM = {
    kdc = kdc01.hadoop.example.com:88
    admin_server = kdc01.hadoop.example.com:749
    default_domain = hadoop.example.com
  }
[domain_realm]
  .hadoop.example.com = HADOOP.EXAMPLE.COM
  hadoop.example.com = HADOOP.EXAMPLE.COM
```

```
  .example.com = EXAMPLE.COM
  example.com = EXAMPLE.COM
```

2. Use one of the following methods to properly translate principal names from the Active Directory realm into the cluster-dedicated KDC realm for the Hadoop cluster.
   - **Using Cloudera Manager:** Configure the **Trusted Kerberos realms** property of the HDFS service:
     1. Open the Cloudera Manager Admin Console.
     2. Go to the HDFS service.
     3. Click the **Configuration** tab.
     4. Expand the **Service-Wide** category and click **Security**.
     5. Scroll down to the **Trusted Kerberos Realms** property, and click on the **Value** field to add the name of your central KDC realm. If you need to use more advanced mappings which do more than just allow principals from another domain, you may enter them in the **Additional Rules to Map Kerberos Principals to Short Names** property. For more information about name mapping rules, see Configuring the Mapping from Kerberos Principals to Short Names on page 112.
   - **Using the Command Line:** Configure the `hadoop.security.auth_to_local` setting in the `core-site.xml` file on all of the cluster hosts. The following example translates all principal names with the realm `EXAMPLE.COM` into the first component of the principal name only. It also preserves the standard translation for the default realm (the cluster realm).

```
<property>
  <name>hadoop.security.auth_to_local</name>
  <value>
    RULE:[1:$1@$0](^.*@EXAMPLE\.COM$)s/^(.*)@EXAMPLE\.COM$/$1/g
    RULE:[2:$1@$0](^.*@EXAMPLE\.COM$)s/^(.*)@EXAMPLE\.COM$/$1/g
    DEFAULT
  </value>
</property>
```

Integrating Hadoop Security with Active Directory

Considerations when using an Active Directory KDC

Performance:

As your cluster grows, so will the volume of Authentication Service (AS) and Ticket Granting Service (TGS) interaction between the services on each cluster server. Consider evaluating the volume of this interaction against the Active Directory domain controllers you have configured for the cluster before rolling this feature out to a production environment. If cluster performance suffers, over time it might become necessary to dedicate a set of AD domain controllers to larger deployments.

Network Proximity:

By default, Kerberos uses UDP for client/server communication. Often, AD services are in a different network than project application services such as Hadoop. If the domain controllers supporting a cluster for Kerberos are not in the same subnet, or they're separated by a firewall, consider using the `udp_preference_limit = 1` setting in the `[libdefaults]` section of the `krb5.conf` used by cluster services. Cloudera strongly recommends *against* using AD domain controller (KDC) servers that are separated from the cluster by a WAN connection, as latency in this service will significantly impact cluster performance.

Process:

Troubleshooting the cluster's operations, especially for Kerberos-enabled services, will need to include AD administration resources. Evaluate your organizational processes for engaging the AD administration team, and how to escalate in case a cluster outage occurs due to issues with Kerberos authentication against AD services. In some situations it might be necessary to enable Kerberos event logging to address desktop and KDC issues within windows environments.

> **Important:** With CDH 5.1 and later, clusters managed by Cloudera Manager 5.1 (and later) do not require a local MIT KDC and are able to integrate directly with an Active Directory KDC. Cloudera recommends you use a direct-to-AD setup. For instructions, see Enabling Kerberos Authentication Using the Wizard on page 17.

If direct integration with AD is not currently possible, use the following instructions to configure a local MIT KDC to trust your AD server:

1. Run an MIT Kerberos KDC and realm local to the cluster and create all service principals in this realm.
2. Set up one-way cross-realm trust from this realm to the Active Directory realm. Using this method, there is no need to create service principals in Active Directory, but Active Directory principals (users) can be authenticated to Hadoop. See Configuring a Local MIT Kerberos Realm to Trust Active Directory on page 121.

Configuring a Local MIT Kerberos Realm to Trust Active Directory

On the Active Directory Server

1. Type the following command to add the local realm trust to Active Directory:

```
netdom trust YOUR-LOCAL-REALM.COMPANY.COM /Domain:AD-REALM.COMPANY.COM /add /realm
/passwordt:<TrustPassword>
```

2. Type the following command to set the proper encryption type:

   On Windows 2003 RC2:

```
ktpass /MITRealmName YOUR-LOCAL-REALM.COMPANY.COM /TrustEncryp <enc_type>
```

   On Windows 2008:

```
ksetup /SetEncTypeAttr YOUR-LOCAL-REALM.COMPANY.COM <enc_type>
```

   where the <enc_type> parameter specifies AES, DES, or RC4 encryption. Refer to the documentation for your version of Windows Active Directory to find the <enc_type> parameter string to use.

   > **Important:** Make sure the encryption type you specify is supported on both your version of Windows Active Directory and your version of MIT Kerberos.

On the MIT KDC Server

Type the following command in the kadmin.local or kadmin shell to add the cross-realm krbtgt principal. Use the same password you used in the `netdom` command on the Active Directory Server.

```
kadmin:  addprinc -e "<enc_type_list>"
krbtgt/YOUR-LOCAL-REALM.COMPANY.COM@AD-REALM.COMPANY.COM
```

where the <enc_type_list> parameter specifies the types of encryption this cross-realm krbtgt principal will support: either AES, DES, or RC4 encryption. You can specify multiple encryption types using the parameter in the command above, what's important is that at least one of the encryption types corresponds to the encryption type found in the tickets granted by the KDC in the remote realm. For example:

```
kadmin:  addprinc -e "rc4-hmac:normal des3-hmac-sha1:normal"
krbtgt/YOUR-LOCAL-REALM.COMPANY.COM@AD-REALM.COMPANY.COM
```

> **Note:** The cross-realm krbtgt principal that you add in this step must have *at least one entry* that uses the same encryption type as the tickets that are issued by the remote KDC. If no entries have the same encryption type, then the problem you will see is that authenticating as a principal in the local realm will allow you to successfully run Hadoop commands, but authenticating as a principal in the remote realm will not allow you to run Hadoop commands.

On All of the Cluster Hosts

1. Verify that both Kerberos realms are configured on all of the cluster hosts. Note that the default realm and the domain realm should remain set as the MIT Kerberos realm which is local to the cluster.

```
[realms]
  AD-REALM.CORP.FOO.COM = {
    kdc = ad.corp.foo.com:88
    admin_server = ad.corp.foo.com:749
    default_domain = foo.com
  }
  CLUSTER-REALM.CORP.FOO.COM = {
    kdc = cluster01.corp.foo.com:88
    admin_server = cluster01.corp.foo.com:749
    default_domain = foo.com
  }
```

2. To properly translate principal names from the Active Directory realm into local names within Hadoop, you must configure the `hadoop.security.auth_to_local` setting in the `core-site.xml` file on all of the cluster machines. The following example translates all principal names with the realm `AD-REALM.CORP.FOO.COM` into the first component of the principal name only. It also preserves the standard translation for the default realm (the cluster realm).

```
<property>
  <name>hadoop.security.auth_to_local</name>
  <value>
    RULE:[1:$1@$0](^.*@AD-REALM\.CORP\.FOO\.COM$)s/^(.*)@AD-REALM\.CORP\.FOO\.COM$/$1/g

    RULE:[2:$1@$0](^.*@AD-REALM\.CORP\.FOO\.COM$)s/^(.*)@AD-REALM\.CORP\.FOO\.COM$/$1/g

    DEFAULT
  </value>
</property>
```

For more information about name mapping rules, see Configuring the Mapping from Kerberos Principals to Short Names on page 112.

Integrating Hadoop Security with Alternate Authentication

One of the ramifications of enabling security on a Hadoop cluster is that every user who interacts with the cluster must have a Kerberos principal configured. For some of the services, specifically Oozie and Hadoop (for example, JobTracker and TaskTracker), it can be convenient to run a mixed form of authentication where Kerberos authentication is used for API or command line access while some other form of authentication (for example, SSO and LDAP) is used for accessing Web UIs. Using an alternate authentication deployment is considered an advanced topic because only a partial implementation is provided in this release: you will have to implement some of the code yourself.

> **Note:** The following instructions assume you already have a Kerberos-enabled cluster.

Proceed as follows:

- Configuring the AuthenticationFilter to use Kerberos on page 123
- Creating an AltKerberosAuthenticationHandler Subclass on page 123

- Enabling Your AltKerberosAuthenticationHandler Subclass on page 123

See also the Example Implementation for Oozie on page 124.

Configuring the AuthenticationFilter to use Kerberos

First, you must do all of the steps in the Server Side Configuration section of the Hadoop Auth, Java HTTP SPNEGO Documentation to configure `AuthenticationFilter` to use Kerberos. You must configure `AuthenticationFilter` to use Kerberos before doing the steps below.

Creating an AltKerberosAuthenticationHandler Subclass

An `AuthenticationHandler` is installed on the server-side to handle authenticating clients and creating an `AuthenticationToken`.

1. Subclass the `org.apache.hadoop.security.authentication.server.AltKerberosAuthenticationHandler` class (in the `hadoop-auth` package).
2. When a client sends a request, the authenticate method will be called. For browsers, `AltKerberosAuthenticationHandler` will call the `alternateAuthenticate` method, which is what you need to implement to interact with the desired authentication mechanism. For non-browsers, `AltKerberosAuthenticationHandler` will follow the Kerberos SPNEGO sequence (this is provided for you).
3. The `alternateAuthenticate(HttpServletRequest request, HttpServletResponse response)` method in your subclass should following these rules:
4. Return `null` if the authentication is still in progress; the `response` object can be used to interact with the client.
5. Throw an `AuthenticationException` if the authentication failed.
6. Return an `AuthenticationToken` if the authentication completed successfully.

Enabling Your AltKerberosAuthenticationHandler Subclass

You can enable the alternate authentication on Hadoop Web UIs, Oozie Web UIs, or both. You will need to include a JAR containing your subclass on the classpath of Hadoop or Oozie. All Kerberos-related configuration properties will still apply.

Enabling Your AltKerberosAuthenticationHandler Subclass on Hadoop Web UIs

1. Stop Hadoop by running the following command on every node in your cluster (as root):

```
$ for x in `cd /etc/init.d ; ls hadoop-*` ; do sudo service $x stop ; done
```

2. Set the following property in `core-site.xml`, where `org.my.subclass.of.AltKerberosAuthenticationHandler` is the classname of your subclass:

```
<property>
  <name>hadoop.http.authentication.type</name>
  <value>org.my.subclass.of.AltKerberosAuthenticationHandler</value>
</property>
```

3. (Optional) You can also specify which user-agents you do not want to be considered as browsers by setting the following property as required (default value is shown). Note that all Java-based programs (such as Hadoop client) will use `java` as their user-agent.

```
<property>
  <name>hadoop.http.authentication.alt-kerberos.non-browser.user-agents</name>
  <value>java,curl,wget,perl</value>
</property>
```

4. Copy the JAR containing your subclass into `/usr/lib/hadoop/lib/`.

5. Start Hadoop by running the following command:

```
$ for x in `cd /etc/init.d ; ls hadoop-*` ; do sudo service $x start ; done
```

Enabling Your AltKerberosAuthenticationHandler Subclass on Oozie Web UI

> **Note:**
>
> These instructions assume you have already performed the installation and configuration steps in Oozie Security Configuration.

1. Stop the Oozie Server:

```
sudo /sbin/service oozie stop
```

2. Set the following property in `oozie-site.xml`, where `org.my.subclass.of.AltKerberosAuthenticationHandler` is the classname of your subclass:

```
<property>
  <name>oozie.authentication.type</name>
  <value>org.my.subclass.of.AltKerberosAuthenticationHandler</value>
</property>
```

3. (Optional) You can also specify which user-agents you do not want to be considered as browsers by setting the following property as required (default value is shown). Note that all Java-based programs (such as Hadoop client) will use `java` as their user-agent.

```
<property>
  <name>oozie.authentication.alt-kerberos.non-browser.user-agents</name>
  <value>java,curl,wget,perl</value>
</property>
```

4. Copy the JAR containing your subclass into `/var/lib/oozie`.
5. Start the Oozie Server:

```
sudo /sbin/service oozie start
```

Example Implementation for Oozie

> **Warning:**
>
> The example implementation is **NOT SECURE**. Its purpose is to be as simple as possible, as an example of how to write your own `AltKerberosAuthenticationHandler` subclass.
>
> **It should NOT be used in a production environment**

An example implementation of `AltKerberosAuthenticationHandler` is included (though not built by default) with Oozie. Also included is a simple Login Server with two implementations. The first one will authenticate any user who is using a username and password that are identical, such as `foo:foo`. The second one can be configured against an LDAP server to use LDAP for authentication.

You can read comprehensive documentation on the example at Creating Custom Authentication.

Important:

If you installed Oozie from the CDH packages and are deploying `oozie-login.war` alongside `oozie.war`, you will also need to run the following commands after you copy the `oozie-login.war` file to `/usr/lib/oozie/oozie-server` (if using YARN or `/usr/lib/oozie/oozie-server-0.20` if using MRv1) because it won't automatically be expanded:

```
jar xvf oozie-login.war
mkdir oozie-login
mv META-INF oozie-login/
mv WEB-INF oozie-login/
```

Configuring LDAP Group Mappings

Important: Cloudera strongly recommends *against* using Hadoop's `LdapGroupsMapping` provider. `LdapGroupsMapping` should only be used in cases where OS-level integration is not possible. Production clusters require an identity provider that works well with all applications, not just Hadoop. Hence, often the preferred mechanism is to use tools such as SSSD, VAS or Centrify to replicate LDAP groups.

When configuring LDAP for group mappings in Hadoop, you must create the users and groups for your Hadoop services in LDAP. When using the default shell-based group mapping provider (`org.apache.hadoop.security.ShellBasedUnixGroupsMapping`), the requisite user and group relationships already exist because they are created during the installation procedure. When you switch to LDAP as the group mapping provider, you must re-create these relationships within LDAP.

Note that if you have modified the **System User** or **System Group** setting within Cloudera Manager for any service, you must use those custom values to provision the users and groups in LDAP.

The table below lists users and their group members for CDH services:

Note: Cloudera Manager 5.3 introduces a new ***single user mode***. In single user mode, the Cloudera Manager Agent and *all the processes run by services managed by Cloudera Manager* are started as a single configured user and group. See Single User Mode Requirements for more information.

Table 1: Users and Groups

Component (Version)	Unix User ID	Groups	Notes
Cloudera Manager (all versions)	`cloudera-scm`	`cloudera-scm`	Cloudera Manager processes such as the Cloudera Manager Server and the monitoring roles run as this user. The Cloudera Manager keytab file must be named `cmf.keytab` since that name is hard-coded in Cloudera Manager. **Note:** Applicable to clusters managed by Cloudera Manager only.

Component (Version)	Unix User ID	Groups	Notes
Apache Accumulo (Accumulo 1.4.3 and higher)	`accumulo`	`accumulo`	Accumulo processes run as this user.
Apache Avro			No special users.
Apache Flume (CDH 4, CDH 5)	`flume`	`flume`	The sink that writes to HDFS as this user must have write privileges.
Apache HBase (CDH 4, CDH 5)	`hbase`	`hbase`	The Master and the RegionServer processes run as this user.
HDFS (CDH 4, CDH 5)	`hdfs`	`hdfs, hadoop`	The NameNode and DataNodes run as this user, and the HDFS root directory as well as the directories used for edit logs should be owned by it.
Apache Hive (CDH 4, CDH 5)	`hive`	`hive`	The HiveServer2 process and the Hive Metastore processes run as this user. A user must be defined for Hive access to its Metastore DB (for example, MySQL or Postgres) but it can be any identifier and does not correspond to a Unix uid. This is `javax.jdo.option.ConnectionUserName` in hive-site.xml.
Apache HCatalog (CDH 4.2 and higher, CDH 5)	`hive`	`hive`	The WebHCat service (for REST access to Hive functionality) runs as the `hive` user.
HttpFS (CDH 4, CDH 5)	`httpfs`	`httpfs`	The HttpFS service runs as this user. See HttpFS Security Configuration for instructions on how to generate the merged `httpfs-http.keytab` file.
Hue (CDH 4, CDH 5)	`hue`	`hue`	Hue services run as this user.
Cloudera Impala (CDH 4.1 and higher, CDH 5)	`impala`	`impala, hadoop, hdfs, hive`	Impala services run as this user.
Apache Kafka (Cloudera Distribution of Kafka 1.2.0)	`kafka`	`kafka`	Kafka services run as this user.
KMS (File) (CDH 5. 2.1 and higher)	`kms`	`kms`	The KMS (File) service runs as this user.
KMS (Navigator Key Trustee) (CDH 5.3 and higher)	`kms`	`kms`	The KMS (Navigator Key Trustee) service runs as this user.
Llama (CDH 5)	`llama`	`llama`	Llama runs as this user.
Apache Mahout			No special users.
MapReduce (CDH 4, CDH 5)	`mapred`	`mapred, hadoop`	Without Kerberos, the JobTracker and tasks run as this user. The LinuxTaskController binary is owned by this user for Kerberos.

Component (Version)	Unix User ID	Groups	Notes
Apache Oozie (CDH 4, CDH 5)	`oozie`	`oozie`	The Oozie service runs as this user.
Parquet			No special users.
Apache Pig			No special users.
Cloudera Search (CDH 4.3 and higher, CDH 5)	`solr`	`solr`	The Solr processes run as this user.
Apache Spark (CDH 5)	`spark`	`spark`	The Spark History Server process runs as this user.
Apache Sentry (incubating) (CDH 5.1 and higher)	`sentry`	`sentry`	The Sentry service runs as this user.
Apache Sqoop (CDH 4, CDH 5)	`sqoop`	`sqoop`	This user is only for the Sqoop1 Metastore, a configuration option that is not recommended.
Apache Sqoop2 (CDH 4.2 and higher, CDH 5)	`sqoop2`	`sqoop, sqoop2`	The Sqoop2 service runs as this user.
Apache Whirr			No special users.
YARN (CDH 4, CDH 5)	`yarn`	`yarn, hadoop`	Without Kerberos, all YARN services and applications run as this user. The LinuxContainerExecutor binary is owned by this user for Kerberos.
Apache ZooKeeper (CDH 4, CDH 5)	`zookeeper`	`zookeeper`	The ZooKeeper processes run as this user. It is not configurable.

Important:

- You can use either Cloudera Manager or the following command-line instructions to complete this configuration.
- This information applies specifically to CDH 5.3.x . If you use an earlier version of CDH, see the documentation for that version located at Cloudera Documentation.

Using Cloudera Manager

Minimum Required Role: Configurator (also provided by **Cluster Administrator, Full Administrator**)

Make the following changes to the HDFS service's security configuration:

1. Open the Cloudera Manager Admin Console and navigate to the **HDFS** service.
2. Click the **Configuration** tab.
3. Modify the following configuration properties under the **Service-Wide > Security** section. The table below lists the properties and the value to be set for each property.

Configuration Property	Value
Hadoop User Group Mapping Implementation	`org.apache.hadoop.security.LdapGroupsMapping`
Hadoop User Group Mapping LDAP URL	`ldap://<server>`
Hadoop User Group Mapping LDAP Bind User	`Administrator@example.com`

Configuration Property	Value
Hadoop User Group Mapping LDAP Bind User Password	`***`
Hadoop User Group Mapping Search Base	`dc=example,dc=com`

Although the above changes are sufficient to configure group mappings for Active Directory, some changes to the remaining default configurations might be required for OpenLDAP.

Using the Command Line

Add the following properties to the `core-site.xml` on the NameNode:

```
<property>
<name>hadoop.security.group.mapping</name>
<value>org.apache.hadoop.security.LdapGroupsMapping</value>
</property>

<property>
<name>hadoop.security.group.mapping.ldap.url</name>
<value>ldap://server</value>
</property>

<property>
<name>hadoop.security.group.mapping.ldap.bind.user</name>
<value>Administrator@example.com</value>
</property>

<property>
<name>hadoop.security.group.mapping.ldap.bind.password</name>
<value>****</value>
</property>

<property>
<name>hadoop.security.group.mapping.ldap.base</name>
<value>dc=example,dc=com</value>
</property>

<property>
<name>hadoop.security.group.mapping.ldap.search.filter.user</name>
<value>(&amp;(objectClass=user)(sAMAccountName={0}))</value>
</property>

<property>
<name>hadoop.security.group.mapping.ldap.search.filter.group</name>
<value>(objectClass=group)</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>
```

Note: In addition:

- If you are using Sentry with Hive, you will also need to add these properties on the HiveServer2 node.
- If you are using Sentry with Impala, add these properties on all hosts

See Users and Groups in Sentry for more information.

Hadoop Users in Cloudera Manager and CDH

A number of special users are created by default when installing and using CDH and Cloudera Manager. Given below is a list of users and groups as of the latest release. Also listed are the corresponding Kerberos principals and keytab files that should be created when you configure Kerberos security on your cluster.

> **Note:** Cloudera Manager 5.3 introduces a new ***single user mode***. In single user mode, the Cloudera Manager Agent and *all the processes run by services managed by Cloudera Manager* are started as a single configured user and group. See Single User Mode Requirements for more information.

Table 2: Users and Groups

Component (Version)	Unix User ID	Groups	Notes
Cloudera Manager (all versions)	`cloudera-scm`	`cloudera-scm`	Cloudera Manager processes such as the Cloudera Manager Server and the monitoring roles run as this user. The Cloudera Manager keytab file must be named `cmf.keytab` since that name is hard-coded in Cloudera Manager. **Note:** Applicable to clusters managed by Cloudera Manager only.
Apache Accumulo (Accumulo 1.4.3 and higher)	`accumulo`	`accumulo`	Accumulo processes run as this user.
Apache Avro			No special users.
Apache Flume (CDH 4, CDH 5)	`flume`	`flume`	The sink that writes to HDFS as this user must have write privileges.
Apache HBase (CDH 4, CDH 5)	`hbase`	`hbase`	The Master and the RegionServer processes run as this user.
HDFS (CDH 4, CDH 5)	`hdfs`	`hdfs, hadoop`	The NameNode and DataNodes run as this user, and the HDFS root directory as well as the directories used for edit logs should be owned by it.
Apache Hive (CDH 4, CDH 5)	`hive`	`hive`	The HiveServer2 process and the Hive Metastore processes run as this user. A user must be defined for Hive access to its Metastore DB (for example, MySQL or Postgres) but it can be any identifier and does not correspond to a Unix uid. This is `javax.jdo.option.ConnectionUserName` in hive-site.xml.
Apache HCatalog (CDH 4.2 and higher, CDH 5)	`hive`	`hive`	The WebHCat service (for REST access to Hive functionality) runs as the `hive` user.

Component (Version)	Unix User ID	Groups	Notes
HttpFS (CDH 4, CDH 5)	`httpfs`	`httpfs`	The HttpFS service runs as this user. See HttpFS Security Configuration for instructions on how to generate the merged `httpfs-http.keytab` file.
Hue (CDH 4, CDH 5)	`hue`	`hue`	Hue services run as this user.
Cloudera Impala (CDH 4.1 and higher, CDH 5)	`impala`	`impala, hadoop, hdfs, hive`	Impala services run as this user.
Apache Kafka (Cloudera Distribution of Kafka 1.2.0)	`kafka`	`kafka`	Kafka services run as this user.
KMS (File) (CDH 5. 2.1 and higher)	`kms`	`kms`	The KMS (File) service runs as this user.
KMS (Navigator Key Trustee) (CDH 5.3 and higher)	`kms`	`kms`	The KMS (Navigator Key Trustee) service runs as this user.
Llama (CDH 5)	`llama`	`llama`	Llama runs as this user.
Apache Mahout			No special users.
MapReduce (CDH 4, CDH 5)	`mapred`	`mapred, hadoop`	Without Kerberos, the JobTracker and tasks run as this user. The LinuxTaskController binary is owned by this user for Kerberos.
Apache Oozie (CDH 4, CDH 5)	`oozie`	`oozie`	The Oozie service runs as this user.
Parquet			No special users.
Apache Pig			No special users.
Cloudera Search (CDH 4.3 and higher, CDH 5)	`solr`	`solr`	The Solr processes run as this user.
Apache Spark (CDH 5)	`spark`	`spark`	The Spark History Server process runs as this user.
Apache Sentry (incubating) (CDH 5.1 and higher)	`sentry`	`sentry`	The Sentry service runs as this user.
Apache Sqoop (CDH 4, CDH 5)	`sqoop`	`sqoop`	This user is only for the Sqoop1 Metastore, a configuration option that is not recommended.
Apache Sqoop2 (CDH 4.2 and higher, CDH 5)	`sqoop2`	`sqoop, sqoop2`	The Sqoop2 service runs as this user.
Apache Whirr			No special users.
YARN (CDH 4, CDH 5)	`yarn`	`yarn, hadoop`	Without Kerberos, all YARN services and applications run as this user. The LinuxContainerExecutor binary is owned by this user for Kerberos.

Component (Version)	Unix User ID	Groups	Notes
Apache ZooKeeper (CDH 4, CDH 5)	`zookeeper`	`zookeeper`	The ZooKeeper processes run as this user. It is not configurable.

Keytabs and Keytab File Permissions

Note:

The Kerberos principal names should be of the format, `username/fully.qualified.domain.name@YOUR-REALM.COM`, where the term `username` refers to the username of an existing UNIX account, such as `hdfs` or `mapred`. The table below lists the usernames to be used for the Kerberos principal names. For example, the Kerberos principal for Apache Flume would be `flume/fully.qualified.domain.name@YOUR-REALM.COM`.

For keytabs with multiple principals, Cloudera Manager merges them appropriately from individual keytabs. If you do not use Cloudera Manager, you must merge the keytabs manually.

Table 3: Clusters Managed by Cloudera Manager

Component (Unix User ID)	Service	Kerberos Principals	Filename (*.keytab)	Keytab File Owner	Keytab File Group	File Permission (octal)
Cloudera Manager (`cloudera-scm`)	NA	cloudera-scm	cmf	cloudera-scm	cloudera-scm	600
Cloudera Management Service (`cloudera-scm`)	cloudera-mgmt-REPORTSMANAGER	hdfs	headlamp	cloudera-scm	cloudera-scm	600
Cloudera Management Service (`cloudera-scm`)	cloudera-mgmt-SERVICEMONITOR, cloudera-mgmt-ACTIVITYMONITOR	hue	cmon	cloudera-scm	cloudera-scm	600
Cloudera Management Service (`cloudera-scm`)	cloudera-mgmt-HOSTMONITOR	N/A	N/A	N/A	N/A	N/A
Apache Accumulo (`accumulo`)	accumulo16-ACCUMULO16_MASTER	accumulo	accumulo16	cloudera-scm	cloudera-scm	600
	accumulo16-ACCUMULO16_TRACER					
	accumulo16-ACCUMULO16_MONITOR					
	accumulo16-ACCUMULO16_GC					
	accumulo16-ACCUMULO16_TSERVER					
Flume (`flume`)	flume-AGENT	flume	flume	cloudera-scm	cloudera-scm	600
HBase (`hbase`)	hbase-REGIONSERVER	hbase	hbase	cloudera-scm	cloudera-scm	600
	hbase-HBASETHRIFTSERVER					
	hbase- HBASERESTSERVER					

Component (Unix User ID)	Service	Kerberos Principals	Filename (*.keytab)	Keytab File Owner	Keytab File Group	File Permission (octal)
	hbase-MASTER					
HDFS (`hdfs`)	hdfs-NAMENODE	hdfs, HTTP	hdfs	cloudera-scm	cloudera-scm	600
	hdfs-DATANODE					
	hdfs-SECONDARYNAMENODE					
Hive (`hive`)	hive-HIVESERVER2	hive	hive	cloudera-scm	cloudera-scm	600
	hive-WEBHCAT	HTTP	HTTP			
	hive-HIVEMETASTORE	hive	hive			
HttpFS (`httpfs`)	hdfs-HTTPFS	httpfs	httpfs	cloudera-scm	cloudera-scm	600
Hue (`hue`)	hue-KT_RENEWER	hue	hue	cloudera-scm	cloudera-scm	600
Impala (`impala`)	impala-STATESTORE	impala	impala	cloudera-scm	cloudera-scm	600
	impala-CATALOGSERVER					
	impala-IMPALAD					
KMS (File) (`kms`)	kms-KMS	HTTP	kms	cloudera-scm	cloudera-scm	600
KMS (Navigator Key Trustee) (`kms`)	keytrustee-KMS_KEYTRUSTEE	HTTP	keytrustee	cloudera-scm	cloudera-scm	600
Llama (`llama`)	impala-LLAMA	llama, HTTP	llama	cloudera-scm	cloudera-scm	600
MapReduce (`mapred`)	mapreduce-JOBTRACKER	mapred, HTTP	mapred	cloudera-scm	cloudera-scm	600
	mapreduce- TASKTRACKER					
Oozie (`oozie`)	oozie-OOZIE_SERVER	oozie, HTTP	oozie	cloudera-scm	cloudera-scm	600
Search (`solr`)	solr-SOLR_SERVER	solr, HTTP	solr	cloudera-scm	cloudera-scm	600
Sentry (`sentry`)	sentry-SENTRY_SERVER	sentry	sentry	cloudera-scm	cloudera-scm	600
Spark (`spark`)	spark_on_yarn-SPARK_YARN_HISTORY_SERVER	spark	spark	cloudera-scm	cloudera-scm	600
YARN (`yarn`)	yarn-NODEMANAGER	yarn, HTTP	yarn	cloudera-scm	cloudera-scm	644
	yarn- RESOURCEMANAGER					600
	yarn-JOBHISTORY					600
ZooKeeper (`zookeeper`)	zookeeper-server	zookeeper	zookeeper	cloudera-scm	cloudera-scm	600

Table 4: CDH Clusters Not Managed by Cloudera Manager

Component (Unix User ID)	Service	Kerberos Principals	Filename (*.keytab)	Keytab File Owner	Keytab File Group	File Permission (octal)
Apache Accumulo (`accumulo`)	accumulo16-ACCUMULO16_MASTER	accumulo	accumulo16	accumulo	accumulo	600
	accumulo16-ACCUMULO16_TRACER					

Component (Unix User ID)	Service	Kerberos Principals	Filename (*.keytab)	Keytab File Owner	Keytab File Group	File Permission (octal)
	accumulo16-ACCUMULO16_MONITOR					
	accumulo16-ACCUMULO16_GC					
	accumulo16-ACCUMULO16_TSERVER					
Flume (`flume`)	flume-AGENT	flume	flume	flume	flume	600
HBase (`hbase`)	hbase-REGIONSERVER	hbase	hbase	hbase	hbase	600
	hbase- HBASETHRIFTSERVER					
	hbase- HBASERESTSERVER					
	hbase-MASTER					
HDFS (`hdfs`)	hdfs-NAMENODE	hdfs, HTTP	hdfs	hdfs	hdfs	600
	hdfs-DATANODE					
	hdfs- SECONDARYNAMENODE					
Hive (`hive`)	hive-HIVESERVER2	hive	hive	hive	hive	600
	hive-WEBHCAT	HTTP	HTTP			
	hive-HIVEMETASTORE	hive	hive			
HttpFS (`httpfs`)	hdfs-HTTPFS	httpfs	httpfs	httpfs	httpfs	600
Hue (hue)	hue-KT_RENEWER	hue	hue	hue	hue	600
Impala (`impala`)	impala-STATESTORE	impala	impala	impala	impala	600
	impala-CATALOGSERVER					
	impala-IMPALAD					
Llama (`llama`)	impala-LLAMA	llama, HTTP	llama	llama	llama	600
KMS (File) (`kms`)	kms-KMS	HTTP	kms	kms	kms	600
MapReduce (`mapred`)	mapreduce-JOBTRACKER	mapred, HTTP	mapred	mapred	hadoop	600
	mapreduce- TASKTRACKER					
Oozie (`oozie`)	oozie-OOZIE_SERVER	oozie, HTTP	oozie	oozie	oozie	600
Search (`solr`)	solr-SOLR_SERVER	solr, HTTP	solr	solr	solr	600
Sentry (`sentry`)	sentry-SENTRY_SERVER	sentry	sentry	sentry	sentry	600
Spark (`spark`)	spark_on_yarn-SPARK_YARN_HISTORY_SERVER	spark	spark	spark	spark	600
YARN (`yarn`)	yarn-NODEMANAGER	yarn, HTTP	yarn	yarn	hadoop	644
	yarn- RESOURCEMANAGER					600
	yarn-JOBHISTORY					600
ZooKeeper (`zookeeper`)	zookeeper-server	zookeeper	zookeeper	zookeeper	zookeeper	600

Authenticating Kerberos Principals in Java Code

This topic provides an example of how to authenticate a Kerberos principal in a Java application using the `org.apache.hadoop.security.UserGroupInformation` class.

The following code snippet authenticates the `cloudera` principal using the `cloudera.keytab` file:

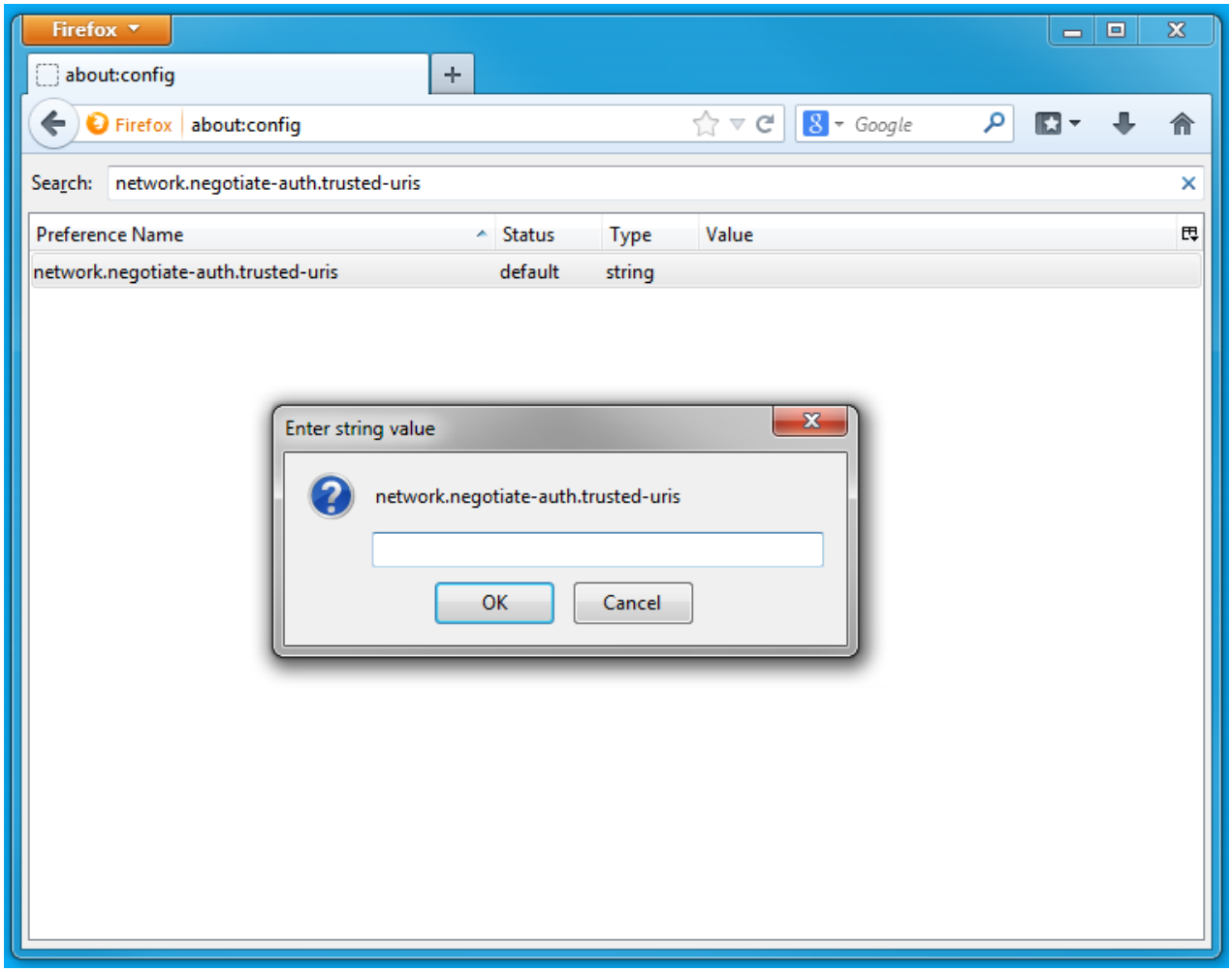
```
// Authenticating Kerberos principal
System.out.println("Principal Authentication: ");
final String user = "cloudera@CLOUDERA.COM";
final String keyPath = "cloudera.keytab";
UserGroupInformation.loginUserFromKeytab(user, keyPath);
```

Using a Web Browser to Access an URL Protected by Kerberos HTTP SPNEGO

To access an URL protected by Kerberos HTTP SPNEGO, use the following instructions for the browser you are using.

To configure Mozilla Firefox:

1. Open the low level Firefox configuration page by loading the `about:config` page.
2. In the **Search** text box, enter: `network.negotiate-auth.trusted-uris`
3. Double-click the `network.negotiate-auth.trusted-uris` preference and enter the hostname or the domain of the web server that is protected by Kerberos HTTP SPNEGO. Separate multiple domains and hostnames with a comma.
4. Click **OK**.

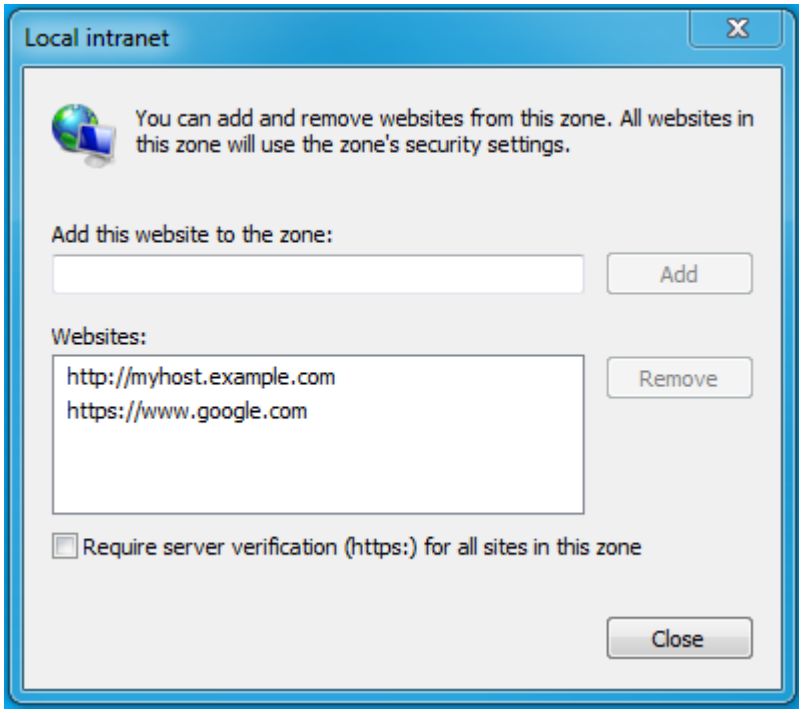


To configure Internet Explorer:

Follow the instructions given below to configure Internet Explorer to access URLs protected by

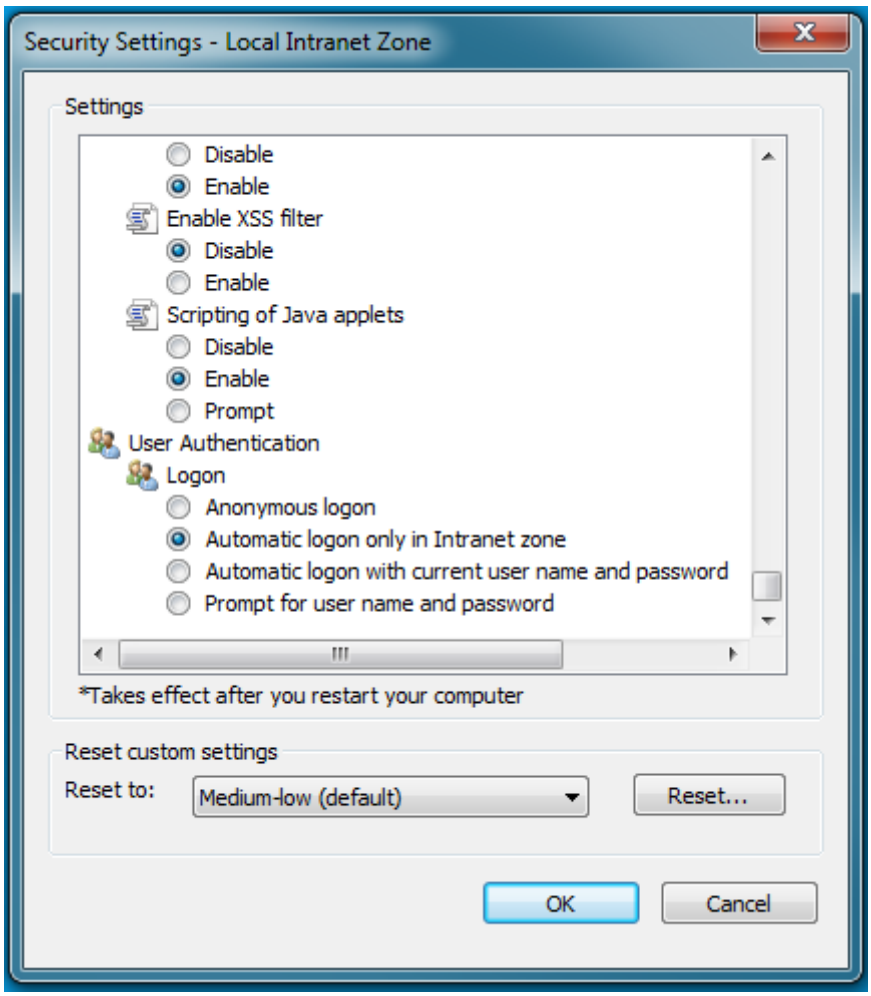
Configuring the Local Intranet Domain

1. Open Internet Explorer and click the Settings "gear" icon in the top-right corner. Select **Internet options**.
2. Select the **Security** tab.
3. Select the **Local Intranet** zone and click the **Sites** button.
4. Make sure that the first two options, **Include all local (intranet) sites not listed in other zones** and **Include all sites that bypass the proxy server** are checked.
5. Click **Advanced** and add the names of the domains that are protected by Kerberos HTTP SPNEGO, one at a time, to the list of websites. For example, `myhost.example.com`. Click **Close**.
6. Click **OK** to save your configuration changes.



Configuring Intranet Authentication

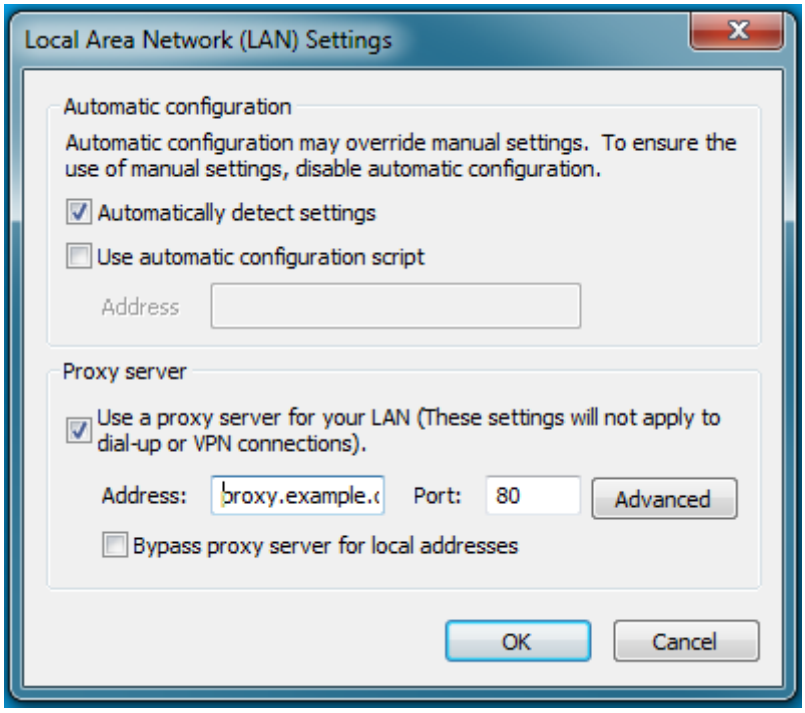
1. Click the Settings "gear" icon in the top-right corner. Select **Internet options**.
2. Select the **Security** tab.
3. Select the **Local Intranet** zone and click the **Custom level...** button to open the **Security Settings - Local Intranet Zone** dialog box.
4. Scroll down to the **User Authentication** options and select **Automatic logon only in Intranet zone**.
5. Click **OK** to save these changes.



Verifying Proxy Settings

You need to perform the following steps only if you have a proxy server already enabled.

1. Click the Settings "gear" icon in the top-right corner. Select **Internet options**.
2. Select the **Connections** tab and click **LAN Settings**.
3. Verify that the proxy server **Address** and **Port** number settings are correct.
4. Click **Advanced** to open the **Proxy Settings** dialog box.
5. Add the Kerberos-protected domains to the **Exceptions** field.
6. Click **OK** to save any changes.



To configure Google Chrome:

If you are using Windows, use the Control Panel to navigate to the **Internet Options** dialog box. Configuration changes required are the same as those described above for Internet Explorer.

On MacOS or Linux, add the `--auth-server-whitelist` parameter to the `google-chrome` command. For example, to run Chrome from a Linux prompt, run the `google-chrome` command as follows,

```
> google-chrome --auth-server-whitelist = "hostname/domain"
```

Troubleshooting Authentication Issues

Typically, if there are problems with security, Hadoop will display generic messages about the cause of the problem. This topic contains some sample Kerberos configuration files for your reference. It also has solutions to potential problems you might face when configuring a secure cluster:

Sample Kerberos Configuration files: krb5.conf, kdc.conf, kadm5.acl

`kdc.conf`:

```
[kdcdefaults]
 kdc_ports = 88
 kdc_tcp_ports = 88

[realms]
  EXAMPLE.COM = {
  #master_key_type = aes256-cts
  max_renewable_life = 7d 0h 0m 0s
  acl_file = /var/kerberos/krb5kdc/kadm5.acl
  dict_file = /usr/share/dict/words
  admin_keytab = /var/kerberos/krb5kdc/kadm5.keytab
# note that aes256 is ONLY supported in Active Directory in a domain / forrest operating
 at a 2008 or greater functional level.
# aes256 requires that you download and deploy the JCE Policy files for your JDK release
 level to provide
```

```
# strong java encryption extension levels like AES256. Make sure to match based on the
 encryption configured within AD for
# cross realm auth, note that RC4 = arcfour when comparing windows and linux enctypes
  supported_enctypes = aes256-cts:normal aes128-cts:normal arcfour-hmac:normal
  default_principal_flags = +renewable, +forwardable
 }
```

krb5.conf:

```
[logging]
 default = FILE:/var/log/krb5libs.log
 kdc = FILE:/var/log/krb5kdc.log
 admin_server = FILE:/var/log/kadmind.log

[libdefaults]
 default_realm = EXAMPLE.COM
 dns_lookup_realm = false
 dns_lookup_kdc = false
 ticket_lifetime = 24h
 renew_lifetime = 7d
 forwardable = true
# udp_preference_limit = 1

# set udp_preference_limit = 1 when TCP only should be
# used. Consider using in complex network environments when
# troubleshooting or when dealing with inconsistent
# client behavior or GSS (63) messages.

# uncomment the following if AD cross realm auth is ONLY providing DES encrypted tickets
# allow-weak-crypto = true

[realms]
 AD-REALM.EXAMPLE.COM = {
  kdc = AD1.ad-realm.example.com:88
  kdc = AD2.ad-realm.example.com:88
  admin_server = AD1.ad-realm.example.com:749
  admin_server = AD2.ad-realm.example.com:749
  default_domain = ad-realm.example.com
 }
 EXAMPLE.COM = {
  kdc = kdc1.example.com:88
  admin_server = kdc1.example.com:749
  default_domain = example.com
 }

# The domain_realm is critical for mapping your host domain names to the kerberos realms
# that are servicing them. Make sure the lowercase left hand portion indicates any
domains or subdomains
# that will be related to the kerberos REALM on the right hand side of the expression.
 REALMs will
# always be UPPERCASE. For example, if your actual DNS domain was test.com but your
kerberos REALM is
# EXAMPLE.COM then you would have,

[domain_realm]
test.com = EXAMPLE.COM
#AD domains and realms are usually the same
ad-domain.example.com = AD-REALM.EXAMPLE.COM
ad-realm.example.com = AD-REALM.EXAMPLE.COM
```

kadm5.acl:

```
*/admin@HADOOP.COM *
cloudera-scm@HADOOP.COM * flume/*@HADOOP.COM
cloudera-scm@HADOOP.COM * hbase/*@HADOOP.COM
cloudera-scm@HADOOP.COM * hdfs/*@HADOOP.COM
cloudera-scm@HADOOP.COM * hive/*@HADOOP.COM
cloudera-scm@HADOOP.COM * httpfs/*@HADOOP.COM
cloudera-scm@HADOOP.COM * HTTP/*@HADOOP.COM
cloudera-scm@HADOOP.COM * hue/*@HADOOP.COM
```

```
cloudera-scm@HADOOP.COM * impala/*@HADOOP.COM
cloudera-scm@HADOOP.COM * mapred/*@HADOOP.COM
cloudera-scm@HADOOP.COM * oozie/*@HADOOP.COM
cloudera-scm@HADOOP.COM * solr/*@HADOOP.COM
cloudera-scm@HADOOP.COM * sqoop/*@HADOOP.COM
cloudera-scm@HADOOP.COM * yarn/*@HADOOP.COM
cloudera-scm@HADOOP.COM * zookeeper/*@HADOOP.COM
```

Potential Security Problems and Their Solutions

This Troubleshooting appendix contains sample Kerberos configuration files, `krb5.conf` and `kdc.conf` for your reference. It also has solutions to potential problems you might face when configuring a secure cluster:

Issues with Generate Credentials

Cloudera Manager uses a command called **Generate Credentials** to create the accounts needed by CDH for enabling authentication using Kerberos. The command is triggered automatically when you are using the Kerberos Wizard or making changes to your cluster that will require new Kerberos principals.

When configuring Kerberos, if CDH services do not start, and on the Cloudera Manager Home page you see a validation error, `Role is missing Kerberos keytab`, it means the **Generate Credentials** command failed. To see the output of the command, navigate to the Home page and click the **All Recent Commands** tab.

Here are some common error messages:

Problems	Possible Causes	Solutions
With Active Directory		
`ldap_sasl_interactive_bind_s: Can't contact LDAP server (-1)`	The Domain Controller specified is incorrect or LDAPS has not been enabled for it.	Verify the KDC configuration by going to the Cloudera Manager Admin Console and navigate to **Administration> Settings> Kerberos**. Also check that LDAPS is enabled for Active Directory.
`ldap_add: Insufficient access (50)`	The Active Directory account you are using for Cloudera Manager does not have permissions to create other accounts.	Use the Delegate Control wizard to grant permission to the Cloudera Manager account to create other accounts. You can also login to Active Directory as the Cloudera Manager user to check that it can create other accounts in your Organizational Unit.
With MIT KDC		
`kadmin: Cannot resolve network address for admin server in requested realm while initializing kadmin interface.`	The hostname for the KDC server is incorrect.	Check the `kdc` field for your default realm in `krb5.conf` and make sure the hostname is correct.

Running any Hadoop command fails after enabling security.

Description:

A user must have a valid Kerberos ticket in order to interact with a secure Hadoop cluster. Running any Hadoop command (such as `hadoop fs -ls`) will fail if you do not have a valid Kerberos ticket in your credentials cache. If you do not have a valid ticket, you will receive an error such as:

```
11/01/04 12:08:12 WARN ipc.Client: Exception encountered while connecting to the server
 : javax.security.sasl.SaslException:
GSS initiate failed [Caused by GSSException: No valid credentials provided (Mechanism
level: Failed to find any Kerberos tgt)]
```

```
Bad connection to FS. command aborted. exception: Call to nn-host/10.0.0.2:8020 failed
 on local exception: java.io.IOException:
javax.security.sasl.SaslException: GSS initiate failed [Caused by GSSException: No valid
 credentials provided (Mechanism level: Failed to find any Kerberos tgt)]
```

Solution:

You can examine the Kerberos tickets currently in your credentials cache by running the `klist` command. You can obtain a ticket by running the `kinit` command and either specifying a keytab file containing credentials, or entering the password for your principal.

Java is unable to read the Kerberos credentials cache created by versions of MIT Kerberos 1.8.1 or higher.

Description:

If you are running MIT Kerberos 1.8.1 or higher, the following error will occur when you attempt to interact with the Hadoop cluster, even after successfully obtaining a Kerberos ticket using `kinit`:

```
11/01/04 12:08:12 WARN ipc.Client: Exception encountered while connecting to the server
 : javax.security.sasl.SaslException:
GSS initiate failed [Caused by GSSException: No valid credentials provided (Mechanism
level: Failed to find any Kerberos tgt)]
Bad connection to FS. command aborted. exception: Call to nn-host/10.0.0.2:8020 failed
 on local exception: java.io.IOException:
javax.security.sasl.SaslException: GSS initiate failed [Caused by GSSException: No valid
 credentials provided (Mechanism level: Failed to find any Kerberos tgt)]
```

Because of a change [1] in the format in which MIT Kerberos writes its credentials cache, there is a bug [2] in the Oracle JDK 6 Update 26 and earlier that causes Java to be unable to read the Kerberos credentials cache created by versions of MIT Kerberos 1.8.1 or higher. Kerberos 1.8.1 is the default in Ubuntu Lucid and higher releases and Debian Squeeze and higher releases. (On RHEL and CentOS, an older version of MIT Kerberos which does not have this issue, is the default.)

Footnotes:

[1] MIT Kerberos change: http://krbdev.mit.edu/rt/Ticket/Display.html?id=6206

[2] Report of bug in Oracle JDK 6 Update 26 and lower:
http://bugs.sun.com/bugdatabase/view_bug.do?bug_id=6979329

Solution:

If you encounter this problem, you can work around it by running `kinit -R` after running `kinit` initially to obtain credentials. Doing so will cause the ticket to be renewed, and the credentials cache rewritten in a format which Java can read. To illustrate this:

```
$ klist
klist: No credentials cache found (ticket cache FILE:/tmp/krb5cc_1000)
$ hadoop fs -ls
11/01/04 13:15:51 WARN ipc.Client: Exception encountered while connecting to the server
 : javax.security.sasl.SaslException:
GSS initiate failed [Caused by GSSException: No valid credentials provided (Mechanism
level: Failed to find any Kerberos tgt)]
Bad connection to FS. command aborted. exception: Call to nn-host/10.0.0.2:8020 failed
 on local exception: java.io.IOException:
javax.security.sasl.SaslException: GSS initiate failed [Caused by GSSException: No valid
 credentials provided (Mechanism level: Failed to find any Kerberos tgt)]
$ kinit
Password for atm@YOUR-REALM.COM:
$ klist
Ticket cache: FILE:/tmp/krb5cc_1000
Default principal: atm@YOUR-REALM.COM

Valid starting     Expires            Service principal
01/04/11 13:19:31  01/04/11 23:19:31  krbtgt/YOUR-REALM.COM@YOUR-REALM.COM
```

```
 renew until 01/05/11 13:19:30
$ hadoop fs -ls
11/01/04 13:15:59 WARN ipc.Client: Exception encountered while connecting to the server
 : javax.security.sasl.SaslException:
GSS initiate failed [Caused by GSSException: No valid credentials provided (Mechanism
level: Failed to find any Kerberos tgt)]
Bad connection to FS. command aborted. exception: Call to nn-host/10.0.0.2:8020 failed
 on local exception: java.io.IOException:
javax.security.sasl.SaslException: GSS initiate failed [Caused by GSSException: No valid
 credentials provided (Mechanism level: Failed to find any Kerberos tgt)]
$ kinit -R
$ hadoop fs -ls
Found 6 items
drwx------   - atm atm          0 2011-01-02 16:16 /user/atm/.staging
```

Note:

This workaround for Problem 2 requires the initial ticket to be renewable. Note that whether or not you can obtain renewable tickets is dependent upon a KDC-wide setting, as well as a per-principal setting for both the principal in question and the Ticket Granting Ticket (TGT) service principal for the realm. A non-renewable ticket will have the same values for its "valid starting" and "renew until" times. If the initial ticket is not renewable, the following error message is displayed when attempting to renew the ticket:

```
kinit: Ticket expired while renewing credentials
```

java.io.IOException: Incorrect permission

Description:

An error such as the following example is displayed if the user running one of the Hadoop daemons has a umask of 0002, instead of 0022:

```
java.io.IOException: Incorrect permission for
/var/folders/B3/B3d2vCm4F+mmWzVPB89W6E+++TI/-Tmp-/tmpYTil84/dfs/data/data1,
expected: rwxr-xr-x, while actual: rwxrwxr-x
        at org.apache.hadoop.util.DiskChecker.checkPermission(DiskChecker.java:107)
        at
org.apache.hadoop.util.DiskChecker.mkdirsWithExistsAndPermissionCheck(DiskChecker.java:144)

        at org.apache.hadoop.util.DiskChecker.checkDir(DiskChecker.java:160)
       at org.apache.hadoop.hdfs.server.datanode.DataNode.makeInstance(DataNode.java:1484)

        at
org.apache.hadoop.hdfs.server.datanode.DataNode.instantiateDataNode(DataNode.java:1432)

        at
org.apache.hadoop.hdfs.server.datanode.DataNode.instantiateDataNode(DataNode.java:1408)

       at org.apache.hadoop.hdfs.MiniDFSCluster.startDataNodes(MiniDFSCluster.java:418)

        at org.apache.hadoop.hdfs.MiniDFSCluster.<init>(MiniDFSCluster.java:279)
        at org.apache.hadoop.hdfs.MiniDFSCluster.<init>(MiniDFSCluster.java:203)
        at
org.apache.hadoop.test.MiniHadoopClusterManager.start(MiniHadoopClusterManager.java:152)

        at
org.apache.hadoop.test.MiniHadoopClusterManager.run(MiniHadoopClusterManager.java:129)
        at
org.apache.hadoop.test.MiniHadoopClusterManager.main(MiniHadoopClusterManager.java:308)

        at sun.reflect.NativeMethodAccessorImpl.invoke0(Native Method)
       at sun.reflect.NativeMethodAccessorImpl.invoke(NativeMethodAccessorImpl.java:39)

        at
```

```
sun.reflect.DelegatingMethodAccessorImpl.invoke(DelegatingMethodAccessorImpl.java:25)
        at java.lang.reflect.Method.invoke(Method.java:597)
        at
org.apache.hadoop.util.ProgramDriver$ProgramDescription.invoke(ProgramDriver.java:68)
        at org.apache.hadoop.util.ProgramDriver.driver(ProgramDriver.java:139)
        at org.apache.hadoop.test.AllTestDriver.main(AllTestDriver.java:83)
        at sun.reflect.NativeMethodAccessorImpl.invoke0(Native Method)
       at sun.reflect.NativeMethodAccessorImpl.invoke(NativeMethodAccessorImpl.java:39)

        at
sun.reflect.DelegatingMethodAccessorImpl.invoke(DelegatingMethodAccessorImpl.java:25)
        at java.lang.reflect.Method.invoke(Method.java:597)
        at org.apache.hadoop.util.RunJar.main(RunJar.java:186)
```

Solution:

Make sure that the `umask` for `hdfs` and `mapred` is `0022`.

A cluster fails to run jobs after security is enabled.

Description:

A cluster that was previously configured to *not* use security may fail to run jobs for certain users on certain TaskTrackers (MRv1) or NodeManagers (YARN) after security is enabled due to the following sequence of events:

1. A cluster is at some point in time configured *without* security enabled.
2. A user X runs some jobs on the cluster, which creates a local user directory on each TaskTracker or NodeManager.
3. Security is enabled on the cluster.
4. User X tries to run jobs on the cluster, and the local user directory on (potentially a subset of) the TaskTrackers or NodeManagers is owned by the wrong user or has overly-permissive permissions.

The bug is that after step 2, the local user directory on the TaskTracker or NodeManager should be cleaned up, but isn't.

If you're encountering this problem, you may see errors in the TaskTracker or NodeManager logs. The following example is for a TaskTracker on MRv1:

```
10/11/03 01:29:55 INFO mapred.JobClient: Task Id : attempt_201011021321_0004_m_000011_0,
 Status : FAILED
Error initializing attempt_201011021321_0004_m_000011_0:
java.io.IOException: org.apache.hadoop.util.Shell$ExitCodeException:
at org.apache.hadoop.mapred.LinuxTaskController.runCommand(LinuxTaskController.java:212)

at
org.apache.hadoop.mapred.LinuxTaskController.initializeUser(LinuxTaskController.java:442)

at
org.apache.hadoop.mapreduce.server.tasktracker.Localizer.initializeUserDirs(Localizer.java:272)

at org.apache.hadoop.mapred.TaskTracker.localizeJob(TaskTracker.java:963)
at org.apache.hadoop.mapred.TaskTracker.startNewTask(TaskTracker.java:2209)
at org.apache.hadoop.mapred.TaskTracker$TaskLauncher.run(TaskTracker.java:2174)
Caused by: org.apache.hadoop.util.Shell$ExitCodeException:
at org.apache.hadoop.util.Shell.runCommand(Shell.java:250)
at org.apache.hadoop.util.Shell.run(Shell.java:177)
at org.apache.hadoop.util.Shell$ShellCommandExecutor.execute(Shell.java:370)
at org.apache.hadoop.mapred.LinuxTaskController.runCommand(LinuxTaskController.java:203)

... 5 more
```

Solution:

Delete the `mapred.local.dir` or `yarn.nodemanager.local-dirs` directories for that user across the cluster.

The NameNode does not start and KrbException Messages (906) and (31) are displayed.

Description:

When you attempt to start the NameNode, a login failure occurs. This failure prevents the NameNode from starting and the following KrbException messages are displayed:

```
Caused by: KrbException: Integrity check on decrypted field failed (31) - PREAUTH_FAILED}}
```

and

```
Caused by: KrbException: Identifier doesn't match expected value (906)
```

> **Note:**
>
> These KrbException error messages are displayed only if you enable debugging output. See Appendix D - Enabling Debugging Output for the Sun Kerberos Classes.

Solution:

Although there are several possible problems that can cause these two KrbException error messages to display, here are some actions you can take to solve the most likely problems:

- If you are using CentOS/Red Hat Enterprise Linux 5.6 or higher, or Ubuntu, which use AES-256 encryption by default for tickets, you must install the Java Cryptography Extension (JCE) Unlimited Strength Jurisdiction Policy File on all cluster and Hadoop user machines. For information about how to verify the type of encryption used in your cluster, see Step 3: If you are Using AES-256 Encryption, Install the JCE Policy File on page 50. Alternatively, you can change your `kdc.conf` or `krb5.conf` to not use AES-256 by removing `aes256-cts:normal` from the `supported_enctypes` field of the `kdc.conf` or `krb5.conf` file. Note that after changing the `kdc.conf` file, you'll need to restart both the KDC and the kadmin server for those changes to take affect. You may also need to recreate or change the password of the relevant principals, including potentially the TGT principal (krbtgt/REALM@REALM).
- In the [realms] section of your `kdc.conf` file, in the realm corresponding to `HADOOP.LOCALDOMAIN`, add (or replace if it's already present) the following variable:

```
supported_enctypes = des3-hmac-sha1:normal arcfour-hmac:normal des-hmac-sha1:normal
des-cbc-md5:normal des-cbc-crc:normal des-cbc-crc:v4 des-cbc-crc:afs3
```

- Recreate the `hdfs` keytab file and `mapred` keytab file using the `-norandkey` option in the `xst` command (for details, see Step 4: Create and Deploy the Kerberos Principals and Keytab Files on page 51).

```
kadmin.local: xst -norandkey -k hdfs.keytab hdfs/fully.qualified.domain.name
HTTP/fully.qualified.domain.name
kadmin.local: xst -norandkey -k mapred.keytab mapred/fully.qualified.domain.name
HTTP/fully.qualified.domain.name
```

The NameNode starts but clients cannot connect to it and error message contains enctype code 18.

Description:

The NameNode keytab file does not have an AES256 entry, but client tickets do contain an AES256 entry. The NameNode starts but clients cannot connect to it. The error message doesn't refer to "AES256", but does contain an enctype code "18".

Solution:

Make sure the "Java Cryptography Extension (JCE) Unlimited Strength Jurisdiction Policy File" is installed or remove `aes256-cts:normal` from the `supported_enctypes` field of the `kdc.conf` or `krb5.conf` file. For more information, see the first suggested solution above for Problem 5.

For more information about the Kerberos encryption types, see http://www.iana.org/assignments/kerberos-parameters/kerberos-parameters.txt.

(MRv1 Only) Jobs won't run and TaskTracker is unable to create a local mapred directory.

Description:

The TaskTracker log contains the following error message:

```
11/08/17 14:44:06 INFO mapred.TaskController: main : user is atm
11/08/17 14:44:06 INFO mapred.TaskController: Failed to create directory
/var/log/hadoop/cache/mapred/mapred/local1/taskTracker/atm - No such file or directory
11/08/17 14:44:06 WARN mapred.TaskTracker: Exception while localization
java.io.IOException: Job initialization failed (20)
        at
org.apache.hadoop.mapred.LinuxTaskController.initializeJob(LinuxTaskController.java:191)

        at org.apache.hadoop.mapred.TaskTracker$4.run(TaskTracker.java:1199)
        at java.security.AccessController.doPrivileged(Native Method)
        at javax.security.auth.Subject.doAs(Subject.java:396)
        at
org.apache.hadoop.security.UserGroupInformation.doAs(UserGroupInformation.java:1127)
        at org.apache.hadoop.mapred.TaskTracker.initializeJob(TaskTracker.java:1174)
        at org.apache.hadoop.mapred.TaskTracker.localizeJob(TaskTracker.java:1089)
        at org.apache.hadoop.mapred.TaskTracker.startNewTask(TaskTracker.java:2257)
       at org.apache.hadoop.mapred.TaskTracker$TaskLauncher.run(TaskTracker.java:2221)
Caused by: org.apache.hadoop.util.Shell$ExitCodeException:
        at org.apache.hadoop.util.Shell.runCommand(Shell.java:255)
        at org.apache.hadoop.util.Shell.run(Shell.java:182)
        at org.apache.hadoop.util.Shell$ShellCommandExecutor.execute(Shell.java:375)
        at
org.apache.hadoop.mapred.LinuxTaskController.initializeJob(LinuxTaskController.java:184)

        ... 8 more
```

Solution:

Make sure the value specified for `mapred.local.dir` is identical in `mapred-site.xml` and `taskcontroller.cfg`. If the values are different, the error message above is returned.

(MRv1 Only) Jobs will not run and TaskTracker is unable to create a Hadoop logs directory.

Description:

The TaskTracker log contains an error message similar to the following :

```
11/08/17 14:48:23 INFO mapred.TaskController: Failed to create directory
/home/atm/src/cloudera/hadoop/build/hadoop-0.23.2-cdh3u1-SNAPSHOT/logs1/userlogs/job_201108171441_0004
 - No such file or directory
11/08/17 14:48:23 WARN mapred.TaskTracker: Exception while localization
java.io.IOException: Job initialization failed (255)
        at
org.apache.hadoop.mapred.LinuxTaskController.initializeJob(LinuxTaskController.java:191)

        at org.apache.hadoop.mapred.TaskTracker$4.run(TaskTracker.java:1199)
        at java.security.AccessController.doPrivileged(Native Method)
        at javax.security.auth.Subject.doAs(Subject.java:396)
        at
org.apache.hadoop.security.UserGroupInformation.doAs(UserGroupInformation.java:1127)
        at org.apache.hadoop.mapred.TaskTracker.initializeJob(TaskTracker.java:1174)
        at org.apache.hadoop.mapred.TaskTracker.localizeJob(TaskTracker.java:1089)
```

```
        at org.apache.hadoop.mapred.TaskTracker.startNewTask(TaskTracker.java:2257)
       at org.apache.hadoop.mapred.TaskTracker$TaskLauncher.run(TaskTracker.java:2221)
Caused by: org.apache.hadoop.util.Shell$ExitCodeException:
        at org.apache.hadoop.util.Shell.runCommand(Shell.java:255)
        at org.apache.hadoop.util.Shell.run(Shell.java:182)
        at org.apache.hadoop.util.Shell$ShellCommandExecutor.execute(Shell.java:375)
        at
org.apache.hadoop.mapred.LinuxTaskController.initializeJob(LinuxTaskController.java:184)

        ... 8 more
```

Solution:

In MRv1, the default value specified for `hadoop.log.dir` in `mapred-site.xml` is `/var/log/hadoop-0.20-mapreduce`. The path must be owned and be writable by the `mapred` user. If you change the default value specified for `hadoop.log.dir`, make sure the value is identical in `mapred-site.xml` and `taskcontroller.cfg`. If the values are different, the error message above is returned.

After you enable cross-realm trust, you can run Hadoop commands in the local realm but not in the remote realm.

Description:

After you enable cross-realm trust, authenticating as a principal in the local realm will allow you to successfully run Hadoop commands, but authenticating as a principal in the remote realm will not allow you to run Hadoop commands. The most common cause of this problem is that the principals in the two realms either do not have the same encryption type, or the cross-realm principals in the two realms do not have the same password. This issue manifests itself because you are able to get Ticket Granting Tickets (TGTs) from both the local and remote realms, but you are unable to get a service ticket to allow the principals in the local and remote realms to communicate with each other.

Solution:

On the local MIT KDC server host, type the following command in the kadmin.local or kadmin shell to add the cross-realm krbtgt principal:

```
kadmin:  addprinc -e "<enc_type_list>"
krbtgt/YOUR-LOCAL-REALM.COMPANY.COM@AD-REALM.COMPANY.COM
```

where the `<enc_type_list>` parameter specifies the types of encryption this cross-realm krbtgt principal will support: AES, DES, or RC4 encryption. You can specify multiple encryption types using the parameter in the command above, what's important is that at least one of the encryption types parameters corresponds to the encryption type found in the tickets granted by the KDC in the remote realm. For example:

```
kadmin:  addprinc -e "aes256-cts:normal rc4-hmac:normal des3-hmac-sha1:normal"
krbtgt/YOUR-LOCAL-REALM.COMPANY.COM@AD-REALM.COMPANY.COM
```

(MRv1 Only) Jobs won't run and cannot access files in mapred.local.dir

Description:

The TaskTracker log contains the following error message:

```
WARN org.apache.hadoop.mapred.TaskTracker: Exception while localization
java.io.IOException: Job initialization failed (1)
```

Solution:

1. Add the `mapred` user to the `mapred` and hadoop groups on all hosts.
2. Restart all TaskTrackers.

Users are unable to obtain credentials when running Hadoop jobs or commands.

Description:

This error occurs because the ticket message is too large for the default UDP protocol. An error message similar to the following may be displayed:

```
13/01/15 17:44:48 DEBUG ipc.Client: Exception encountered while connecting to the server
 : javax.security.sasl.SaslException:
GSS initiate failed [Caused by GSSException: No valid credentials provided (Mechanism
level: Fail to create credential.
(63) - No service creds)]
```

Solution:

Force Kerberos to use TCP instead of UDP by adding the following parameter to `libdefaults` in the `krb5.conf` file on the client(s) where the problem is occurring.

```
[libdefaults]
udp_preference_limit = 1
```

If you choose to manage `krb5.conf` through Cloudera Manager, this will automatically get added to `krb5.conf`.

> **Note:**
>
> When sending a message to the KDC, the library will try using TCP before UDP if the size of the ticket message is larger than the setting specified for the `udp_preference_limit` property. If the ticket message is smaller than `udp_preference_limit` setting, then UDP will be tried before TCP. Regardless of the size, both protocols will be tried if the first attempt fails.

`Request is a replay` exceptions in the logs.

Description:

Symptom: The following exception shows up in the logs for one or more of the Hadoop daemons:

```
2013-02-28 22:49:03,152 INFO  ipc.Server (Server.java:doRead(571)) - IPC Server listener
 on 8020: readAndProcess threw exception javax.security.sasl.SaslException: GSS initiate
 failed [Caused by GSSException: Failure unspecified at GSS-API level (Mechanism l
javax.security.sasl.SaslException: GSS initiate failed [Caused by GSSException: Failure
 unspecified at GSS-API level (Mechanism level: Request is a replay (34))]
        at
com.sun.security.sasl.gsskerb.GssKrb5Server.evaluateResponse(GssKrb5Server.java:159)
       at org.apache.hadoop.ipc.Server$Connection.saslReadAndProcess(Server.java:1040)

        at org.apache.hadoop.ipc.Server$Connection.readAndProcess(Server.java:1213)
        at org.apache.hadoop.ipc.Server$Listener.doRead(Server.java:566)
        at org.apache.hadoop.ipc.Server$Listener$Reader.run(Server.java:363)
Caused by: GSSException: Failure unspecified at GSS-API level (Mechanism level: Request
 is a replay (34))
        at sun.security.jgss.krb5.Krb5Context.acceptSecContext(Krb5Context.java:741)
        at sun.security.jgss.GSSContextImpl.acceptSecContext(GSSContextImpl.java:323)
        at sun.security.jgss.GSSContextImpl.acceptSecContext(GSSContextImpl.java:267)
        at
com.sun.security.sasl.gsskerb.GssKrb5Server.evaluateResponse(GssKrb5Server.java:137)
        ... 4 more
Caused by: KrbException: Request is a replay (34)
        at sun.security.krb5.KrbApReq.authenticate(KrbApReq.java:300)
        at sun.security.krb5.KrbApReq.<init>(KrbApReq.java:134)
       at sun.security.jgss.krb5.InitSecContextToken.<init>(InitSecContextToken.java:79)

        at sun.security.jgss.krb5.Krb5Context.acceptSecContext(Krb5Context.java:724)
        ... 7 more
```

In addition, this problem can manifest itself as performance issues for all clients in the cluster, including dropped connections, timeouts attempting to make RPC calls, and so on.

Likely causes:

- **Multiple services in the cluster are using the same kerberos principal.** All secure clients that run on multiple machines should use unique kerberos principals for each machine. For example, rather than connecting as a service principal `myservice@EXAMPLE.COM`, services should have per-host principals such as `myservice/host123.example.com@EXAMPLE.COM`.
- **Clocks not in synch:** All hosts should run NTP so that clocks are kept in synch between clients and servers.

CDH services fail to start

Possible Causes: Check that the encryption types are matched between your KDC and `krb5.conf` on all hosts.

Solution: If you are using AES-256, follow the instructions at Step 2: If You are Using AES-256 Encryption, Install the JCE Policy File on page 20 to deploy the JCE policy file on all hosts.

Encryption

The goal of encryption is to ensure that only authorized users can view, use, or contribute to a data set. These security controls add another layer of protection against potential threats by end-users, administrators and other malicious actors on the network. Data protection can be applied at a number of levels within Hadoop:

- **OS Filesystem-level** - Encryption can be applied at the Linux operating system file system level to cover all files in a volume. An example of this approach is Cloudera Navigator Encrypt (formerly Gazzang zNcrypt) which is available for Cloudera customers licensed for Cloudera Navigator. Navigator Encrypt operates at the Linux volume level, so it can encrypt cluster data inside and outside HDFS, such as temp/spill files, configuration files and metadata databases (to be used only for data related to a CDH cluster). Navigator Encrypt must be used with Navigator Key Trustee (formerly Gazzang zTrustee).
- **HDFS-level** - Encryption applied by the HDFS client software. HDFS Data At Rest Encryption on page 187 operates at the HDFS folder level, enabling encryption to be applied only to the HDFS folders where it is needed. Cannot encrypt any data outside HDFS. To ensure reliable key storage (so that data is not lost), Navigator Key Trustee should be used, while the default Java keystore can be used for test purposes.
- **Network-level** - Encryption can be applied to encrypt data just before it gets sent across a network and to decrypt it as soon as it is received. In Hadoop this means coverage for data sent from client user interfaces as well as service-to-service communication like remote procedure calls (RPCs). This protection uses industry-standard protocols such as SSL/TLS.

SSL Certificates Overview

This topic will guide you through the different certificate strategies that you can employ on your cluster to allow SSL clients to securely connect to servers using trusted certificates or certificates issued by trusted authorities. The set of certificates required depends upon the certificate provisioning strategy you implement. The following strategies, among others, are possible:

- **Certificate per host:** In this strategy, you obtain one certificate for each host on which at least one SSL daemon role is running. All services on a given host will share this single certificate.
- **Certificate for multiple hosts:** Using the `SubjectAltName` extension, it is possible to obtain a certificate that is bound to a list of specific DNS names. One such certificate could be used to protect all hosts in the cluster, or some subset of the cluster hosts. The advantage of this approach over a wildcard certificate is that it allows you to limit the scope of the certificate to a specific set of hosts. The disadvantage is that it requires you to update and redeploy the certificate whenever a host is added or removed from the cluster.
- **Wildcard certificate:** You may also choose to obtain a single wildcard certificate to be shared by all services on all hosts in the cluster. This strategy requires that all hosts belong to the same domain. For example, if the hosts in the cluster have DNS names `node1.example.com` ... `node100.example.com`, you can obtain a certificate for `*.example.com`. Note that only one level of wildcarding is allowed; a certificate bound to `*.example.com` will not work for a daemon running on `node1.subdomain.example.com`.

> **Note:** Wildcard domain certificates and certificates using the SubjectAlternativeName extension are not supported at this time.

When choosing an approach to certificate provisioning, bear in mind that SSL must be enabled for all core Hadoop services (HDFS, MapReduce, and YARN) as a group. For example, if you are running HDFS and YARN on your cluster, you cannot choose to enable SSL for HDFS, but not for YARN. You must enable it for both services, which implies that you must make certificates available to all daemon roles of both services. With a certificate-per-host strategy, for example, you will need to obtain a certificate for each host on which an HDFS or YARN daemon role is running.

Creating Certificates

The following sections will walk you through obtaining certificates from commercial Certificate Authorities and creating self-signed test certificates.

Using Keytool

`Keytool` is a utility for creating and managing certificates and cryptographic keys, and is part of the standard JDK distribution. The `keytool` executable usually resides in `$JAVA_HOME/bin`.

`Keytool` stores certificates and keys in a file known as a *keystore*. While several different keystore types are supported, by default `keytool` uses the Java KeyStore (JKS) format.

Java-based services such as HDFS, MapReduce, and YARN use the JKS format by default. For this reason it is particularly convenient to use `keytool` for managing keys and certificates for these services. In the following topics, we assume you are using `keytool`.

For additional information on `keytool`, refer the keytool documentation.

Using OpenSSL

Python-based services such as Hue expect certificates and keys to be stored in `PEM` format. When managing certificates and keys for such services, you may find it convenient to use the `openssl` tool.

Refer the `openssl` documentation for more information.

Obtaining a Production Certificate from a Commercial CA

Once you have decided on a certificate-provisioning strategy, and have determined which hosts require certificates, you will typically purchase the necessary certificates from a commercial Certificate Authority (CA). The procedure for applying for a certificate varies from one CA to another, but typically involves providing some form of proof that you are the legitimate owner of the domain name for which you are requesting a certificate, generating a key pair, and submitting a Certificate Signing Request (CSR) to the CA.

As noted above, you may find it convenient to use the Java `keytool` utility to generate your key pair and CSR, and to manage your certificates. The CA you choose will provide instructions for obtaining and installing a certificate; typically, there will be separate sets of instructions for different web and application servers. The instructions for Java-based servers (Tomcat, for example), will usually describe the following process comprising three `keytool` commands to obtain a certificate:

1. `keytool -genkeypair` to generate a public/private key pair and create the keystore.
2. `keytool -certreq` to create the CSR.
3. `keytool -importcert` to import the signed certificate into the keystore.

For example, to generate a public/private key pair for the domain name `node1.example.com`, you would use a command similar to the one shown below:

```
$ keytool -genkeypair -keystore node1.keystore -alias node1 \
-dname "CN=node1.example.com,O=Hadoop" -keyalg RSA \
-keysize 2048 -storepass changeme -keypass changeme
```

This command generates a pair of 2048-bit keys using the RSA key algorithm, one of several available. The keys are stored in a keystore file called `node1.keystore`, in a keystore entry identified by by the alias `node1`. The keystore password (which protects the keystore as a whole) and the key password (which protects the private key stored in the node1 entry) are set using the `-storepass` and `-keypass` options (respectively). `-keypass` must be set to the same password value as `-storepass` for Cloudera Manager to access the keystore.

To create a CSR, you would use a command similar to the following:

```
$ keytool -certreq -keystore node1.keystore -alias node1 \
-storepass changeme -keypass changeme -file node1.csr
```

This command generates the CSR, and stores it in a file called `node1.csr`. Once you've submitted your CSR to the CA, and received the CA's reply (containing the signed certificate), you will use the following `keytool -importcert` command to import the reply into your keystore:

```
$ keytool -importcert -keystore node1.keystore -alias node1 \
-storepass changeme -keypass changeme -trustcacerts -file node1.crt
```

Here we assume that the CA's reply is stored in the file `node1.crt`.

> **Important:** This section describes a generic procedure using `keytool` to obtain a certificate from a commercial Certificate Authority. This procedure will differ from one CA to another and Cloudera recommends you consult your CA's documentation for more specifics.

Creating Self-Signed Test Certificates

> **Important:** Cloudera strongly recommends against the use of self-signed certificates in production clusters.

It is also possible to create your own test certificates. These certificates are typically self-signed; that is, they are signed by your own private key, rather than that of an external CA. Such test certificates are useful during testing and bringup of a cluster.

To generate a self-signed certificate, use `keytool -genkeypair`. (In addition to creating a public/private key pair, this command wraps the public key into a self-signed certificate.) For example, the following command creates a self-signed test certificate for the host `node1.example.com`, and stores it in a keystore named `node1.keystore`:

```
$ keytool -genkeypair -keystore node1.keystore -keyalg RSA \
-alias node1 -dname "CN=node1.example.com,O=Hadoop" \
-storepass changeme -keypass changeme
```

By default, self-signed certificates as created above are only valid for 90 days. To increase this period, use the `-validity <val_days>` parameter to specify the number of days for which the certificate should be considered valid.

Creating Java Keystores and Truststores

Typically, a keystore is used in one of two distinct ways:

- The keystore contains private keys and certificates used by SSL servers to authenticate themselves to SSL clients. By convention, such files are referred to as keystores.
- When used as a *truststore*, the file contains certificates of trusted SSL servers, or of Certificate Authorities trusted to identify servers. There are no private keys in the truststore.

> **Note:** The foregoing assumes that certificate-based authentication is being used in one direction only—that is, SSL servers are using certificates to authenticate themselves to clients. It is also possible for clients to authenticate themselves to servers using certificates. (This is known as mutual authentication.) Throughout this document, we assume that client certificates are not in use.

While all SSL clients must have access to a truststore, it is not always necessary to create and deploy truststores across a cluster. The standard JDK distribution includes a default truststore which is pre-provisioned with the root certificates of a number of well-known Certificate Authorities. If you do not provide a custom truststore, the Hadoop daemons load this default truststore. Therefore, if you are using certificates issued by a CA in the default truststore, you do not need to provide custom truststores. However, you must consider the following before you decide to use the default truststore:

- If you choose to use the default truststore, it is your responsibility to maintain it. You may need to remove the certificates of CAs you do not deem trustworthy, or add or update the certificates of CAs you trust. Use the `keytool` utility to perform these actions.

Security Considerations for Keystores and Truststores

> **Note:** While the strategy for certificate deployment you select will ultimately depend upon the security policies you wish to implement, the following guidelines may prove useful.

Because keystores contain private keys, while truststores do not, the security requirements for keystores are more stringent. In particular:

- Hadoop SSL requires that truststores and the truststore password be stored, in plaintext, in a configuration file that is readable by all.
- Keystore and key passwords are stored, in plaintext, in a file that is readable only by members of the appropriate group.

These considerations should inform your choice of which keys and certificates to store in the keystores and truststores you will deploy across your cluster.

- Keystores should contain a minimal set of keys and certificates. A reasonable strategy would be to create a unique keystore for each host, which would contain only the keys and certificates needed by the Hadoop SSL services running on the host. In most cases, the keystore would contain a single key/certificate entry.
- On the other hand, because truststores do not contain sensitive information, it is reasonable to create a single truststore for an entire cluster. On a production cluster, such a truststore would often contain a single CA certificate (or certificate chain), since you would typically choose to have all certificates issued by a single CA.
- > **Important:** *Do not use the same password for truststores and keystores/keys.*

  Since truststore passwords are stored in the clear in files readable by all, doing so would compromise the security of the private keys in the keystore.

Creating Keystores

Once you have settled on a storage plan for your keys and certificates, you can use `keytool` to create or update the necessary keystores and truststores. To create a new keystore with a certificate see Creating Certificates on page 150.

In many cases, you will already have created the set of keystores that you need. If you have followed the approach of creating a separate keystore for each private key and certificate, and wish to maintain this arrangement when deploying the keystores, no additional steps are required to prepare the keystores for deployment. If you wish to reorganize your keys and certificates into a different set of keystores, you can use `keytool -importkeystore` to transfer entries from one keystore to another.

Creating Truststores

The steps involved in preparing the truststores to be used in your deployment depend on whether you have decided to use the default Java truststore, or to create custom truststores:

- If you are using the default truststore, you may need to add CA certificates (or certificate chains) to the truststore, or delete them from the truststore.
- If you are creating custom truststores, you will need to build the truststores by importing trusted certificates into new truststores. The trusted certificates can be CA certificates (typically downloaded from the CA's website), or self-signed certificates that you have created.

As shown in the examples below, when creating a truststore you must select a password. All truststore passwords for a given service must be the same. In practice, this restriction rarely comes into play, since it is only relevant when you wish to create distinct custom truststores for each host.

The following sections provide examples of the steps required for several common scenarios:

Example 1: Adding a CA Certificate to the alternative Default Truststore

In this example, we assume that you have chosen to use the default Java truststore, but have obtained a certificate from a CA not included in the truststore. (This situation can also arise if the CA that issued your certificate has an entry in the default truststore, but the particular certificate product you purchased requires an alternate CA certificate chain.)

1. Locate the default truststore on your system. The default truststore is located in the `$JAVA_HOME/jre/lib/security/cacerts` file. This contains the default CA information shipped with the JDK. Create an alternate default file called `jssecacerts` in the same location as the `cacerts` file. You can now safely append CA certificates for any private or public CAs not present in the default `cacerts` file, while keeping the original file intact.

   The alternate file will always be read unless the `javax.net.ssl.trustStore` flag is set in the arguments for the startup of the `java` process.

   For our example, we will be following this recommendation by copying the default `cacerts` file into the new `jssecacerts` file.

```
$ cp $JAVA_HOME/jre/lib/security/cacerts \
  $JAVA_HOME/jre/lib/security/jssecacerts
```

   If you use a copy of the `cacerts` file, remember the default keystore password is `changeit`.

2. Import the CA certificate into the default truststore. Assuming that the file `myCA-root.cer` contains the CA's certificate, which you have previously downloaded from the CA's web site, the following command imports this certificate into the alternative default truststore.

   Note: Test the trust relationship before you import any intermediary CA certificates. Trust should be derived from the root CA only. Import intermediary CA certificates only if necessary.

```
$ keytool -importcert -file myCA-root.cer -alias myCA \
-keystore /usr/java/default/jre/lib/security/jssecacerts \
-storepass changeit
```

   When you give this command, you will be prompted to confirm that you trust the certificate. Be sure to verify that the certificate is genuine before importing it.

   Important: Any updates you make to the default truststore must be made on all hosts in the cluster.

Example 2: Creating a Custom Truststore Containing a Single CA Certificate Chain

In this example, we demonstrate how to use `keytool` to create a custom truststore. We assume all certificates were issued by a single CA, so a truststore containing the certificate chain for that CA will serve for all hosts in the cluster.

Our example certificate chain consists of a root certificate and a single intermediate certificate. We assume that you have downloaded these and saved them in the files `myCA-root.cer` and `myCA-intermediate.cer` (respectively). The steps below show the commands needed to build a custom truststore containing the root and intermediate certificates.

1. Import the root certificate and create the truststore:

```
$ keytool -importcert -keystore my.truststore -alias myCA-root \
-storepass trustchangeme -file myCA-root.cer
```

   You will be prompted to confirm that the root certificate is trustworthy. Be sure to verify that the certificate is genuine before you import it.

2. Import the intermediate certificate into the truststore created in Step 1:

```
$ keytool -importcert -keystore my.truststore \
-alias myCA-intermediate -storepass trustchangeme \
-file myCA-intermediate.cer
```

Example 3: Creating a Custom Truststore Containing Self-Signed Test Certificates

> **Important:** Cloudera strongly recommends against the use of self-signed certificates in production clusters.

This example is particularly relevant when setting up a test cluster. We assume that you have generated a set of self-signed test certificates for the hosts in the cluster, and wish to create a single truststore that can be deployed on all hosts. Because the certificates are self-signed, we cannot simply construct a truststore containing a single certificate chain, as in the previous example. When a client receives a self-signed certificate from a server during the SSL handshake, it must be able to find the server's certificate in the truststore, since no other signing certificate exists to establish trust. Therefore, the truststore must contain all the test certificates.

We assume that the test certificates reside in keystores named `node1.keystore` … `node100.keystore`, which were created following the steps described in Creating Self-Signed Test Certificates.

1. Export the test certificate for `node1.example.com`:

```
$ keytool -exportcert -keystore node1.keystore -alias node1 \
-storepass changeme -file node1.cer
```

2. Import the test certificate into the truststore:

```
keytool -importcert -keystore my.truststore -alias node1 \
-storepass trustchangeme -file node1.cer -noprompt
```

   Here we specify the `-noprompt` option to suppress the prompt asking you to confirm that the certificate is trustworthy. Since you created the certificate yourself, this confirmation is unnecessary.
3. Repeat Steps 1 and 2 for `node2.keystore` … `node100.keystore`.

Private Key and Certificate Reuse Across Java Keystores and OpenSSL

This topic provides a quick tutorial on exporting/importing private keys for reuse from a Java keystore to OpenSSL and vice versa. Regardless of the procedure followed to create host private keys and certificates, sometimes it becomes necessary to reuse those private keys and certificates by other services on the same host. For example, if you used OpenSSL to create private keys and certificates for a service, you can reuse those keys for a Java-based service on the same host by converting them to the Java keystore format.

The documentation for Configuring TLS Security for Cloudera Manager describes both approaches to creating private keys, using Java keystore, and OpenSSL.

Why Reuse a Private Key?

Certificate authorities generally revoke previous generations of certificates issued to a host. Hence, a host cannot have 2 sets of CA-issued certificates and have both be valid. Once a certificate is issued to a host, it then becomes necessary to reuse the private key that requested the certificate, and the CA-issued certificate across different services, Java-based and otherwise.

> **Note:** This following sections assume the default paths set up in Configuring TLS Encryption Only for Cloudera Manager.

Conversion from Java Keystore to OpenSSL

First, use `keytool` to export the private key and certificate to a PKCS12 file as a transitional file format that can then be split up into individual key and certificate files by the `openssl` command line. Replace `cmhost` and `hostname` in the commands below with the actual hostname of the server that is managing the certificate and keys.

```
$ keytool -importkeystore -srckeystore /opt/cloudera/security/jks/hostname-keystore.jks
 \
-srcstorepass password -srckeypass password -destkeystore /tmp/hostname-keystore.p12 \

-deststoretype PKCS12 -srcalias hostname -deststorepass password -destkeypass password
```

Now use `openssl` to split the PKCS12 file created above into first, the certificate file, and then the private key file. While the CA-issued certificate can be used as is, the command has been provided here for completeness.

```
$ openssl pkcs12 -in /tmp/hostname-keystore.p12 -passin pass:password  -nokeys \
-out /opt/cloudera/security/x509/hostname.pem

$ openssl pkcs12 -in /tmp/hostname-keystore.p12 -passin pass:password -nocerts \
-out /opt/cloudera/security/x509/hostname.key -passout pass:password
```

Note that the method above generates a key with a password. For services such as Hue and Impala that require keys without passwords, you can use the following command:

```
$ openssl rsa -in /opt/cloudera/security/x509/hostname.key \
-passin pass:password -out /opt/cloudera/security/x509/hostname.pem
```

Conversion from OpenSSL to Java Keystore

First, convert the `openssl` private key and certificate files into a PKCS12 file. The PKCS12 file can then be imported into a Java keystore file. Replace `hostname` in the commands below with the FQDN for the host whose certificate is being imported.

```
$ openssl pkcs12 -export -in /opt/cloudera/security/x509/hostname.pem \
-inkey /opt/cloudera/security/x509/hostname.key -out /tmp/hostname.p12 \
-name hostname -passin pass:password -passout pass:password

$ keytool -importkeystore -srckeystore /tmp/hostname.p12 -srcstoretype PKCS12 \
-srcstorepass password -alias hostname -deststorepass password
-destkeypass password -destkeystore /opt/cloudera/security/jks/hostname-keystore.jks
```

Configuring TLS Security for Cloudera Manager

Important:

- Cloudera strongly recommends that you set up a fully functional CDH cluster and Cloudera Manager before you configure the Cloudera Manager Server and Agents to use TLS.
- When TLS is enabled, Cloudera Manager continues to accept HTTP requests on port 7180 (default) but immediately redirects clients to port 7183 for HTTPS connectivity.
- When Level 3 TLS is configured, to add new hosts running Agents, you must manually deploy the Cloudera Manager agent and daemon packages for your platform, issue a new certificate for the host, configure `/etc/cloudera-scm-agent/config.ini` to use SSL/TLS, and then bring the host online.

  Or, you can disable TLS to add the host, configure the new host for TLS, and then re-enable with the proper configuration in place. Either approach is valid, based on your needs.

- For all hosts running Agents, Cloudera recommends that you first create the keystore in Java, and then export the key and certificate using openSSL for use by the Agent or Hue.

Transport Layer Security (TLS) provides encryption and authentication in communication between the Cloudera Manager Server and Agents. Encryption prevents snooping, and authentication helps prevent problems caused by malicious servers or agents.

Cloudera Manager supports three levels of TLS security.

- Level 1 (Good) - This level encrypts communication between the browser and Cloudera Manager, and between Agents and the Cloudera Manager Server. See Configuring TLS Encryption Only for Cloudera Manager on page 156 followed by Level 1: Configuring TLS Encryption for Cloudera Manager Agents on page 160 for instructions. Level 1 encryption prevents snooping of commands and controls ongoing communication between Agents and Cloudera Manager.
- Level 2 (Better) - This level encrypts communication between the Agents and the Server, and provides strong verification of the Cloudera Manager Server certificate by Agents. See Level 2: Configuring TLS Verification of Cloudera Manager Server by the Agents on page 161. Level 2 provides Agents with additional security by verifying trust for the certificate presented by the Cloudera Manager Server.
- Level 3 (Best) - This includes encrypted communication between the Agents and the Server, strong verification of the Cloudera Manager Server certificate by the Agents, *and* authentication of Agents to the Cloudera Manager Server using self-signed or CA-signed certs. See Level 3: Configuring TLS Authentication of Agents to the Cloudera Manager Server on page 163. Level 3 TLS prevents cluster Servers from being spoofed by untrusted Agents running on a host. Cloudera recommends that you configure Level 3 TLS encryption for untrusted network environments before enabling Kerberos authentication. This provides secure communication of keytabs between the Cloudera Manager Server and verified Agents across the cluster.

> **Important:** You must finish configuring Level 1 and Level 2 TLS to configure Level 3 encryption. To enable TLS encryption for all connections between your Web browser running the Cloudera Manager Admin Console and the Cloudera Manager Server, see the first 2 steps of Level 1: Configuring TLS Encryption for Cloudera Manager Agents on page 160.

For details on how HTTPS communication is handled Cloudera Manager Agents and Cloudera Management Services daemons, see HTTPS Communication in Cloudera Manager on page 167.

Configuring TLS Encryption Only for Cloudera Manager

Minimum Required Role: Cluster Administrator (also provided by **Full Administrator**)

> **Important:** The sequence of steps described in the following topics to configure Level 1 through 3 TLS will each build upon the steps of the previous level. The procedure and examples provided in these topics, are based on this concept.

Before enabling TLS security for Cloudera Manager, you must create a keystore, submit a certificate-signing request, and install the issued certificate for the Server. You do this using the Oracle JDK `keytool` command-line tool. If you are using a Private CA, append its certificate (and any required intermediary certificates) to the alternate default truststore provided with the JDK for inherent trust. This process is described in detail in Configuring SSL Encryption in Cloudera Manager Deployments.

The table below shows the paths for managing certificates in the following examples . These paths persist during any upgrades and should be removed manually if the host is removed from a CDH cluster. Note that the folders and filepaths listed here can reside anywhere on the system and must be created on every host, especially as later sections move on to creating certificates for each host.

> **Note:** Set permissions on the paths such that `scm-user`, hue, Hadoop service users (or groups), and `root` users can read the private key, certificate, and keystore and truststore files.

Example Property Values	Description
`cmhost.sec.cloudera.com`	FQDN for Cloudera Manager Server host.

Example Property Values	Description
`/opt/cloudera/security`	Base location for security-related files.
`/opt/cloudera/security/x509`	Location for `openssl key/, cert/` and `cacerts/` files to be used by the Cloudera Manager Agent and Hue.
`/opt/cloudera/security/jks`	Location for the Java-based `keystore/` and `truststore/` files for use by Cloudera Manager and Java-based cluster services.
`/opt/cloudera/security/CAcerts`	Location for CA certificates (root and intermediary/subordinate CAs). One PEM file per CA in the chain is required.

Important:

- You must use the Oracle JDK `keytool`. The following procedure requires use of the Cloudera-installed Oracle JDK (or JDK downloaded from Oracle). Do not use both `keytool` and OpenJDK, or varying versions of JDK command line tools like `keytool`. If necessary, set your PATH so that the Oracle JDK is first. For example:

```
$ export JAVA_HOME=/usr/java/jdk1.7.0_67-cloudera
$ export PATH=$JAVA_HOME/bin:$PATH
```

- Set `-keypass` to the same value as `-storepass`. Cloudera Manager assumes that the same password is used to access both the key and the keystore, and therefore, does not support separate values for `-keypass` and `-storepass`.

Step 1: Create the Cloudera Manager Server Keystore, Generate a Certificate Request, and Install the Certificate

The following procedure assumes that a private Certificate Authority is used, and therefore trust must be established for that private CA. If a known public CA such as Verisign or GeoTrust is used, you may not need to explicitly establish trust for the issued certificates. Newer public CAs might not be present yet in the JDK default `cacerts` file. If you have problems with the import process (such as keytool error: `java.lang.Exception: Failed to establish chain from reply`), follow the steps for trusting private CAs below.

1. Assuming the paths documented in the table above have been created, use `keytool` to generate a Java keystore and Certificate Signing Request (CSR) for the Cloudera Manager Server. Replace `cmhost` and `cmhost.sec.cloudera.com` in the commands below with your hostname and FQDN. For example:

```
$ keytool -genkeypair -alias cmhost -keyalg RSA -keystore \
/opt/cloudera/security/jks/cmhost-keystore.jks -keysize 2048 -dname \
"CN=cmhost.sec.cloudera.com,OU=Support,O=Cloudera,L=Denver,ST=Colorado,C=US" \
-storepass password -keypass password
```

- `-alias` is a label used only in the keystore. In this example, the hostname is used for easy tracking and management of the key and certificate. Ensure that `-alias` is consistent across all your commands.
- `-keyalg` is the algorithm used to generate the key. RSA allows key lengths greater than 1024 bits for certificate requests.
- `-dname` allows you to provide the certificate subject as a single line. If not specified, you will be prompted for the values of the certificate subject information. In that case, use the host FQDN that agents and browsers will use to connect to in the subject **First and Last name (CN)** question prompt.
- `/opt/cloudera/security/jks/cmhost-keystore.jks` is an example path to the keystore where you store the keystore file and where the Cloudera Manager Server host can access it.
- `-keypass` must be set to the same password value as `-storepass` for Cloudera Manager to access the keystore.

2. Generate a certificate signing request for the host (in this example, `cmhost`).

```
$ keytool -certreq -alias cmhost \
-keystore /opt/cloudera/security/jks/cmhost-keystore.jks \
-file /opt/cloudera/security/x509/cmhost.csr -storepass password \
-keypass password
```

3. Submit the `.csr` file created by the `-certreq` command to your Certificate Authority to obtain a server certificate. When possible, work with certificates in the default Base64 (ASCII) format. You can easily modify Base64-encoded files from .CER or .CRT to .PEM. The file is in ASCII format if you see the opening and closing lines as follows:

```
-----BEGIN CERTIFICATE-----
( the encoded certificate is represented by multiple lines of exactly 64 characters,
except
for the last line which can contain 64 characters or less.)
-----END CERTIFICATE-----
```

If your issued certificate is in binary (DER) format, adjust the commands according to the keytool documentation.

4. Copy the root CA certificate and any intermediary or subordinate CA certificates to `/opt/cloudera/security/CAcerts/`.

> **Important:** For a private CA, you must import the private CA and intermediary or subordinate CA certificates into an alternative default JDK truststore `jssecacerts`, *before* importing them to your Java keystore.

a. Import the root CA certificate first, followed by any intermediary or subordinate CA certificates. Substitute `$JAVA_HOME` in the command below with the path for your Oracle JDK.

```
$ sudo cp $JAVA_HOME/jre/lib/security/cacerts $JAVA_HOME/jre/lib/security/jssecacerts

$ sudo keytool -importcert -alias RootCA -keystore $JAVA_HOME/jre/lib/security/jssecacerts
 \
-file /opt/cloudera/security/CAcerts/RootCA.cer -storepass changeit

$ sudo keytool -importcert -alias SubordinateCA -keystore \
$JAVA_HOME/jre/lib/security/jssecacerts \
-file /opt/cloudera/security/CAcerts/SubordinateCA.cer -storepass changeit
```

Repeat for as many subordinate or intermediary CA certificates as needed. The default `-storepass` for the `cacerts` file is `changeit`. After completing this step, copy the `jssecacerts` file created to the same path on all cluster hosts.

b. Import the Private CA certificates into your Java keystore file. Import the root CA certificate first.

```
$ keytool -importcert -trustcacerts -alias RootCA -keystore \
/opt/cloudera/security/jks/<cmhost-keystore>.jks -file \
/opt/cloudera/security/CAcerts/RootCA.cer -storepass password

$ keytool -importcert -trustcacerts -alias SubordinateCA -keystore \
/opt/cloudera/security/jks/<cmhost-keystore>.jks -file \
/opt/cloudera/security/CAcerts/SubordinateCA.cer -storepass password
```

Repeat for as many subordinate/intermediary CA certificates as needed.

5. Copy the signed certificate file provided to a location where it can be used by the Cloudera Manager Agents (and Hue if necessary).

```
$ cp certificate-file.cer  /opt/cloudera/security/x509/cmhost.pem
```

Install it with the following `keytool` command:

```
$ keytool -importcert -trustcacerts -alias cmhost \
-file /opt/cloudera/security/x509/cmhost.pem \
-keystore /opt/cloudera/security/jks/cmhost-keystore.jks -storepass password
```

You *must* see the following response verifying that the certificate has been properly imported against its private key.

```
Certificate reply was installed in keystore
```

Because the issued certificate has been imported by the Java keystore, the original certificate-signing request (.CSR) and certificate files are no longer needed by Java services on that host, and the certificate and private key are now accessed through the keystore.

However, you still must export the private key from the Jave keystore to make the certificate usable by Hue and the Cloudera Manager Agent. For instructions on reusing certificates, see Private Key and Certificate Reuse Across Java Keystores and OpenSSL on page 154.

Step 2: Enable HTTPS for the Cloudera Manager Admin Console and Specify Server Keystore Properties

1. Log into the Cloudera Manager Admin Console.
2. Select **Administration** > **Settings**.
3. Click the **Security** category.
4. Configure the following TLS settings:

Property	Description
Path to TLS Keystore File	The complete path to the keystore file. In the example, this path would be: `/opt/cloudera/security/jks/cmhost-keystore.jks`
Keystore Password	The password for keystore: `password`
Use TLS Encryption for Admin Console	Check this box to enable TLS encryption for Cloudera Manager.

5. Click **Save Changes** to save the settings.
6.

Step 3: Specify SSL Truststore Properties for Cloudera Management Services

When enabling TLS for the Cloudera Manager UI, you must set the Java truststore location and password in the Cloudera Management Services configuration. If this is not done, services such as the Host Monitor and Service Monitor will be unable to connect to Cloudera Manager and will not start.

1. Open the Cloudera Manager Admin Console and navigate to the **Cloudera Management Service**.
2. Click **Configuration**.
3. In the Search field, type **SSL** to show the SSL properties (found under the **Service-Wide** > **Security** category).
4. Edit the following SSL properties according to your cluster configuration.

Property	Description
SSL Client Truststore File Location	Path to the client truststore file used in HTTPS communication. The contents of this truststore can be modified without restarting the Cloudera Management Service roles. By default, changes to its contents are picked up within ten seconds.
SSL Client Truststore File Password	Password for the client truststore file.

5. Click **Save Changes**.

6. Restart the Cloudera Management Service. For more information, see HTTPS Communication in Cloudera Manager on page 167.

Step 4: Restart the Cloudera Manager Server

Restart the Cloudera Manager Server by running `service cloudera-scm-server restart` from the Cloudera Manager host command prompt.

You should now be able to connect to the Cloudera Manager Admin Console using an HTTPS browser connection. If a private CA certificate or self-signed certificate is used, you must establish trust in the browser for your certificate. Certificates issued by public commercial CAs should be trusted by your browser and other Java or OpenSSL-based services.

For more information on establishing trust for certificates, see SSL Certificates Overview on page 149 or the relevant JDK documentation.

Level 1: Configuring TLS Encryption for Cloudera Manager Agents

Minimum Required Role: Cluster Administrator (also provided by **Full Administrator**)

Prerequisite:

You must have completed the steps described at Configuring TLS Encryption Only for Cloudera Manager.

Step 1: Enable Agent Connections to Cloudera Manager to use TLS

In this step, you enable TLS properties for Cloudera Manager Agents and their connections to the Cloudera Manager Server. To configure agents to connect to Cloudera Manager over TLS, log into the Cloudera Manager Admin Console.

> **Note:** If you are using a private certificate authority to sign certificate requests, see information on establishing trust for this CA in Configuring TLS Encryption Only for Cloudera Manager on page 156.

1. Log into the Cloudera Manager Admin Console.
2. Select **Administration** > **Settings**.
3. Click the **Security** category.
4. Configure the following TLS settings in the Cloudera Manager Server:

Property	Description
Use TLS Encryption for Agents	Enable TLS encryption for Agents connecting to the Server. The Agents will still connect to the defined agent listener port for Cloudera Manager (default: `7182`). This property negotiates TLS connections to the service at this point.

5. Click Save Changes.

Step 2: Enable and Configure TLS on the Agent Hosts

To enable and configure TLS, you must specify values for the TLS properties in the `/etc/cloudera-scm-agent/config.ini` configuration file on all Agent hosts.

1. On the Agent host, open the `/etc/cloudera-scm-agent/config.ini` configuration file and edit the following property:

Property	Description
use_tls	Specify `1` to enable TLS on the Agent, or `0` (zero) to disable TLS.

2. Repeat this step on every Agent host. You can copy the Agent's `config.ini` file across all hosts since this file by default does not have host specific information within it. If you modify properties such as `listening_hostname` or `listening_ip address` in `config.ini`, you must configure the file individually for each host.

Step 3: Restart the Cloudera Manager Server

Restart the Cloudera Manager Server with the following command to activate the TLS configuration settings.

```
$ sudo service cloudera-scm-server restart
```

Step 4: Restart the Cloudera Manager Agents

On every Agent host, restart the Agent:

```
$ sudo service cloudera-scm-agent restart
```

Step 5: Verify that the Server and Agents are Communicating

In the Cloudera Manager Admin Console, open the **Hosts** page. If the Agents heartbeat successfully, TLS encryption is working properly.

Level 2: Configuring TLS Verification of Cloudera Manager Server by the Agents

Minimum Required Role: Cluster Administrator (also provided by **Full Administrator**)

This level of TLS security requires that you provide a server certificate that is signed, either directly or through a chain, by a trusted root certificate authority (CA), to the Cloudera Manager Server. You must also provide the certificate of the CA that signed the Server certificate. For test environments, you can use a self-signed server certificate.

> **Note:** If the Cloudera Manager Server certificate or the associated CA certificate is missing or expired, Agents will not communicate with the Cloudera Manager Server.

Step 1: Configure TLS encryption

If you have not done so, configure TLS encryption to use this level of security. For instructions, see Configuring TLS Encryption Only for Cloudera Manager on page 156 and Level 1: Configuring TLS Encryption for Cloudera Manager Agents on page 160.

Step 2: Copy the CA Certificate or Cloudera Manager Server's .pem file to the Agents

1. Agents can verify the Cloudera Manager Server using either the Server certificate or the associated root CA certificate. Pick any one of the following approaches to proceed:
   - **Copy the Cloudera Manager Server .pem file to the Agent host**
     1. For verification by the Agent, copy the Server .pem file (for example, `cmhost.pem`) to any directory on the Agent host. In the examples, this path is `/opt/cloudera/security/x509/cmhost.pem`.
     2. On the Agent host, open the `/etc/cloudera-scm-agent/config.ini` configuration file and edit the following properties.

Property	Description
`verify_cert_file`	Point this property to the copied .pem file on the Agent host; in this example, `/opt/cloudera/security/x509/cmhost-cert.pem`.
`use_tls`	Set this property to `1`.

OR

   - **Copy the CA certificates to the Agent host**

1. If you have a CA-signed certificate, copy the root CA or intermediate CA certificates in PEM format to the Agent host. In the example, the CA certificates are copied to `/opt/cloudera/security/CAcerts/*`.
2. On the Agent host, open the `/etc/cloudera-scm-agent/config.ini` configuration file and edit the following properties.

Property	Description
`verify_cert_dir`	Point this property to the directory on the Agent host with the copied CA certificates; in the example, `/opt/cloudera/security/CAcerts/`.
`use_tls`	Set this property to `1`.

Note: When configuring the `verify_cert_dir` property, the `openss-perl` package is required to provide the `c_rehash` command that is necessary to generate the Subject Name hash values that need to be linked to the certificates to make them usable. See the comments in the `config.ini` file for more information.

The following example is for RHEL-compatible systems. The package name for Debian-based systems is the same.Afterthe package is installed, go to the CA certificate path and run the `c_rehash` command. This generates symbolic links to the certificate in that location, with "`.`" being the current path, as follows:

```
$ yum -y install openssl-perl
$ cd /opt/cloudera/security/CAcerts/
$ c_rehash .
   Doing .
   w2k8-1-root.pem => 4507f087.0
   w2k8-2-intermediary.pem => 082ba6df.0
$ls -l
total 8.0K
lrwxrwxrwx 1 root root   23 Oct  6 22:44 082ba6df.0 ->
w2k8-2-intermediary.pem
lrwxrwxrwx 1 root root   15 Oct  6 22:44 4507f087.0 ->
w2k8-1-root.pem
-rw-r----- 1 root root 2.1K Oct  6 17:23 w2k8-1-root.pem
-rw-r----- 1 root root 2.8K Oct  6 17:23
w2k8-2-intermediary.pem
```

2. Based on the approach you select in step 1, repeat the steps on every Agent host. You can copy the Agent's `config.ini` file across all hosts. However, if you modify properties such as `listening_hostname` or `listening_ip address` in `config.ini`, you must configure `config.ini` for each host individually.

Step 3: Restart the Cloudera Manager Agents

On every Agent host, restart the Agent:

```
$ sudo service cloudera-scm-agent restart
```

Step 4: Restart the Cloudera Management Services

To restart the Cloudera Management Service from the Cloudera Manager Admin Console:

1. On the Home page, click

   to the right of the service name and select **Restart**.
2. Click **Start** on the next screen to confirm. When you see a **Finished** status, the service has restarted.

Step 5: Verify that the Server and Agents are communicating

In the Cloudera Manager Admin Console, open the **Hosts** page. If the Agents heartbeat successfully, the Server and Agents are communicating. If not, check the Agent log `/var/log/cloudera-scm-agent/cloudera-scm-agent.log`, which shows errors if the connection fails.

Level 3: Configuring TLS Authentication of Agents to the Cloudera Manager Server

Minimum Required Role: Cluster Administrator (also provided by **Full Administrator**)

This is the highest level of TLS security supported for Cloudera Manager Server-Agent communications, and requires you to create private keys and Certificate Signing Requests (CSR) for each cluster node. A Certificate Authority (CA) can then sign the CSR, thus providing a server certificate for each host. Agents will then need to authenticate themselves to Cloudera Manager using this server certificate.

This can be completed one of two ways, depending on the approach you choose to configuring TLS on your cluster.

- Approach A - Use OpenSSL to create private keys and request CA-signed certificates for every Agent on your cluster. Approach A is faster if you only need to enable TLS for Cloudera Manager Server-Agent communication.
- Approach B - Create a Java truststore file that contains the Agent and CA certificates, and authenticate Agents against this truststore file. If you plan to enable TLS communication for all CDH services cluster-wide, including Java-based components, consider using Approach B.

Steps for creating self-signed certificates are not included. Self-signed certificates are not recommended for production environments.

> **Note:** Wildcard domain certificates and certificates using the SubjectAlternativeName extension are not supported at this time.

Step 1: Configure TLS encryption

If you have not already done so, you must configure TLS encryption to use this third level of security. For instructions, see Configuring TLS Encryption Only for Cloudera Manager on page 156 and Configuring TLS Encryption for Cloudera Manager.

Step 2: Configure TLS Verification of Server Trust by Agents

If you have not already done so, you must configure TLS Verification of Server Trust by Agents. For instructions, see Configuring TLS Authentication of Server to Agents.

> **Important:**
>
> Steps 3, 4, and 5 can be completed one of two ways, depending on the approach you choose to configuring TLS on your cluster.
>
> - Approach A - Use OpenSSL to create private keys and request CA-signed certificates for every Agent on your cluster. Approach A is faster if you only need to enable TLS for Cloudera Manager Server-Agent communication.
> - Approach B - Create a Java truststore file that contains the Agent and CA certificates, and authenticate Agents against this truststore file. If you plan to enable TLS communication for all CDH services cluster-wide, including Java-based components, consider using Approach B.
>
> Irrespective of the path you select, it will still be possible to reuse OpenSSL private keys and certificates by exporting to a Java keystore and vice versa. For instructions, see Private Key and Certificate Reuse Across Java Keystores and OpenSSL on page 154.
>
> *After choosing an approach, follow steps 3-5 for all hosts in your cluster.*

Approach A: Using OpenSSL to Create Private Keys and Request Agent Certificates

If the Cloudera Manager Server is running Management Services or CDH components (and therefore, has a Cloudera Manager Agent installed), you do not need to re-create a private key for the Server host. Follow the steps in Private Key and Certificate Reuse Across Java Keystores and OpenSSL on page 154 to reuse the host certificate. Follow steps 3-5 for all remaining cluster hosts.

Approach A Step 3: Generate the private key and certificate signing request for the Agent using OpenSSL.

Run the following command on the Agent, replacing `hostname` with your actual hostname. The `-subj` command line option allows you to provide the certificate subject as a single line.If you do not specify the certificate subject as an argument, you are prompted for the values of the certificate subject information. In that case, use the host FQDN that Agents will use to connect from in the subject **First and Last name (CN)** question prompt. Country (C) requires a 2 letter country code. The `"/"` is replaced with `", "` in the actual CSR and private key file.

```
$  openssl req -subj
'/CN=hostname.sec.cloudera.com/OU=Support/O=Cloudera/L=Denver/ST=Colorado/C=US' \
-out /opt/cloudera/security/x509/hostname.csr -new -newkey rsa:2048 \
-keyout /opt/cloudera/security/x509/hostname.key -passout pass:password
```

`password` provides a password to protect the private key file. Keep the password in a safe place; you must provide a key password file to the Agent to complete configuration.

Approach A Step 4: Submit the certificate signing request to your CA and distribute the issued certificates.

The CSR file created (`/opt/cloudera/security/x509/hostname.csr`) is collected from cluster hosts for submission to the certificate authority (CA) for signing. In the example paths, you copy the issued CA-signed certificate file to `/opt/cloudera/security/x509` on each cluster host. For easy management and tracking of files, name the files in the `hostname.pem` format, replacing `hostname` with the actual hostname.

> **Note:** Certificate file extensions of .cer, .crt, and .pem are interchangeable. Rename the files so they have a .pem extension, and can therefore be used by the Agent and Hue (or any other Python-based component).

The CSR can be examined with the following command:

```
$  openssl req -text -noout -verify -in  /opt/cloudera/security/x509/hostname.csr
```

The issued certificate file can be examined with the following command:

```
$ openssl x509 -in /opt/cloudera/security/x509/hostname.pem -text -noout
```

Approach A Step 5 (Optional): Import the OpenSSL private key and certificate into the per-host Java keystore.

Follow the steps in Private Key and Certificate Reuse Across Java Keystores and OpenSSL on page 154 for this step.

> **Important: If you are using Approach A, skip to step 6 to continue.**

Approach B: Creating a Java Keystore and Importing Signed Agent Certificates into it

If the Cloudera Manager Server is running Management Services or CDH components (and therefore, has a Cloudera Manager Agent installed), you do not need to re-create a private key for the Server host. Follow the steps in Private Key and Certificate Reuse Across Java Keystores and OpenSSL on page 154 to reuse the host certificate. Follow steps 3-5 for all remaining cluster hosts.

Approach B - Step 3: Create a Java Keystore and private key for a host

Create a Java Keystore and private key files for an Agent host as follows:

```
$ keytool -genkeypair -alias hostname -keyalg RSA -keystore \
/opt/cloudera/security/jks/hostname-keystore.jks -keysize 2048 -dname \
"CN=cmhost.sec.cloudera.com,OU=Support,O=Cloudera,L=Denver,ST=Colorado,C=US" \
-storepass password -keypass password
```

`password` provides a password to protect the private key file. Note the password in a safe place; you must provide a key password file to the Agent to complete configuration.

Approach B - Step 4: Generate a certificate signing request and install the issued certificate into the Java Keystore

1. Generate a certificate signing request (CSR) and submit it to your CA for a signed certificate.

```
$ keytool -certreq -alias hostname \
-keystore /opt/cloudera/security/jks/hostname-keystore.jks \
-file /opt/cloudera/security/x509/hostname.csr \
-storepass password -keypass password
```

2. If you are using a Private CA, first import the root CA certificate followed by the intermediary/subordinate CA certificates into the Java keystore created previously.

```
 $ keytool -importcert -trustcacerts -alias RootCA -keystore \
/opt/cloudera/security/jks/hostname-keystore.jks -file \
/opt/cloudera/security/CAcerts/RootCA.cer -storepass password
```

   Repeat the following for all subordinate/intermediary CA certificates presented.

```
$ keytool -importcert -trustcacerts -alias SubordinateCA -keystore \
/opt/cloudera/security/jks/hostname-keystore.jks -file \
/opt/cloudera/security/CAcerts/SubordinateCA.cer -storepass password
```

3. Copy the issued signed certificate file provided by your CA to the location from where it will be imported by the Cloudera Manager Agent and possibly Hue.

```
$ cp certificate-file.cer /opt/cloudera/security/x509/hostname.pem
```

4. Import the issued certificate file into the previously created Java keystore (.jks) with the following command:

```
$ keytool -import -trustcacerts -alias <hostname> \
-keystore /opt/cloudera/security/jks/<hostname>-keystore.jks \
-file /opt/cloudera/security/x509/<hostname>.pem -storepass password
```

Approach B - Step 5: Export the private key from the Java keystore and convert it with OpenSSL for reuse by Agent

Follow the steps in Private Key and Certificate Reuse Across Java Keystores and OpenSSL on page 154.

Step 6: Create a File that Contains the Password for the Key

The Agent reads the password from a text file, not from the command line. The password file allows you to use file permissions to protect the password. For our example the password file was created at, `/etc/cloudera-scm-agent/agentkey.pw`.

Step 7: Configure the Agent with its Private Key and Certificate

1. On the Agent host, open the `/etc/cloudera-scm-agent/config.ini` configuration file and edit the following properties:

Property	Description
`client_key_file`	Name of the client key file.
`client_keypw_file`	Name of the client key password file, `agentkey.pw`.
`client_cert_file`	Name of the client certificate file.

2. Repeat these steps on every Agent host.

Step 8: Verify that steps 3-7 Were Completed for every Agent Host in Your Cluster

Important: EachAgent's private key and certificate that you import into the Cloudera Manager Server's truststore must be unique.

Step 9: Create a Truststore by Importing CA and Agent Certificates

Perform this step on the Cloudera Manager server, where the new truststore is used to authenticate Agents.

Create a new truststore file (`/opt/cloudera/security/jks/truststore.jks`) and import the CA root and intermediary/subordinate certificates to this truststore. The new truststore functions like a keystore, containing only certificates and no private key.

Note: Alternatively, you can use the existing Cloudera Manager keystore, containing the CA intermediate and root certificates, as the truststore. However, Cloudera recommends separating the two files, because the new truststore can be copied to and used by all hosts in the cluster when enabling SSL/TLS for CDH services.

1. Create a trusted keystore using the `keytool` command and import the root CA certificate to this truststore.

```
$ keytool -importcert -noprompt -keystore /opt/cloudera/security/jks/truststore.jks \
-alias root_CA -file root.crt -storepass password
```

2. Import any remaining intermediary/subordinate CA certificates into the truststore.

```
$ keytool -importcert -noprompt -keystore /opt/cloudera/security/jks/truststore.jks
-alias int_CA -file intermediate-CA.pem -storepass password
```

3. Collect all `hostname.pem` certificate files from all cluster hosts in a location that Cloudera Manager Server can access. You can now import all the host certificates (`hostname.pem`) into the new truststore.

```
$ keytool -keystore /opt/cloudera/security/jks/truststore.jks \
-importcert -alias hostname -file hostname.pem -storepass password
```

Consider creating a `for` loop on a list of hostnames to speed up this process.

```
$ for HOST in 'cat hostlist.txt'; do keytool -keystore
/opt/cloudera/security/jks/truststore.jks \
-importcert -alias $HOST -file $HOST.pem -storepass password
```

Step 10: Enable Agent Authentication and Configure the Cloudera Manager Server to Use the New Truststore

1. Log into the Cloudera Manager Admin Console.
2. Select **Administration** > **Settings**.
3. Click the **Security** category.
4. Configure the following TLS settings:

Setting	Description
Use TLS Authentication of Agents to Server	Select this option to enable TLS authentication of Agents to the Server.
Path to Truststore	Specify the full filesystem path to the truststore located on the Cloudera Manager Server host; in the example, `/opt/cloudera/security/jks/truststore.jks`
Truststore Password	Specify the password for the truststore.

5. Click **Save Changes** to save the settings.

Step 12: Restart the Cloudera Manager Server

```
$ sudo service cloudera-scm-server restart
```

Step 13: Restart the Cloudera Manager Agents

On every Agent host, restart the Agent:

```
$ sudo service cloudera-scm-agent restart
```

Step 14: Verify that the Server and Agents Are Communicating

In Cloudera Manager Admin Console, open the **Hosts** page. If the Agents heartbeat successfully, the Server and Agents are communicating. If they are not, you may see an error in the Server, such as a `null CA chain` error. This implies that either the truststore does not contain the Agent certificate, or the Agent is not presenting the certificate. Check all of your settings, and check the Server log to verify that TLS and Agent validation have been enabled correctly.

HTTPS Communication in Cloudera Manager

Both the Cloudera Manager Agent and the roles that make up the Cloudera Management Service use HTTPS to communicate with Cloudera Manager and CDH services. This topic aims to explain how the various aspects of HTTPS communication are handled by the Cloudera Manager Agents and the Cloudera Management Service roles.

Cloudera Manager Agents use HTTPS to communicate with HBase, HDFS,Impala, MapReduce, and YARN to collect monitoring data.

Cloudera Manager Agent

Configuring TLS communication between the Cloudera Manager Server and Agents is outlined in Configuring TLS Security for Cloudera Manager on page 155. You can configure the certificates available for server certificate verification using the `verify_cert_dir` parameter in the Agent `config.ini` file. See the comments in the `config.ini` file for a detailed explanation of this property. You can also use the existing value for the `verify_cert_file` parameter.

When the Cloudera Manager Agent communicates with CDH services using HTTPS:

- If `verify_cert_file` or `verify_cert_dir` are configured in the Agent `config.ini`, the Agent uses these settings to verify the server certificates. If these settings are not configured, no certificate verification occurs. If certificate verification is performed for the Cloudera Manager Server, it must also be performed for CDH daemons.
- An Agent never participates in mutual TLS authentication with any CDH service. Instead, each service has its own authentication scheme. Most services use Kerberos authentication, but Impala uses HTTP digest.

User Impact

This depends on how you use certificates.

- If you do not need certificate verification, do not configure `verify_cert_file` or `verify_cert_dir`. However, this leaves you vulnerable to man-in-the-middle attacks.

- If you are using a CA-signed certificate, configure the Agent accordingly. Adding new services or enabling SSL/TLS on a service requires no changes to the Agent configuration because the CA verifies the certificates used by any new servers brought online.
- If you are using self-signed certificates, the certificate for each new service that uses HTTPS must be available to the Agent. Modify the file pointed to by `verify_cert_file` (Agent restart required), or the directory pointed to by `verify_cert_dir`, to contain the new certificate.

Cloudera Management Services

Some Cloudera Management Service roles act as HTTPS clients when communicating with Cloudera Manager entities and CDH services.

You can verify server certificates in two ways:

- Configure a truststore through Cloudera Manager to perform certificate verification on the certificates of the servers with which it communicates. If this truststore is configured, it is used to verify server certificates.

  OR

- If no truststore is configured through Cloudera Manager, the default Java truststore (`cacerts`) is used to verify certificates.

The following table shows Cloudera Management Service roles that act as HTTPS clients as Cloudera Manager entities, and CDH services that communicate with them as HTTPS servers.This table does not depict the entirety of the roles' communication, only communications over HTTPS.

Table 5: HTTPS Communication Between Cloudera Management Service Roles and Cloudera Manager Entities

Roles as HTTPS Clients	Communicating HTTPS Servers
Activity Monitor	• Cloudera Manager Server • JobTracker Web Server • Oozie server (may involve the load balancer in an HA configuration)
Host Monitor	• Cloudera Manager Server
Service Monitor	• Cloudera Manager Server • NameNode(s) Web Server(s) • Impala StateStore Web Server • YARN ResourceManager(s) Web Server(s) • YARN JobHistory Web Server • Oozie server (directly, not through the load balancer)
Event Server	• Cloudera Manager Server
Reports Manager	• Cloudera Manager Server • NameNode(s) Web Servers

Note: The Cloudera Navigator roles also act as HTTPS clients, but are outside the scope of this document.

The Cloudera Management Service roles communicate using HTTPS as follows:

- If the Cloudera Management Service **SSL Client Truststore File Location** parameter is configured, the roles use this truststore to verify server certificates. If this parameter is not set, the default Java truststore is used to verify certificates. Without using safety valves, you cannot verify certificates for some Cloudera Management Service roles but not for others. Nor can you verify certificates for only a subset of the HTTPS communication by a role.

- The Cloudera Management Service roles never participate in mutual TLS authentication with any CDH service or with the Cloudera Manager Server. Instead, each service has its own authentication scheme: Kerberos for most services, HTTP digest for Impala. For the Cloudera Manager Server, this authentication is session-based.

User Impact

This depends on how you use certificates:

- If you use a CA-signed certificate, configure the Cloudera Management Service **SSL Client Truststore File Location** parameter to point to a truststore that contains the CA certificate. Adding a new service or enabling TLS on an existing service requires no changes to the Cloudera Management Service configuration because the CA certificate verifies the certificates used by any new servers brought online. Alternatively, this CA-signed certificate can be added to the default Java truststore.
- If you are using self-signed certificates, the certificate for each new service that uses HTTPS must be available to the Agent.. You must modify the truststore pointed to by the Cloudera Management Service **SSL Client Truststore File Location** parameter. Truststore changes are required on each host on which a Cloudera Management Service daemon is running. Changes to the truststore do not require a role restart, and should be picked up within 10 seconds by default.

  If the Cloudera Management Service **SSL Client Truststore File Location** is not used, the certificate must be made available in the default Java truststore. The Cloudera Management Service role must be restarted for this change to take effect.

Troubleshooting SSL/TLS Connectivity

The `openssl` tool can be run from the host that is running the Cloudera Manager Agent or client service that should be inspected for connectivity issues. You should also test whether the certificate in use by the host is recognized by a trusted CA during the TLS/SSL negotiation.

Use the following command to inspect the connection.

```
$ openssl s_client -connect [host.fqdn.name]:[port]
```

For example:

```
$ openssl s_client -connect test1.sec.cloudera.com:7183
```

A return code 0 means `openssl` was able to establish trust of the server through its library of trusted public CAs. If the certificate was self-signed or provided by a private CA it might be necessary to add the private CA or self-signed certificate to the truststore using the openssl command. Adding the path to the root CA, `-CAfile </path/to/root-ca.pem>`, should allow `openssl` to verify your self-signed or private CA-signed certificate as follows:

```
$ openssl s_client -connect test1.sec.cloudera.com:7183 -CAfile \
/opt/cloudera/security/CAcerts/RootCA.pem
```

Note that providing only the Root CA certificate is necessary to eatablish trust for this test. The result from the command is successful when you see the return code `0` as follows:

```
...
 Verify return code: 0 (ok)
---
```

By default, the Cloudera Manager Server writes logs to the `/etc/cloudera-scm-server/cloudera-scm-server.log` file on startup. Successful start of the server process with the certificate will show logs similar to the following:

```
2014-10-06 21:33:47,515 INFO WebServerImpl:org.mortbay.log: jetty-6.1.26.cloudera.2
2014-10-06 21:33:47,572 INFO WebServerImpl:org.mortbay.log: Started
SslSelectChannelConnector@0.0.0.0:7183
```

```
2014-10-06 21:33:47,573 INFO WebServerImpl:org.mortbay.log: Started
SelectChannelConnector@0.0.0.0:7180
2014-10-06 21:33:47,573 INFO WebServerImpl:com.cloudera.server.cmf.WebServerImpl: Started
 Jetty server.
```

Configuring SSL for Cloudera Navigator

> **Important:** The following instructions assume you have a Java keystore set up on the Navigator Metadata Server host.

1. Open the Cloudera Manager Admin Console and navigate to the **Cloudera Management Service**.
2. Click **Configuration**.
3. Go to the **Navigator Metadata Server Default Group** > **Advanced** category, and add the following strings to the **Navigator Metadata Server Advanced Configuration Snippet (Safety Valve) for cloudera-navigator.properties** property.

```
nav.http.enable_ssl=true
nav.ssl.keyStore=<path to jks keystore with signed server certificate installed>
nav.ssl.keyStorePassword=<password>
```

4. Click **Save Changes**.
5. Restart the Navigator Metadata server.

> **Note:** Once you have enabled SSL, the Quick Links in Cloudera Manager pointing to the Cloudera Navigator UI will not work as they use HTTP, not HTTPS.

Configuring SSL/TLS Encryption for CDH Services

This section describes how to configure encryption for CDH services (HDFS, MapReduce, YARN, HBase, Hive, Impala, Hue and Oozie) focusing on SSL.

Prerequisites

- Cloudera recommends securing a cluster using Kerberos authentication before enabling encryption such as SSL on a cluster. If you enable SSL for a cluster that does not already have Kerberos authentication configured, a warning will be displayed.
- The following sections assume that you have created all the certificates required for SSL communication. If not, for information on how to do this, see Creating Certificates.
- The certificates and keys to be deployed in your cluster should be organized into the appropriate set of keystores and truststores. For more information, see Creating Java Keystores and Truststores on page 151.

Hadoop Services as SSL Servers and Clients

Hadoop services differ in their use of TLS/SSL as follows:

- HDFS, MapReduce, and YARN daemons act as both TLS/SSL servers and clients.
- HBase daemons act as TLS/SSL servers only.
- Oozie daemons act as TLS/SSL servers only.
- Hue acts as an TLS/SSL client to all of the above.

Daemons that act as TLS/SSL servers load the keystores when starting up. When a client connects to an TLS/SSL server daemon, the server transmits the certificate loaded at startup time to the client, which then uses its truststore to validate the server's certificate.

Configuring SSL for HDFS, YARN and MapReduce

Minimum Required Role: Configurator (also provided by **Cluster Administrator, Full Administrator**)

Before You Begin

- Before enabling SSL, keystores containing certificates bound to the appropriate domain names will need to be accessible on all hosts on which at least one HDFS, MapReduce, or YARN daemon role is running.
- Since HDFS, MapReduce, and YARN daemons act as SSL clients as well as SSL servers, they must have access to truststores. In many cases, the most practical approach is to deploy truststores to all hosts in the cluster, as it may not be desirable to determine in advance the set of hosts on which clients will run.
- Keystores for HDFS, MapReduce and YARN must be owned by the `hadoop` group, and have permissions `0440` (that is, readable by owner and group). Truststores must have permissions `0444` (that is, readable by all)
- Cloudera Manager supports SSL configuration for HDFS, MapReduce and YARN at the service level. For each of these services, you must specify absolute paths to the keystore and truststore files. These settings apply to all hosts on which daemon roles of the service in question run. Therefore, the paths you choose must be valid on all hosts.

  An implication of this is that the keystore file names for a given service must be the same on all hosts. If, for example, you have obtained separate certificates for HDFS daemons on hosts `node1.example.com` and `node2.example.com`, you might have chosen to store these certificates in files called `hdfs-node1.keystore` and `hdfs-node2.keystore` (respectively). When deploying these keystores, you must give them both the same name on the target host — for example, `hdfs.keystore`.

- Multiple daemons running on a host can share a certificate. For example, in case there is a DataNode and an Oozie server running on the same host, they can use the same certificate.

Configuring SSL for HDFS

1. Navigate to the **HDFS** service and click **Configuration**.
2. In the Search field, type **SSL** to show the SSL properties (found under the **Service-Wide** > **Security** category).
3. Edit the following properties according to your cluster configuration:

Property	Description
SSL Server Keystore File Location	Path to the keystore file containing the server certificate and private key.
SSL Server Keystore File Password	Password for the server keystore file.
SSL Server Keystore Key Password	Password that protects the private key contained in the server keystore.

4. If you are not using the default truststore, configure SSL client truststore properties:

   Important: The HDFS properties below define a cluster-wide default truststore that can be overridden by YARN and MapReduce (see the **Configuring SSL for YARN and MapReduce** section below).

Property	Description
Cluster-Wide Default SSL Client Truststore Location	Path to the client truststore file. This truststore contains certificates of trusted servers, or of Certificate Authorities trusted to identify servers.
Cluster-Wide Default SSL Client Truststore Password	Password for the client truststore file.

5. Cloudera recommends you enable Web UI authentication for the HDFS service.

Enter **web consoles** in the Search field to bring up the **Enable Authentication for HTTP Web-Consoles** property (found under the **Service-Wide**>**Security** category). Check the property to enable web UI authentication.

Enable Authentication for HTTP Web-Consoles	Enables authentication for hadoop HTTP web-consoles for all roles of this service. **Note:** This is effective only if security is enabled for the HDFS service.

6. Click **Save Changes**.
7. Follow the procedure described in the following **Configuring TLS/SSL for YARN and MapReduce** section, at the end of which you will be instructed to restart all the affected services (HDFS, MapReduce and YARN).

Configuring SSL for YARN and MapReduce

Perform the following steps to configure SSL for the YARN or MapReduce services:

1. Navigate to the **YARN** or **MapReduce** service and click **Configuration**.
2. In the Search field, type **SSL** to show the SSL properties (found under the **Service-Wide** > **Security** category).
3. Edit the following properties according to your cluster configuration:

Property	Description
SSL Server Keystore File Location	Path to the keystore file containing the server certificate and private key.
SSL Server Keystore File Password	Password for the server keystore file.
SSL Server Keystore Key Password	Password that protects the private key contained in the server keystore.

4. Configure the following SSL client truststore properties for MRv1 or YARN only if you want to override the cluster-wide defaults set by the HDFS properties configured above.

Property	Description
SSL Client Truststore File Location	Path to the client truststore file. This truststore contains certificates of trusted servers, or of Certificate Authorities trusted to identify servers.
SSL Client Truststore File Password	Password for the client truststore file.

5. Cloudera recommends you enable Web UI authentication for the service in question.

   Enter **web consoles** in the Search field to bring up the **Enable Authentication for HTTP Web-Consoles** property (found under the **Service-Wide**>**Security** category). Check the property to enable web UI authentication.

Enable Authentication for HTTP Web-Consoles	Enables authentication for hadoop HTTP web-consoles for all roles of this service. **Note:** This is effective only if security is enabled for the HDFS service.

6. Click **Save Changes**.
7. Navigate to the **HDFS** service and in the Search field, type **Hadoop SSL Enabled**. Click the value for the **Hadoop SSL Enabled** property and select the checkbox to enable SSL communication for HDFS, MapReduce, and YARN.

Property	Description
Hadoop SSL Enabled	Enable SSL encryption for HDFS, MapReduce, and YARN web UIs, as well as encrypted shuffle for MapReduce and YARN.

8. Click **Save Changes** to commit the changes.
9. Restart all affected services (HDFS, MapReduce and YARN), as well as their dependent services.

Configuring SSL for HBase

Minimum Required Role: Configurator (also provided by **Cluster Administrator, Full Administrator**)

Before You Begin

- Before enabling SSL, ensure that keystores containing certificates bound to the appropriate domain names will need to be accessible on all hosts on which at least one HBase daemon role is running.
- Keystores for HBase must be owned by the `hbase` group, and have permissions `0440` (that is, readable by owner and group).
- You must specify absolute paths to the keystore and truststore files. These settings apply to all hosts on which daemon roles of the HBase service run. Therefore, the paths you choose must be valid on all hosts.
- Cloudera Manager supports the SSL configuration for HBase at the service level. Ensure you specify absolute paths to the keystore and truststore files. These settings apply to all hosts on which daemon roles of the service in question run. Therefore, the paths you choose must be valid on all hosts.

  An implication of this is that the keystore file names for a given service must be the same on all hosts. If, for example, you have obtained separate certificates for HBase daemons on hosts `node1.example.com` and `node2.example.com`, you might have chosen to store these certificates in files called `hbase-node1.keystore` and `hbase-node2.keystore` (respectively). When deploying these keystores, you must give them both the same name on the target host — for example, `hbase.keystore`.

Procedure

The steps for configuring and enabling SSL for HBase are similar to those for HDFS, YARN and MapReduce:

1. Navigate to the **HBase service** and click **Configuration**.
2. In the Search field, type **SSL** to show the HBase SSL properties (found under the **Service-Wide** > **Security** category).
3. Edit the following SSL properties according to your cluster configuration:

   Table 6: HBase SSL Properties

Property	Description
SSL Server Keystore File Location	Path to the keystore file containing the server certificate and private key.
SSL Server Keystore File Password	Password for the server keystore file.
SSL Server Keystore Key Password	Password that protects the private key contained in the server keystore.

4. Check the **Web UI SSL Encryption Enabled** property.

Web UI SSL Encryption Enabled	Enable SSL encryption for the HBase Master, Region Server, Thrift Server, and REST Server web UIs.

5. Click **Save Changes**.
6. Restart the HBase service.

Configuring SSL for Oozie

Minimum Required Role: Configurator (also provided by **Cluster Administrator, Full Administrator**)

Before You Begin

- Keystores for Oozie must be readable by the `oozie` user. This could be a copy of the Hadoop services' keystore with permissions `0440` and owned by the `oozie` group.
- Truststores must have permissions `0444` (that is, readable by all).
- Specify absolute paths to the keystore and truststore files. These settings apply to all hosts on which daemon roles of the Oozie service run. Therefore, the paths you choose must be valid on all hosts.
- In case there is a DataNode and an Oozie server running on the same host, they can use the same certificate.

For more information on obtaining signed certificates and creating keystores, see SSL Certificates Overview on page 149. You can also view the upstream documentation located here.

> **Important:**
> - You can use either Cloudera Manager or the following command-line instructions to complete this configuration.
> - This information applies specifically to CDH 5.3.x . If you use an earlier version of CDH, see the documentation for that version located at Cloudera Documentation.

Using Cloudera Manager

The steps for configuring and enabling Hadoop SSL for Oozie are as follows:

1. Open the Cloudera Manager Admin Console and navigate to the **Oozie service**.
2. Click **Configuration**.
3. In the Search field, type **SSL** to show the Oozie SSL properties (found under the **Service-Wide** > **Security** category).
4. Edit the following SSL properties according to your cluster configuration.

Table 7: Oozie SSL Properties

Property	Description
Use SSL	Check this field to enable SSL for Oozie.
Oozie SSL Keystore File	Location of the keystore file on the local file system.
Oozie SSL Keystore Password	Password for the keystore.

5. Click **Save Changes**.
6. Restart the Oozie service.

Using the Command Line

To configure the Oozie server to use SSL:

1. Stop Oozie by running

```
sudo /sbin/service oozie stop
```

2. To enable SSL, set the MapReduce version that the Oozie server should work with using the `alternatives` command.

> **Note:** The `alternatives` command is only available on RHEL systems. For SLES, Ubuntu and Debian systems, the command is `update-alternatives`.

For RHEL systems, to use YARN with SSL:

```
alternatives --set oozie-tomcat-conf /etc/oozie/tomcat-conf.https
```

For RHEL systems, to use MapReduce (MRv1) with SSL:

```
alternatives --set oozie-tomcat-conf /etc/oozie/tomcat-conf.https.mr1
```

> **Important:**
>
> The `OOZIE_HTTPS_KEYSTORE_PASS` variable must be the same as the password used when creating the keystore file. If you used a password other than `password`, you'll have to change the value of the `OOZIE_HTTPS_KEYSTORE_PASS` variable in this file.

3. Start Oozie by running

```
sudo /sbin/service oozie start
```

Connect to the Oozie Web UI using SSL (HTTPS)

Use `https://oozie.server.hostname:11443/oozie` though most browsers should automatically redirect you if you use `http://oozie.server.hostname:11000/oozie`.

Additional Considerations when Configuring SSL for Oozie HA

Configure the load balancer to perform SSL pass-through. This will allow clients talking to Oozie to use the SSL certificate provided by the Oozie servers (so the load balancer will not need one). Consult your load balancer's documentation on how to configure this. Make sure to point the load balancer at the `https://HOST:HTTPS_PORT` addresses for your Oozie servers. Clients can then connect to the load balancer at `https://LOAD_BALANCER_HOST:PORT`.

Configuring Encrypted Communication Between Hive and Client Drivers

This topic describes how to set up encrypted communication between HiveServer2 and its clients. Encrypting Hive communication depends on whether you are using Kerberos authentication for communications between HiveServer2 and JDBC/ODBC client drivers.

With Kerberos Enabled

With Kerberos authentication enabled, traffic between the Hive JDBC or ODBC drivers and HiveServer2 can be encrypted using SASL-QOP which allows you to preserve both data integrity (using checksums to validate message integrity) and confidentiality (by encrypting messages). For instructions, see Configuring Encrypted Client/Server Communication for Kerberos-enabled HiveServer2 Connections on page 175.

Without Kerberos Enabled

If you are using any alternate means of authentication, such as LDAP, between HiveServer2 and its clients, you can configure Secure Socket Layer (SSL) communication between them. For instructions, see Configuring Encrypted Client/Server Communication for non-Kerberos HiveServer2 Connections on page 176. For more information on configuring SSL truststores and keystores, see SSL Certificates Overview on page 149.

Configuring Encrypted Client/Server Communication for Kerberos-enabled HiveServer2 Connections

With Kerberos authentication enabled, traffic between the Hive JDBC or ODBC drivers and HiveServer2 can be encrypted which allows you to preserve data integrity (using checksums to validate message integrity) and confidentiality (by

encrypting messages). This can be enabled by setting the `hive.server2.thrift.sasl.qop` property in `hive-site.xml`. For example,

```
<property>
<name>hive.server2.thrift.sasl.qop</name>
<value>auth-conf</value>
<description>Sasl QOP value; one of 'auth', 'auth-int' and 'auth-conf'</description>
</property>
```

Valid settings for the `value` field are:

- `auth`: Authentication only (default)
- `auth-int`: Authentication with integrity protection
- `auth-conf`: Authentication with confidentiality protection

The parameter value that you specify above in the HiveServer2 configuration, should match that specified in the Beeline client connection JDBC URL. For example:

```
!connect jdbc:hive2://ip-10-5-15-197.us-west-2.compute.internal:10000/default;  \
principal=hive/_HOST@US-WEST-2.COMPUTE.INTERNAL;sasl.qop=auth-conf
```

Configuring Encrypted Client/Server Communication for non-Kerberos HiveServer2 Connections

You can use either Cloudera Manager or the command-line to enable SSL encryption for non-Kerberized client connections to HiveServer2.

Using Cloudera Manager

The steps for configuring and enabling SSL for Hive are as follows:

1. Open the Cloudera Manager Admin Console and navigate to the Hive service.
2. Click **Configuration**.
3. In the Search field, type **SSL** to show the Hive SSL properties (found under the **Service-Wide** > **Security** category).
4. Edit the following SSL properties according to your cluster configuration.

   Table 8: Hive SSL Properties

Property	Description
Enable SSL for HiveServer	Enable support for encrypted client-server communication using Secure Socket Layer (SSL) for HiveServer2 connections. Not applicable for Kerberos-enabled connections.
Keystore File Path	Path to the SSL keystore.
Keystore Password	Password for the keystore.

5. Click **Save Changes**.
6. Restart the Hive service.

Using the Command Line

- To enable SSL, add the following configuration parameters to `hive-site.xml` :

```
<property>
  <name>hive.server2.use.SSL</name>
  <value>true</value>
  <description>enable/disable SSL </description>
</property>

<property>
  <name>hive.server2.keystore.path</name>
  <value>keystore-file-path</value>
  <description>path to keystore file</description>
```

```
</property>

<property>
  <name>hive.server2.keystore.password</name>
  <value>keystore-file-password</value>
  <description>keystore password</description>
</property>
```

- The `keystore` must contain the server's certificate.
- The JDBC client must add the following properties in the connection URL when connecting to a HiveServer2 using SSL:

```
;ssl=true[;sslTrustStore=<Trust-Store-Path>;trustStorePassword=<Trust-Store-password>]
```

- Make sure one of the following is true:
  - *Either:* `sslTrustStore` points to the trust store file containing the server's certificate; for example:

```
jdbc:hive2://localhost:10000/default;ssl=true;\
sslTrustStore=/home/usr1/ssl/trust_store.jks;trustStorePassword=xyz
```

  - *or:* the Trust Store arguments are set using the Java system properties `javax.net.ssl.trustStore` and `javax.net.ssl.trustStorePassword`; for example:

```
java -Djavax.net.ssl.trustStore=/home/usr1/ssl/trust_store.jks
-Djavax.net.ssl.trustStorePassword=xyz \
 MyClass jdbc:hive2://localhost:10000/default;ssl=true
```

For more information on using self-signed certificates and the Trust Store, see the Oracle Java SE keytool page.

Configuring SSL for Hue

Minimum Required Role: Configurator (also provided by **Cluster Administrator, Full Administrator**)

Hue as an SSL Client

Hue acts as an SSL client when communicating with Oozie, HBase and core Hadoop services. This means it may have to authenticate authenticate HDFS, MapReduce, and YARN daemons, as well as the HBase Thrift Server, and will need their certificates (or the relevant CA certificate) in its truststore.

Deploying the Hue Truststore:

You can create the Hue truststore by consolidating certificates of all SSL-enabled servers (or a single CA certificate chain) that Hue communicates with into one file. This will generally include certificates of all the HDFS, MapReduce and YARN daemons, and other SSL-enabled services such as Oozie..

The Hue truststore must be in `PEM` format whereas other services use JKS format by default. Hence, to populate the Hue truststore, you will need to extract the certificates from Hadoop's JKS keystores and convert them to `PEM` format. The following example assumes that `hadoop-server.keystore` contains the server certificate identified by alias `foo-1.example.com` and password `example123`.

```
$ keytool -exportcert -keystore hadoop-server.keystore -alias foo-1.example.com \
        -storepass example123 -file foo-1.cert
$ openssl x509 -inform der -in foo-1.cert > foo-1.pem
```

Once you've done this for each host in the cluster, you can concatenate the `PEM` files into one `PEM` file that can serve as the Hue truststore.

```
cat foo-1.pem foo-2.pem ... > huetrust.pem
```

> **Note:** Ensure the final PEM truststore is deployed in a location that is accessible by the Hue service.

In Cloudera Manager, set REQUESTS_CA_BUNDLE to the path of the consolidated PEM file, huetrust.pem created above. To do this:

1. Open the Cloudera Manager Admin Console and navigate to the **Hue service**.
2. Click **Configuration**.
3. In the Search field, type **Hue Service Environment** to show the **Hue Service Environment Advanced Configuration Snippet (Safety Valve)** property (found under the **Service-Wide** > **Advanced** category).
4. Click the **Value** column to enable editing, and add the REQUESTS_CA_BUNDLE property set to the path of the Hue truststore in PEM format.
5. Click **Save Changes**.
6. Restart the Hue service.

Hue as an SSL Server

Hue expects certificates and keys to be stored in PEM format. When managing certificates and keys for such services, using the openssl tool may be more convenient. To configure Hue to use HTTPS, you must generate a private key and a self-signed SSL certificate that does not require a passphrase.

```
openssl genrsa 4096 > server.key
openssl req -new -x509 -nodes -sha1 -key server.key > server.cert
```

> **Note:** You must configure your browser to "trust" this self-signed Hue server certificate.

Ensure secure session cookies for Hue have been enabled in hue.ini under [desktop]>[[session]].

```
[desktop]
  [[session]]
  secure=true
```

For more details on configuring Hue with SSL, see this blog post.

Enabling SSL for the Hue Server using the Command Line

If you are not using Cloudera Manager, update the following properties in hue.ini under [desktop].

```
[desktop]
 ssl_certificate=/path/to/server.cert
 ssl_private_key=/path/to/server.key
```

Enabling SSL for the Hue Server in Cloudera Manager

Perform the following steps in Cloudera Manager to enable SSL for the Hue web server.

1. Open the Cloudera Manager Admin Console and navigate to the **Hue service**.
2. Click **Configuration**.
3. In the Search field, type **SSL** to show the Hue SSL properties (found under the **Hue Server Default Group** category).
4. Edit the following SSL properties according to your cluster configuration.

Property	Description
Enable HTTPS	Enable HTTPS for the Hue web server.
Local Path to SSL Certificate	Path to the SSL certificate on the host running the Hue web server.

Property	Description
Local Path to SSL Private Key	Path to the SSL private key on the host running the Hue web server. Hue only supports a key without a passphrase.

5. Click **Save Changes**.
6. Restart the Hue service.

Configuring SSL for Impala

Impala supports SSL network encryption, between Impala and client programs, and between the Impala-related daemons running on different nodes in the cluster. This feature is important when you also use other features such as Kerberos authentication or Sentry authorization, where credentials are being transmitted back and forth.

Important:

- You can use either Cloudera Manager or the following command-line instructions to complete this configuration.
- This information applies specifically to CDH 5.3.x . If you use an earlier version of CDH, see the documentation for that version located at Cloudera Documentation.

Using Cloudera Manager

To configure Impala to listen for Beeswax and HiveServer2 requests on SSL-secured ports:

1. Open the Cloudera Manager Admin Console and navigate to the **Impala** service.
2. Click **Configuration**.
3. In the Search field, type **SSL** to show the Impala SSL properties (found under the **Service-Wide** > **Security** category).
4. If you are ***not*** using Kerberos, edit the following SSL properties according to your cluster configuration.

Table 9: Impala SSL Properties

Property	Description
Enable SSL for Impala Client Services	Enable support for encrypted client-server communication using Secure Socket Layer (SSL) for Impala client services.
SSL Server Certificate	Local path to the SSL server certificate file.
SSL Server Private Key	Local path to the SSL private key file.

Cloudera Manager 5.3 does not have the ability to configure SSL when Kerberos is enabled. Hence, if you are using Kerberos, add the following properties to the **Impala Command Line Argument Advanced Configuration Snippet (Safety Valve)** property (found under the **Service-Wide** > **Advanced** category):

```
--ssl_server_certificate=/etc/impala/<certificate>.pem
--ssl_private_key=/etc/impala/<private_key>.pem
```

5. Click **Save Changes**.
6. Restart the Impala service.

For information on configuring SSL communication with the `impala-shell` interpreter, see the section below on **Configuring SSL Communication for the Impala Shell**.

Using the Command Line

To enable SSL for Impala network communication, add both of the following flags to the `impalad` startup options:

- `--ssl_server_certificate`: the full path to the server certificate, on the local filesystem.

- `--ssl_private_key`: the full path to the server private key, on the local filesystem.

If either of these flags are set, both must be set. In that case, Impala starts listening for Beeswax and HiveServer2 requests on SSL-secured ports only. (The port numbers stay the same; see Ports Used by Impala for details.)

Configuring SSL Communication for the Impala Shell

Typically, a client program has corresponding configuration properties in Cloudera Manager to verify that it is connecting to the right server. For example, with SSL enabled for Impala, you use the following options when starting the `impala-shell` interpreter:

- `--ssl`: enables SSL for `impala-shell`.
- `--ca_cert`: the local pathname pointing to the third-party CA certificate, or to a copy of the server certificate for self-signed server certificates.

If `--ca_cert` is not set, `impala-shell` enables SSL, but does not validate the server certificate. This is useful for connecting to a known-good Impala that is only running over SSL, when a copy of the certificate is not available (such as when debugging customer installations).

Using TLS/SSL with Business Intelligence Tools

You can use Kerberos authentication, TLS/SSL encryption, or both to secure connections from JDBC and ODBC applications to Impala. See Configuring Impala to Work with JDBC and Configuring Impala to Work with ODBC for details.

Currently, the Hive JDBC driver does not support connections that use both Kerberos authentication and SSL encryption. To use both of these security features with Impala through a JDBC application, use the Cloudera JDBC Connector as the JDBC driver.

Configuring HttpFS to use SSL

Configure the HttpFS Server to use SSL (HTTPS)

1. Stop HttpFS by running

```
sudo /sbin/service hadoop-httpfs stop
```

2. To enable SSL, change which configuration the HttpFS server should work with using the `alternatives` command.

> **Note:** The `alternatives` command is only available on RHEL systems. For SLES, Ubuntu and Debian systems, the command is `update-alternatives`.

For RHEL systems, to use SSL:

```
alternatives --set hadoop-httpfs-tomcat-conf /etc/hadoop-httpfs/tomcat-conf.https
```

> **Important:**
>
> The `HTTPFS_SSL_KEYSTORE_PASS` variable must be the same as the password used when creating the keystore file. If you used a password other than `password`, you'll have to change the value of the `HTTPFS_SSL_KEYSTORE_PASS` variable in `/etc/hadoop-httpfs/conf/httpfs-env.sh`.

3. Start HttpFS by running

```
sudo /sbin/service hadoop-httpfs start
```

Connect to the HttpFS Web UI using SSL (HTTPS)

Use `https://<httpfs_server_hostname>:14000/webhdfs/v1/` though most browsers should automatically redirect you if you use `http://<httpfs_server_hostname>:14000/webhdfs/v1/`

> **Important:**
>
> If using a Self-Signed Certificate, your browser will warn you that it can't verify the certificate or something similar. You will probably have to add your certificate as an exception.

Encrypted Shuffle and Encrypted Web UIs

Once you've enabled Kerberos, which provides for strong authentication, you can optionally enable network encryption if you so desire. CDH 5 supports the Encrypted Shuffle and Encrypted Web UIs feature that allows encryption of the MapReduce shuffle and web server ports using HTTPS with optional client authentication (also known as bi-directional HTTPS, or HTTPS with client certificates). It includes:

- Hadoop configuration setting for toggling the shuffle between HTTP and HTTPS.
- Hadoop configuration setting for toggling the Web UIs to use either HTTP or HTTPS.
- Hadoop configuration settings for specifying the keystore and truststore properties (location, type, passwords) that are used by the shuffle service, web server UIs and the reducers tasks that fetch shuffle data.
- A way to re-read truststores across the cluster (when a node is added or removed).

CDH 5 supports Encrypted Shuffle for both MRv1 and MRv2 (YARN), with common configuration properties used for both versions. The only configuration difference is in the parameters used to enable the features:

- For **MRv1**, setting the `hadoop.ssl.enabled` parameter in the `core-site.xml` file enables both the Encrypted Shuffle and the Encrypted Web UIs. In other words, the encryption toggling is coupled for the two features.
- For **MRv2**, setting the `hadoop.ssl.enabled` parameter enables the Encrypted Web UI feature; setting the `mapreduce.shuffle.ssl.enabled` parameter in the `mapred-site.xml` file enables the Encrypted Shuffle feature.

All other configuration properties apply to both the Encrypted Shuffle and Encrypted Web UI functionality.

When the Encrypted Web UI feature is enabled, all Web UIs for Hadoop components are served over HTTPS. If you configure the systems to require client certificates, browsers must be configured with the appropriate client certificates in order to access the Web UIs.

> **Important:**
>
> When the Web UIs are served over HTTPS, you must specify `https://` as the protocol; there is no redirection from `http://`. If you attempt to access an HTTPS resource over HTTP, your browser will probably show an empty screen with no warning.

Most components that run on top of MapReduce automatically use Encrypted Shuffle when it is configured.

Configuring Encrypted Shuffle and Encrypted Web UIs

To configure Encrypted Shuffle and Encrypted Web UIs, set the appropriate property/value pairs in the following:

- `core-site.xml` enables these features and defines the implementation
- `mapred-site.xml` enables Encrypted Shuffle for MRv2
- `ssl-server.xml` stores keystone and truststore settings for the server
- `ssl-client.xml` stores keystone and truststore settings for the client

core-site.xml Properties

To configure encrypted shuffle, set the following properties in the `core-site.xml` files of all nodes in the cluster:

Property	Default Value	Explanation
`hadoop.ssl.enabled`	`false`	For MRv1, setting this value to `true` enables both the Encrypted Shuffle and the Encrypted Web UI features. For MRv2, this property only enables the Encrypted WebUI; Encrypted Shuffle is enabled with a property in the `mapred-site.xml` file as described below.
`hadoop.ssl.require.client.cert`	`false`	When this property is set to `true`, client certificates are required for all shuffle operations and all browsers used to access Web UIs. Cloudera recommends that this be set to `false`. See Client Certificates on page 186.
`hadoop.ssl.hostname.verifier`	`DEFAULT`	The hostname verifier to provide for HttpsURLConnections. Valid values are: `DEFAULT`, `STRICT`, `STRICT_I6`, `DEFAULT_AND_LOCALHOST` and `ALLOW_ALL`.
`hadoop.ssl.keystores.factory.class`	`org.apache.hadoop.security.ssl.FileBasedKeyStoresFactory`	The `KeyStoresFactory` implementation to use.
`hadoop.ssl.server.conf`	`ssl-server.xml`	Resource file from which ssl server keystore information is extracted. This file is looked up in the `classpath`; typically it should be in the `/etc/hadoop/conf/` directory.
`hadoop.ssl.client.conf`	`ssl-client.xml`	Resource file from which ssl server keystore information is extracted. This file is looked up in the `classpath`; typically it should be in the `/etc/hadoop/conf/` directory.

Note:

All these properties should be marked as final in the cluster configuration files.

Example

```
<configuration>
    ...
    <property>
      <name>hadoop.ssl.require.client.cert</name>
      <value>false</value>
      <final>true</final>
    </property>

    <property>
```

```
      <name>hadoop.ssl.hostname.verifier</name>
      <value>DEFAULT</value>
      <final>true</final>
    </property>

    <property>
      <name>hadoop.ssl.keystores.factory.class</name>
      <value>org.apache.hadoop.security.ssl.FileBasedKeyStoresFactory</value>
      <final>true</final>
    </property>

    <property>
      <name>hadoop.ssl.server.conf</name>
      <value>ssl-server.xml</value>
      <final>true</final>
    </property>

    <property>
      <name>hadoop.ssl.client.conf</name>
      <value>ssl-client.xml</value>
      <final>true</final>
    </property>

    <property>
      <name>hadoop.ssl.enabled</name>
      <value>true</value>
    </property>
    ...
</configuration>
```

The cluster should be configured to use the Linux Task Controller in MRv1 and Linux container executor in MRv2 to run job tasks so that they are prevented from reading the server keystore information and gaining access to the shuffle server certificates. Refer to Appendix B - Information about Other Hadoop Security Programs for more information.

mapred-site.xml Property (MRv2 only)

To enable Encrypted Shuffle for MRv2, set the following property in the `mapred-site.xml` file on every node in the cluster:

Property	Default Value	Explanation
`mapreduce.shuffle.ssl.enabled`	`false`	If this property is set to `true`, encrypted shuffle is enabled. If this property is not specified, it defaults to the value of `hadoop.ssl.enabled`. This value can be `false` when `hadoop.ssl.enabled` is `true` but cannot be `true` when `hadoop.ssl.enabled` is `false`

This property should be marked as `final` in the cluster configuration files.

Example:

```
<configuration>
    ...
    <property>
      <name>mapreduce.shuffle.ssl.enabled</name>
      <value>true</value>
      <final>true</final>
    </property>
    ...
</configuration>
```

Keystore and Truststore Settings

`FileBasedKeyStoresFactory` is the only `KeyStoresFactory` that is currently implemented. It uses properties in the `ssl-server.xml` and `ssl-client.xml` files to configure the keystores and truststores.

ssl-server.xml (Shuffle server and Web UI) Configuration

Use the following settings to configure the keystores and truststores in the `ssl-server.xml` file.

> **Note:**
> The `ssl-server.xml` should be owned by the `hdfs` or `mapred` Hadoop system user, belong to the hadoop group, and it should have 440 permissions. Regular users should not belong to the hadoop group.

Property	Default Value	Description
`ssl.server.keystore.type`	`jks`	Keystore file type
`ssl.server.keystore.location`	NONE	Keystore file location. The `mapred` user should own this file and have exclusive read access to it.
`ssl.server.keystore.password`	NONE	Keystore file password
`ssl.server.keystore.keypassword`	NONE	Key password
`ssl.server.truststore.type`	`jks`	Truststore file type
`ssl.server.truststore.location`	NONE	Truststore file location. The `mapred` user should own this file and have exclusive read access to it.
`ssl.server.truststore.password`	NONE	Truststore file password
`ssl.server.truststore.reload.interval`	10000	Truststore reload interval, in milliseconds

Example

```
<configuration>
<!-- Server Certificate Store -->
<property>
  <name>ssl.server.keystore.type</name>
     <value>jks</value>
</property>
<property>
  <name>ssl.server.keystore.location</name>
  <value>${user.home}/keystores/server-keystore.jks</value>
</property>
<property>
  <name>ssl.server.keystore.password</name>
  <value>serverfoo</value>
</property>
<property>
  <name>ssl.server.keystore.keypassword</name>
  <value>serverfoo</value>
</property>

<!-- Server Trust Store -->
<property>
  <name>ssl.server.truststore.type</name>
  <value>jks</value>
</property>
<property>
  <name>ssl.server.truststore.location</name>
  <value>${user.home}/keystores/truststore.jks</value>
```

```
</property>
<property>
  <name>ssl.server.truststore.password</name>
  <value>clientserverbar</value>
</property>
<property>
  <name>ssl.server.truststore.reload.interval</name>
  <value>10000</value>
</property>
</configuration>
```

ssl-client.xml (Reducer/Fetcher) Configuration

Use the following settings to configure the keystores and truststores in the `ssl-client.xml` file. This file should be owned by the `mapred` user for MRv1 and by the `yarn` user for MRv2; the file permissions should be 444 (read access for all users).

Property	Default Value	Description
`ssl.client.keystore.type`	`jks`	Keystore file type
`ssl.client.keystore.location`	NONE	Keystore file location. The mapred user should own this file and it should have default permissions.
`ssl.client.keystore.password`	NONE	Keystore file password
`ssl.client.keystore.keypassword`	NONE	Key password
`ssl.client.truststore.type`	`jks`	Truststore file type
`ssl.client.truststore.location`	NONE	Truststore file location. The mapred user should own this file and it should have default permissions.
`ssl.client.truststore.password`	NONE	Truststore file password
`ssl.client.truststore.reload.interval`	10000	Truststore reload interval, in milliseconds

Example

```
<configuration>

  <!-- Client certificate Store -->
  <property>
    <name>ssl.client.keystore.type</name>
    <value>jks</value>
  </property>
  <property>
    <name>ssl.client.keystore.location</name>
    <value>${user.home}/keystores/client-keystore.jks</value>
  </property>
  <property>
    <name>ssl.client.keystore.password</name>
    <value>clientfoo</value>
  </property>
  <property>
    <name>ssl.client.keystore.keypassword</name>
    <value>clientfoo</value>
  </property>

  <!-- Client Trust Store -->
  <property>
    <name>ssl.client.truststore.type</name>
    <value>jks</value>
```

```
  </property>
  <property>
    <name>ssl.client.truststore.location</name>
    <value>${user.home}/keystores/truststore.jks</value>
  </property>
  <property>
    <name>ssl.client.truststore.password</name>
    <value>clientserverbar</value>
  </property>
  <property>
    <name>ssl.client.truststore.reload.interval</name>
    <value>10000</value>
  </property>
</configuration>
```

Activating Encrypted Shuffle

When you have made the above configuration changes, activate Encrypted Shuffle by re-starting all TaskTrackers in MRv1 and all NodeManagers in YARN.

> **Important:**
>
> Encrypted shuffle has a significant performance impact. You should benchmark this before implementing it in production. In many cases, one or more additional cores are needed to maintain performance.

Client Certificates

Client Certificates are supported but they do not guarantee that the client is a reducer task for the job. The Client Certificate keystore file that contains the private key must be readable by all users who submit jobs to the cluster, which means that a rogue job could read those keystore files and use the client certificates in them to establish a secure connection with a Shuffle server. The JobToken mechanism that the Hadoop environment provides is a better protector of the data; each job uses its own JobToken to retrieve only the shuffle data that belongs to it. Unless the rogue job has a proper JobToken, it cannot retrieve Shuffle data from the Shuffle server.

> **Important:**
>
> If your certificates are signed by a certificate authority (CA), you must include the complete chain of CA certificates in the keystore that has the server's key.

Reloading Truststores

By default, each truststore reloads its configuration every 10 seconds. If a new truststore file is copied over the old one, it is re-read, and its certificates replace the old ones. This mechanism is useful for adding or removing nodes from the cluster, or for adding or removing trusted clients. In these cases, the client, TaskTracker or NodeManager certificate is added to (or removed from) all the truststore files in the system, and the new configuration is picked up without requiring that the TaskTracker in MRv1 and NodeManager in YARN daemons are restarted.

> **Note:**
>
> The keystores are not automatically reloaded. To change a keystore for a TaskTracker in MRv1 or a NodeManager in YARN, you must restart the TaskTracker or NodeManager daemon.

The reload interval is controlled by the `ssl.client.truststore.reload.interval` and `ssl.server.truststore.reload.interval` configuration properties in the `ssl-client.xml` and `ssl-server.xml` files described above.

Debugging

> **Important:**
>
> Enable debugging only for troubleshooting, and then only for jobs running on small amounts of data. Debugging is very verbose and slows jobs down significantly.

To enable SSL debugging in the reducers, set `-Djavax.net.debug=all` in the `mapred.reduce.child.java.opts` property; for example:

```
<configuration>
    ...
    <property>
        <name>mapred.reduce.child.java.opts</name>
        <value>-Xmx200m -Djavax.net.debug=all</value>
    </property>
    ...
</configuration>
```

You can do this on a per-job basis, or by means of a cluster-wide setting in `mapred-site.xml`.

To set this property in TaskTrackersfor MRv1, set it in `hadoop-env.sh`:

```
HADOOP_TASKTRACKER_OPTS="-Djavax.net.debug=all $HADOOP_TASKTRACKER_OPTS"
```

To set this property in NodeManagers for YARN, set it in `hadoop-env.sh`:

```
YARN_OPTS="-Djavax.net.debug=all $YARN_OPTS"
```

HDFS Data At Rest Encryption

> **Important:** Cloudera provides two solutions:
>
> - **Navigator Encrypt** is production ready and available to Cloudera customers licensed for Cloudera Navigator. Navigator Encrypt operates at the Linux volume level, so it can encrypt cluster data inside and outside HDFS. Consult your Cloudera account team for more information.
> - **HDFS Encryption** is production ready and operates at the HDFS directory level, enabling encryption to be applied only to HDFS folders where needed.

HDFS Encryption implements transparent, end-to-end encryption of data read from and written to HDFS, without requiring changes to application code. Because the encryption is end-to-end, data can be encrypted and decrypted only by the client. HDFS does not store or have access to unencrypted data or encryption keys. This supports both, at-rest encryption (data on persistent media, such as a disk) and in-transit encryption (data traveling over a network).

Use Cases

Data encryption is required by a number of different government, financial, and regulatory entities. For example, the healthcare industry has HIPAA regulations, the card payment industry has PCI DSS regulations, and the United States government has FISMA regulations. Transparent encryption in HDFS makes it easier for organizations to comply with these regulations. Encryption can also be performed at the application-level, but by integrating it into HDFS, existing applications can operate on encrypted data without changes. This integrated architecture implements stronger encrypted file semantics and better coordination with other HDFS functions.

Architecture

Encryption Zones

An encryption zone is a directory in HDFS with all of its contents, that is, every file and subdirectory in it, encrypted. The files in this directory will be transparently encrypted upon write and transparently decrypted upon read. Each encryption zone is associated with a key which is specified when the zone is created. Each file within an encryption zone also has its own encryption/decryption key, called the Data Encryption Key (DEK). These DEKs are never stored persistently unless they are encrypted with the encryption zone's key. This encrypted DEK is known as the EDEK. The EDEK is then stored persistently as part of the file's metadata on the NameNode.

A key can have multiple key versions, where each key version has its own distinct key material (that is, the portion of the key used during encryption and decryption). Key rotation is achieved by modifying the encryption zone's key, that is, bumping up its version. Per-file key rotation is then achieved by re-encrypting the file's DEK with the new encryption zone key to create new EDEKs. An encryption key can be fetched either by its key name, returning the latest version of the key, or by a specific key version.

Key Management Server

A new service needs to be added to your cluster to store, manage, and access encryption keys, called the **Hadoop Key Management Server** (KMS). The KMS service is a proxy that interfaces with a backing key store on behalf of HDFS daemons and clients. Both the backing key store and the KMS implement the Hadoop KeyProvider client API.

Encryption and decryption of EDEKs happens entirely on the KMS. More importantly, the client requesting creation or decryption of an EDEK never handles the EDEK's encryption key (that is, the encryption zone key). When a new file is created in an encryption zone, the NameNode asks the KMS to generate a new EDEK encrypted with the encryption zone's key. When reading a file from an encryption zone, the NameNode provides the client with the file's EDEK and the encryption zone key version that was used to encrypt the EDEK. The client then asks the KMS to decrypt the EDEK, which involves checking that the client has permission to access the encryption zone key version. Assuming that is successful, the client uses the DEK to decrypt the file's contents. All the steps for read and write take place automatically through interactions between the DFSClient, the NameNode, and the KMS.

Access to encrypted file data and metadata is controlled by normal HDFS filesystem permissions. Typically, the backing key store is configured to only allow end-user access to the encryption zone keys used to encrypt DEKs. This means that EDEKs can be safely stored and handled by HDFS, since the `hdfs` user will not have access to EDEK encryption keys. This means that if HDFS is compromised (for example, by gaining unauthorized access to a superuser account), a malicious user only gains access to the ciphertext and EDEKs. This does not pose a security threat since access to encryption zone keys is controlled by a separate set of permissions on the KMS and key store.

For more details on configuring the KMS, see Configuring the Key Management Server (KMS) on page 195.

Navigator Key Trustee

By default, the current implementation of HDFS encryption uses a local Java keystore for key management. This KMS (File) service may not be sufficient for large enterprises where a more robust and secure key management solution is required. Navigator Key Trustee is a keystore server for managing encryption keys, certificates, and passwords that is completely integrated into Cloudera Navigator.

In order to leverage the manageable, highly-available key management capabilities of the Navigator Key Trustee server, the KMS (Navigator Key Trustee) service uses a Key Trustee-specific plugin called the TrusteeKeyProvider.

For more information on integrating Navigator Key Trustee with HDFS encryption, you can contact your Cloudera account team.

`crypto` Command Line Interface

createZone

Use this command to create a new encryption zone.

```
-createZone -keyName <keyName> -path <path>
```

Where:

- `path`: The path of the encryption zone to be created. It must be an empty directory.
- `keyName`: Name of the key to use for the encryption zone.

listZones

List all encryption zones. This command requires superuser permissions.

```
-listZones
```

Enabling HDFS Encryption on a Cluster

Minimum Required Role: Full Administrator

The following sections will guide you through enabling HDFS encryption on your cluster, using the default Java keystore-based KMS:

> **Important:** If you want to use Navigator Key Trustee, and not the default JavaKeyStoreProvider, to store HDFS encryption keys, you will need to add the **KMS (Navigator Key Trustee)** service to Cloudera Manager and install the Navigator Key Trustee Server. Contact your Cloudera account team for more information.

Adding the KMS Service

1. On the Home page, click to the right of the cluster name and select **Add a Service**. A list of service types display. You can add one type of service at a time.
2. Select the **KMS (File)** service and click **Continue**.
3. Customize the assignment of role instances to hosts. You can click the **View By Host** button for an overview of the role assignment by hostname ranges.

   Click the field below the Key Management Server (KMS) role to display a dialog containing a list of hosts. Select the host for the new KMS role and click **OK**.
4. Review and modify the **JavaKeyStoreProvider Directory** configuration setting if required and click **Continue**. The KMS (File) service is started.
5. Click **Continue**, then click **Finish**. You are returned to the Home page.
6. Verify that the new KMS (File) service has started properly by checking its health status. If the Health Status is **Good**, then the service started properly.

Enabling KMS for the HDFS Service

1. Go to the HDFS service.
2. Click the **Configuration** tab.
3. Go to the **Service-Wide** category.
4. Click the **Value** field for the **KMS Service** property and select **KMS (File)**.
5. Click **Save Changes**.
6. Restart your cluster.
   a. On the Home page, click to the right of the cluster name and select **Restart**.
   b. Click **Restart** that appears in the next screen to confirm. The **Command Details** window shows the progress of stopping services.

When **All services successfully started** appears, the task is complete and you can close the **Command Details** window.

7. Deploy client configuration.
   a. On the Home page, click

   to the right of the cluster name and select **Deploy Client Configuration**.
   b. Click **Deploy Client Configuration**.

Optimizing HDFS Encryption at Rest

CDH 5.3 implements the ***Advanced Encryption Standard New Instructions*** (AES-NI), which provide substantial performance improvements. To get these improvements, you need a recent version of `libcrypto.so` on HDFS and MapReduce client hosts -- that is, any host from which you originate HDFS or MapReduce requests. Many OS versions have an older version of the library that does not support AES-NI. The instructions that follow tell you what you need to do for each OS version that CDH 5.3 supports.

RHEL/CentOS 6.5 or later

The installed version of `libcrypto.so` supports AES-NI, but you need to install the `openssl-devel` package on all clients:

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

RHEL/CentOS 6.4 or earlier 6.x versions, or SLES 11

Download and extract a newer version of `libcrypto.so` from a CentOS 6.5 repository and install it on all clients in `/var/lib/hadoop/extra/native/`:

1. Download the latest version of the `openssl` package:

```
$ wget
http://mirror.centos.org/centos/6/os/x86_64/Packages/openssl-1.0.1e-30.el6.x86_64.rpm
```

> **Note:**
> The `libcrypto.so` in this package can be used on SLES 11 as well as RHEL/CentOS

2. Decompress the files in the package, but **do not** install it:

```
$ rpm2cpio openssl-1.0.1e-30.el6.x86_64.rpm | cpio -idmv
```

3. If you are using parcels, create the `/var/lib/hadoop/extra/native/` directory:

```
$ sudo mkdir -p /var/lib/hadoop/extra/native
```

4. Copy the shared library into `/var/lib/hadoop/extra/native/` .

> **Important:**
> Name the target file `libcrypto.so`, with no suffix at the end, exactly as in the command that follows.

```
$ sudo cp ./usr/lib64/libcrypto.so.1.0.1e /var/lib/hadoop/extra/native/libcrypto.so
```

RHEL/CentOS 5

In this case, you need to build `libcrypto.so` and copy it to all clients:

1. On one client, compile and install `openssl` from source:

```
$ wget http://www.openssl.org/source/openssl-1.0.1j.tar.gz
$ cd openssl-1.0.1j
$ ./config --shared --prefix=/opt/openssl-1.0.1j
$ sudo make install
```

2. If you are using parcels, create the `/var/lib/hadoop/extra/native/` directory:

```
$ sudo mkdir -p /var/lib/hadoop/extra/native
```

3. Copy the files into `/var/lib/hadoop/extra/native/`:

```
$ sudo cp /opt/openssl-1.0.1j/lib/libcrypto.so /var/lib/hadoop/extra/native
```

4. Copy the files to the remaining clients using a utility such as `rsync`

Debian Wheezy

The installed version of `libcrypto.so` supports AES-NI, but you need to install the `libssl-devel` package on all clients:

```
$ sudo apt-get install libssl-dev
```

Ubuntu Precise and Ubuntu Trusty

Install the `libssl-devel` package on all clients:

```
$ sudo apt-get install libssl-dev
```

Testing if encryption optimization works

To verify that a client host is ready to make use of the AES-NI instruction set optimization for HDFS encryption at rest, use the following command:

```
hadoop checknative
```

You should see a response such as the following:

```
14/12/12 13:48:39 INFO bzip2.Bzip2Factory: Successfully loaded & initialized native-bzip2
library system-native14/12/12 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 right version.

Configuring Encryption Properties for the HDFS and NameNode

Configure the following properties to select the encryption algorithm and KeyProvider that will be used during encryption. If you do not modify these properties, the default values will use AES-CTR to encrypt your data.

> **Note:** You may notice many of the properties described in the table below are absent from your `kms-site.xml`. In such a case, the default values listed here are being used.
>
> For a managed cluster, since the properties listed below have not been exposed in Cloudera Manager, use the corresponding safety valves if you want to specify a different value.

Property	Description
Selecting an Encryption Algorithm: Set the following properties in the `core-site.xml` safety valve and redeploy client configuration.	
`hadoop.security.crypto.codec.classes.EXAMPLECIPHERSUITE`	The prefix for a given crypto codec, contains a comma-separated list of implementation classes for a given crypto codec (for example, `EXAMPLECIPHERSUITE`). The first implementation will be used if available, others are fallbacks. By default, the cipher suite used is `AES/CTR/NoPadding` and its default classes are `org.apache.hadoop.crypto.OpensslAesCtrCryptoCodec` and `org.apache.hadoop.crypto.JceAesCtrCryptoCodec` as described in the following properties.
`hadoop.security.crypto.cipher.suite`	Cipher suite for crypto codec. Default: `AES/CTR/NoPadding`
`hadoop.security.crypto.codec.classes.aes.ctr.nopadding`	Comma-separated list of crypto codec implementations for the default cipher suite: `AES/CTR/NoPadding`. The first implementation will be used if available, others are fallbacks. Default: `org.apache.hadoop.crypto.OpensslAesCtrCryptoCodec`, `org.apache.hadoop.crypto.JceAesCtrCryptoCodec`
`hadoop.security.crypto.jce.provider`	The JCE provider name used in CryptoCodec. Default: None
`hadoop.security.crypto.buffer.size`	The buffer size used by CryptoInputStream and CryptoOutputStream. Default: `8192`
KeyProvider Configuration: Set this property in the `hdfs-site.xml` safety valve and restart the NameNode.	
`dfs.encryption.key.provider.uri`	The KeyProvider to be used when interacting with encryption keys that are used to read and write to an encryption zone. If you have a managed cluster, Cloudera Manager will point to the KMS server you have enabled above.
NameNode Configuration: Set this property in the `hdfs-site.xml` safety valve and restart the NameNode.	
`dfs.namenode.list.encryption.zones.num.responses`	When listing encryption zones, the maximum number of zones that will be returned in a batch. Fetching the list incrementally in batches improves NameNode performance.

Property	Description
	Default: `100`

Creating Encryption Zones

Once a KMS has been set up and the NameNode and HDFS clients have been correctly configured, an admin user can use the `hadoop key` and `hdfs crypto` command-line tools to create encryption keys and set up new encryption zones.

- Create an encryption key for your zone as the application user that will be using the key. For example, if you are creating an encryption zone for HBase, create the key as the `hbase` user as follows:

```
$ sudo -u hbase hadoop key create <key_name>
```

- Create a new empty directory and make it an encryption zone using the key created above.

```
$ hadoop fs -mkdir /zone
$ hdfs crypto -createZone -keyName <key_name> -path /zone
```

You can verify creation of the new encryption zone by running the `-listZones` command. You should see the encryption zone along with its key listed as follows:

```
$ sudo -u hdfs hdfs crypto -listZones
/zone    <key_name>
```

Warning: Do not delete an encryption key as long as it is still in use for an encryption zone. This will result in loss of access to data in that zone.

For more information and recommendations on creating encryption zones for each CDH component, see Configuring CDH Services for HDFS Encryption on page 205.

Adding Files to an Encryption Zone

Existing data can be encrypted by coping it copied into the new encryption zones using tools like distcp. See the **DistCp Considerations** section below for information on using DistCp with encrypted data files.

You can add files to an encryption zone by copying them over to the encryption zone. For example:

```
sudo -u hdfs hadoop distcp /user/dir /user/enczone
```

Additional Information:

- For more information on KMS setup and high availability configuration, see Configuring the Key Management Server (KMS) on page 195.
- For instructions on securing the KMS using Kerberos, SSL communication and ACLs, see Securing the Key Management Server (KMS) on page 198.
- If you want to use the KMS to encrypt data used by other CDH services, see Configuring CDH Services for HDFS Encryption on page 205 for information on recommended encryption zones for each service.

DistCp Considerations

A common usecase for DistCp is to replicate data between clusters for backup and disaster recovery purposes. This is typically performed by the cluster administrator, who is an HDFS superuser. To retain this workflow when using HDFS encryption, a new virtual path prefix has been introduced, `/.reserved/raw/`, that gives superusers direct access to the underlying block data in the filesystem. This allows superusers to `distcp` data without requiring access to encryption

keys, and avoids the overhead of decrypting and re-encrypting data. It also means the source and destination data will be byte-for-byte identical, which would not have been true if the data was being re-encrypted with a new EDEK.

> **Warning:**
>
> When using `/.reserved/raw/` to `distcp` encrypted data, it's important to preserve extended attributes with the `-px` flag. This is because encrypted attributes such as the EDEK are exposed through extended attributes and *must* be preserved to be able to decrypt the file.
>
> This means that if the `distcp` is initiated at or above the encryption zone root, it will automatically create a new encryption zone at the destination if it does not already exist. Hence, Cloudera recommends you first create identical encryption zones on the destination cluster to avoid any potential mishaps.

Copying between encrypted and unencrypted locations

By default, `distcp` compares checksums provided by the filesystem to verify that data was successfully copied to the destination. When copying between an unencrypted and encrypted location, the filesystem checksums will not match since the underlying block data is different.

In this case, you can specify the `-skipcrccheck` and `-update` flags to avoid verifying checksums.

Attack Vectors

Type of Exploit	Issue	Mitigation
Hardware Access Exploit		
These exploits assume the attacker has gained physical access to hard drives from cluster machines, that is, DataNodes and NameNodes.	*Access to swap files of processes containing DEKs*. This exploit does not expose cleartext, as it also requires access to encrypted block files.	It can be mitigated by disabling swap, using encrypted swap, or using mlock to prevent keys from being swapped out.
	Access to encrypted block files. This exploit does not expose cleartext, as it also requires access to the DEKs.	It can only be mitigated by restricting physical access to the cluster machines.
Root Access Exploits		
These exploits assume the attacker has gained root shell access to cluster machines running datanodes and namenodes. Many of these exploits cannot be addressed in HDFS, since a malicious root user has access to the in-memory state of processes holding encryption keys and cleartext. For these exploits, the only mitigation technique is carefully restricting and monitoring root shell access.	Access to encrypted block files. By itself, this does not expose cleartext, as it also requires access to encryption keys.	No mitigation required.
	Dump memory of client processes to obtain DEKs, delegation tokens, cleartext.	No mitigation.
	Recording network traffic to sniff encryption keys and encrypted data in transit. By itself, insufficient to read cleartext without the EDEK encryption key.	No mitigation required.
	Dump memory of datanode process to obtain encrypted block data. By itself, insufficient to read cleartext without the DEK.	No mitigation required.

Type of Exploit	Issue	Mitigation
	Dump memory of namenode process to obtain encrypted data encryption keys. By itself, insufficient to read cleartext without the EDEK's encryption key and encrypted block files.	No mitigation required.
HDFS Admin Exploits		
These exploits assume that the attacker has compromised HDFS, but does not have root or hdfs user shell access.	Access to encrypted block files. By itself, insufficient to read cleartext without the EDEK and EDEK encryption key.	No mitigation required.
	Access to encryption zone and encrypted file metadata (including encrypted data encryption keys), using -fetchImage. By itself, insufficient to read cleartext without EDEK encryption keys.	No mitigation required.
Root Access Exploits		
	A rogue user can collect keys to which they have access, and use them later to decrypt encrypted data.	This can be mitigated through periodic key rolling policies.

Configuring the Key Management Server (KMS)

Hadoop KMS is a cryptographic key management server based on Hadoop's **KeyProvider** API. It provides a client which is a KeyProvider implementation that interacts with the KMS using the HTTP REST API. Both the KMS and its client support HTTP SPNEGO Kerberos authentication and SSL-secured communication. The KMS is a Java-based web application which runs using a pre-configured Tomcat server bundled with the Hadoop distribution.

For instructions on securing the KMS, see Securing the Key Management Server (KMS) on page 198.

Setup Configuration

KeyProvider Configuration

Configure the KMS backing KeyProvider properties in the `/etc/hadoop-kms/conf/kms-site.xml` configuration file:

```
<property>
    <name>hadoop.kms.key.provider.uri</name>
    <value>jceks://file@/${user.home}/kms.keystore</value>
</property>

<property>
    <name>hadoop.security.keystore.java-keystore-provider.password-file</name>
    <value>kms.keystore.password</value>
</property>
```

The password file is looked up in Hadoop's configuration directory using `CLASSPATH`.

Restart the KMS for configuration changes to take effect.

KMS Cache

KMS caches keys for short periods of time to avoid excessive hits to the underlying key provider. The cache is enabled by default and can be disabled by setting the `hadoop.kms.cache.enable` property to `false`.

The cache is used with the following methods only: `getCurrentKey()`, `getKeyVersion()` and `getMetadata()`.

For the `getCurrentKey()` method, cached entries are kept for a maximum of `30000` milliseconds regardless of the number of times the key is accessed. This is to prevent stale keys from being considered current.

For the `getKeyVersion()` method, cached entries are kept with a default inactivity timeout of `600000` milliseconds (10 minutes). The cache and its timeout value is configurable using the following properties in the `/etc/hadoop-kms/conf/kms-site.xml` configuration file:

```
<property>
    <name>hadoop.kms.cache.enable</name>
    <value>true</value>
</property>

<property>
    <name>hadoop.kms.cache.timeout.ms</name>
    <value>600000</value>
</property>

<property>
    <name>hadoop.kms.current.key.cache.timeout.ms</name>
    <value>30000</value>
</property>
```

KMS Client Configuration

The KMS client KeyProvider uses the **kms** scheme, and the embedded URL must be the URL of the KMS.

For example, for a KMS running on `http://localhost:16000/kms`, the KeyProvider URI is `kms://http@localhost:16000/kms`. And for a KMS running on `https://localhost:16000/kms`, the KeyProvider URI is `kms://https@localhost:16000/kms`.

Starting/Stopping the KMS

To start or stop KMS use KMS's `bin/kms.sh` script. For example, to start the KMS:

```
hadoop-3.0.0 $ sbin/kms.sh start
```

Invoking the script without any parameters will list all possible parameters.

KMS Aggregated Audit logs

Audit logs are aggregated for API accesses to the `GET_KEY_VERSION, GET_CURRENT_KEY, DECRYPT_EEK`, and `GENERATE_EEK` operations.

Entries are grouped by the <user,key,operation> for a configurable aggregation interval after which the number of accesses to the specified end-point by the user for a given key is flushed to the audit log.

The aggregation interval is configured using the following property:

```
<property>
    <name>hadoop.kms.aggregation.delay.ms</name>
    <value>10000</value>
</property>
```

Configuring the Embedded Tomcat Server

The embedded Tomcat server can be configured using the `/etc/hadoop-kms/tomcat-conf.http/conf/server.xml.conf` file. KMS pre-configures the HTTP and Admin ports in Tomcat's `server.xml.conf` to `16000` and `16001`. Tomcat logs are also preconfigured to go to Hadoop's `logs/` directory.

The following environment variables can be set in KMS's `/etc/hadoop-kms/conf/kms-env.sh` script and can be used to alter the default ports and log directory:

- `KMS_HTTP_PORT`
- `KMS_ADMIN_PORT`
- `KMS_LOG`

Restart the KMS for the configuration changes to take effect.

Configuring KMS High Availability/Multiple KMSs

KMS supports multiple KMS instances behind a load balancer or VIP for scalability and HA purposes. These instances must be specially configured to work properly as a single logical service. When using multiple KMS instances, requests from the same user may be handled by different KMS instances.

HTTP Kerberos Principals Configuration

When KMS instances are behind a load balancer or VIP, clients will use the hostname of the VIP. For Kerberos SPNEGO authentication, the VIP hostname is used to construct the Kerberos principal for the server, `HTTP/<FQDN-VIP>`. This means for client communication, all KMS instances must have the load balancer or VIP's principal.

However, in order to allow clients to directly access a specific KMS instance, the KMS instance must also have a Kerberos principal with its own hostname.

Both the Kerberos service principals (for the load balancer/VIP and the actual KMS host) must be in the keytab file. The principal name specified in the `kms-site.xml` configuration file must be '*' as follows:

```
<property>
    <name>hadoop.kms.authentication.kerberos.principal</name>
    <value>*</value>
</property>
```

If using HTTPS, the SSL certificate used by the KMS instance must be configured to support multiple hostnames (see Java 7 keytool SAN extension support for details on how to do this).

HTTP Authentication Signature

KMS uses Hadoop Authentication for HTTP authentication. Hadoop Authentication issues a signed HTTP Cookie once a client has been authenticated successfully. This HTTP Cookie has an expiration time, after which it triggers a new authentication sequence. This is done to avoid requiring authentication for every HTTP request of a client.

A KMS instance must verify the HTTP Cookie signatures signed by other KMS instances. To do this all KMS instances must share the signing secret which can be configured by the `hadoop.kms.authentication.signer.secret.provider` property.

This secret can be shared using a Zookeeper service which must be configured in the `kms-site.xml`:

```
<property>
    <name>hadoop.kms.authentication.signer.secret.provider</name>
    <value>zookeeper</value>
    <description>
      Indicates how the secret to sign the authentication cookies will be
      stored. Options are 'random' (default), 'string' and 'zookeeper'.
      If using a setup with multiple KMS instances, 'zookeeper' should be used.
    </description>
</property>

<property>
    <name>hadoop.kms.authentication.signer.secret.provider.zookeeper.path</name>
    <value>/hadoop-kms/hadoop-auth-signature-secret</value>
    <description>
      The Zookeeper ZNode path where the KMS instances will store and retrieve
      the secret from.
    </description>
</property>

<property>
```

```
<name>hadoop.kms.authentication.signer.secret.provider.zookeeper.connection.string</name>

    <value>#HOSTNAME#:#PORT#,...</value>
    <description>
      The Zookeeper connection string, a list of hostnames and port comma
      separated.
    </description>
</property>

<property>
    <name>hadoop.kms.authentication.signer.secret.provider.zookeeper.auth.type</name>
    <value>kerberos</value>
    <description>
      The Zookeeper authentication type, 'none' or 'sasl' (Kerberos).
    </description>
</property>

<property>

<name>hadoop.kms.authentication.signer.secret.provider.zookeeper.kerberos.keytab</name>

    <value>/etc/hadoop/conf/kms.keytab</value>
    <description>
      The absolute path for the Kerberos keytab with the credentials to
      connect to Zookeeper.
    </description>
</property>

<property>

<name>hadoop.kms.authentication.signer.secret.provider.zookeeper.kerberos.principal</name>

    <value>kms/#HOSTNAME#</value>
    <description>
      The Kerberos service principal used to connect to Zookeeper.
    </description>
</property>
```

Securing the Key Management Server (KMS)

This topic contains information on securing the Hadoop KMS using Kerberos, SSL-communication and Access Control Lists for operations on encryption keys.

Enabling Kerberos Authentication

Important:

- You can use either Cloudera Manager or the following command-line instructions to complete this configuration.
- This information applies specifically to CDH 5.3.x . If you use an earlier version of CDH, see the documentation for that version located at Cloudera Documentation.

Using Cloudera Manager

Minimum Required Role: Full Administrator

To enable Kerberos for the KMS using Cloudera Manager:

1. Open the Cloudera Manager Admin Console and navigate to the KMS service.
2. Click **Configuration**.
3. Set the **Authentication Type** property to `kerberos`.
4. Click **Save Changes**.
5. Since Cloudera Manager does not automatically create the principal and keytab file for the KMS, you will need to manually run the **Generate Credentials** command. Using the top navigation bar, go to **Administration** > **Kerberos** > **Credentials** and click **Generate Credentials**.

> **Note:** No new principals will be created since KMS uses its host's HTTP principal.

Kerberos Mapping Rules:

The `hadoop.kms.authentication.kerberos.name.rules` property can only be changed using the **Key Management Server Advanced Configuration Snippet (Safety Valve) for kms-site.xml** property.

To modify, the value should be copied from the `hadoop.security.auth_to_local` configuration property, found in the `core-site.xml` file of the service.

Using the Command Line

Configure the `etc/krb5.conf` file with information for your KDC server. Create an HTTP principal and keytab file for the KMS.

Configure the `etc/hadoop/kms-site.xml` with the following properties:

```
<property>
    <name>hadoop.kms.authentication.type</name>
    <value>kerberos</value>
</property>

<property>
    <name>hadoop.kms.authentication.kerberos.keytab</name>
    <value>${user.home}/kms.keytab</value>
</property>

<property>
    <name>hadoop.kms.authentication.kerberos.principal</name>
    <value>HTTP/localhost</value>
</property>

<property>
    <name>hadoop.kms.authentication.kerberos.name.rules</name>
    <value>DEFAULT</value>
</property>
```

Restart the KMS for the configuration changes to take effect.

Configuring the KMS Proxyuser

Each proxyuser must be configured in `etc/hadoop/kms-site.xml` using the following properties:

```
<property>
    <name>hadoop.kms.proxyuser.#USER#.users</name>
    <value>*</value>
</property>

<property>
    <name>hadoop.kms.proxyuser.#USER#.groups</name>
    <value>*</value>
</property>

<property>
    <name>hadoop.kms.proxyuser.#USER#.hosts</name>
    <value>*</value>
</property>
```

Where #USER# is the username of the proxyuser to be configured.

The `hadoop.kms.proxyuser.#USER#.users` property indicates the users that can be impersonated. The `hadoop.kms.proxyuser.#USER#.groups` property indicates the groups to which the users being impersonated must belong. At least one of these properties must be defined. If both are defined, then the configured proxyuser will be able to impersonate any user in the `users` list and any user belonging to a group listed in the `groups` list.

The `hadoop.kms.proxyuser.#USER#.hosts` property indicates the host from which the proxyuser can make impersonation requests. "*" means there are no restrictions for the #USER# regarding users, groups or hosts.

Configuring SSL for the KMS

> **Important:**
> - You can use either Cloudera Manager or the following command-line instructions to complete this configuration.
> - This information applies specifically to CDH 5.3.x . If you use an earlier version of CDH, see the documentation for that version located at Cloudera Documentation.

Using Cloudera Manager

Minimum Required Role: Configurator (also provided by **Cluster Administrator, Full Administrator**)

The steps for configuring and enabling Hadoop SSL for the KMS are as follows:

1. Open the Cloudera Manager Admin Console and navigate to the **KMS service**.
2. Click **Configuration**.
3. In the Search field, type **SSL** to show the KMS SSL properties (found under the **Key Management Server Default Group** > **Security** category).
4. Edit the following SSL properties according to your cluster configuration.

   Table 10: KMS SSL Properties

Property	Description
Key Management Server SSL Enabled	Enable SSL for requests to Key Management Server.
Key Management Server SSL Server JKS Keystore File Password	Location of the keystore file on the local file system.
Key Management Server SSL Server JKS Keystore File Location	Password for the keystore.

5. Click **Save Changes**.
6. Restart the KMS service.

Using the Command Line

To configure KMS to work over HTTPS by setting the following properties in the `etc/hadoop/kms_env.sh` script:

- `KMS_SSL_KEYSTORE_FILE`
- `KMS_SSL_KEYSTORE_PASS`

In the KMS `tomcat/conf` directory, replace the `server.xml` file with the provided `ssl-server.xml` file.

You need to create an SSL certificate for the KMS. As the `kms` 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 `kms` user's home directory. The password you enter for the keystore must match the value of the `KMS_SSL_KEYSTORE_PASS` environment variable set in the `kms-env.sh` script in the configuration directory.

The answer to "What is your first and last name?" (CN) must be the hostname of the machine where the KMS will be running.

> **Note:** Restart the KMS for the configuration changes to take effect.

Configuring Access Control Lists for the KMS

By default, the KMS ACLs are fully permissive, that is, everyone is allowed to perform all operations. However, a key design requirement is to be able to restrict HDFS superusers from having access to key material. This prevents a malicious superuser from having access to both (a) all the key material and (b) all the encrypted data, and thus being able to decrypt everything.

The KMS supports both whitelist and blacklist ACLs where the blacklist overrides the whitelist. A user accessing KMS is first checked for inclusion in the ACL for the requested operation and then checked for exclusion in the blacklist for the operation before access is granted. Hence, add HDFS superusers to the blacklist, while allowing everyone else.

> **Important:**
> - You can use either Cloudera Manager or the following command-line instructions to complete this configuration.
> - This information applies specifically to CDH 5.3.x . If you use an earlier version of CDH, see the documentation for that version located at Cloudera Documentation.

Using Cloudera Manager

Minimum Required Role: Configurator (also provided by **Cluster Administrator, Full Administrator**)

> **Important:** See related Known Issue and listed workaround: KMS and Key Trustee ACLs do not work in Cloudera Manager 5.3.

1. Open the Cloudera Manager Admin Console and navigate to the **KMS service**.
2. Click **Configuration**.
3. In the Search field, type **acl** to show the KMS ACL properties (found under the **Key Management Server Default Group** category).
4. Edit the ACL properties according to your cluster configuration.
5. Click **Save Changes**.
6. Restart the KMS service.

Using the Command Line

KMS ACLs configuration are defined in the KMS `etc/hadoop/kms-acls.xml` configuration file. This file is hot-reloaded when it changes.

```
<property>
    <name>hadoop.kms.acl.CREATE</name>
    <value>*</value>
    <description>
      ACL for create-key operations.
      If the user does is not in the GET ACL, the key material is not returned
      as part of the response.
    </description>
</property>

<property>
    <name>hadoop.kms.blacklist.CREATE</name>
    <value>hdfs,foo</value>
    <description>
      Blacklist for create-key operations.
```

```
      If the user does is in the Blacklist, the key material is not returned
      as part of the response.
    </description>
</property>

<property>
    <name>hadoop.kms.acl.DELETE</name>
    <value>*</value>
    <description>
      ACL for delete-key operations.
    </description>
</property>

<property>
    <name>hadoop.kms.blacklist.DELETE</name>
    <value>hdfs,foo</value>
    <description>
      Blacklist for delete-key operations.
    </description>
</property>

<property>
    <name>hadoop.kms.acl.ROLLOVER</name>
    <value>*</value>
    <description>
      ACL for rollover-key operations.
      If the user does is not in the GET ACL, the key material is not returned
      as part of the response.
    </description>
</property>

<property>
    <name>hadoop.kms.blacklist.ROLLOVER</name>
    <value>hdfs,foo</value>
    <description>
      Blacklist for rollover-key operations.
    </description>
</property>

<property>
    <name>hadoop.kms.acl.GET</name>
    <value>*</value>
    <description>
      ACL for get-key-version and get-current-key operations.
    </description>
</property>

<property>
    <name>hadoop.kms.blacklist.GET</name>
    <value>hdfs,foo</value>
    <description>
      ACL for get-key-version and get-current-key operations.
    </description>
</property>

<property>
    <name>hadoop.kms.acl.GET_KEYS</name>
    <value>*</value>
    <description>
      ACL for get-keys operation.
    </description>
</property>

<property>
    <name>hadoop.kms.blacklist.GET_KEYS</name>
    <value>hdfs,foo</value>
    <description>
      Blacklist for get-keys operation.
    </description>
</property>

<property>
    <name>hadoop.kms.acl.GET_METADATA</name>
```

```
    <value>*</value>
    <description>
      ACL for get-key-metadata and get-keys-metadata operations.
    </description>
</property>

<property>
    <name>hadoop.kms.blacklist.GET_METADATA</name>
    <value>hdfs,foo</value>
    <description>
      Blacklist for get-key-metadata and get-keys-metadata operations.
    </description>
</property>

 <property>
    <name>hadoop.kms.acl.SET_KEY_MATERIAL</name>
    <value>*</value>
    <description>
        Complimentary ACL for CREATE and ROLLOVER operation to allow the client
        to provide the key material when creating or rolling a key.
    </description>
</property>

<property>
    <name>hadoop.kms.blacklist.SET_KEY_MATERIAL</name>
    <value>hdfs,foo</value>
    <description>
        Complimentary Blacklist for CREATE and ROLLOVER operation to allow the client
        to provide the key material when creating or rolling a key.
    </description>
</property>

<property>
    <name>hadoop.kms.acl.GENERATE_EEK</name>
    <value>*</value>
    <description>
      ACL for generateEncryptedKey
      CryptoExtension operations
    </description>
</property>

<property>
    <name>hadoop.kms.blacklist.GENERATE_EEK</name>
    <value>hdfs,foo</value>
    <description>
      Blacklist for generateEncryptedKey
      CryptoExtension operations
    </description>
</property>

<property>
    <name>hadoop.kms.acl.DECRYPT_EEK</name>
    <value>*</value>
    <description>
      ACL for decrypt EncryptedKey
      CryptoExtension operations
    </description>
</property>

<property>
    <name>hadoop.kms.blacklist.DECRYPT_EEK</name>
    <value>hdfs,foo</value>
    <description>
      Blacklist for decrypt EncryptedKey
      CryptoExtension operations
    </description>
  </property>
```

Configuring Key Access Control

KMS supports access control for all non-read operations at the Key level. All Key Access operations are classified as:

- `MANAGEMENT - createKey, deleteKey, rolloverNewVersion`

- `GENERATE_EEK - generateEncryptedKey, warmUpEncryptedKeys`
- `DECRYPT_EEK - decryptEncryptedKey`
- `READ - getKeyVersion, getKeyVersions, getMetadata, getKeysMetadata, getCurrentKey`
- `ALL` - all of the above.

These can be defined in the `etc/hadoop/kms-acls.xml` as follows:

```
<property>
  <name>key.acl.testKey1.MANAGEMENT</name>
  <value>*</value>
  <description>
    ACL for create-key, deleteKey and rolloverNewVersion operations.
  </description>
</property>

<property>
  <name>key.acl.testKey2.GENERATE_EEK</name>
  <value>*</value>
  <description>
    ACL for generateEncryptedKey operations.
  </description>
</property>

<property>
  <name>key.acl.testKey3.DECRYPT_EEK</name>
  <value>*</value>
  <description>
    ACL for decryptEncryptedKey operations.
  </description>
</property>

<property>
  <name>key.acl.testKey4.READ</name>
  <value>*</value>
  <description>
    ACL for getKeyVersion, getKeyVersions, getMetadata, getKeysMetadata,
    getCurrentKey operations
  </description>
</property>

<property>
  <name>key.acl.testKey5.ALL</name>
  <value>*</value>
  <description>
    ACL for ALL operations.
  </description>
</property>

<property>
  <name>default.key.acl.MANAGEMENT</name>
  <value>user1,user2</value>
  <description>
    default ACL for MANAGEMENT operations for all keys that are not
    explicitly defined.
  </description>
</property>

<property>
  <name>default.key.acl.GENERATE_EEK</name>
  <value>user1,user2</value>
  <description>
    default ACL for GENERATE_EEK operations for all keys that are not
    explicitly defined.
  </description>
</property>

<property>
  <name>default.key.acl.DECRYPT_EEK</name>
  <value>user1,user2</value>
  <description>
    default ACL for DECRYPT_EEK operations for all keys that are not
    explicitly defined.
```

```
    </description>
  </property>

  <property>
    <name>default.key.acl.READ</name>
    <value>user1,user2</value>
    <description>
      default ACL for READ operations for all keys that are not
      explicitly defined.
    </description>
  </property>
```

For all keys for which an ACL has not been explicitly configured, you can configure a default key ACL for a subset of the operation types.

If no ACL is configured for a specific key, *and* no default ACL is configured for the requested operation, access will be denied.

> **Note:** The default ACL does not support the `ALL` operation qualifier.

KMS Delegation Token Configuration

KMS delegation token secret manager can be configured using the following properties:

```
  <property>
    <name>hadoop.kms.authentication.delegation-token.update-interval.sec</name>
    <value>86400</value>
    <description>
      How often the master key is rotated, in seconds. Default value 1 day.
    </description>
  </property>

  <property>
    <name>hadoop.kms.authentication.delegation-token.max-lifetime.sec</name>
    <value>604800</value>
    <description>
      Maximum lifetime of a delegation token, in seconds. Default value 7 days.
    </description>
  </property>

  <property>
    <name>hadoop.kms.authentication.delegation-token.renew-interval.sec</name>
    <value>86400</value>
    <description>
      Renewal interval of a delegation token, in seconds. Default value 1 day.
    </description>
  </property>

  <property>
    <name>hadoop.kms.authentication.delegation-token.removal-scan-interval.sec</name>
    <value>3600</value>
    <description>
      Scan interval to remove expired delegation tokens.
    </description>
  </property>
```

Configuring CDH Services for HDFS Encryption

The following topics contain recommendations for setting up HDFS encryption with various CDH services.

Hive

HDFS encryption has been designed such that files cannot be moved from one encryption zone to another encryption zone or from encryption zones to unencrypted directories. Hence, the landing zone for data when using the `LOAD DATA INPATH` command should always be inside the destination encryption zone.

If you want to use HDFS encryption with Hive, ensure you are using *one* of the following configurations:

Single Encryption Zone

With this configuration, you can use HDFS encryption by having all Hive data inside the same encryption zone. Additionally, in Cloudera Manager, configure the Hive Scratch Directory (`hive.exec.scratchdir`) to be inside the encryption zone.

Recommended HDFS Path: `/user/hive`

For example, to configure a single encryption zone for the entire Hive warehouse, you can rename `/user/hive` to `/user/hive-old`, create an encryption zone at `/user/hive`, and then `distcp` all the data from `/user/hive-old` to `/user/hive`.

Additionally, in Cloudera Manager, configure the Hive Scratch Directory (`hive.exec.scratchdir`) to be inside the encryption zone by setting it to `/user/hive/tmp`, ensuring the permissions are `1777` on `/user/hive/tmp`.

Multiple Encryption Zones

With this configuration, you can use encrypted databases or tables with different encryption keys. The only limitation is that in order to read data from read-only encrypted tables, users must have access to a temporary directory which is encrypted with at least as strong encryption as the table.

For example, you can configure two encrypted tables, `ezTbl1` and `ezTbl2`. Create two new encryption zones, `/data/ezTbl1` and `/data/ezTbl2`. Load data to the tables in Hive as usual using `LOAD` statements. See the **Changed Behavior after HDFS Encryption is Enabled** section below for more information.

Other Encrypted Directories

- **`LOCALSCRATCHDIR`**: The MapJoin optimization in Hive writes HDFS tables out to a local directory and then uploads them to the distributed cache. If you want to enable encryption, you will either need to disable MapJoin or encrypt the *local* Hive Scratch directory (`hive.exec.local.scratchdir`).
- **`DOWNLOADED_RESOURCES_DIR`**: Jars which 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 is local to the HiveServer2.
- **NodeManager Local Directory List:** Since Hive stores Jars and MapJoin files in the distributed cache, 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, you can disable MapJoin by setting `hive.auto.convert.join` to `false`.

Changed Behavior after HDFS Encryption is Enabled

- Loading data from one encryption zone to another will result in a copy of the data. Distcp will be used to speed up the process if the size of the files being copied is higher than the value specified by `HIVE_EXEC_COPYFILE_MAXSIZE`. The minimum size limit for `HIVE_EXEC_COPYFILE_MAXSIZE` is 32 MB, which can be modified by changing the value for the `hive.exec.copyfile.maxsize` configuration property.
- When loading data to encrypted tables, Cloudera strongly recommends using a landing zone inside the same encryption zone as the table.
  - **Example 1: Loading unencrypted data to an encrypted table** - There are 2 approaches to doing this.
    - If you're loading new unencrypted data to an encrypted table, just load the data using the `LOAD DATA ...` statement. Since the source data does not reside inside the encryption zone, the `LOAD` statement will result in a copy. This is why Cloudera recommends landing data (that you expect to encrypt) inside the destination encryption zone. However, this approach may use `distcp` to speed up the copying process if your data is inside HDFS.

– If the data to be loaded is already inside a Hive table, you can 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 that the location specified in the `CREATE TABLE` statement above needs to be inside an encryption zone. Creating a table pointing `LOCATION` to an unencrypted directory will not encrypt your source data. You must copy your data to an encryption zone, and then point `LOCATION` to that encryption zone.

– **Example 2: Loading encrypted data to an encrypted table** - If the data to be loaded is already encrypted, use the `CREATE TABLE` statement pointing `LOCATION` to the encrypted source directory where your data is. This is the fastest way to create encrypted tables.

```
CREATE TABLE encrypted_table [STORED AS] LOCATION ... AS SELECT * FROM
<encrypted_source_directory>
```

- Users reading data from encrypted tables which are read-only, must have access to a temp directory which is encrypted with at least as strong encryption as the table.
- Temp data is now written to a directory named `.hive-staging` within each table or partition
- 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.

Impala

Recommendations

- If HDFS encryption is enabled, configure Impala to encrypt data spilled to local disk.
- Impala does not support the `LOAD DATA` statement when the source and destination are in different encryption zones. If you need to use `LOAD DATA`, copy the data to the table's encryption zone prior to running the statement.
- Use Cloudera Navigator to lock down the local directory where Impala UDFs are copied during execution. By default, Impala copies UDFs into `/tmp`, and you can configure this location through the `--local_library_dir` startup flag for the `impalad` daemon.
- Limit the rename operations for internal tables once encryption zones are set up. Impala cannot do an `ALTER TABLE RENAME` operation to move an internal table from one database to another, if the root directories for those databases are in different encryption zones. If the encryption zone covers a table directory but not the parent directory associated with the database, Impala cannot do an `ALTER TABLE RENAME` operation to rename an internal table even within the same database.
- Avoid structuring partitioned tables where different partitions reside in different encryption zones, or where any partitions reside in an encryption zone that is different from the root directory for the table. Impala cannot do an `INSERT` operation into any partition that is not in the same encryption zone as the root directory of the overall table.

Steps

Start every `impalad` process with the `--disk_spill_encryption=true` flag set. This encrypts all spilled data using AES-256-CFB. Set this flag using the Impala service configuration property, **Impala Daemon Command Line Argument Advanced Configuration Snippet (Safety Valve)**, found under **Impala Daemon Default Group** > **Advanced**.

> **Important:** Impala does not selectively encrypt data based on whether the source data is already encrypted in HDFS. This will result in at most 15 percent performance degradation when data is spilled.

HBase

Recommendations

Make `/hbase` an encryption zone. Do not create encryption zones as subdirectories under `/hbase`, as HBase may need to rename files across those subdirectories.

Steps

On a cluster without HBase currently installed, create the `/hbase` directory and make that an encryption zone.

On a cluster with HBase already installed, perform the following steps:

1. Stop the HBase service.
2. Move data from the `/hbase` directory to `/hbase-tmp`.
3. Create an empty `/hbase` directory and make it an encryption zone.
4. Distcp all data from `/hbase-tmp` to `/hbase` preserving user-group permissions and extended attributes.
5. Start the HBase service and verify that it is working as expected.
6. Remove the `/hbase-tmp` directory.

Search

Recommendations

Make `/solr` an encryption zone.

Steps

On a cluster without Solr currently installed, create the `/solr` directory and make that an encryption zone. On a cluster with Solr already installed, create an empty `/solr-tmp` directory, make `/solr-tmp` an encryption zone, `distcp` all data from `/solr` into `/solr-tmp`, remove `/solr` and rename `/solr-tmp` to `/solr`.

Sqoop

Recommendations

- **For Hive support:** Ensure that you are using Sqoop with the `--target-dir` parameter set to a directory that is inside the Hive encryption zone. For more details, see Hive on page 205
- **For append/incremental support:** Make sure that the `sqoop.test.import.rootDir` property points to the same encryption zone as the above `--target-dir` argument.
- **For HCatalog support:** No special configuration should be required

Hue

Recommendations

Make `/user/hue` an encryption zone since that's where Oozie workflows and other Hue specific data are stored by default.

Steps

On a cluster without Hue currently installed, create the `/user/hue` directory and make that an encryption zone. On a cluster with Hue already installed, create an empty `/user/hue-tmp` directory, make `/user/hue-tmp` an encryption zone, `distcp` all data from `/user/hue` into `/user/hue-tmp`, remove `/user/hue` and rename `/user/hue-tmp` to `/user/hue`.

Spark

Recommendations

- By default, application event logs are stored at `/user/spark/applicationHistory` which can be made into an encryption zone.
- Spark also optionally caches its jar file at `/user/spark/share/lib` (by default), but encrypting this directory is not necessary.
- Spark does not encrypt shuffle data. However, if that is desired, you should configure Spark's local directory, `spark.local.dir` (in Standalone mode), to reside on an encrypted disk. For YARN mode, make the corresponding YARN configuration changes.

MapReduce and YARN
MapReduce v1

Recommendations

MRv1 stores both history and logs on local disks by default. Even if you do configure history to be stored on HDFS, the files are not renamed. Hence, no special configuration is required.

MapReduce v2 (YARN)

Recommendations

Make `/user/history` a single encryption zone, since history files are moved between the `intermediate` and done directories, and HDFS encryption does not allow moving encrypted files across encryption zones.

Steps

On a cluster with MRv2 (YARN) installed, create the `/user/history` directory and make that an encryption zone. If `/user/history` already exists and is not empty, create an empty `/user/history-tmp` directory, make `/user/history-tmp` an encryption zone, `distcp` all data from `/user/history` into `/user/history-tmp`, remove `/user/history` and rename `/user/history-tmp` to `/user/history`.

HDFS Encryption Troubleshooting

This topic contains HDFS Encryption-specific troubleshooting information in the form of issues you might face when encrypting HDFS files/directories and their workarounds.

Retrieval of encryption keys fails

Description

You see the following error when trying to list encryption keys

```
user1@example-sles-4:~> hadoop key list
Cannot list keys for KeyProvider: KMSClientProvider[https:
//example-sles-2.example.com:16000/kms/v1/]: Retrieval of all keys failed.
```

Solution

Make sure your truststore has been updated with the relevant certificate(s), such as the Key Trustee server certificate.

DistCp between unencrypted and encrypted locations fails

Description

By default, DistCp compares checksums provided by the filesystem to verify that data was successfully copied to the destination. However, when copying between unencrypted and encrypted locations, the filesystem checksums will not match since the underlying block data is different.

Solution

Specify the `-skipcrccheck` and `-update` distcp flags to avoid verifying checksums.

Cannot move encrypted files to trash

Description

With HDFS encryption enabled, you cannot move encrypted files or directories to the trash directory.

Solution

To remove encrypted files/directories, use the following command with the `-skipTrash` flag specified to bypass trash.

```
rm -r -skipTrash /testdir
```

NameNode - KMS communication fails after long periods of inactivity

Description

Encrypted files and encryption zones cannot be created if a long period of time (by default, 20 hours) has passed since the last time the KMS and NameNode communicated.

Solution

> **Important:** Upgrading your cluster to the latest CDH 5 release will fix this problem. For instructions, see Upgrading from an Earlier CDH 5 Release to the Latest Release.

For lower CDH 5 releases, there are two possible workarounds to this issue :

- You can increase the KMS authentication token validity period to a very high number. Since the default value is 10 hours, this bug will only be encountered after 20 hours of no communication between the NameNode and the KMS. Add the following property to the `kms-site.xml`Safety Valve:

  ```
  <property>
  <name>hadoop.kms.authentication.token.validity</name>
  <value>SOME VERY HIGH NUMBER</value>
  </property>
  ```

- You can switch the KMS signature secret provider to the string secret provider by adding the following property to the `kms-site.xml` Safety Valve:

  ```
  <property>
  <name>hadoop.kms.authentication.signature.secret</name>
  <value>SOME VERY SECRET STRING</value>
  </property>
  ```

Configuring Encrypted HDFS Data Transport

This topic describes how to configure encrypted HDFS data transport using both, Cloudera Manager, and the command line.

Using Cloudera Manager

Minimum Required Role: Full Administrator

To enable encryption of data transfered between DataNodes and clients, and among DataNodes, proceed as follows:

1. Enable Hadoop security using Kerberos.
2. Select the HDFS service.
3. Click the **Configuration** tab.

4. Expand the **Service-Wide** category and click the **Security** subcategory. Configure the following properties:

Property	Description
Enable Data Transfer Encryption	Check this field to enable wire encryption.
Data Transfer Encryption Algorithm	Optionally configure the algorithm used to encrypt data.
Hadoop RPC Protection	Select **privacy**.

5. Click **Save Changes**.
6. Restart the HDFS service.

Using the Command Line

Important:

- You can use either Cloudera Manager or the following command-line instructions to complete this configuration.
- This information applies specifically to CDH 5.3.x . If you use an earlier version of CDH, see the documentation for that version located at Cloudera Documentation.

To enable encrypted data transport using the command line, proceed as follows:

1. Enable the Hadoop Security using Kerberos, following these instructions.
2. Set the optional RPC encryption by setting `hadoop.rpc.protection` to "`privacy`" in the `core-site.xml` file in both client and server configurations.

   Note:

   If RPC encryption is not enabled, transmission of other HDFS data is also insecure.

3. Set `dfs.encrypt.data.transfer` to `true` in the `hdfs-site.xml` file on all server systems.
4. Restart all daemons.

Authorization

Authorization is concerned with who or what has access or control over a given resource or service. Since Hadoop merges together the capabilities of multiple varied, and previously separate IT systems as an enterprise data hub that stores and works on all data within an organization, it requires multiple authorization controls with varying granularities. In such cases, Hadoop management tools simplify setup and maintenance by:

- Tying all users to groups, which can be specified in existing LDAP or AD directories.
- Providing role-based access control for similar interaction methods, like batch and interactive SQL queries. For example, Apache Sentry permissions apply to Hive (HiveServer2) and Impala.

CDH currently provides the following forms of access control:

- Traditional POSIX-style permissions for directories and files, where each directory and file is assigned a single owner and group. Each assignment has a basic set of permissions available; file permissions are simply read, write, and execute, and directories have an additional permission to determine access to child directories.
- Extended Access Control Lists (ACLs) for HDFS that provide fine-grained control of permissions for HDFS files by allowing you to set different permissions for specific named users or named groups.
- Apache HBase uses ACLs to authorize various operations (`READ`, `WRITE`, `CREATE`, `ADMIN`) by column, column family, and column family qualifier. HBase ACLs are granted and revoked to both users and groups.
- Role-based access control with Apache Sentry.As of Cloudera Manager 5.1.x, Sentry permissions can be configured using either policy files or the database-backed Sentry service.
  - The Sentry service is the preferred way to set up Sentry permissions. See The Sentry Service on page 217 for more information.
  - For the policy file approach to configuring Sentry, see Sentry Policy File Authorization on page 240.

Cloudera Manager User Roles

Minimum Required Role: User Administrator (also provided by **Full Administrator**)

Access to Cloudera Manager features is controlled by user accounts. For more information about user accounts, see Cloudera Manager User Accounts. Among the properties of a user account is the *user role*, which determines the Cloudera Manager features visible to the user and the actions the user can perform. All the tasks in the Cloudera Manager documentation indicate which role is required to perform the task.

> **Note:** The full set of roles are available with Cloudera Enterprise; Cloudera Express supports only the Read-Only and Full Administrator roles. When a Cloudera Enterprise Data Hub Edition trial license expires, only users with Read-Only and Full Administrator roles are allowed to log in. A Full Administrator must change the role of any other user to Read-Only or Full Administrator before that user can log in.

User Roles

A Cloudera Manager user account can be assigned one of the following roles with associated permissions:

- **Auditor**
  - View data in Cloudera Manager.
  - View audit events.
- **Read-Only** - Allows the user to:
  - View data in Cloudera Manager.
  - View service and monitoring information.

The Read-Only role does not allow the user to add services or take any actions that affect the state of the cluster.

- **Limited Operator**
  - View data in Cloudera Manager.
  - View service and monitoring information.
  - Decommission hosts (except hosts running Cloudera Management Service roles).

  The Limited Operator role does not allow the user to add services or take any other actions that affect the state of the cluster.

- **Operator**
  - View data in Cloudera Manager.
  - View service and monitoring information.
  - Stop, start, and restart clusters, services (except the Cloudera Management Service), and roles.
  - Decommission and recommission hosts (except hosts running Cloudera Management Service roles).
  - Decommission and recommission roles (except Cloudera Management Service roles).
  - Start, stop, and restart KMS.

  The Operator role does not allow the user to add services, roles, or hosts, or take any other actions that affect the state of the cluster.

- **Configurator**
  - View data in Cloudera Manager.
  - Perform all Operator operations.
  - Configure services (except the Cloudera Management Service).
  - Enter and exit maintenance mode.
  - Manage dashboards (including Cloudera Management Service dashboards).

- **Cluster Administrator** - View all data and perform all actions *except* the following:
  - Administer Cloudera Navigator.
  - View replication schedules and snapshot policies.
  - View audit events.
  - Manage user accounts and configuration of external authentication.

- **BDR Administrator**
  - View data in Cloudera Manager.
  - View service and monitoring information.
  - Perform replication and define snapshot operations.

- **Navigator Administrator**
  - View data in Cloudera Manager.
  - View service and monitoring information.
  - Administer Cloudera Navigator.
  - View audit events.

- **User Administrator** - Allows the user to:
  - View data in Cloudera Manager.
  - View service and monitoring information.
  - Manage user accounts and configuration of external authentication.

- **Key Administrator**
  - View data in Cloudera Manager.
  - Configure HDFS encryption, administer Key Trustee Server, and manage encryption keys.

  - Start, stop, and restart KMS

- **Full Administrator** - Full Administrators have permissions to view all data and do all actions, including reconfiguring and restarting services, and administering other users.

Determining the Role of the Currently Logged in User

1. Click the logged-in username at the far right of the top navigation bar. The role displays right under the username. For example:

Full Administrator

Removing the Full Administrator User Role

In some organizations, security policies may prohibit the use of the Full Administrator role. The Full Administrator role is created during Cloudera Manager installation, but you can remove it as long as you have at least one remaining user account with User Administrator privileges.

To remove the Full Administrator user role, perform the following steps.

1. Add at least one user account with User Administrator privileges, or ensure that at least one such user account already exists.
2. Ensure that there is only a single user account with Full Administrator privileges.
3. While logged in as the single remaining Full Administrator user, select your own user account and either delete it or assign it a new user role.

> **Warning:** After you delete the last Full Administrator account, you will be logged out immediately and will not be able to log in unless you have access to another user account. Also, it will no longer be possible to create or assign Full Administrators.

A consequence of removing the Full Administrator role is that some tasks may require collaboration between two or more users with different user roles. For example:

- If the machine that the Cloudera Navigator roles are running on needs to be replaced, the Cluster Administrator will want to move all the roles running on that machine to a different machine. The Cluster Administrator can move any non-Navigator roles by deleting and re-adding them, but would need a Navigator Administrator to perform the stop, delete, add, and start actions for the Cloudera Navigator roles.
- In order to take HDFS snapshots, snapshots must be enabled on the cluster by a Cluster Administrator, but the snapshots themselves must be taken by a BDR Administrator.

Cloudera Navigator User Roles

User roles determine the Cloudera Navigator features visible to the user and the actions the user can perform.

User Roles

A Cloudera Navigator user account can be assigned one of the following user roles:

- **Lineage Viewer** - Search for entities, view metadata, and view lineage.
- **Auditing Viewer** - View audit events and create audit reports.
- **Policy Viewer** - View metadata policies.
- **Metadata Administrator** - Search for entities, view metadata, view lineage, and edit metadata.
- **Policy Administrator** - View, create, update, and delete metadata policies.
- **User Administrator** - Administer role assignments to groups.

- **Full Administrator** - Full access, including role assignments to groups.

The user roles and associated permissions are summarized as follows:

Table 11: Cloudera Navigator User Roles

User Role	Read Audit	Read Lineage	Read Metadata (Search)	Write Metadata (Edit)	Read Policies	Write Policies	Administer Role Group Mapping
Full Administrator	■	■	■	■	■	■	■
User Administrator							■
Auditing Viewer	■						
Lineage Viewer		■	■				
Metadata Administrator		■	■	■			
Policy Viewer					■		
Policy Administrator			■	■	■	■	

Determining the Roles of the Currently Logged in User

To display the Cloudera Navigator user roles for the currently logged-in user:

1. Click the username in the upper right.
2. Click **My roles**.
3. The **Roles** pop-up window will appear, displaying all roles assigned to the LDAP or Active Directory groups to which the current user belongs.

The selection of menus displayed in the upper right indicates the user's access to Cloudera Navigator features, as determined by the roles associated with the user's LDAP or Active Directory groups. For example, a Full Administrator will see the **Search**, **Audits**, **Policies**, and **Administration** menus, while a user with the Policy Administrator role will only see the **Search** and **Policies** menus.

Enabling HDFS Extended ACLs

As of CDH 5.1, HDFS supports POSIX Access Control Lists (ACLs), as well as the traditional POSIX permissions model already supported. ACLs control access of HDFS files by providing a way to set different permissions for specific named users or named groups.

Enabling ACLs

By default, ACLs are disabled on a cluster. To enable them, set the `dfs.namenode.acls.enabled` property to `true` in the NameNode's `hdfs-site.xml`.

> **Important:** Ensure that all users and groups resolve on the NameNode for ACLs to work as expected.

```
<property>
<name>dfs.namenode.acls.enabled</name>
<value>true</value>
</property>
```

Commands

To set and get file access control lists (ACLs), use the file system shell commands, `setfacl` and `getfacl`.

getfacl

```
hdfs dfs -getfacl [-R] <path>

<!-- COMMAND OPTIONS
<path>: Path to the file or directory for which ACLs should be listed.
-R: Use this option to recursively list ACLs for all files and directories.
-->
```

Examples:

```
<!-- To list all ACLs for the file located at /user/hdfs/file -->
hdfs dfs -getfacl /user/hdfs/file

<!-- To recursively list ACLs for /user/hdfs/file -->
hdfs dfs -getfacl -R /user/hdfs/file
```

setfacl

```
hdfs dfs -setfacl [-R] [-b|-k -m|-x <acl_spec> <path>]|[--set <acl_spec> <path>]

<!-- COMMAND OPTIONS
<path>: Path to the file or directory for which ACLs should be set.
-R: Use this option to recursively list ACLs for all files and directories.
-b: Revoke all permissions except the base ACLs for user, groups and others.
-k: Remove the default ACL.
-m: Add new permissions to the ACL with this option. Does not affect existing permissions.
-x: Remove only the ACL specified.
<acl_spec>: Comma-separated list of ACL permissions.
--set: Use this option to completely replace the existing ACL for the path specified.
       Previous ACL entries will no longer apply.
-->
```

Examples:

```
<!-- To give user ben read & write permission over /user/hdfs/file -->
hdfs dfs -setfacl -m user:ben:rw- /user/hdfs/file

<!-- To remove user alice's ACL entry for /user/hdfs/file -->
hdfs dfs -setfacl -x user:alice /user/hdfs/file

<!-- To give user hadoop read & write access, and group or others read-only access -->
hdfs dfs -setfacl --set user::rw-,user:hadoop:rw-,group::r--,other::r-- /user/hdfs/file
```

More details about using this feature can be found in the HDFS Permissions Guide on the Apache website.

The Sentry Service

> **Important:** This is the documentation for the Sentry service introduced in CDH 5.1. If you want to use Sentry's previous policy file approach to secure your data, see Sentry Policy File Authorization on page 240.

The Sentry service is a RPC server that stores the authorization metadata in an underlying relational database and provides RPC interfaces to retrieve and manipulate privileges. It supports secure access to services using Kerberos. The service serves authorization metadata from the database backed storage; it does not handle actual privilege validation. The Hive and Impala services are clients of this service and will enforce Sentry privileges when configured to use Sentry.

The motivation behind introducing a new Sentry service is to make it easier to handle user privileges than the existing policy file approach. Providing a database instead, allows you to use the more traditional `GRANT/REVOKE` statements to modify privileges.

For more information on installing, upgrading and configuring the Sentry service, see:

Prerequisites

- CDH 5.1.x (or higher) managed by Cloudera Manager 5.1.x (or higher). See the Cloudera Manager Administration Guide and Cloudera Installation Guide for instructions.
- HiveServer2 and the Hive Metastore running with strong authentication. For HiveServer2, strong authentication is either Kerberos or LDAP. For the Hive Metastore, only Kerberos is considered strong authentication (to override, see Securing the Hive Metastore on page 230).
- Impala 1.4.0 (or higher) running with strong authentication. With Impala, either Kerberos or LDAP can be configured to achieve strong authentication.
- Implement Kerberos authentication on your cluster. For instructions, see Enabling Kerberos Authentication Using the Wizard on page 17.

Terminologies

- An **object** is an entity protected by Sentry's authorization rules. The objects supported in the current release are `server`, `database`, `table`, and `URI`.
- A **role** is a collection of rules for accessing a given Hive object.
- A **privilege** is granted to a role to govern access to an object. Supported privileges are:

Table 12: Valid privilege types and the objects they apply to

Privilege	Object
INSERT	SEVER, DB, TABLE
SELECT	DB, TABLE
ALL	SERVER, TABLE, DB, URI

- A user is an entity that is permitted by the authentication subsystem to access the Hive service. This entity can be a Kerberos principal, an LDAP `userid`, or an artifact of some other pluggable authentication system supported by HiveServer2.
- A group connects the authentication system with the authorization system. It is a collection of one or more users who have been granted one or more authorization roles. Sentry allows a set of roles to be configured for a group.
- A configured group provider determines a user's affiliation with a group. The current release supports HDFS-backed groups and locally configured groups.

Privilege Model

Sentry uses a role-based privilege model with the following characteristics.

- Allows any user to execute `show function`, `desc function`, and `show locks`.
- Allows the user to see only those tables and databases for which this user has privileges.
- Requires a user to have the necessary privileges on the URI to execute HiveQL operations that take in a location. Examples of such operations include `LOAD`, `IMPORT`, and `EXPORT`.

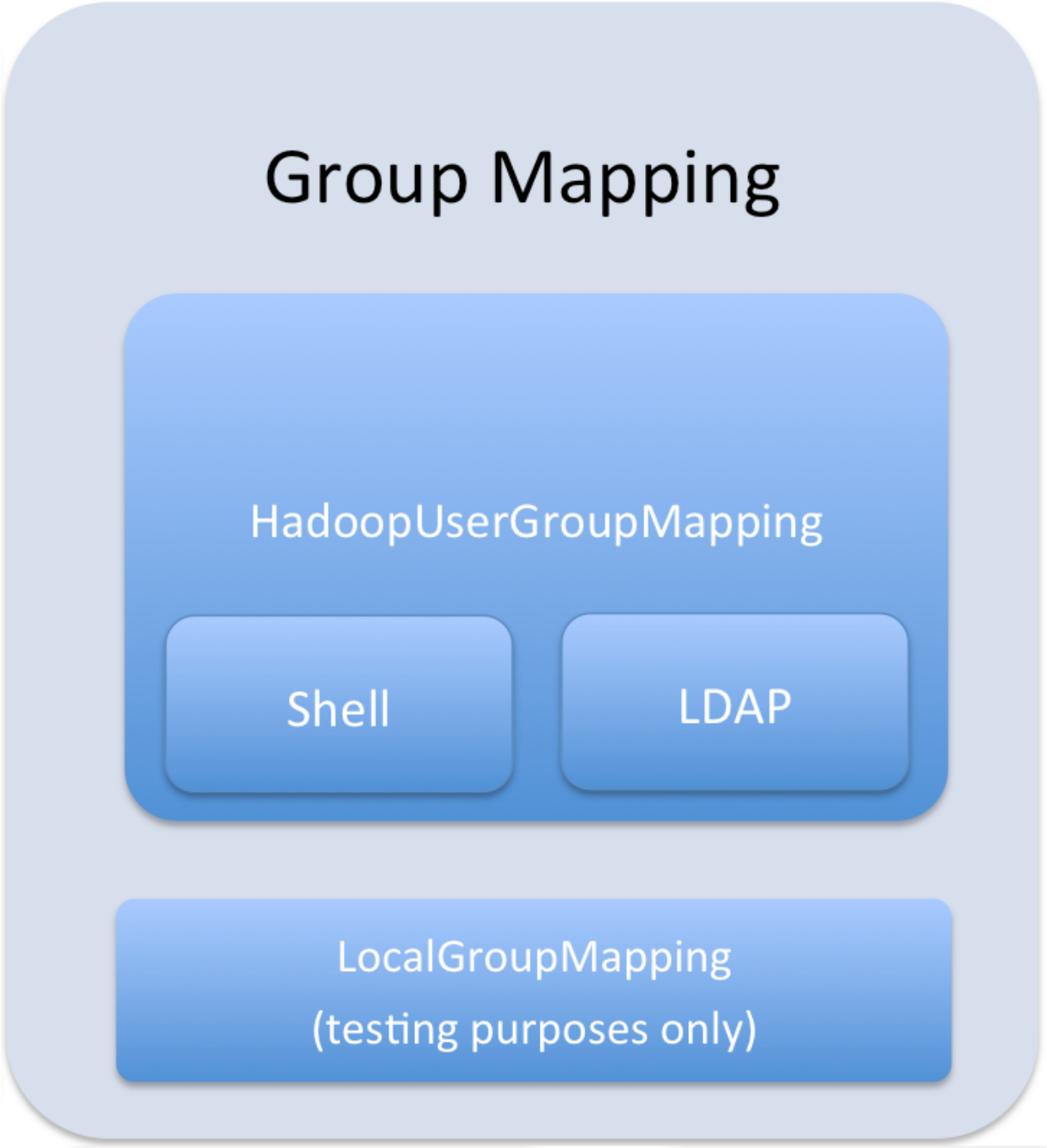
> **Important:**
>
> - When Sentry is enabled, you must use Beeline to execute Hive queries. Hive CLI is not supported with Sentry.
> - When Sentry is enabled, a user with no privileges on a database will not be allowed to connect to HiveServer2. This is because the `use <database>` command is now executed as part of the connection to HiveServer2, which is why the connection fails. See HIVE-4256.

For more information, see Appendix: Authorization Privilege Model for Hive and Impala on page 220.

User to Group Mapping

Minimum Required Role: Configurator (also provided by **Cluster Administrator, Full Administrator**)

Group mappings in Sentry can be summarized as in the figure below.



The Sentry service only uses HadoopUserGroup mappings. You can refer Configuring LDAP Group Mappings on page 125 for details on configuring LDAP group mappings in Hadoop.

> **Important:** Cloudera strongly recommends *against* using Hadoop's `LdapGroupsMapping` provider. `LdapGroupsMapping` should only be used in cases where OS-level integration is not possible. Production clusters require an identity provider that works well with all applications, not just Hadoop. Hence, often the preferred mechanism is to use tools such as SSSD, VAS or Centrify to replicate LDAP groups.

Appendix: Authorization Privilege Model for Hive and Impala

Privileges can be granted on different objects in the Hive warehouse. Any privilege that can be granted is associated with a level in the object hierarchy. If a privilege is granted on a container object in the hierarchy, the base object automatically inherits it. For instance, if a user has ALL privileges on the database scope, then (s)he has ALL privileges on all of the base objects contained within that scope.

> **Important:**
>
> Note that because of this object hierarchy, it is possible for a user to read data from a database that the user does not have access to. For example, you have two roles:
>
> - *role1* - full access to *database1* and *database2*
> - *role2* - full access to *database1*, no access to *database2*
>
> A user with *role1* can create a view in *database1* based on a table in *database2*. Because *role2* has access to *database1*, a user with *role2* can read the data in that view from *database2*.

Object Hierarchy in Hive

```
Server
     URI
     Database
         Table
             Partition
             Columns
         View
         Index
     Function/Routine
     Lock
```

Table 13: Valid privilege types and objects they apply to

Privilege	Object
INSERT	DB, TABLE
SELECT	DB, TABLE
ALL	SERVER, TABLE, DB, URI

Note that when you grant ALL on a URI, those permissions extend into the subdirectories in that path. For example, if a role has ALL on the following URI:

- `hdfs://host:port/directory_A/directory_B`

That role will also have ALL on these directories:

- `hdfs://host:port/directory_A/directory_B/directory_C`
- `hdfs://host:port/directory_A/directory_B/directory_C/directory_D`
- `hdfs://host:port/directory_A/directory_B/directory_E`

URI permissions do not affect HDFS ACL's.

Table 14: Privilege hierarchy

Base Object	Granular privileges on object	Container object that contains the base object	Privileges on container object that implies privileges on the base object
DATABASE	ALL	SERVER	ALL

Base Object	Granular privileges on object	Container object that contains the base object	Privileges on container object that implies privileges on the base object
TABLE	INSERT	DATABASE	ALL
TABLE	SELECT	DATABASE	ALL
VIEW	SELECT	DATABASE	ALL

Table 15: Privilege table for Hive & Impala operations

Operation	Scope	Privileges	URI	Others
CREATE DATABASE	SERVER	ALL		
DROP DATABASE	DATABASE	ALL		
CREATE TABLE	DATABASE	ALL		
DROP TABLE	TABLE	ALL		
CREATE VIEW	DATABASE; SELECT on TABLE	ALL		SELECT on TABLE
DROP VIEW	VIEW/TABLE	ALL		
ALTER TABLE .. ADD COLUMNS	TABLE	ALL		
ALTER TABLE .. REPLACE COLUMNS	TABLE	ALL		
ALTER TABLE .. CHANGE column	TABLE	ALL		
ALTER TABLE .. RENAME	TABLE	ALL		
ALTER TABLE .. SET TBLPROPERTIES	TABLE	ALL		
ALTER TABLE .. SET FILEFORMAT	TABLE	ALL		
ALTER TABLE .. SET LOCATION	TABLE	ALL	URI	
ALTER TABLE .. ADD PARTITION	TABLE	ALL		
ALTER TABLE .. ADD PARTITION location	TABLE	ALL	URI	
ALTER TABLE .. DROP PARTITION	TABLE	ALL		
ALTER TABLE .. PARTITION SET FILEFORMAT	TABLE	ALL		
SHOW CREATE TABLE	TABLE	SELECT/INSERT		
SHOW PARTITIONs	TABLE	SELECT/INSERT		
DESCRIBE TABLE	TABLE	SELECT/INSERT		

Operation	Scope	Privileges	URI	Others
LOAD DATA	TABLE	INSERT	URI	
SELECT	TABLE	SELECT		
INSERT OVERWRITE TABLE	TABLE	INSERT		
CREATE TABLE .. AS SELECT	DATABASE; SELECT on TABLE	ALL		SELECT on TABLE
USE <dbName>	Any			
CREATE FUNCTION	SERVER	ALL		
ALTER TABLE .. SET SERDEPROPERTIES	TABLE	ALL		
ALTER TABLE .. PARTITION SET SERDEPROPERTIES	TABLE	ALL		
Hive-Only Operations				
INSERT OVERWRITE DIRECTORY	TABLE	INSERT	URI	
Analyze TABLE	TABLE	SELECT + INSERT		
IMPORT TABLE	DATABASE	ALL	URI	
EXPORT TABLE	TABLE	SELECT	URI	
ALTER TABLE TOUCH	TABLE	ALL		
ALTER TABLE TOUCH PARTITION	TABLE	ALL		
ALTER TABLE .. CLUSTERED BY SORTED BY	TABLE	ALL		
ALTER TABLE .. ENABLE/DISABLE	TABLE	ALL		
ALTER TABLE .. PARTITION ENABLE/DISABLE	TABLE	ALL		
ALTER TABLE .. PARTITION. . RENAME TO PARTITION	TABLE	ALL		
MSCK REPAIR TABLE	TABLE	ALL		
ALTER DATABASE	DATABASE	ALL		
DESCRIBE DATABASE	DATABASE	SELECT/INSERT		
SHOW COLUMNS	TABLE	SELECT/INSERT		
CREATE INDEX	TABLE	ALL		
DROP INDEX	TABLE	ALL		
SHOW INDEXES	TABLE	SELECT/INSERT		
GRANT PRIVILEGE	Allowed only for Sentry admin users			
REVOKE PRIVILEGE	Allowed only for Sentry admin users			

Operation	Scope	Privileges	URI	Others
SHOW GRANT	Allowed only for Sentry admin users			
SHOW TBLPROPERTIES	TABLE	SELECT/INSERT		
DESCRIBE TABLE .. PARTITION	TABLE	SELECT/INSERT		
ADD JAR	Not Allowed			
ADD FILE	Not Allowed			
DFS	Not Allowed			
Impala-Only Operations				
EXPLAIN	TABLE	SELECT		
INVALIDATE METADATA	SERVER	ALL		
INVALIDATE METADATA <table name>	TABLE	SELECT/INSERT		
REFRESH <table name>	TABLE	SELECT/INSERT		
DROP FUNCTION	SERVER	ALL		
COMPUTE STATS	TABLE	ALL		

Installing and Upgrading the Sentry Service

This topic describes how to install and upgrade the Sentry service. If you are migrating from Sentry policy files to the database-backed Sentry service, see Migrating from Sentry Policy Files to the Sentry Service on page 226.

Adding the Sentry Service

Use one of the following sections to add/install the Sentry service:

Adding the Sentry Service Using Cloudera Manager

Minimum Required Role: Cluster Administrator (also provided by **Full Administrator**)

1. On the Home page, click to the right of the cluster name and select **Add a Service**. A list of service types display. You can add one type of service at a time.
2. Select the **Sentry** service and click **Continue**.
3. Select the radio button next to the services on which the new service should depend and click **Continue**.
4. Customize the assignment of role instances to hosts. The wizard evaluates the hardware configurations of the hosts to determine the best hosts for each role. These assignments are typically acceptable, but you can reassign role instances to hosts of your choosing, if desired.

   Click a field below a role to display a dialog box containing a pageable list of hosts. If you click a field containing multiple hosts, you can also select **All Hosts** to assign the role to all hosts or **Custom** to display the pageable hosts dialog box.

   The following shortcuts for specifying hostnames are supported:

   - Range of hostnames (without the domain portion)

Range Definition	Matching Hosts
10.1.1.[1-4]	10.1.1.1, 10.1.1.2, 10.1.1.3, 10.1.1.4
host[1-3].company.com	host1.company.com, host2.company.com, host3.company.com
host[07-10].company.com	host07.company.com, host08.company.com, host09.company.com, host10.company.com

- IP addresses
- Rack name

Click the **View By Host** button for an overview of the role assignment by host ranges.

5. Configure database settings. You can use either an embedded or a custom database.
   a. Choose the database type:
      - Leave the default setting of **Use Embedded Database** to have Cloudera Manager create and configure required databases. Make a note of the auto-generated passwords.
      - Select **Use Custom Databases** to specify external databases.
        1. Enter the database host, database type, database name, username, and password for the database that you created when you set up the database. See the Creating Databases documentation for Sentry Server database requirements.
   b. Click **Test Connection** to confirm that Cloudera Manager can communicate with the database using the information you have supplied. If the test succeeds in all cases, click **Continue**; otherwise check and correct the information you have provided for the database and then try the test again. (For some servers, if you are using the embedded database, you will see a message saying the database will be created at a later step in the installation process.) The Review Changes page displays.
6. Click **Continue** then click **Finish**. You are returned to the Home page.
7. Verify the new service is started properly by checking the health status for the new service. If the Health Status is **Good**, then the service started properly.
8. To use the Sentry service, begin by enabling Hive and Impala for the service.

Installing Sentry Using the Command Line

Use the following the instructions, depending on your operating system, to install the latest version of Sentry.

> **Important: Configuration files**
>
> - If you install a newer version of a package that is already on the system, configuration files that you have modified will remain intact.
> - If you uninstall a package, the package manager renames any configuration files you have modified from `<file>` to `<file>.rpmsave`. If you then re-install the package (probably to install a new version) the package manager creates a new `<file>` with applicable defaults. You are responsible for applying any changes captured in the original configuration file to the new configuration file. In the case of Ubuntu and Debian upgrades, you will be prompted if you have made changes to a file for which there is a new version; for details, see Automatic handling of configuration files by `dpkg`.

OS	Command
RHEL	`$ sudo yum install sentry`
SLES	`$ sudo zypper install sentry`

OS	Command
Ubuntu or Debian	`$ sudo apt-get update;` `$ sudo apt-get install sentry`

Starting the Sentry Service

Perform the following steps to start the Sentry service on your cluster.

1. Set the `SENTRY_HOME` and `HADOOP_HOME` parameters.
2. Create the Sentry database schema using the Sentry schematool. Sentry, by default, does not initialize the schema. The schematool is a built-in way for you to deploy the backend schema required by the Sentry service. For example, the following command uses the schematool to initialize the schema for a MySQL database.

```
bin/sentry --command schema-tool --conffile <sentry-site.xml> --dbType mysql --initSchema
```

   Alternatively, you can set the `sentry.verify.schema.version` configuration property to `false`. However, this is not recommended.
3. Start the Sentry service.

```
bin/sentry --command service --conffile <sentry-site.xml>
```

Upgrading the Sentry Service

Use one of the following sections to upgrade the Sentry service:

Upgrading the Sentry Service Using Cloudera Manager

If you have a cluster managed by Cloudera Manager, go to Upgrading CDH and Managed Services Using Cloudera Manager and follow the instructions depending on the version of CDH you are upgrading to. If you are upgrading from CDH 5.1, you will notice an extra step in the procedure to upgrade the Sentry database schema.

For command-line instructions, continue reading.

Upgrading the Sentry Service Using the Command Line

1. Stop the Sentry service by identifying the PID of the Sentry Service and use the `kill` command to end the process:

```
ps -ef | grep sentry
kill -9 <PID>
```

   Replace `<PID>` with the PID of the Sentry Service.
2. Remove the previous version of Sentry.

OS	Command
RHEL	`$ sudo yum remove sentry`
SLES	`$ sudo zypper remove sentry`
Ubuntu or Debian	`$ sudo apt-get remove sentry`

3. Install the new version of Sentry.

OS	Command
RHEL	`$ sudo yum install sentry`
SLES	`$ sudo zypper install sentry`
Ubuntu or Debian	`$ sudo apt-get update;` `$ sudo apt-get install sentry`

4. **(From CDH 5.1 to CDH 5.x) Upgrade Sentry Database Schema**

   Use the Sentry `schematool` to upgrade the database schema as follows:

```
bin/sentry --command schema-tool --conffile <sentry-site.xml> --dbType <db-type>
--upgradeSchema
```

   Where `<db-type>` should be either `mysql`, `postgres` or `oracle`.

5. **Start the Sentry Service**

   a. Set the `SENTRY_HOME` and `HADOOP_HOME` parameters.
   b. Run the following command:

```
bin/sentry --command service --conffile <sentry-site.xml>
```

Migrating from Sentry Policy Files to the Sentry Service

Minimum Required Role: Cluster Administrator (also provided by **Full Administrator**)

The following steps describe how you can upgrade from Sentry's policy file-based approach to the new database-backed Sentry service.

1. If you haven't already done so, upgrade your cluster to the latest version of CDH and Cloudera Manager. Refer the Cloudera Manager Administration Guide for instructions.
2. Disable the existing Sentry policy file for any Hive or Impala services on the cluster. To do this:
   a. Navigate to the Hive or Impala service.
   b. Click the **Configuration** tab.
   c. Under the **Service-Wide** > **Policy File Based Sentry** category, uncheck the **Enable Sentry Authorization using Policy Files** checkbox. Cloudera Manager will throw a validation error if you attempt to configure the Sentry service while this property is checked.
   d. Repeat for any remaining Hive or Impala services.
3. Add the new Sentry service to your cluster. For instructions, see Adding the Sentry Service on page 223.
4. To begin using the Sentry service, see Enabling the Sentry Service Using Cloudera Manager on page 226 and Configuring Impala as a Client for the Sentry Service on page 229.
5. Use the command-line interface Beeline to issue grants to the Sentry service to match the contents of your old policy file(s). For more details on the Sentry service and examples on using Grant/Revoke statements to match your policy file, see Hive SQL Syntax for Use with Sentry on page 232.

Configuring the Sentry Service

This topic describes how to enable the Sentry service for Hive and Impala, and configuring the Hive metastore to communicate with the service.

Enabling the Sentry Service Using Cloudera Manager

Minimum Required Role: Configurator (also provided by **Cluster Administrator, Full Administrator**)

Before Enabling the Sentry Service

- Ensure you satisfy all the Prerequisites on page 217 for the Sentry service.
- The Hive warehouse directory (`/user/hive/warehouse` or any path you specify as `hive.metastore.warehouse.dir` in your `hive-site.xml`) must be owned by the Hive user and group.
  - Permissions on the warehouse directory must be set as follows (see following Note for caveats):
    - **771** on the directory itself (for example, `/user/hive/warehouse`)
    - **771** on all subdirectories (for example, `/user/hive/warehouse/mysubdir`)

– All files and subdirectories should be owned by hive:hive

For example:

```
$ sudo -u hdfs hdfs dfs -chmod -R 771 /user/hive/warehouse
$ sudo -u hdfs hdfs dfs -chown -R hive:hive /user/hive/warehouse
```

Note:

- If you set `hive.warehouse.subdir.inherit.perms` to `true` in `hive-site.xml`, the permissions on the subdirectories will be set when you set permissions on the warehouse directory itself.
- If a user has access to any object in the warehouse, that user will be able to execute `use default`. This ensures that `use default` commands issued by legacy applications work when Sentry is enabled.

Important: These instructions override the recommendations in the Hive section of the CDH 5 Installation Guide.

- Disable impersonation for HiveServer2 in the Cloudera Manager Admin Console:
  1. Go to the Hive service.
  2. Click the **Configuration** tab.
  3. Under the **HiveServer2** role group, uncheck the **HiveServer2 Enable Impersonation** property, and click **Save Changes**.
- If you are using MapReduce, enable the Hive user to submit MapReduce jobs.
  1. Open the Cloudera Manager Admin Console and go to the MapReduce service.
  2. Click the **Configuration** tab.
  3. Under a TaskTracker role group go to the **Security** category.
  4. Set the **Minimum User ID for Job Submission** property to zero (the default is 1000) and click **Save Changes**.
  5. Repeat steps 1-4 for *every* TaskTracker role group for the MapReduce service that is associated with Hive, if more than one exists.
  6. Restart the MapReduce service.
- If you are using YARN, enable the Hive user to submit YARN jobs.
  1. Open the Cloudera Manager Admin Console and go to the YARN service.
  2. Click the **Configuration** tab.
  3. Under a NodeManager role group go to the **Security** category.
  4. Ensure the **Allowed System Users** property includes the `hive` user. If not, add `hive` and click **Save Changes**.
  5. Repeat steps 1-4 for *every* NodeManager role group for the YARN service that is associated with Hive, if more than one exists.
  6. Restart the YARN service.

Important: Ensure you have unchecked the **Enable Sentry Authorization using Policy Files** configuration property for *both* Hive and Impala under the **Service-Wide** > **Policy File Based Sentry** category before you proceed.

Enabling the Sentry Service for Hive

1. Go to the Hive service.
2. Click the **Configuration** tab.

3. In the **Service-Wide** category, set the **Sentry Service** property to `Sentry`.
4. Restart the Hive service.

Enabling the Sentry Service for Impala

1. Enable the Sentry service for Hive (as instructed above).
2. Go to the Impala service.
3. Click the **Configuration** tab.
4. In the **Service-Wide** category, set the **Sentry Service** property to `Sentry`.
5. Restart Impala.

Enabling the Sentry Service for Hue

To interact with Sentry using Hue, enable the Sentry service as follows:

1. Enable the Sentry service for Hive and Impala (as instructed above).
2. Go to the Hue service.
3. Click the **Configuration** tab.
4. In the **Service-Wide** category, set the **Sentry Service** property to `Sentry`.
5. Restart Hue.

Enabling the Sentry Service Using the Command Line

Important:

- If you use Cloudera Manager, do not use these command-line instructions.
- This information applies specifically to CDH 5.3.x . If you use an earlier version of CDH, see the documentation for that version located at Cloudera Documentation.

Before Enabling the Sentry Service

- The Hive warehouse directory (`/user/hive/warehouse` or any path you specify as `hive.metastore.warehouse.dir` in your `hive-site.xml`) must be owned by the Hive user and group.
  - Permissions on the warehouse directory must be set as follows (see following Note for caveats):
    - **771** on the directory itself (for example, `/user/hive/warehouse`)
    - **771** on all subdirectories (for example, `/user/hive/warehouse/mysubdir`)
    - All files and subdirectories should be owned by hive:hive

    For example:

```
$ sudo -u hdfs hdfs dfs -chmod -R 771 /user/hive/warehouse
$ sudo -u hdfs hdfs dfs -chown -R hive:hive /user/hive/warehouse
```

Note:

- If you set `hive.warehouse.subdir.inherit.perms` to `true` in `hive-site.xml`, the permissions on the subdirectories will be set when you set permissions on the warehouse directory itself.
- If a user has access to any object in the warehouse, that user will be able to execute `use default`. This ensures that `use default` commands issued by legacy applications work when Sentry is enabled.

Important: These instructions override the recommendations in the Hive section of the CDH 5 Installation Guide.

- HiveServer2 impersonation must be turned off.
- If you are using MapReduce, you must enable the Hive user to submit MapReduce jobs. You can ensure that this is true by setting the minimum user ID for job submission to 0. Edit the `taskcontroller.cfg` file and set `min.user.id=0`.

  If you are using YARN, you must enable the Hive user to submit YARN jobs, add the user `hive` to the `allowed.system.users` configuration property. Edit the `container-executor.cfg` file and add `hive` to the `allowed.system.users` property. For example,

```
allowed.system.users = nobody,impala,hive
```

> **Important:** You must restart the cluster and HiveServer2 after changing these values.

Configuring HiveServer2 for the Sentry Service

Add the following properties to `hive-site.xml` to allow the Hive service to communicate with the Sentry service.

```
<property>
   <name>hive.security.authorization.task.factory</name>
   <value>org.apache.sentry.binding.hive.SentryHiveAuthorizationTaskFactoryImpl</value>
</property>
<property>
   <name>hive.server2.session.hook</name>
   <value>org.apache.sentry.binding.hive.HiveAuthzBindingSessionHook</value>
</property>
<property>
   <name>hive.sentry.conf.url</name>
   <value>file:///{{PATH/TO/DIR}}/sentry-site.xml</value>
</property>
```

Configuring the Hive Metastore for the Sentry Service

Add the following properties to `hive-site.xml` to allow the Hive metastore to communicate with the Sentry service.

```
<property>
    <name>hive.metastore.client.impl</name>
    <value>org.apache.sentry.binding.metastore.SentryHiveMetaStoreClient</value>
    <description>Sets custom Hive metastore client which Sentry uses to filter out
metadata.</description>
</property>

<property>
    <name>hive.metastore.pre.event.listeners</name>
    <value>org.apache.sentry.binding.metastore.MetastoreAuthzBinding</value>
    <description>list of comma separated listeners for metastore events.</description>
</property>

<property>
    <name>hive.metastore.event.listeners</name>
   <value>org.apache.sentry.binding.metastore.SentryMetastorePostEventListener</value>

    <description>list of comma separated listeners for metastore, post
events.</description>
</property>
```

Configuring Impala as a Client for the Sentry Service

Set the following configuration properties in `sentry-site.xml`.

```
<property>
   <name>sentry.service.client.server.rpc-port</name>
   <value>3893</value>
</property>
<property>
```

```
   <name>sentry.service.client.server.rpc-address</name>
   <value>hostname</value>
</property>
<property>
   <name>sentry.service.client.server.rpc-connection-timeout</name>
   <value>200000</value>
</property>
<property>
   <name>sentry.service.security.mode</name>
   <value>none</value>
</property>
```

You must also add the following configuration properties to Impala's `/etc/default/impala` file. For more information , see Configuring Impala Startup Options through the Command Line.

- On the `catalogd` and the `impalad`.

```
--sentry_config=<absolute path to sentry service configuration file>
```

- On the `impalad`.

```
--server_name=<server name>
```

If the `--authorization_policy_file` flag is set, Impala will use the policy file-based approach. Otherwise, the database-backed approach will be used to implement authorization.

Configuring Pig and HCatalog for the Sentry Service

Once you have the Sentry service up and running, and Hive has been configured to use the Sentry service, there are some configuration changes you must make to your cluster to allow Pig, MapReduce (using HCatLoader, HCatStorer) and WebHCat queries to access Sentry-secured data stored in Hive.

Since the Hive warehouse directory is owned by `hive:hive`, with its permissions set to **`771`**, with these settings, other user requests such as commands coming through Pig jobs, WebHCat queries, and MapReduce jobs, may fail. To give these users access, perform the following configuration changes:

- Use HDFS ACLs to define permissions on a specific directory or file of HDFS. This directory/file is generally mapped to a database, table, partition, or a data file.
- Users running these jobs should have the required permissions in Sentry to add new metadata or read metadata from the Hive Metastore Server. For instructions on how to set up the required permissions, see Hive SQL Syntax for Use with Sentry on page 232. You can use HiveServer2's command line interface, Beeline to update the Sentry database with the user privileges.

Examples:

- A user who is using Pig HCatLoader will require read permissions on a specific table or partition. In such a case, you can `GRANT` read access to the user in Sentry and set the ACL to read and execute, on the file being accessed.
- A user who is using Pig HCatStorer will require ALL permissions on a specific table. In this case, you `GRANT ALL` access to the user in Sentry and set the ACL to write and execute, on the table being used.

Securing the Hive Metastore

It's important that the Hive metastore be secured. If you want to override the Kerberos prerequisite for the Hive metastore, set the `sentry.hive.testing.mode` property to `true` to allow Sentry to work with weaker authentication mechanisms. Add the following property to the HiveServer2 and Hive metastore's `sentry-site.xml`:

```
<property>
  <name>sentry.hive.testing.mode</name>
  <value>true</value>
</property>
```

Impala does not require this flag to be set.

> **Warning:** Cloudera strongly recommends against enabling this property in production. Use Sentry's testing mode only in test environments.

You can also set the property in Cloudera Manager. Go to the Hive service and open the **Configuration** tab. Search for the **Hive Service Advanced Configuration Snippet (Safety Valve) for sentry-site.xml**. Click the plus sign (+) to add a new property with the following values:

- **Name:** sentry.hive.testing.mode
- **Value:** true

You canturn on Hive metastore security using the instructions in Cloudera Security. To secure the Hive metastore; see Hive Metastore Server Security Configuration on page 81.

Using User-Defined Functions with HiveServer2

The `ADD JAR` command does *not* work with HiveServer2 and the Beeline client when Beeline runs on a different host. As an alternative to `ADD JAR`, Hive's *auxiliary paths* functionality should be used. There are some differences in the procedures for creating permanent functions and temporary functions when Sentry is enabled. For detailed instructions, see:

- User-Defined Functions (UDFs) with HiveServer2 Using Cloudera Manager

  OR

- User-Defined Functions (UDFs) with HiveServer2 Using the Command Line

.

Sentry Debugging and Failure Scenarios

This topic describes how Sentry deals with conflicting policies, how to debug Sentry authorization request failures and how different CDH components respond when the Sentry service fails.

Resolving Policy Conflicts

Sentry treats all policies independently. Hence, for any operation, if Sentry can find a policy that allows it, that operation will be allowed. Consider an example with a table, `test_db.test_tbl`, whose HDFS directory is located at `hdfs://user/hive/warehouse/test_db.db/test_tbl`, and grant the following conflicting privileges to a user with the role, `test_role`. That is, you are granting `ALL` privilege to the role `test_role` on the URI, but only the `SELECT` privilege on the table itself.

```
GRANT ALL ON URI 'hdfs:///user/hive/warehouse/test_db.db/test_tbl' to role test_role;
USE test_db;
GRANT SELECT ON TABLE test_tbl to role test_role;
```

With these privileges, all users with the role `test_role` will be able to carry out the `EXPORT TABLE` operation even though they should only have `SELECT` privileges on `test_db.test_tbl`:

```
EXPORT TABLE <another-table-user-can-read> TO
'hdfs:///user/hive/warehouse/test_db.db/test_tbl'
```

Debugging Failed Sentry Authorization Requests

Sentry logs all facts that lead up to authorization decisions at the debug level. If you do not understand why Sentry is denying access, the best way to debug is to temporarily turn on debug logging:

- In Cloudera Manager, add `log4j.logger.org.apache.sentry=DEBUG` to the logging settings for your service through the corresponding **Logging Safety Valve** field for Impala or HiveServer2.
- On systems not managed by Cloudera Manager, add `log4j.logger.org.apache.sentry=DEBUG` to the `log4j.properties` file on each host in the cluster, in the appropriate configuration directory for each service.

Specifically, look for exceptions and messages such as:

```
FilePermission server..., RequestPermission server...., result [true|false]
```

which indicate each evaluation Sentry makes. The `FilePermission` is from the policy file, while `RequestPermission` is the privilege required for the query. A `RequestPermission` will iterate over all appropriate `FilePermission` settings until a match is found. If no matching privilege is found, Sentry returns `false` indicating "Access Denied" .

Sentry Service Failure Scenarios

If the Sentry service fails and you attempt to access the Hive warehouse, Hive, Impala and HDFS will behave as follows:

- **Hive:** Queries to the Hive warehouse will fail with an authentication error.
- **Impala:** The Impala Catalog server caches Sentry privileges. If Sentry goes down, Impala queries will continue to work and will be authorized against this cached copy of the metadata. However, authorization DDLs such as `CREATE ROLE` or `GRANT ROLE` will fail.
- **HDFS/Sentry Synchronized Permissions:** Affected HDFS files will continue to use a cached copy of the synchronized ACLs for a configurable period of time (by default, 60 seconds), after which they will fall back to NameNode ACLs.
- **Solr:** Solr does not use the Sentry service, hence there will be no impact.

Hive SQL Syntax for Use with Sentry

Sentry permissions can be configured through Grant and Revoke statements issued either interactively or programmatically through the HiveServer2 SQL command line interface, Beeline (documentation available here). Note that since Hive CLI does not work with HiveServer2, it cannot be used to configure Sentry permissions.

In HUE, the Sentry Admin that creates roles and grants privileges must belong to a group that has ALL privileges on the server. For example, you can create a role for the group that contains the hive or impala user, and grant ALL ON SERVER to that role:

```
CREATE ROLE <admin role>;
GRANT ALL ON SERVER <server1> TO ROLE <admin_role>;
GRANT ROLE <admin role> TO GROUP <hive>;
```

Important:

- When Sentry is enabled, you must use Beeline to execute Hive queries. Hive CLI is not supported with Sentry.
- Note that there are some differences in syntax between Hive and the corresponding Impala SQL statements. For the Impala syntax, see SQL Statements.

CREATE ROLE Statement

The `CREATE ROLE` statement creates a role to which privileges can be granted. Privileges can be granted to roles, which can then be assigned to users. A user that has been assigned a role will only be able to exercise the privileges of that role.

Only users that have administrative privileges can create/drop roles. By default, the `hive`, `impala` and hue users have admin privileges in Sentry.

```
CREATE ROLE [role_name];
```

DROP ROLE Statement

The `DROP ROLE` statement can be used to remove a role from the database. Once dropped, the role will be revoked for all users to whom it was previously assigned. Queries that are already executing will not be affected. However,

since Hive checks user privileges before executing each query, active user sessions in which the role has already been enabled will be affected.

```
DROP ROLE [role_name];
```

GRANT ROLE Statement

The `GRANT ROLE` statement can be used to grant roles to groups. Only Sentry admin users can grant roles to a group.

```
GRANT ROLE role_name [, role_name]
    TO GROUP <groupName> [,GROUP <groupName>]
```

REVOKE ROLE Statement

The `REVOKE ROLE` statement can be used to revoke roles from groups. Only Sentry admin users can revoke the role from a group.

```
REVOKE ROLE role_name [, role_name]
    FROM GROUP <groupName> [,GROUP <groupName>]
```

GRANT <PRIVILEGE> Statement

In order to grant privileges on an object to a role, the user must be a Sentry admin user.

```
GRANT
    <PRIVILEGE> [, <PRIVILEGE> ]
    ON <OBJECT> <object_name>
    TO ROLE <roleName> [,ROLE <roleName>]
```

REVOKE <PRIVILEGE> Statement

Since only authorized admin users can create roles, consequently only Sentry admin users can revoke privileges from a group.

```
REVOKE
    <PRIVILEGE> [, <PRIVILEGE> ]
    ON <OBJECT> <object_name>
    FROM ROLE <roleName> [,ROLE <roleName>]
```

GRANT <PRIVILEGE> ... WITH GRANT OPTION

With CDH 5.2, you can delegate granting and revoking privileges to other roles. For example, a role that is granted a privilege `WITH GRANT OPTION` can `GRANT/REVOKE` the same privilege to/from other roles. Hence, if a role has the ALL privilege on a database and the `WITH GRANT OPTION` set, users granted that role can execute `GRANT/REVOKE` statements only for that database or child tables of the database.

```
GRANT
    <PRIVILEGE>
    ON <OBJECT> <object_name>
    TO ROLE <roleName>
    WITH GRANT OPTION
```

Only a role with `GRANT` option on a specific privilege or its parent privilege can revoke that privilege from other roles. Once the following statement is executed, all privileges with and without grant option are revoked.

```
REVOKE
    <PRIVILEGE>
    ON <OBJECT> <object_name>
    FROM ROLE <roleName>
```

Hive does not currently support revoking only the `WITH GRANT OPTION` from a privilege previously granted to a role. To remove the `WITH GRANT OPTION`, revoke the privilege and grant it again without the `WITH GRANT OPTION` flag.

SET ROLE Statement

The `SET ROLE` statement can be used to specify a role to be enabled for the current session. A user can only enable a role that has been granted to them. Any roles not listed and not already enabled are disabled for the current session. If no roles are enabled, the user will have the privileges granted by any of the roles that (s)he belongs to.

To enable a specific role:

```
SET ROLE <roleName>;
```

To enable all roles:

```
SET ROLE ALL;
```

No roles enabled:

```
SET ROLE NONE;
```

SHOW Statement

To list all the roles in the system (only for sentry admin users):

```
SHOW ROLES;
```

To list all the roles in effect for the current user session:

```
SHOW CURRENT ROLES;
```

To list all the roles assigned to the given `<groupName>` (only allowed for Sentry admin users and others users that are part of the group specified by `<groupName>`):

```
SHOW ROLE GRANT GROUP <groupName>;
```

The `SHOW` statement can also be used to list the privileges that have been granted to a role or all the grants given to a role for a particular object.

To list all the grants for the given `<roleName>` (only allowed for Sentry admin users and other users that have been granted the role specified by `<roleName>`):

```
SHOW GRANT ROLE <roleName>;
```

To list all the grants for a role on the given `<objectName>` (only allowed for Sentry admin users and other users that have been granted the role specified by `<roleName>`):

```
SHOW GRANT ROLE <roleName> on OBJECT <objectName>;
```

Example: Using Grant/Revoke Statements to Match an Existing Policy File

Here is a sample policy file:

```
[groups]
# Assigns each Hadoop group to its set of roles
manager = analyst_role, junior_analyst_role
analyst = analyst_role
jranalyst = junior_analyst_role
customers_admin = customers_admin_role
admin = admin_role

[roles] # The uris below define a define a landing skid which
# the user can use to import or export data from the system.
# Since the server runs as the user "hive" files in that directory
# must either have the group hive and read/write set or
```

```
# be world read/write.
analyst_role = server=server1->db=analyst1, \
    server=server1->db=jranalyst1->table=*->action=select
    server=server1->uri=hdfs://ha-nn-uri/landing/analyst1
junior_analyst_role = server=server1->db=jranalyst1, \
    server=server1->uri=hdfs://ha-nn-uri/landing/jranalyst1

# Implies everything on server1.
admin_role = server=server1
```

The following sections show how you can use the new GRANT statements to assign privileges to roles (and assign roles to groups) to match the sample policy file above.

Grant privileges to `analyst_role`:

```
CREATE ROLE analyst_role;
GRANT ALL ON DATABASE analyst1 TO ROLE analyst_role;
GRANT SELECT ON DATABASE jranalyst1 TO ROLE analyst_role;
GRANT ALL ON URI 'hdfs://ha-nn-uri/landing/analyst1' \
TO ROLE analyst_role;
```

Grant privileges to `junior_analyst_role`:

```
CREATE ROLE junior_analyst_role;
GRANT ALL ON DATABASE jranalyst1 TO ROLE junior_analyst_role;
GRANT ALL ON URI 'hdfs://ha-nn-uri/landing/jranalyst1' \
TO ROLE junior_analyst_role;
```

Grant privileges to `admin_role`:

```
CREATE ROLE admin_role
GRANT ALL ON SERVER server TO ROLE admin_role;
```

Grant roles to groups:

```
GRANT ROLE admin_role TO GROUP admin;
GRANT ROLE analyst_role TO GROUP analyst;
GRANT ROLE jranalyst_role TO GROUP jranalyst;
```

Synchronizing HDFS ACLs and Sentry Permissions

This topic introduces an HDFS-Sentry plugin that allows you to configure synchronization of Sentry privileges with HDFS ACLs for specific HDFS directories.

Previously, when Sentry was used to secure data in Hive or Impala, it was difficult to securely share the same HDFS data files with other components such as Pig, MapReduce, Spark, HDFS client and so on. You had two options:

- You could set ownership for the entire Hive warehouse to `hive:hive` and not allow other components any access to the data. While this is secure, it does not allow for sharing.
- Use HDFS ACLs and synchronize Sentry privileges and HDFS ACLs manually. For example, if a user only has the Sentry SELECT privilege on a table, that user should only be able to read the table data files, and not write to those HDFS files.

Introduction

To solve the problem stated above, CDH 5.3 introduces integration of Sentry and HDFS permissions that will automatically keep HDFS ACLs in sync with the privileges configured with Sentry. This feature offers the easiest way to share data between Hive, Impala and other components such as MapReduce, Pig, and so on, while setting permissions for that data with just one set of rules through Sentry. It maintains the ability of Hive and Impala to set permissions on views, in addition to tables, while access to data outside of Hive and Impala (for example, reading files off HDFS) requires table permissions. HDFS permissions for some or all of the files that are part of tables defined in the Hive Metastore will now be controlled by Sentry.

This change consists of three components:

- An HDFS NameNode plugin
- A Sentry-Hive Metastore plugin
- A Sentry Service plugin

Sentry will translate table permissions to the appropriate corresponding ACL on the underlying table files in HDFS. For example, if a user group is assigned to a Sentry role that has SELECT permission on a particular table, then that user group will also have read access to the HDFS files that are part of that table. When listing those files in HDFS, you will see this permission listed as an HDFS ACL. The mapping of Sentry privileges to HDFS ACL permissions is as follows:

- SELECT privilege -> Read access on the file.
- INSERT privilege -> Write access on the file.
- ALL privilege -> Read and Write access on the file.

Important:

- With synchronization enabled, your ability to set HDFS permissions for those files is disabled. Permissions for those particular files can be set only through Sentry, and when examined through HDFS these permissions appear as HDFS ACLs.

  A configurable set of users (such as `hive` and `impala`) will have full access to the files automatically. This ensures that a key requirement of using Sentry with Hive and Impala — giving these processes full access to regulate permissions on underlying data files — is met automatically.

- Tables that are not associated with Sentry (that is, have no user with Sentry privileges to access them) will retain their old ACLs.
- There are no HDFS permissions associated with table views. That is, even if you have a view giving you access to some columns of a table, this does not translate to any HDFS permissions on the file(s).
- Synchronized privileges are not persisted to HDFS. This means that when this feature is disabled, HDFS privileges will return to their original values.
- If the Sentry service fails, affected HDFS files will continue to use a cached copy of the synchronized ACLs for a configurable period of time (by default, 60 seconds), after which they will fall back to NameNode ACLs.
- Sentry HDFS synchronization does not support Hive metastore HA.

Prompting HDFS ACL Changes

URIs do not have an impact on the HDFS-Sentry plugin. Therefore, you cannot manage all of your HDFS ACLs with the HDFS-Sentry plugin and you must continue to use standard HDFS ACLs for data outside of Hive.

HDFS ACL changes are triggered on:

- Hive DATABASE object LOCATION (HDFS) when a role is granted to the object
- Hive TABLE object LOCATION (HDFS) when a role is granted to the object

HDFS ACL changes are not triggered by:

- Hive URI LOCATION (HDFS) when a role is granted to a URI
- Hive SERVER object when a role is granted to the object. HDFS ACLs are not updated if a role is assigned to the SERVER. The privileges are inherited by child objects in standard Sentry interactions, but the plugin does not trickle the privileges down.
- Permissions granted on views. Views are not synchronized as objects in the HDFS file system.

Prerequisites

- CDH 5.3.0 (or later) managed by Cloudera Manager 5.3.0 (or later)

- (Strongly Recommended) Implement Kerberos authentication on your cluster.

The following conditions must be also be true when enabling Sentry-HDFS synchronization. Failure to comply with any of these will result in validation errors.

- You must use the Sentry service, not policy file-based authorization.
- Enabling HDFS Extended Access Control Lists (ACLs) is required.
- There must be exactly one Sentry service dependent on HDFS.
- The Sentry service must have exactly one Sentry Server role.
- The Sentry service must have exactly one dependent Hive service.
- The Hive service must have exactly one Hive Metastore role (that is, High Availability should not be enabled).

Enabling the HDFS-Sentry Plugin Using Cloudera Manager

1. Go to the HDFS service.
2. Click the **Configuration** tab.
3. Navigate to the **Service-Wide** category and check the **Check HDFS Permissions** checkbox.
4. Under the **Service-Wide** category go to **Security**.
5. Check the **Enable Sentry Synchronization** checkbox.
6. Use the **Sentry Synchronization Path Prefixes** property to list HDFS path prefixes where Sentry permissions should be enforced. Multiple HDFS path prefixes can be specified. By default, this property points to `user/hive/warehouse` and must always be non-empty. HDFS privilege synchronization will not occur for tables located outside the HDFS regions listed here.
7. Click **Save Changes**.
8. Restart the cluster. Note that it may take an additional two minutes after cluster restart for privilege synchronization to take effect.

Enabling the HDFS-Sentry Plugin Using the Command Line

Important:

- If you use Cloudera Manager, do not use these command-line instructions.
- This information applies specifically to CDH 5.3.x. If you use an earlier version of CDH, see the documentation for that version located at Cloudera Documentation.

To enable the Sentry plugins on an unmanaged cluster, you must explicitly allow the `hdfs` user to interact with Sentry, and install the plugin packages as described in the following sections.

Allowing the `hdfs` user to connect with Sentry

For an unmanaged cluster, add `hdfs` to the `sentry.service.allow.connect` property in `sentry-site.xml`.

```
<property>
    <name>sentry.service.allow.connect</name>
    <value>impala,hive,hue,hdfs</value>
</property>
```

Installing the HDFS-Sentry Plugin

Note: Install Cloudera Repository

Before using the instructions on this page to install the package, install the Cloudera `yum`, `zypper/YaST` or `apt` repository, and install or upgrade CDH 5 and make sure it is functioning correctly. For instructions, see Installing the Latest CDH 5 Release.

Use the following the instructions, depending on your operating system, to install the `sentry-hdfs-plugin` package. The package must be installed (at a minimum) on the following hosts:

- The host running the NameNode and Secondary NameNode
- The host running the Hive Metastore
- The host running the Sentry Service

OS	Command
RHEL-compatible	`$ sudo yum install sentry-hdfs-plugin`
SLES	`$ sudo zypper install sentry-hdfs-plugin`
Ubuntu or Debian	`$ sudo apt-get install sentry-hdfs-plugin`

Configuring the HDFS NameNode Plugin

Add the following properties to the `hdfs-site.xml` file on the NameNode host.

```
<property>
<name>dfs.namenode.acls.enabled</name>
<value>true</value>
</property>

<property>
<name>dfs.namenode.authorization.provider.class</name>
<value>org.apache.sentry.hdfs.SentryAuthorizationProvider</value>
</property>

<property>
<name>dfs.permissions</name>
<value>true</value>
</property>

<!-- Comma-separated list of HDFS path prefixes where Sentry permissions should be
enforced. -->
<!-- Privilege synchronization will occur only for tables located in HDFS regions
specified here. -->
<property>
<name>sentry.authorization-provider.hdfs-path-prefixes</name>
<value>/user/hive/warehouse</value>
</property>

<property>
<name>sentry.hdfs.service.security.mode</name>
<value>kerberos</value>
</property>

<property>
<name>sentry.hdfs.service.server.principal</name>
<value> SENTRY_SERVER_PRINCIPAL (for eg :  sentry/_HOST@VPC.CLOUDERA.COM )</value>
</property>

<property>
<name>sentry.hdfs.service.client.server.rpc-port</name>
<value>SENTRY_SERVER_PORT</value>
</property>

<property>
<name>sentry.hdfs.service.client.server.rpc-address</name>
<value>SENTRY_SERVER_HOST</value>
</property>
```

Configuring the Hive Metastore Plugin

Add the following properties to `hive-site.xml` on the Hive Metastore Server host.

```
<property>
<name>sentry.metastore.plugins</name>
<value>org.apache.sentry.hdfs.MetastorePlugin</value>
</property>

<property>
```

```
<name>sentry.hdfs.service.client.server.rpc-port</name>
<value> SENTRY_SERVER_PORT </value>
</property>

<property>
<name>sentry.hdfs.service.client.server.rpc-address</name>
<value> SENTRY_SERVER_HOSTNAME </value>
</property>

<property>
<name>sentry.hdfs.service.client.server.rpc-connection-timeout</name>
<value>200000</value>
</property>

<property>
<name>sentry.hdfs.service.security.mode</name>
<value>kerberos</value>
</property>

<property>
<name>sentry.hdfs.service.server.principal</name>
<value> SENTRY_SERVER_PRINCIPAL (for eg :  sentry/_HOST@VPC.CLOUDERA.COM )</value>
</property>
```

Configuring the Sentry Service Plugin

Add the following properties to the `sentry-site.xml` file on the NameNode host.

```
<property>
<name>sentry.service.processor.factories</name>
<value>org.apache.sentry.provider.db.service.thrift.SentryPolicyStoreProcessorFactory,
org.apache.sentry.hdfs.SentryHDFSServiceProcessorFactory</value>
</property>

<property>
<name>sentry.policy.store.plugins</name>
<value>org.apache.sentry.hdfs.SentryPlugin</value>
</property>
```

> **Important:** Once all the configuration changes are complete, restart your cluster. Note that it may take an additional two minutes after cluster restart for privilege synchronization to take effect.

Testing the Sentry Synchronization Plugins

The following tasks should help you make sure that Sentry-HDFS synchronization has been enabled and configured correctly:

For a folder that has been enabled for the plugin, such as the Hive warehouse, try accessing the files in that folder outside Hive and Impala. For this, you should know what tables those HDFS files belong to and the Sentry permissions on those tables. Attempt to view or modify the Sentry permissions settings over those tables using one of the following tools:

- **(Recommended)** Hue's Security application
- HiveServer2 CLI
- Impala CLI
- Access the table files directly in HDFS. For example:
  - List files inside the folder and verify that the file permissions shown in HDFS (including ACLs) match what was configured in Sentry.
  - Run a MapReduce, Pig or Spark job that accesses those files. Pick any tool besides HiveServer2 and Impala

Reporting Metrics for the Sentry Service

Metrics for the Sentry service can now be reported using either JMX or console. To obtain the metrics in JSON format, you can use the Sentry Web Server which by default, listens on port 51000. Use the following properties to enable and configure metric reports.

`sentry.service.reporter`	Specify the tool being used to report metrics. Value: `jmx` or `console`
`sentry.service.web.enable`	Set this property to `true` to enable reporting of metrics by the Sentry Web Server. Default: `false`
`sentry.service.web.port`	Configure the port on which the Sentry Web Server listens for metrics. Default: `51000`

Sentry Policy File Authorization

Important: This is the documentation for configuring Sentry using the policy file approach. Cloudera recommends you use the database-backed Sentry service introduced in CDH 5.1 to secure your data. See The Sentry Service on page 217 for more information.

Sentry enables role-based, fine-grained authorization for HiveServer2, Cloudera Impala and Cloudera Search.

For more information on installing, upgrading and configuring policy file authorization, see:

Prerequisites

Sentry depends on an underlying authentication framework to reliably identify the requesting user. It requires:

- CDH 4.3.0 or higher.
- HiveServer2 and the Hive Metastore running with strong authentication. For HiveServer2, strong authentication is either Kerberos or LDAP. For the Hive Metastore, only Kerberos is considered strong authentication (to override, see Securing the Hive Metastore on page 257).
- Impala 1.2.1 (or higher) running with strong authentication. With Impala, either Kerberos or LDAP can be configured to achieve strong authentication. Auditing of authentication failures is supported only with CDH 4.4.0 and Impala 1.2.1 or higher.
- Implement Kerberos authentication on your cluster. This is to prevent a user bypassing the authorization and gaining direct access to the underlying data.

Terminologies

- An **object** is an entity protected by Sentry's authorization rules. The objects supported in the current release are `server`, `database`, `table`, and `URI`.
- A **role** is a collection of rules for accessing a given Hive object.
- A **privilege** is granted to a role to govern access to an object. Supported privileges are:

Table 16: Valid privilege types and the objects they apply to

Privilege	Object
INSERT	SEVER, DB, TABLE
SELECT	DB, TABLE
ALL	SERVER, TABLE, DB, URI

- A user is an entity that is permitted by the authentication subsystem to access the Hive service. This entity can be a Kerberos principal, an LDAP `userid`, or an artifact of some other pluggable authentication system supported by HiveServer2.
- A group connects the authentication system with the authorization system. It is a collection of one or more users who have been granted one or more authorization roles. Sentry allows a set of roles to be configured for a group.
- A configured group provider determines a user's affiliation with a group. The current release supports HDFS-backed groups and locally configured groups.

Privilege Model

Sentry uses a role-based privilege model with the following characteristics.

- Allows any user to execute `show function`, `desc function`, and `show locks`.
- Allows the user to see only those tables and databases for which this user has privileges.
- Requires a user to have the necessary privileges on the URI to execute HiveQL operations that take in a location. Examples of such operations include `LOAD`, `IMPORT`, and `EXPORT`.

> **Important:**
>
> - When Sentry is enabled, you must use Beeline to execute Hive queries. Hive CLI is not supported with Sentry.
> - When Sentry is enabled, a user with no privileges on a database will not be allowed to connect to HiveServer2. This is because the `use <database>` command is now executed as part of the connection to HiveServer2, which is why the connection fails. See HIVE-4256.

For more information, see Authorization Privilege Model for Hive and Impala on page 247.

Granting Privileges

For example, a rule for the `Select` privilege on table `customers` from database `sales` would be formulated as follows:

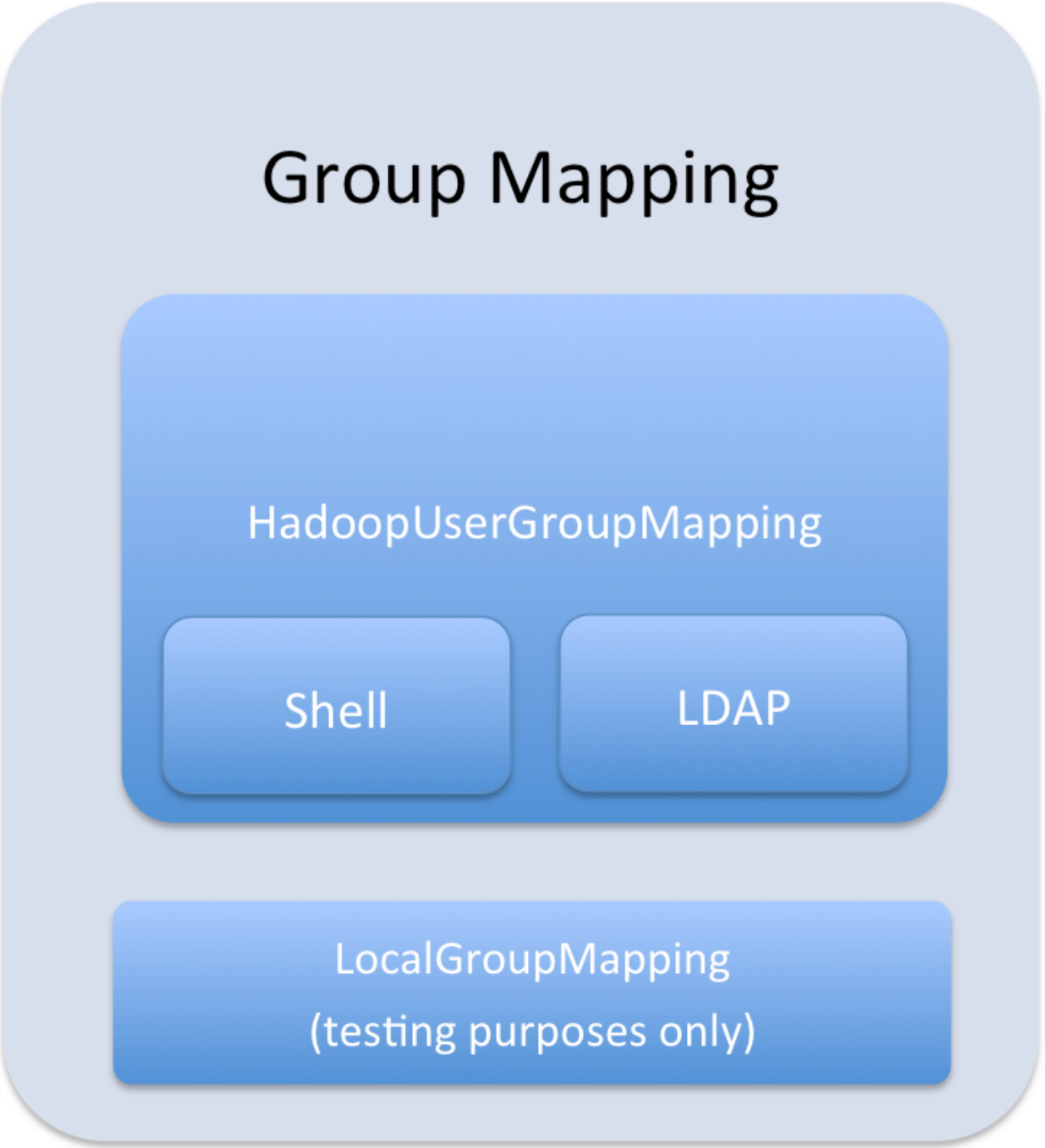
```
server=server1->db=sales->table=customer->action=Select
```

Each object must be specified as a hierarchy of the containing objects, from server to table, followed by the privilege granted for that object. A role can contain multiple such rules, separated by commas. For example, a role might contain the `Select` privilege for the `customer` and `items` tables in the `sales` database, and the `Insert` privilege for the `sales_insights` table in the `reports` database. You would specify this as follows:

```
sales_reporting =
\server=server1->db=sales->table=customer->action=Select,
\server=server1->db=sales->table=items>action=Select,
\server=server1->db=reports->table=sales_insights>action=Insert
```

User to Group Mapping

You can configure Sentry to use either Hadoop groups or groups defined in the policy file. By default, Sentry looks up groups locally, but it can be configured to look up Hadoop groups using LDAP (for Active Directory). Local groups will be looked up on the host Sentry runs on. For Hive, this will be the host running HiveServer2. Group mappings in Sentry can be summarized as in the figure below:



Important: You can use either Hadoop groups or local groups, but not both at the same time. Local groups are traditionally used for a quick proof-of-concept, while Hadoop groups are more commonly used in production. Refer Configuring LDAP Group Mappings on page 125 for details on configuring LDAP group mappings in Hadoop.

Policy File

The sections that follow contain notes on creating and maintaining the policy file, and using URIs to load external data and JARs.

> **Warning:** An invalid policy file will be ignored while logging an exception. This will lead to a situation where users will lose access to all Sentry-protected data, since default Sentry behaviour is *deny* unless a user has been explicitly granted access. (Note that if only the per-DB policy file is invalid, it will invalidate only the policies in that file.)

Storing the Policy File

Considerations for storing the policy file(s) in HDFS include:

1. Replication count - Because the file is read for each query in Hive and read once every five minutes by all Impala daemons, you should increase this value; since it is a small file, setting the replication count equal to the number of slave nodes in the cluster is reasonable.
2. Updating the file - Updates to the file are reflected immediately, so you should write them to a temporary copy of the file first, and then replace the existing file with the temporary one after all the updates are complete. This avoids race conditions caused by reads on an incomplete file.

Defining Roles

Keep in mind that role definitions are not cumulative; the definition that is further down in the file replaces the older one. For example, the following results in `role1` having `privilege2`, not `privilege1` and `privilege2`.

```
role1 = privilege1
role1 = privilege2
```

Role names are scoped to a specific file. For example, if you give `role1` the `ALL` privilege on `db1` in the global policy file and give `role1` `ALL` on `db2` in the per-db db2 policy file, the user will be given both privileges.

URIs

Any command which references a URI such as `CREATE TABLE EXTERNAL`, `LOAD`, `IMPORT`, `EXPORT`, and more, in addition to `CREATE TEMPORARY FUNCTION` requires the `URI` privilege. This is an important security control because without this users could simply create an external table over an existing table they do not have access to and bypass Sentry.

URIs must start with either `hdfs://` or `file://`. If a URI starts with anything else, it will cause an exception and the policy file will be invalid.

When defining URIs for HDFS, you must also specify the NameNode. For example:

```
data_read = server=server1->uri=file:///path/to/dir,\
server=server1->uri=hdfs://namenode:port/path/to/dir
```

> **Important:** Because the NameNode host and port must be specified, Cloudera strongly recommends you use High Availability (HA). This ensures that the URI will remain constant even if the NameNode changes.

Loading Data

Data can be loaded using a landing skid, either in HDFS or using a local/NFS directory where HiveServer2/Impala run. The following privileges can be used to grant a role access to a loading skid:

- **Load data from a local/NFS directory:**

```
server=server1->uri=file:///path/to/nfs/local/to/nfs
```

- **Load data from HDFS (MapReduce, Pig, and so on):**

```
server=server1->uri=hdfs://ha-nn-uri/data/landing-skid
```

In addition to the privilege in Sentry, the `hive` or `impala` user will require the appropriate file permissions to access the data being loaded. Groups can be used for this purpose. For example, create a group `hive-users`, and add the `hive` and `impala` users along with the users who will be loading data, to this group.

The example `usermod` and `groupadd` commands below are only applicable to locally defined groups on the NameNode, JobTracker, and ResourceManager. If you use another system for group management, equivalent changes should be made in your group management system.

```
$ groupadd hive-users
$ usermod -G someuser,hive-users someuser
$ usermod -G hive,hive-users hive
```

External Tables

External tables require the `ALL@database` privilege in addition to the `URI` privilege. When data is being inserted through the `EXTERNAL TABLE` statement, or is referenced from an HDFS location outside the normal Hive database directories, the user needs appropriate permissions on the URIs corresponding to those HDFS locations. This means that the URI location must either be owned by the `hive:hive` user OR the `hive/impala` users must be members of the group that owns the directory.

You can configure access to the directory using a URI as follows:

```
[roles]
someuser_home_dir_role = server=server1->uri=hdfs://ha-nn-uri/user/someuser
```

You should now be able to create an external table:

```
CREATE EXTERNAL TABLE ...
LOCATION 'hdfs://ha-nn-uri/user/someuser/mytable';
```

Sample Sentry Configuration Files

This section provides a sample configuration.

Policy Files

The following is an example of a policy file with a per-DB policy file. In this example, the first policy file, `sentry-provider.ini` would exist in HDFS; `hdfs://ha-nn-uri/etc/sentry/sentry-provider.ini` might be an appropriate location. The per-DB policy file is for the customer's database. It is located at `hdfs://ha-nn-uri/etc/sentry/customers.ini`.

sentry-provider.ini

```
[databases]
# Defines the location of the per DB policy file for the customers DB/schema
customers = hdfs://ha-nn-uri/etc/sentry/customers.ini

[groups]
# Assigns each Hadoop group to its set of roles
manager = analyst_role, junior_analyst_role
analyst = analyst_role
jranalyst = junior_analyst_role
customers_admin = customers_admin_role
admin = admin_role

[roles]
# The uris below define a define a landing skid which
# the user can use to import or export data from the system.
```

```
# Since the server runs as the user "hive" files in that directory
# must either have the group hive and read/write set or
# be world read/write.
analyst_role = server=server1->db=analyst1, \
    server=server1->db=jranalyst1->table=*->action=select
    server=server1->uri=hdfs://ha-nn-uri/landing/analyst1
junior_analyst_role = server=server1->db=jranalyst1, \
    server=server1->uri=hdfs://ha-nn-uri/landing/jranalyst1

# Implies everything on server1 -> customers. Privileges for
# customers can be defined in the global policy file even though
# customers has its only policy file. Note that the Privileges from
# both the global policy file and the per-DB policy file
# are merged. There is no overriding.
customers_admin_role = server=server1->db=customers

# Implies everything on server1.
admin_role = server=server1
```

customers.ini

```
[groups]
manager = customers_insert_role, customers_select_role
analyst = customers_select_role

[roles]
customers_insert_role = server=server1->db=customers->table=*->action=insert
customers_select_role = server=server1->db=customers->table=*->action=select
```

Important: Sentry does not support using the `view` keyword in policy files. If you want to define a role against a view, use the keyword `table` instead. For example, to define the role `analyst_role` against the view `col_test_view`:

```
[roles]
analyst_role =
server=server1->db=default->table=col_test_view->action=select
```

Sentry Configuration File

The following is an example of a `sentry-site.xml` file.

Important: If you are using Cloudera Manager 4.6 (or earlier), make sure you **do not** store `sentry-site.xml` in `/etc/hive/conf` ; that directory is regenerated whenever the Hive client configurations are redeployed. Instead, use a directory such as `/etc/sentry` to store the `sentry` file.

If you are using Cloudera Manager 4.7 (or higher), Cloudera Manager will create and deploy `sentry-site.xml` for you.See The Sentry Service on page 217 for more details on configuring Sentry with Cloudera Manager.

sentry-site.xml

```
<configuration>
  <property>
    <name>hive.sentry.provider</name>

<value>org.apache.sentry.provider.file.HadoopGroupResourceAuthorizationProvider</value>

  </property>

  <property>
    <name>hive.sentry.provider.resource</name>
```

```
    <value>/path/to/authz-provider.ini</value>
    <!--
       If the hdfs-site.xml points to HDFS, the path will be in HDFS;
       alternatively you could specify a full path, e.g.:
       hdfs://namenode:port/path/to/authz-provider.ini
       file:///path/to/authz-provider.ini
    -->
  </property>

  <property>
    <name>sentry.hive.server</name>
    <value>server1</value>
  </property>
</configuration>
```

Accessing Sentry-Secured Data Outside Hive/Impala

When Sentry is enabled, the `hive` user owns all data within the Hive warehouse. However, unlike traditional database systems the enterprise data hub allows for multiple engines to execute over the same dataset.

> **Note:** Cloudera strongly recommends you use Hive/Impala SQL queries to access data secured by Sentry, as opposed to accessing the data files directly.

However, there are scenarios where fully vetted and reviewed jobs will also need to access the data stored in the Hive warehouse. A typical scenario would be a secured MapReduce transformation job that is executed automatically as an application user. In such cases it's important to know that the user executing this job will also have full access to the data in the Hive warehouse.

Scenario One: Authorizing Jobs

Problem

A reviewed, vetted, and automated job requires access to the Hive warehouse and cannot use Hive/Impala to access the data.

Solution

Create a group which contains `hive`, `impala`, and the user executing the automated job. For example, if the `etl` user is executing the automated job, you can create a group called `hive-users` which contains the `hive`, `impala`, and `etl` users.

The example `usermod` and groupadd commands below are only applicable to locally defined groups on the NameNode, JobTracker, and ResourceManager. If you use another system for group management, equivalent changes should be made in your group management system.

```
$ groupadd hive-users
$ usermod -G hive,impala,hive-users hive
$ usermod -G hive,impala,hive-users impala
$ usermod -G etl,hive-users etl
```

Once you have added users to the `hive-users` group, change directory permissions in the HDFS:

```
$ hadoop fs -chgrp -R hive:hive-users /user/hive/warehouse
$ hadoop fs -chmod -R 770 /user/hive/warehouse
```

Scenario Two: Authorizing Group Access to Databases

Problem

One group of users, `grp1` should have full access to the database, `db1`, outside of Sentry. The database, `db1` should not be accessible to any other groups, outside of Sentry. Sentry should be used for all other authorization needs.

Solution

Place the `hive` and `impala` users in `grp1`.

```
$ usermod -G hive,impala,grp1 hive
$ usermod -G hive,impala,grp1 impala
```

Then change group ownerships of all directories and files in `db1` to `grp1`, and modify directory permissions in the HDFS. This example is only applicable to local groups on a single host.

```
$ hadoop fs -chgrp -R hive:grp1 /user/hive/warehouse/db1.db
$ hadoop fs -chmod -R 770 /user/hive/warehouse/db1.db
```

Debugging Failed Sentry Authorization Requests

Sentry logs all facts that lead up to authorization decisions at the debug level. If you do not understand why Sentry is denying access, the best way to debug is to temporarily turn on debug logging:

- In Cloudera Manager, add `log4j.logger.org.apache.sentry=DEBUG` to the logging settings for your service through the corresponding **Logging Safety Valve** field for the Impala, Hive Server 2, or Solr Server services.
- On systems not managed by Cloudera Manager, add `log4j.logger.org.apache.sentry=DEBUG` to the `log4j.properties` file on each host in the cluster, in the appropriate configuration directory for each service.

Specifically, look for exceptions and messages such as:

```
FilePermission server..., RequestPermission server...., result [true|false]
```

which indicate each evaluation Sentry makes. The `FilePermission` is from the policy file, while `RequestPermission` is the privilege required for the query. A `RequestPermission` will iterate over all appropriate `FilePermission` settings until a match is found. If no matching privilege is found, Sentry returns `false` indicating “Access Denied” .

Authorization Privilege Model for Hive and Impala

Privileges can be granted on different objects in the Hive warehouse. Any privilege that can be granted is associated with a level in the object hierarchy. If a privilege is granted on a container object in the hierarchy, the base object automatically inherits it. For instance, if a user has `ALL` privileges on the database scope, then (s)he has `ALL` privileges on all of the base objects contained within that scope.

Object Hierarchy in Hive

```
Server
     URI
     Database
         Table
             Partition
             Columns
         View
         Index
     Function/Routine
     Lock
```

Table 17: Valid privilege types and objects they apply to

Privilege	Object
INSERT	DB, TABLE
SELECT	DB, TABLE
ALL	SERVER, TABLE, DB, URI

Note that when you grant ALL on a URI, those permissions extend into the subdirectories in that path. For example, if a role has ALL on the following URI:

- `hdfs://host:port/directory_A/directory_B`

That role will also have ALL on these directories:

- `hdfs://host:port/directory_A/directory_B/directory_C`
- `hdfs://host:port/directory_A/directory_B/directory_C/directory_D`
- `hdfs://host:port/directory_A/directory_B/directory_E`

URI permissions do not affect HDFS ACL's.

Table 18: Privilege hierarchy

Base Object	Granular privileges on object	Container object that contains the base object	Privileges on container object that implies privileges on the base object
DATABASE	ALL	SERVER	ALL
TABLE	INSERT	DATABASE	ALL
TABLE	SELECT	DATABASE	ALL
VIEW	SELECT	DATABASE	ALL

Table 19: Privilege table for Hive & Impala operations

Operation	Scope	Privileges	URI	Others
CREATE DATABASE	SERVER	ALL		
DROP DATABASE	DATABASE	ALL		
CREATE TABLE	DATABASE	ALL		
DROP TABLE	TABLE	ALL		
CREATE VIEW	DATABASE; SELECT on TABLE	ALL		SELECT on TABLE
DROP VIEW	VIEW/TABLE	ALL		
ALTER TABLE .. ADD COLUMNS	TABLE	ALL		
ALTER TABLE .. REPLACE COLUMNS	TABLE	ALL		
ALTER TABLE .. CHANGE column	TABLE	ALL		
ALTER TABLE .. RENAME	TABLE	ALL		
ALTER TABLE .. SET TBLPROPERTIES	TABLE	ALL		
ALTER TABLE .. SET FILEFORMAT	TABLE	ALL		
ALTER TABLE .. SET LOCATION	TABLE	ALL	URI	
ALTER TABLE .. ADD PARTITION	TABLE	ALL		

Operation	Scope	Privileges	URI	Others
ALTER TABLE .. ADD PARTITION location	TABLE	ALL	URI	
ALTER TABLE .. DROP PARTITION	TABLE	ALL		
ALTER TABLE .. PARTITION SET FILEFORMAT	TABLE	ALL		
SHOW CREATE TABLE	TABLE	SELECT/INSERT		
SHOW PARTITIONs	TABLE	SELECT/INSERT		
DESCRIBE TABLE	TABLE	SELECT/INSERT		
LOAD DATA	TABLE	INSERT	URI	
SELECT	TABLE	SELECT		
INSERT OVERWRITE TABLE	TABLE	INSERT		
CREATE TABLE .. AS SELECT	DATABASE; SELECT on TABLE	ALL		SELECT on TABLE
USE <dbName>	Any			
CREATE FUNCTION	SERVER	ALL		
ALTER TABLE .. SET SERDEPROPERTIES	TABLE	ALL		
ALTER TABLE .. PARTITION SET SERDEPROPERTIES	TABLE	ALL		
Hive-Only Operations				
INSERT OVERWRITE DIRECTORY	TABLE	INSERT	URI	
Analyze TABLE	TABLE	SELECT + INSERT		
IMPORT TABLE	DATABASE	ALL	URI	
EXPORT TABLE	TABLE	SELECT	URI	
ALTER TABLE TOUCH	TABLE	ALL		
ALTER TABLE TOUCH PARTITION	TABLE	ALL		
ALTER TABLE .. CLUSTERED BY SORTED BY	TABLE	ALL		
ALTER TABLE .. ENABLE/DISABLE	TABLE	ALL		
ALTER TABLE .. PARTITION ENABLE/DISABLE	TABLE	ALL		
ALTER TABLE .. PARTITION. . RENAME TO PARTITION	TABLE	ALL		
MSCK REPAIR TABLE	TABLE	ALL		
ALTER DATABASE	DATABASE	ALL		
DESCRIBE DATABASE	DATABASE	SELECT/INSERT		

Operation	Scope	Privileges	URI	Others
SHOW COLUMNS	TABLE	SELECT/INSERT		
CREATE INDEX	TABLE	ALL		
DROP INDEX	TABLE	ALL		
SHOW INDEXES	TABLE	SELECT/INSERT		
GRANT PRIVILEGE	Allowed only for Sentry admin users			
REVOKE PRIVILEGE	Allowed only for Sentry admin users			
SHOW GRANT	Allowed only for Sentry admin users			
SHOW TBLPROPERTIES	TABLE	SELECT/INSERT		
DESCRIBE TABLE .. PARTITION	TABLE	SELECT/INSERT		
ADD JAR	Not Allowed			
ADD FILE	Not Allowed			
DFS	Not Allowed			
Impala-Only Operations				
EXPLAIN	TABLE	SELECT		
INVALIDATE METADATA	SERVER	ALL		
INVALIDATE METADATA <table name>	TABLE	SELECT/INSERT		
REFRESH <table name>	TABLE	SELECT/INSERT		
DROP FUNCTION	SERVER	ALL		
COMPUTE STATS	TABLE	ALL		

Installing and Upgrading Sentry for Policy File Authorization

Sentry stores the configuration as well as privilege policies in files. The `sentry-site.xml` file contains configuration options such as `group association provider`, `privilege policy file location`, and so on. The policy file contains the privileges and groups. It has a `.ini` file format and can be stored on a local file system or HDFS.

Sentry is plugged into Hive as session hooks which you configure in `hive-site.xml`. The `vsentry` package must be installed; it contains the required JAR files. You must also configure properties in the Sentry Configuration File on page 245.

> **Important:**
>
> If you have not already done so, install the Cloudera `yum`, `zypper/YaST` or `apt` repository before using the following commands. For instructions, see Installing the Latest CDH 5 Release.

Installing Sentry

Use the following the instructions, depending on your operating system, to install the latest version of Sentry.

Important: Configuration files

- If you install a newer version of a package that is already on the system, configuration files that you have modified will remain intact.
- If you uninstall a package, the package manager renames any configuration files you have modified from `<file>` to `<file>.rpmsave`. If you then re-install the package (probably to install a new version) the package manager creates a new `<file>` with applicable defaults. You are responsible for applying any changes captured in the original configuration file to the new configuration file. In the case of Ubuntu and Debian upgrades, you will be prompted if you have made changes to a file for which there is a new version; for details, see Automatic handling of configuration files by `dpkg`.

OS	Command
RHEL	`$ sudo yum install sentry`
SLES	`$ sudo zypper install sentry`
Ubuntu or Debian	`$ sudo apt-get update;` `$ sudo apt-get install sentry`

Upgrading Sentry

If you are upgrading from CDH 5.x to the latest CDH release, see Installing Sentry on page 250 to install the latest version.

Configuring Sentry Policy File Authorization Using Cloudera Manager

This topic describes how to configure Sentry policy files and enable policy file authorization for CDH services using Cloudera Manager.

Configuring User to Group Mappings

Minimum Required Role: Configurator (also provided by **Cluster Administrator, Full Administrator**)

Hadoop Groups

1. Go to the Hive service.
2. Click the **Configuration** tab.
3. Under the **Service-Wide** category, go to the **Policy File Based Sentry** section.
4. Set the **Sentry User to Group Mapping Class** property to `org.apache.sentry.provider.file.HadoopGroupResourceAuthorizationProvider`.
5. Click **Save Changes**.
6. Restart the Hive service.

Local Groups

1. Define local groups in the `[users]` section of the Policy File on page 242. For example:

```
[users]
user1 = group1, group2, group3
user2 = group2, group3
```

2. Modify Sentry configuration as follows:
   a. Go to the Hive service.
   b. Click the **Configuration** tab.
   c. Under the **Service-Wide** category, go to the **Policy File Based Sentry** section.
   d. Set the **Sentry User to Group Mapping Class** property to `org.apache.sentry.provider.file.LocalGroupResourceAuthorizationProvider`.

e. Click **Save Changes**.
f. Restart the Hive service.

Enabling URIs for Per-DB Policy Files

The ADD JAR command does *not* work with HiveServer2 and the Beeline client when Beeline runs on a different host. As an alternative to ADD JAR, Hive's *auxiliary paths* functionality should be used as described in the following steps.

> **Important:** Enabling URIs in per-DB policy files introduces a security risk by allowing the owner of the db-level policy file to grant himself/herself load privileges to anything the `hive` user has read permissions for in HDFS (including data in other databases controlled by different db-level policy files).

Add the following string to the Java configuration options for HiveServer2 during startup.

```
-Dsentry.allow.uri.db.policyfile=true
```

Using User-Defined Functions with HiveServer2

Minimum Required Role: Configurator (also provided by **Cluster Administrator, Full Administrator**)

The ADD JAR command does *not* work with HiveServer2 and the Beeline client when Beeline runs on a different host. As an alternative to ADD JAR, Hive's *auxiliary paths* functionality should be used. There are some differences in the procedures for creating permanent functions and temporary functions. For detailed instructions, see User-Defined Functions (UDFs) with HiveServer2 Using Cloudera Manager.

Enabling Policy File Authorization for Hive

Minimum Required Role: Configurator (also provided by **Cluster Administrator, Full Administrator**)

1. Ensure the Prerequisites on page 240 have been satisfied.
2. The Hive warehouse directory (`/user/hive/warehouse` or any path you specify as `hive.metastore.warehouse.dir` in your `hive-site.xml`) must be owned by the Hive user and group.
   - Permissions on the warehouse directory must be set as follows (see following Note for caveats):
     - **771** on the directory itself (for example, `/user/hive/warehouse`)
     - **771** on all subdirectories (for example, `/user/hive/warehouse/mysubdir`)
     - All files and subdirectories should be owned by hive:hive

     For example:

```
$ sudo -u hdfs hdfs dfs -chmod -R 771 /user/hive/warehouse
$ sudo -u hdfs hdfs dfs -chown -R hive:hive /user/hive/warehouse
```

> **Note:**
> - If you set `hive.warehouse.subdir.inherit.perms` to `true` in `hive-site.xml`, the permissions on the subdirectories will be set when you set permissions on the warehouse directory itself.
> - If a user has access to any object in the warehouse, that user will be able to execute `use default`. This ensures that `use default` commands issued by legacy applications work when Sentry is enabled. Note that you can protect objects in the default database (or any other database) by means of a policy file.

> **Important:** These instructions override the recommendations in the Hive section of the CDH 5 Installation Guide.

3. Disable impersonation for HiveServer2:
   a. Go to the Hive service.
   b. Click the **Configuration** tab.
   c. Under the **HiveServer2** role group, uncheck the **HiveServer2 Enable Impersonation** property, and click **Save Changes**.
4. Create the Sentry policy file, `sentry-provider.ini`, as an HDFS file.
5. Enable the Hive user to submit MapReduce jobs.
   a. Go to the MapReduce service.
   b. Click the **Configuration** tab.
   c. Under a TaskTracker role group go to the **Security** category.
   d. Set the **Minimum User ID for Job Submission** property to zero (the default is 1000) and click **Save Changes**.
   e. Repeat steps 5.a-5.d for *every* TaskTracker role group for the MapReduce service that is associated with Hive, if more than one exists.
   f. Restart the MapReduce service.
6. Enable the Hive user to submit YARN jobs.
   a. Go to the YARN service.
   b. Click the **Configuration** tab.
   c. Under a NodeManager role group go to the **Security** category.
   d. Ensure the **Allowed System Users** property includes the `hive` user. If not, add `hive` and click **Save Changes**.
   e. Repeat steps 6.a-6.d for *every* NodeManager role group for the YARN service that is associated with Hive, if more than one exists.
   f. Restart the YARN service.
7. Go to the Hive service.
8. Click the **Configuration** tab.
9. Under the **Service-Wide** category, go to the **Policy File Based Sentry** section.
10. Check **Enable Sentry Authorization Using Policy Files**, then click **Save Changes**.
11. You must restart the cluster and HiveServer2 after changing these values, whether you use Cloudera Manager or not.

Configuring Group Access to the Hive Metastore

Minimum Required Role: Configurator (also provided by **Cluster Administrator, Full Administrator**)

You can configure the Hive Metastore to reject connections from users not listed in the Hive group proxy list (in HDFS). If you do not configure this override, the Hive Metastore will use the value in the core-site HDFS configuration. To configure the Hive group proxy list:

1. Go to the Hive service.
2. Click the **Configuration** tab.
3. Click the **Proxy** category.
4. In the **Hive Metastore Access Control and Proxy User Groups Override** property, specify a list of groups whose users are allowed to access the Hive Metastore. If you do not specify "*" (wildcard), you will be warned if the groups do not include hive and impala (if the Impala service is configured) in the list of groups.
5. Click **Save Changes**.
6. Restart the Hive service.

Enabling Policy File Authorization for Impala

For a cluster managed by Cloudera Manager, perform the following steps to enable policy file authorization for Impala.

1. Enable Sentry's policy file based authorization for Hive. For details, see Enabling Policy File Authorization for Hive on page 252.
2. Go to the Cloudera Manager Admin Console and navigate to the Impala service.

3. Click the **Configuration** tab.
4. Under the **Service-Wide** category, go to the **Policy File Based Sentry** section.
5. Check **Enable Sentry Authorization Using Policy Files**, then click **Save Changes**.
6. Restart the Impala service.

For more details, see Starting the impalad Daemon with Sentry Authorization Enabled on page 258.

Enabling Sentry Authorization for Solr

Minimum Required Role: Full Administrator

1. Ensure the following requirements are satisfied:
   - Cloudera Search 1.1.1 or higher or CDH 5 or higher.
   - A secure Hadoop cluster.
2. Create the policy file `sentry-provider.ini` as an HDFS file. When you create the policy file `sentry-provider.ini` follow the instructions in the Policy File section in Configuring Sentry for Search (CDH 4) orSearch Authentication on page 104. The file must be owned by owned by the `solr` user in the `solr` group, with perms=600. By default Cloudera Manager assumes the policy file is in the HDFS location `/user/solr/sentry`. To configure the location:
   a. Go to the Solr service.
   b. Click the **Configuration** tab.
   c. Under the **Service-Wide** category, select **Sentry** and modify the path in the **Sentry Global Policy File** property.
   d. Click **Save Changes**.
3. Under the **Service-Wide** category, go to the **Policy File Based Sentry** section.
4. Check **Enable Sentry Authorization Using Policy Files**, then click **Save Changes**.
5. Restart the Solr service.

For more details, see Enabling Sentry Authorization for Search on page 268.

Configuring Sentry to Enable BDR Replication

Cloudera recommends the following steps when configuring Sentry and data replication is enabled.

- Group membership should be managed outside of Sentry (as typically OS groups, LDAP groups, and so on are managed) and replication for them also should be handled outside of Cloudera Manager.
- In Cloudera Manager, set up HDFS replication for the Sentry files of the databases that are being replicated (separately using Hive replication).
- On the source cluster:
  - Use a separate Sentry policy file for every database
  - Avoid placing any group or role info (except for server admin info) in the global Sentry policy file (to avoid manual replication/merging with the global file on the target cluster)
  - To avoid manual fix up of URI privileges, ensure that the URIs for the data are the same on both the source and target cluster
- On the target cluster:
  - In the global Sentry policy file, manually add the *DB name - DB file* mapping entries for the databases being replicated
  - Manually copy the server admin info from the global Sentry policy file on the source to the policy on the target cluster
  - For the databases being replicated, avoid adding more privileges (adding tables specific to target cluster may sometimes require adding extra privileges to allow access to those tables). If any target cluster specific privileges absolutely need to be added for a database, add them to the global Sentry policy file on the target cluster since the per database files would be overwritten periodically with source versions during scheduled replication.

Configuring Sentry Policy File Authorization Using the Command Line

This topic describes how to configure Sentry policy files and enable policy file authorization for unmanaged CDH services using the command line.

Configuring User to Group Mappings

Hadoop Groups

Set the `hive.sentry.provider` property in `sentry-site.xml`.

```
<property>
<name>hive.sentry.provider</name>
<value>org.apache.sentry.provider.file.HadoopGroupResourceAuthorizationProvider</value>
</property>
```

Local Groups

1. Define local groups in the `[users]` section of the Policy File on page 242. For example:

```
[users]
user1 = group1, group2, group3
user2 = group2, group3
```

2. Modify Sentry configuration as follows:

   In `sentry-site.xml`, set `hive.sentry.provider` as follows:

```
<property>
<name>hive.sentry.provider</name>
<value>org.apache.sentry.provider.file.LocalGroupResourceAuthorizationProvider</value>
</property>
```

Enabling URIs for Per-DB Policy Files

The `ADD JAR` command does *not* work with HiveServer2 and the Beeline client when Beeline runs on a different host. As an alternative to `ADD JAR`, Hive's *auxiliary paths* functionality should be used as described in the following steps.

> **Important:** Enabling URIs in per-DB policy files introduces a security risk by allowing the owner of the db-level policy file to grant himself/herself load privileges to anything the `hive` user has read permissions for in HDFS (including data in other databases controlled by different db-level policy files).

Add the following string to the Java configuration options for HiveServer2 during startup.

```
-Dsentry.allow.uri.db.policyfile=true
```

Using User-Defined Functions with HiveServer2

The `ADD JAR` command does *not* work with HiveServer2 and the Beeline client when Beeline runs on a different host. As an alternative to `ADD JAR`, Hive's *auxiliary paths* functionality should be used as described in the following steps. There are some differences in the procedures for creating permanent functions and temporary functions. For detailed instructions, see User-Defined Functions (UDFs) with HiveServer2 Using the Command Line.

Enabling Policy File Authorization for Hive

Prerequisites

In addition to the Prerequisites on page 240 above, make sure that the following are true:

- The Hive warehouse directory (`/user/hive/warehouse` or any path you specify as `hive.metastore.warehouse.dir` in your `hive-site.xml`) must be owned by the Hive user and group.

- Permissions on the warehouse directory must be set as follows (see following Note for caveats):
  - **771** on the directory itself (for example, `/user/hive/warehouse`)
  - **771** on all subdirectories (for example, `/user/hive/warehouse/mysubdir`)
  - All files and subdirectories should be owned by hive:hive

  For example:

```
$ sudo -u hdfs hdfs dfs -chmod -R 771 /user/hive/warehouse
$ sudo -u hdfs hdfs dfs -chown -R hive:hive /user/hive/warehouse
```

Note:

- If you set `hive.warehouse.subdir.inherit.perms` to `true` in `hive-site.xml`, the permissions on the subdirectories will be set when you set permissions on the warehouse directory itself.
- If a user has access to any object in the warehouse, that user will be able to execute `use default`. This ensures that `use default` commands issued by legacy applications work when Sentry is enabled. Note that you can protect objects in the default database (or any other database) by means of a policy file.

Important: These instructions override the recommendations in the Hive section of the CDH 5 Installation Guide.

- HiveServer2 impersonation must be turned off.
- The Hive user must be able to submit MapReduce jobs. You can ensure that this is true by setting the minimum user ID for job submission to 0. Edit the `taskcontroller.cfg` file and set `min.user.id=0`.

  To enable the Hive user to submit YARN jobs, add the user `hive` to the `allowed.system.users` configuration property. Edit the `container-executor.cfg` file and add `hive` to the `allowed.system.users` property. For example,

```
allowed.system.users = nobody,impala,hive
```

Important:

- You must restart the cluster and HiveServer2 after changing this value, whether you use Cloudera Manager or not.
- These instructions override the instructions under Configuring MRv1 Security on page 61
- These instructions override the instructions under Configuring YARN Security on page 64

Configuration Changes Required

To enable Sentry, add the following properties to `hive-site.xml`:

```
<property>
<name>hive.server2.session.hook</name>
<value>org.apache.sentry.binding.hive.HiveAuthzBindingSessionHook</value>
</property>

<property>
<name>hive.sentry.conf.url</name>
<value></value>
<description>sentry-site.xml file location</description>
</property>
```

```
<property>
<name>hive.metastore.client.impl</name>
<value>org.apache.sentry.binding.metastore.SentryHiveMetaStoreClient</value>
<description>Sets custom Hive Metastore client which Sentry uses to filter out
metadata.</description>
</property>
```

Securing the Hive Metastore

It's important that the Hive metastore be secured. If you want to override the Kerberos prerequisite for the Hive metastore, set the `sentry.hive.testing.mode` property to `true` to allow Sentry to work with weaker authentication mechanisms. Add the following property to the HiveServer2 and Hive metastore's `sentry-site.xml`:

```
<property>
  <name>sentry.hive.testing.mode</name>
  <value>true</value>
</property>
```

Impala does not require this flag to be set.

> **Warning:** Cloudera strongly recommends against enabling this property in production. Use Sentry's testing mode only in test environments.

You canturn on Hive metastore security using the instructions in Cloudera Security. To secure the Hive metastore; see Hive Metastore Server Security Configuration on page 81.

Enabling Policy File Authorization for Impala

First, enable Sentry's policy file based authorization for Hive. For details, see Enabling Policy File Authorization for Hive on page 255.

See Enabling Sentry Authorization for Impala on page 257 for details on configuring Impala to work with Sentry policy files.

Enabling Sentry in Cloudera Search

See Enabling Sentry in Cloudera Search for CDH 5 on page 271 for details on securing Cloudera Search with Sentry.

Enabling Sentry Authorization for Impala

Authorization determines which users are allowed to access which resources, and what operations they are allowed to perform. In Impala 1.1 and higher, you use the Sentry open source project for authorization. Sentry adds a fine-grained authorization framework for Hadoop. By default (when authorization is not enabled), Impala does all read and write operations with the privileges of the `impala` user, which is suitable for a development/test environment but not for a secure production environment. When authorization is enabled, Impala uses the OS user ID of the user who runs `impala-shell` or other client program, and associates various privileges with each user.

> **Note:** Sentry is typically used in conjunction with Kerberos authentication, which defines which hosts are allowed to connect to each server. Using the combination of Sentry and Kerberos prevents malicious users from being able to connect by creating a named account on an untrusted machine. See Enabling Kerberos Authentication for Impala on page 96 for details about Kerberos authentication.

The Sentry Privilege Model

Privileges can be granted on different objects in the schema. Any privilege that can be granted is associated with a level in the object hierarchy. If a privilege is granted on a container object in the hierarchy, the child object automatically inherits it. This is the same privilege model as Hive and other database systems such as MySQL.

The object hierarchy covers Server, URI, Database, and Table. (The Table privileges apply to views as well; anywhere you specify a table name, you can specify a view name instead.) Currently, you cannot assign privileges at the partition or column level. The way you implement column-level or partition-level privileges is to create a view that queries just the relevant columns or partitions, and assign privileges to the view rather than the underlying table or tables.

A restricted set of privileges determines what you can do with each object:

SELECT privilege

Lets you read data from a table or view, for example with the `SELECT` statement, the `INSERT...SELECT` syntax, or `CREATE TABLE...LIKE`. Also required to issue the `DESCRIBE` statement or the `EXPLAIN` statement for a query against a particular table. Only objects for which a user has this privilege are shown in the output for `SHOW DATABASES` and `SHOW TABLES` statements. The `REFRESH` statement and `INVALIDATE METADATA` statements only access metadata for tables for which the user has this privilege.

INSERT privilege

Lets you write data to a table. Applies to the `INSERT` and `LOAD DATA` statements.

ALL privilege

Lets you create or modify the object. Required to run DDL statements such as `CREATE TABLE`, `ALTER TABLE`, or `DROP TABLE` for a table, `CREATE DATABASE` or `DROP DATABASE` for a database, or `CREATE VIEW`, `ALTER VIEW`, or `DROP VIEW` for a view. Also required for the URI of the "location" parameter for the `CREATE EXTERNAL TABLE` and `LOAD DATA` statements.

Privileges can be specified for a table or view before that object actually exists. If you do not have sufficient privilege to perform an operation, the error message does not disclose if the object exists or not.

Originally, privileges were encoded in a policy file, stored in HDFS. This mode of operation is still an option, but the emphasis of privilege management is moving towards being SQL-based. Although currently Impala does not have `GRANT` or `REVOKE` statements, Impala can make use of privileges assigned through `GRANT` and `REVOKE` statements done through Hive. The mode of operation with `GRANT` and `REVOKE` statements instead of the policy file requires that a special Sentry service be enabled; this service stores, retrieves, and manipulates privilege information stored inside the metastore database.

Starting the impalad Daemon with Sentry Authorization Enabled

To run the `impalad` daemon with authorization enabled, you add one or more options to the `IMPALA_SERVER_ARGS` declaration in the `/etc/default/impala` configuration file:

- The `-server_name` option turns on Sentry authorization for Impala. The authorization rules refer to a symbolic server name, and you specify the name to use as the argument to the `-server_name` option.
- If you specify just `-server_name`, Impala uses the Sentry service for authorization, relying on the results of `GRANT` and `REVOKE` statements issued through Hive. (This mode of operation is available in Impala 1.4.0 and higher.) Prior to Impala 1.4.0, or if you want to continue storing privilege rules in the policy file, also specify the `-authorization_policy_file` option as in the following item.
- Specifying the `-authorization_policy_file` option in addition to `-server_name` makes Impala read privilege information from a policy file, rather than from the metastore database. The argument to the `-authorization_policy_file` option specifies the HDFS path to the policy file that defines the privileges on different schema objects.

For example, you might adapt your `/etc/default/impala` configuration to contain lines like the following. To use the Sentry service rather than the policy file:

```
IMPALA_SERVER_ARGS=" \
-server_name=server1 \
...
```

Or to use the policy file, as in releases prior to Impala 1.4:

```
IMPALA_SERVER_ARGS=" \
-authorization_policy_file=/user/hive/warehouse/auth-policy.ini \
```

```
-server_name=server1 \
...
```

The preceding examples set up a symbolic name of `server1` to refer to the current instance of Impala. This symbolic name is used in the following ways:

- In an environment managed by Cloudera Manager, the server name is specified through **Impala > Service-Wide > Advanced > Server Name for Sentry Authorization** and **Hive > Service-Wide > Advanced > Server Name for Sentry Authorization**. The values must be the same for both, so that Impala and Hive can share the privilege rules. Restart the Impala and Hive services after setting or changing this value.
- In an environment not managed by Cloudera Manager, you specify this value for the `sentry.hive.server` property in the `sentry-site.xml` configuration file for Hive, as well as in the `-server_name` option for `impalad`.

  If the `impalad` daemon is not already running, start it as described in Starting Impala. If it is already running, restart it with the command `sudo /etc/init.d/impala-server restart`. Run the appropriate commands on all the nodes where `impalad` normally runs.
- If you use the mode of operation using the policy file, the rules in the `[roles]` section of the policy file refer to this same `server1` name. For example, the following rule sets up a role `report_generator` that lets users with that role query any table in a database named `reporting_db` on a node where the `impalad` daemon was started up with the `-server_name=server1` option:

```
[roles]
report_generator = server=server1->db=reporting_db->table=*->action=SELECT
```

When `impalad` is started with one or both of the `-server_name=server1` and `-authorization_policy_file` options, Impala authorization is enabled. If Impala detects any errors or inconsistencies in the authorization settings or the policy file, the daemon refuses to start.

Using Impala with the Sentry Service (CDH 5.1 or higher only)

When you use the Sentry service rather than the policy file, you set up privileges through GRANT and REVOKE statement in either Impala or Hive, then both components use those same privileges automatically. (Impala added the GRANT and REVOKE statements in Impala 2.0.0 / CDH 5.2.0.)

Hive already had GRANT and REVOKE statements prior to CDH 5.1, but those statements were not production-ready. CDH 5.1 is the first release where those statements use the Sentry framework and are considered GA level. If you used the Hive GRANT and REVOKE statements prior to CDH 5.1, you must set up these privileges with the CDH 5.1 versions of GRANT and REVOKE to take advantage of Sentry authorization.

For information about using the updated Hive GRANT and REVOKE statements, see Sentry service topic in the *CDH 5 Security Guide*.

Using Impala with the Sentry Policy File

The policy file is a file that you put in a designated location in HDFS, and is read during the startup of the `impalad` daemon when you specify both the `-server_name` and `-authorization_policy_file` startup options. It controls which objects (databases, tables, and HDFS directory paths) can be accessed by the user who connects to `impalad`, and what operations that user can perform on the objects.

> **Note:**
>
> In CDH 5 and higher, Cloudera recommends managing privileges through SQL statements, as described in Using Impala with the Sentry Service (CDH 5.1 or higher only) on page 259. If you are still using policy files, plan to migrate to the new approach some time in the future.

The location of the policy file is listed in the `auth-site.xml` configuration file. To minimize overhead, the security information from this file is cached by each `impalad` daemon and refreshed automatically, with a default interval of

5 minutes. After making a substantial change to security policies, restart all Impala daemons to pick up the changes immediately.

Policy File Location and Format

The policy file uses the familiar `.ini` format, divided into the major sections `[groups]` and `[roles]`. There is also an optional `[databases]` section, which allows you to specify a specific policy file for a particular database, as explained in Using Multiple Policy Files for Different Databases on page 264. Another optional section, `[users]`, allows you to override the OS-level mapping of users to groups; that is an advanced technique primarily for testing and debugging, and is beyond the scope of this document.

In the `[groups]` section, you define various categories of users and select which roles are associated with each category. The group and user names correspond to Linux groups and users on the server where the `impalad` daemon runs.

The group and user names in the `[groups]` section correspond to Linux groups and users on the server where the `impalad` daemon runs. When you access Impala through the `impalad` interpreter, for purposes of authorization, the user is the logged-in Linux user and the groups are the Linux groups that user is a member of. When you access Impala through the ODBC or JDBC interfaces, the user and password specified through the connection string are used as login credentials for the Linux server, and authorization is based on that user name and the associated Linux group membership.

In the `[roles]` section, you a set of roles. For each role, you specify precisely the set of privileges is available. That is, which objects users with that role can access, and what operations they can perform on those objects. This is the lowest-level category of security information; the other sections in the policy file map the privileges to higher-level divisions of groups and users. In the `[groups]` section, you specify which roles are associated with which groups. The group and user names correspond to Linux groups and users on the server where the `impalad` daemon runs. The privileges are specified using patterns like:

```
server=server_name->db=database_name->table=table_name->action=SELECT
server=server_name->db=database_name->table=table_name->action=CREATE
server=server_name->db=database_name->table=table_name->action=ALL
```

For the *server_name* value, substitute the same symbolic name you specify with the `impalad -server_name` option. You can use `*` wildcard characters at each level of the privilege specification to allow access to all such objects. For example:

```
server=impala-host.example.com->db=default->table=t1->action=SELECT
server=impala-host.example.com->db=*->table=*->action=CREATE
server=impala-host.example.com->db=*->table=audit_log->action=SELECT
server=impala-host.example.com->db=default->table=t1->action=*
```

When authorization is enabled, Impala uses the policy file as a *whitelist*, representing every privilege available to any user on any object. That is, only operations specified for the appropriate combination of object, role, group, and user are allowed; all other operations are not allowed. If a group or role is defined multiple times in the policy file, the last definition takes precedence.

To understand the notion of whitelisting, set up a minimal policy file that does not provide any privileges for any object. When you connect to an Impala node where this policy file is in effect, you get no results for `SHOW DATABASES`, and an error when you issue any `SHOW TABLES`, `USE database_name`, `DESCRIBE table_name`, `SELECT`, and or other statements that expect to access databases or tables, even if the corresponding databases and tables exist.

The contents of the policy file are cached, to avoid a performance penalty for each query. The policy file is re-checked by each `impalad` node every 5 minutes. When you make a non-time-sensitive change such as adding new privileges or new users, you can let the change take effect automatically a few minutes later. If you remove or reduce privileges, and want the change to take effect immediately, restart the `impalad` daemon on all nodes, again specifying the `-server_name` and `-authorization_policy_file` options so that the rules from the updated policy file are applied.

Examples of Policy File Rules for Security Scenarios

The following examples show rules that might go in the policy file to deal with various authorization-related scenarios. For illustration purposes, this section shows several very small policy files with only a few rules each. In your environment, typically you would define many roles to cover all the scenarios involving your own databases, tables, and applications, and a smaller number of groups, whose members are given the privileges from one or more roles.

A User with No Privileges

If a user has no privileges at all, that user cannot access any schema objects in the system. The error messages do not disclose the names or existence of objects that the user is not authorized to read.

This is the experience you want a user to have if they somehow log into a system where they are not an authorized Impala user. In a real deployment with a filled-in policy file, a user might have no privileges because they are not a member of any of the relevant groups mentioned in the policy file.

Examples of Privileges for Administrative Users

When an administrative user has broad access to tables or databases, the associated rules in the `[roles]` section typically use wildcards and/or inheritance. For example, in the following sample policy file, `db=*` refers to all databases and `db=*->table=*` refers to all tables in all databases.

Omitting the rightmost portion of a rule means that the privileges apply to all the objects that could be specified there. For example, in the following sample policy file, the `all_databases` role has all privileges for all tables in all databases, while the `one_database` role has all privileges for all tables in one specific database. The `all_databases` role does not grant privileges on URIs, so a group with that role could not issue a `CREATE TABLE` statement with a `LOCATION` clause. The `entire_server` role has all privileges on both databases and URIs within the server.

```
[groups]
supergroup = all_databases

[roles]
read_all_tables = server=server1->db=*->table=*->action=SELECT
all_tables = server=server1->db=*->table=*
all_databases = server=server1->db=*
one_database = server=server1->db=test_db
entire_server = server=server1
```

A User with Privileges for Specific Databases and Tables

If a user has privileges for specific tables in specific databases, the user can access those things but nothing else. They can see the tables and their parent databases in the output of `SHOW TABLES` and `SHOW DATABASES`, `USE` the appropriate databases, and perform the relevant actions (`SELECT` and/or `INSERT`) based on the table privileges. To actually create a table requires the `ALL` privilege at the database level, so you might define separate roles for the user that sets up a schema and other users or applications that perform day-to-day operations on the tables.

The following sample policy file shows some of the syntax that is appropriate as the policy file grows, such as the # comment syntax, \ continuation syntax, and comma separation for roles assigned to groups or privileges assigned to roles.

```
[groups]
cloudera = training_sysadmin, instructor
visitor = student

[roles]
training_sysadmin = server=server1->db=training, \
server=server1->db=instructor_private, \
server=server1->db=lesson_development
instructor = server=server1->db=training->table=*->action=*, \
server=server1->db=instructor_private->table=*->action=*, \
server=server1->db=lesson_development->table=lesson*
# This particular course is all about queries, so the students can SELECT but not INSERT
 or CREATE/DROP.
student = server=server1->db=training->table=lesson_*->action=SELECT
```

Privileges for Working with External Data Files

When data is being inserted through the `LOAD DATA` statement, or is referenced from an HDFS location outside the normal Impala database directories, the user also needs appropriate permissions on the URIs corresponding to those HDFS locations.

In this sample policy file:

- The `external_table` role lets us insert into and query the Impala table, `external_table.sample`.
- The `staging_dir` role lets us specify the HDFS path `/user/cloudera/external_data` with the `LOAD DATA` statement. Remember, when Impala queries or loads data files, it operates on all the files in that directory, not just a single file, so any Impala `LOCATION` parameters refer to a directory rather than an individual file.
- We included the IP address and port of the Hadoop name node in the HDFS URI of the `staging_dir` rule. We found those details in `/etc/hadoop/conf/core-site.xml`, under the `fs.default.name` element. That is what we use in any roles that specify URIs (that is, the locations of directories in HDFS).
- We start this example after the table `external_table.sample` is already created. In the policy file for the example, we have already taken away the `external_table_admin` role from the `cloudera` group, and replaced it with the lesser-privileged `external_table` role.
- We assign privileges to a subdirectory underneath `/user/cloudera` in HDFS, because such privileges also apply to any subdirectories underneath. If we had assigned privileges to the parent directory `/user/cloudera`, it would be too likely to mess up other files by specifying a wrong location by mistake.
- The `cloudera` under the `[groups]` section refers to the `cloudera` group. (In the demo VM used for this example, there is a `cloudera` user that is a member of a `cloudera` group.)

Policy file:

```
[groups]
cloudera = external_table, staging_dir

[roles]
external_table_admin = server=server1->db=external_table
external_table = server=server1->db=external_table->table=sample->action=*
staging_dir =
server=server1->uri=hdfs://127.0.0.1:8020/user/cloudera/external_data->action=*
```

`impala-shell` session:

```
[localhost:21000] > use external_table;
Query: use external_table
[localhost:21000] > show tables;
Query: show tables
Query finished, fetching results ...
+--------+
| name   |
+--------+
| sample |
+--------+
Returned 1 row(s) in 0.02s

[localhost:21000] > select * from sample;
Query: select * from sample
Query finished, fetching results ...
+-----+
| x   |
+-----+
| 1   |
| 5   |
| 150 |
+-----+
Returned 3 row(s) in 1.04s

[localhost:21000] > load data inpath '/user/cloudera/external_data' into table sample;
Query: load data inpath '/user/cloudera/external_data' into table sample
Query finished, fetching results ...
+----------------------------------------------------------+
| summary                                                  |
```

```
+-----------------------------------------------------------+
| Loaded 1 file(s). Total files in destination location: 2 |
+-----------------------------------------------------------+
Returned 1 row(s) in 0.26s
[localhost:21000] > select * from sample;
Query: select * from sample
Query finished, fetching results ...
+-------+
| x     |
+-------+
| 2     |
| 4     |
| 6     |
| 8     |
| 64738 |
| 49152 |
| 1     |
| 5     |
| 150   |
+-------+
Returned 9 row(s) in 0.22s

[localhost:21000] > load data inpath '/user/cloudera/unauthorized_data' into table
sample;
Query: load data inpath '/user/cloudera/unauthorized_data' into table sample
ERROR: AuthorizationException: User 'cloudera' does not have privileges to access:
hdfs://127.0.0.1:8020/user/cloudera/unauthorized_data
```

Controlling Access at the Column Level through Views

If a user has `SELECT` privilege for a view, they can query the view, even if they do not have any privileges on the underlying table. To see the details about the underlying table through `EXPLAIN` or `DESCRIBE FORMATTED` statements on the view, the user must also have `SELECT` privilege for the underlying table.

> **Important:**
>
> The types of data that are considered sensitive and confidential differ depending on the jurisdiction the type of industry, or both. For fine-grained access controls, set up appropriate privileges based on all applicable laws and regulations.
>
> Be careful using the `ALTER VIEW` statement to point an existing view at a different base table or a new set of columns that includes sensitive or restricted data. Make sure that any users who have `SELECT` privilege on the view do not gain access to any additional information they are not authorized to see.

The following example shows how a system administrator could set up a table containing some columns with sensitive information, then create a view that only exposes the non-confidential columns.

```
[localhost:21000] > create table sensitive_info
                  > (
                  >   name string,
                  >   address string,
                  >   credit_card string,
                  >   taxpayer_id string
                  > );
[localhost:21000] > create view name_address_view as select name, address from
sensitive_info;
```

Then the following policy file specifies read-only privilege for that view, without authorizing access to the underlying table:

```
[groups]
cloudera = view_only_privs

[roles]
view_only_privs = server=server1->db=reports->table=name_address_view->action=SELECT
```

Thus, a user with the `view_only_privs` role could access through Impala queries the basic information but not the sensitive information, even if both kinds of information were part of the same data file.

You might define other views to allow users from different groups to query different sets of columns.

Separating Administrator Responsibility from Read and Write Privileges

Remember that to create a database requires full privilege on that database, while day-to-day operations on tables within that database can be performed with lower levels of privilege on specific table. Thus, you might set up separate roles for each database or application: an administrative one that could create or drop the database, and a user-level one that can access only the relevant tables.

For example, this policy file divides responsibilities between users in 3 different groups:

- Members of the `supergroup` group have the `training_sysadmin` role and so can set up a database named `training`.
- Members of the `cloudera` group have the `instructor` role and so can create, insert into, and query any tables in the `training` database, but cannot create or drop the database itself.
- Members of the `visitor` group have the `student` role and so can query those tables in the `training` database.

```
[groups]
supergroup = training_sysadmin
cloudera = instructor
visitor = student

[roles]
training_sysadmin = server=server1->db=training
instructor = server=server1->db=training->table=*->action=*
student = server=server1->db=training->table=*->action=SELECT
```

Using Multiple Policy Files for Different Databases

For an Impala cluster with many databases being accessed by many users and applications, it might be cumbersome to update the security policy file for each privilege change or each new database, table, or view. You can allow security to be managed separately for individual databases, by setting up a separate policy file for each database:

- Add the optional `[databases]` section to the main policy file.
- Add entries in the `[databases]` section for each database that has its own policy file.
- For each listed database, specify the HDFS path of the appropriate policy file.

For example:

```
[databases]
# Defines the location of the per-DB policy files for the 'customers' and 'sales'
databases.
customers = hdfs://ha-nn-uri/etc/access/customers.ini
sales = hdfs://ha-nn-uri/etc/access/sales.ini
```

To enable URIs in per-DB policy files, add the following string in the Cloudera Manager field **Impala Service Environment Advanced Configuration Snippet (Safety Valve)**:

```
JAVA_TOOL_OPTIONS="-Dsentry.allow.uri.db.policyfile=true"
```

> **Important:** Enabling URIs in per-DB policy files introduces a security risk by allowing the owner of the db-level policy file to grant himself/herself load privileges to anything the `impala` user has read permissions for in HDFS (including data in other databases controlled by different db-level policy files).

Setting Up Schema Objects for a Secure Impala Deployment

Remember that in your role definitions, you specify privileges at the level of individual databases and tables, or all databases or all tables within a database. To simplify the structure of these rules, plan ahead of time how to name your schema objects so that data with different authorization requirements is divided into separate databases.

If you are adding security on top of an existing Impala deployment, remember that you can rename tables or even move them between databases using the `ALTER TABLE` statement. In Impala, creating new databases is a relatively inexpensive operation, basically just creating a new directory in HDFS.

You can also plan the security scheme and set up the policy file before the actual schema objects named in the policy file exist. Because the authorization capability is based on whitelisting, a user can only create a new database or table if the required privilege is already in the policy file: either by listing the exact name of the object being created, or a * wildcard to match all the applicable objects within the appropriate container.

Privilege Model and Object Hierarchy

Privileges can be granted on different objects in the schema. Any privilege that can be granted is associated with a level in the object hierarchy. If a privilege is granted on a container object in the hierarchy, the child object automatically inherits it. This is the same privilege model as Hive and other database systems such as MySQL.

The kinds of objects in the schema hierarchy are:

```
Server
URI
Database
  Table
```

The server name is specified by the `-server_name` option when `impalad` starts. Specify the same name for all `impalad` nodes in the cluster.

URIs represent the HDFS paths you specify as part of statements such as `CREATE EXTERNAL TABLE` and `LOAD DATA`. Typically, you specify what look like UNIX paths, but these locations can also be prefixed with `hdfs://` to make clear that they are really URIs. To set privileges for a URI, specify the name of a directory, and the privilege applies to all the files in that directory and any directories underneath it.

There are not separate privileges for individual table partitions or columns. To specify read privileges at this level, you create a view that queries specific columns and/or partitions from a base table, and give `SELECT` privilege on the view but not the underlying table. See Overview of Impala Views for details about views in Impala.

URIs must start with either `hdfs://` or `file://`. If a URI starts with anything else, it will cause an exception and the policy file will be invalid. When defining URIs for HDFS, you must also specify the NameNode. For example:

```
data_read = server=server1->uri=file:///path/to/dir, \
server=server1->uri=hdfs://namenode:port/path/to/dir
```

Warning:

Because the NameNode host and port must be specified, Cloudera strongly recommends you use High Availability (HA). This ensures that the URI will remain constant even if the NameNode changes.

```
data_read = server=server1->uri=file:///path/to/dir,\
server=server1->uri=hdfs://ha-nn-uri/path/to/dir
```

Table 20: Valid privilege types and objects they apply to

Privilege	Object
INSERT	DB, TABLE
SELECT	DB, TABLE
ALL	SERVER, TABLE, DB, URI

Note:

Although this document refers to the `ALL` privilege, currently if you use the policy file mode, you do not use the actual keyword `ALL` in the policy file. When you code role entries in the policy file:

- To specify the `ALL` privilege for a server, use a role like `server=server_name`.
- To specify the `ALL` privilege for a database, use a role like `server=server_name->db=database_name`.
- To specify the `ALL` privilege for a table, use a role like `server=server_name->db=database_name->table=table_name->action=*`.

Operation	Scope	Privileges	URI	Others
EXPLAIN	TABLE	SELECT		
LOAD DATA	TABLE	INSERT	URI	
CREATE DATABASE	SERVER	ALL		
DROP DATABASE	DATABASE	ALL		
CREATE TABLE	DATABASE	ALL		
DROP TABLE	TABLE	ALL		
DESCRIBE TABLE	TABLE	SELECT/INSERT		
ALTER TABLE .. ADD COLUMNS	TABLE	ALL		
ALTER TABLE .. REPLACE COLUMNS	TABLE	ALL		
ALTER TABLE .. CHANGE column	TABLE	ALL		
ALTER TABLE .. RENAME	TABLE	ALL		
ALTER TABLE .. SET TBLPROPERTIES	TABLE	ALL		
ALTER TABLE .. SET FILEFORMAT	TABLE	ALL		
ALTER TABLE .. SET LOCATION	TABLE	ALL	URI	
ALTER TABLE .. ADD PARTITION	TABLE	ALL		
ALTER TABLE .. ADD PARTITION location	TABLE	ALL	URI	
ALTER TABLE .. DROP PARTITION	TABLE	ALL		
ALTER TABLE .. PARTITION SET FILEFORMAT	TABLE	ALL		
ALTER TABLE .. SET SERDEPROPERTIES	TABLE	ALL		
CREATE VIEW	DATABASE; SELECT on TABLE	ALL		SELECT on TABLE

Operation	Scope	Privileges	URI	Others
DROP VIEW	VIEW/TABLE	ALL		
ALTER VIEW	You need ALL privilege on the named view and the parent database, plus SELECT privilege for any tables or views referenced by the view query. Once the view is created or altered by a high-privileged system administrator, it can be queried by a lower-privileged user who does not have full query privileges for the base tables. (This is how you implement column-level security.)	ALL, SELECT		
ALTER TABLE .. SET LOCATION	TABLE	ALL	URI	
CREATE EXTERNAL TABLE	Database (ALL), URI (SELECT)	ALL, SELECT		
SELECT	TABLE	SELECT		
USE <dbName>	Any			
CREATE FUNCTION	SERVER	ALL		
DROP FUNCTION	SERVER	ALL		
REFRESH <table name>	TABLE	SELECT/INSERT		
INVALIDATE METADATA	SERVER	ALL		
INVALIDATE METADATA <table name>	TABLE	SELECT/INSERT		
COMPUTE STATS	TABLE	ALL		
SHOW TABLE STATS, SHOW PARTITIONS	TABLE	SELECT/INSERT		
SHOW COLUMN STATS	TABLE	SELECT/INSERT		
SHOW FUNCTIONS	DATABASE	SELECT		
SHOW TABLES		No special privileges needed to issue the statement, but only shows objects you are authorized for		
SHOW DATABASES, SHOW SCHEMAS		No special privileges needed to issue the statement, but only shows objects you are authorized for		

Debugging Failed Sentry Authorization Requests

Sentry logs all facts that lead up to authorization decisions at the debug level. If you do not understand why Sentry is denying access, the best way to debug is to temporarily turn on debug logging:

- In Cloudera Manager, add `log4j.logger.org.apache.sentry=DEBUG` to the logging settings for your service through the corresponding **Logging Safety Valve** field for the Impala, Hive Server 2, or Solr Server services.
- On systems not managed by Cloudera Manager, add `log4j.logger.org.apache.sentry=DEBUG` to the `log4j.properties` file on each host in the cluster, in the appropriate configuration directory for each service.

Specifically, look for exceptions and messages such as:

```
FilePermission server..., RequestPermission server...., result [true|false]
```

which indicate each evaluation Sentry makes. The `FilePermission` is from the policy file, while `RequestPermission` is the privilege required for the query. A `RequestPermission` will iterate over all appropriate `FilePermission` settings until a match is found. If no matching privilege is found, Sentry returns `false` indicating "Access Denied" .

Managing Sentry for Impala through Cloudera Manager

To enable the Sentry service for Impala and Hive, set **Hive/Impala > Service-Wide > Sentry Service** parameter to the Sentry service. Then restart Impala and Hive. Simply adding Sentry service as a dependency and restarting enables Impala and Hive to use the Sentry service.

To set the server name to use when granting server level privileges, set the **Hive > Service-Wide > Advanced > Server Name for Sentry Authorization** parameter. When using Sentry with the Hive Metastore, you can specify the list of users that are allowed to bypass Sentry Authorization in Hive Metastore using **Hive > Service-Wide > Security > Bypass Sentry Authorization Users**. These are usually service users that already ensure all activity has been authorized.

> **Note:** The **Hive/Impala > Service-Wide > Policy File Based Sentry** tab contains parameters only relevant to configuring Sentry using policy files. In particular, make sure that **Enable Sentry Authorization using Policy Files** parameter is unchecked when using the Sentry service. Cloudera Manager throws a validation error if you attempt to configure the Sentry service and policy file at the same time.

The DEFAULT Database in a Secure Deployment

Because of the extra emphasis on granular access controls in a secure deployment, you should move any important or sensitive information out of the `DEFAULT` database into a named database whose privileges are specified in the policy file. Sometimes you might need to give privileges on the `DEFAULT` database for administrative reasons; for example, as a place you can reliably specify with a `USE` statement when preparing to drop a database.

Enabling Sentry Authorization for Search

Sentry enables role-based, fine-grained authorization for Cloudera Search. Sentry can apply a range of restrictions to various tasks, such accessing data or creating collections. These restrictions are consistently applied, regardless of the way users attempt to complete actions. For example, restricting access to data in a collection restricts that access whether queries come from the command line, from a browser, or through the admin console.

Follow the instructions below to configure Sentry under CDH 4.5 or later or CDH 5. Sentry is included in the Search installation.

> **Note:** Sentry for Search depends on Kerberos authentication. For additional information on using Kerberos with Search, see Search Authentication on page 104.

Note that this document is for configuring Sentry for Cloudera Search. For information about alternate ways to configure Sentry or for information about installing Sentry for other services, see:

- Setting Up Search Authorization with Sentry for instructions for using Cloudera Manager 4 to install and configure Search Authorization with Sentry.
- Impala Security for instructions on using Impala with Sentry.
- Sentry Installation to install the version of Sentry that was provided with CDH 4.
- Sentry Installation to install the version of Sentry that was provided with CDH 5.

Roles and Collection-Level Privileges

Sentry uses a role-based privilege model. A role is a set of rules for accessing a given Solr collection. Access to each collection is governed by privileges: `Query`, `Update`, or `All (*)`.

For example, a rule for the `Query` privilege on collection `logs` would be formulated as follows:

```
collection=logs->action=Query
```

A role can contain multiple such rules, separated by commas. For example the `engineer_role` might contain the `Query` privilege for `hive_logs` and `hbase_logs` collections, and the `Update` privilege for the `current_bugs` collection. You would specify this as follows:

```
engineer_role = collection=hive_logs->action=Query, collection=hbase_logs->action=Query,
 collection=current_bugs->action=Update
```

Users and Groups

- Auseris an entity that is permitted by the Kerberos authentication system to access the Search service.
- Agroupconnects the authentication system with the authorization system. It is a set of one or more users who have been granted one or more authorization roles. Sentry allows a set of roles to be configured for a group.
- A configured group providerdetermines a user's affiliation with a group. The current release supports HDFS-backed groups and locally configured groups. For example,

```
dev_ops = dev_role, ops_role
```

Here the group `dev_ops` is granted the roles `dev_role` and `ops_role`. The members of this group can complete searches that are allowed by these roles.

User to Group Mapping

You can configure Sentry to use either Hadoop groups or groups defined in the policy file.

> **Important:** You can use either Hadoop groups or local groups, but not both at the same time. Use local groups if you want to do a quick proof-of-concept. For production, use Hadoop groups.

To configure Hadoop groups:

Set the `sentry.provider` property in `sentry-site.xml` to `org.apache.sentry.provider.file.HadoopGroupResourceAuthorizationProvider`.

> **Note:** Note that, by default, this uses local shell groups. See the Group Mapping section of the HDFS Permissions Guide for more information.

OR

To configure local groups:

1. Define local groups in a `[users]` section of the Sentry Configuration File on page 271, `sentry-site.xml`. For example:

```
[users]
user1 = group1, group2, group3
user2 = group2, group3
```

2. In `sentry-site.xml`, set `search.sentry.provider` as follows:

```
<property>
    <name>sentry.provider</name>
   <value>org.apache.sentry.provider.file.LocalGroupResourceAuthorizationProvider</value>

  </property>
```

Setup and Configuration

This release of Sentry stores the configuration as well as privilege policies in files. The `sentry-site.xml` file contains configuration options such as privilege policy file location. The Policy File on page 270 contains the privileges and groups. It has a `.ini` file format and should be stored on HDFS.

Sentry is automatically installed when you install Cloudera Search for CDH or Cloudera Search 1.1.0 or later.

Policy File

The sections that follow contain notes on creating and maintaining the policy file.

> **Warning:** An invalid configuration disables all authorization while logging an exception.

Storing the Policy File

Considerations for storing the policy file(s) include:

1. Replication count - Because the file is read for each query, you should increase this; 10 is a reasonable value.
2. Updating the file - Updates to the file are only reflected when the Solr process is restarted.

Defining Roles

Keep in mind that role definitions are not cumulative; the newer definition replaces the older one. For example, the following results in `role1` having `privilege2`, not `privilege1` and `privilege2`.

```
role1 = privilege1
role1 = privilege2
```

Sample Configuration

This section provides a sample configuration.

> **Note:** Sentry with CDH Search does not support multiple policy files. Other implementations of Sentry such as Sentry for Hive do support different policy files for different databases, but Sentry for CDH Search has no such support for multiple policies.

Policy File

The following is an example of a CDH Search policy file. The `sentry-provider.ini` would exist in an HDFS location such as `hdfs://ha-nn-uri/user/solr/sentry/sentry-provider.ini`.

> **Note:** Use separate policy files for each Sentry-enabled service. Using one file for multiple services results in each service failing on the other services' entries. For example, with a combined Hive and Search file, Search would fail on Hive entries and Hive would fail on Search entries.

sentry-provider.ini

```
[groups]
# Assigns each Hadoop group to its set of roles
engineer = engineer_role
ops = ops_role
dev_ops = engineer_role, ops_role
hbase_admin = hbase_admin_role

[roles]
# The following grants all access to source_code.
# "collection = source_code" can also be used as syntactic
# sugar for "collection = source_code->action=*"
engineer_role = collection = source_code->action=*

# The following imply more restricted access.
ops_role = collection = hive_logs->action=Query
dev_ops_role = collection = hbase_logs->action=Query

#give hbase_admin_role the ability to create/delete/modify the hbase_logs collection
hbase_admin_role = collection=admin->action=*, collection=hbase_logs->action=*
```

Sentry Configuration File

The following is an example of a `sentry-site.xml` file.

sentry-site.xml

```
<configuration>
  <property>
    <name>hive.sentry.provider</name>

<value>org.apache.sentry.provider.file.HadoopGroupResourceAuthorizationProvider</value>

  </property>

  <property>
    <name>sentry.solr.provider.resource</name>
    <value>/path/to/authz-provider.ini</value>
    <!--
        If the HDFS configuration files (core-site.xml, hdfs-site.xml)
        pointed to by SOLR_HDFS_CONFIG in /etc/default/solr
        point to HDFS, the path will be in HDFS;
        alternatively you could specify a full path,
        e.g.:hdfs://namenode:port/path/to/authz-provider.ini
    -->
  </property>
```

Enabling Sentry in Cloudera Search for CDH 5

Enabling Sentry is achieved by adding two properties to `/etc/default/solr`. If your Search installation is managed by Cloudera Manager, then these properties are added automatically. If your Search installation is not managed by Cloudera Manager, you must make these changes yourself. The variable `SOLR_AUTHORIZATION_SENTRY_SITE` specifies the path to `sentry-site.xml`. The variable `SOLR_AUTHORIZATION_SUPERUSER` specifies the first part of `SOLR_KERBEROS_PRINCIPAL`. This is `solr` for the majority of users, as `solr` is the default. Settings are of the form:

```
SOLR_AUTHORIZATION_SENTRY_SITE=/location/to/sentry-site.xml
SOLR_AUTHORIZATION_SUPERUSER=solr
```

To enable sentry collection-level authorization checking on a new collection, the instancedir for the collection must use a modified version of `solrconfig.xml` with Sentry integration. The command `solrctl instancedir --generate` generates two versions of `solrconfig.xml`: the standard `solrconfig.xml` without sentry integration,

and the sentry-integrated version called `solrconfig.xml.secure`. To use the sentry-integrated version, replace `solrconfig.xml` with `solrconfig.xml.secure` before creating the instancedir.

If you have an existing collection using the standard `solrconfig.xml` called `foo` and an instancedir of the same name, perform the following steps:

```
# generate a fresh instancedir
solrctl instancedir --generate foosecure
# download the existing instancedir from ZK into subdirectory foo
solrctl instancedir --get foo foo
# replace the existing solrconfig.xml with the sentry-enabled one
cp foosecure/conf/solrconfig.xml.secure foo/conf/solrconfig.xml
# update the instancedir in ZK
solrctl instancedir --update foo foo
# reload the collection
solrctl collection --reload foo
```

If you have an existing collection using a version of `solrconfig.xml` that you have modified, contact Support for assistance.

Providing Document-Level Security Using Sentry

For role-based access control of a collection, an administrator modifies a Sentry role so it has query, update, or administrative access, as described above.

Collection-level authorization is useful when the access control requirements for the documents in the collection are the same, but users may want to restrict access to a subset of documents in a collection. This finer-grained restriction could be achieved by defining separate collections for each subset, but this is difficult to manage, requires duplicate documents for each collection, and requires that these documents be kept synchronized.

Document-level access control solves this issue by associating authorization tokens with each document in the collection. This enables granting Sentry roles access to sets of documents in a collection.

Document-Level Security Model

Document-level security depends on a chain of relationships between users, groups, roles, and documents.

- Users are assigned to groups.
- Groups are assigned to roles.
- Roles are stored as "authorization tokens" in a specified field in the documents.

Document-level security supports restricting which documents can be viewed by which users. Access is provided by adding roles as "authorization tokens" to a specified document field. Conversely, access is implicitly denied by omitting roles from the specified field. In other words, in a document-level security enabled environment, a user might submit a query that matches a document; if the user is not part of a group that has a role has been granted access to the document, the result is not returned.

For example, Alice might belong to the administrators group. The administrators group may belong to the doc-mgmt role. A document could be ingested and the doc-mgmt role could be added at ingest time. In such a case, if Alice submitted a query that matched the document, Search would return the document, since Alice is then allowed to see any document with the "doc-mgmt" authorization token.

Similarly, Bob might belong to the guests group. The guests group may belong to the public-browser role. If Bob tried the same query as Alice, but the document did not have the public-browser role, Search would not return the result because Bob does not belong to a group that is associated with a role that has access.

Note that collection-level authorization rules still apply, if enabled. Even if Alice is able to view a document given document-level authorization rules, if she is not allowed to query the collection, the query will fail.

Roles are typically added to documents when those documents are ingested, either via the standard Solr APIs or, if using morphlines, the `setValues` morphline command.

Enabling Document-Level Security

Cloudera Search supports document-level security in Search for CDH 5.1 and later. Document-level security is disabled by default, so the first step in using document-level security is to enable the feature by modifying the `solrconfig.xml.secure` file. Remember to replace the `solrconfig.xml` with this file, as described in Enabling Sentry in Cloudera Search for CDH 5 on page 271.

To enable document-level security, change `solrconfig.xml.secure`. The default file contents are as follows:

```
<searchComponent name="queryDocAuthorization">
    <!-- Set to true to enabled document-level authorization -->

    <bool name="enabled">false</bool>


    <!-- Field where the auth tokens are stored in the document -->
    <str name="sentryAuthField">sentry_auth</str>


    <!-- Auth token defined to allow any role to access the document.
         Uncomment to enable. -->

    <!--<str name="allRolesToken">*</str>-->

</searchComponent>
```

- The `enabled` Boolean determines whether document-level authorization is enabled. To enable document level security, change this setting to `true`.
- The `sentryAuthField` string specifies the name of the field that is used for storing authorization information. You can use the default setting of `sentry_auth` or you can specify some other string that you will use for assigning values on ingest.

  > **Note:** This field must exist as an explicit or dynamic field in the schema. `sentry_auth` exists in the default `schema.xml`.

- The `allRolesToken` string represents a special token defined to allow any role access to the document. By default, this feature is disabled. To enable this feature, uncomment the specification and specify the token. This token should be different from the name of any sentry role to avoid collision. By default it is "*". This feature is useful when first configuring document level security or it can be useful in granting all roles access to a document when the set of roles may change. See Best Practices for additional information.

Best Practices

Using `allRolesToken`

You may want to grant every user that belongs to a role access to certain documents. One way to accomplish this is to specify all known roles in the document, but this requires updating or re-indexing the document if you add a new role. Alternatively, an `allUser` role, specified in the Sentry `.ini` file, could contain all valid groups, but this role would need to be updated every time a new group was added to the system. Instead, specifying `allRolesToken` allows any user that belongs to a valid role to access the document. This access requires no updating as the system evolves.

In addition, `allRolesToken` may be useful for transitioning a deployment to use document-level security. Instead of having to define all the roles upfront, all the documents can be specified with `allRolesToken` and later modified as the roles are defined.

Consequences of Document-Level Authorization Only Affecting Queries

Document-level security does not prevent users from modifying documents or performing other update operations on the collection. Update operations are only governed by collection-level authorization.

Document-level security can be used to prevent documents being returned in query results. If users are not granted access to a document, those documents are not returned even if that user submits a query that matches those documents. This does not have affect attempted updates.

Consequently, it is possible for a user to not have access to a set of documents based on document-level security, but to still be able to modify the documents via their collection-level authorization update rights. This means that a user can delete all documents in the collection. Similarly, a user might modify all documents, adding their authorization token to each one. After such a modification, the user could access any document via querying. Therefore, if you are restricting access using document-level security, consider granting collection-level update rights only to those users you trust and assume they will be able to access every document in the collection.

Limitations on Query Size

By default queries support up to 1024 Boolean clauses. As a result, queries containing more that 1024 clauses may cause errors. Because authorization information is added by Sentry as part of a query, using document-level security can increase the number of clauses. In the case where users belong to many roles, even simple queries can become quite large. If a query is too large, an error of the following form occurs:

```
org.apache.lucene.search.BooleanQuery$TooManyClauses: maxClauseCount is set to 1024
```

To change the supported number of clauses, edit the `maxBooleanClauses` setting in `solrconfig.xml`. For example, to allow 2048 clauses, you would edit the setting so it appears as follows:

```
<maxBooleanClauses>2048</maxBooleanClauses>
```

For `maxBooleanClauses` to be applied as expected, make any change to this value to all collections and then restart the service. You must make this change to all collections because this option modifies a global Lucene property, affecting all Solr cores. If different `solrconfig.xml` files have different values for this property, the effective value is determined per host, based on the first Solr core to be initialized.

Enabling Secure Impersonation

Secure Impersonation is a feature that allows a user to make requests as another user in a secure way. For example, to allow the following impersonations:

- User hue can make requests as any user from any host.
- User `foo` can make requests as any member of group `bar`, from `host1` or `host2`.

  Configure the following properties in `/etc/default/solr`:

```
SOLR_SECURITY_ALLOWED_PROXYUSERS=hue,foo
SOLR_SECURITY_PROXYUSER_hue_HOSTS=*
SOLR_SECURITY_PROXYUSER_hue_GROUPS=*
SOLR_SECURITY_PROXYUSER_foo_HOSTS=host1,host2
SOLR_SECURITY_PROXYUSER_foo_GROUPS=bar
```

`SOLR_SECURITY_ALLOWED_PROXYUSERS` lists all of the users allowed to impersonate. For a user x in `SOLR_SECURITY_ALLOWED_PROXYUSERS`, `SOLR_SECURITY_PROXYUSER_x_HOSTS` list the hosts x is allowed to connect from in order to impersonate, and `SOLR_SECURITY_PROXYUSERS_x_GROUPS` lists the groups that the users is allowed to impersonate members of. Both `GROUPS` and `HOSTS` support the wildcard * and both `GROUPS` and `HOSTS` must be defined for a specific user.

> **Note:** Cloudera Manager has its own management of secure impersonation for Hue. To add additional users for Secure Impersonation, use the environment variable safety value for Solr to set the environment variables as above. Be sure to include hue in `SOLR_SECURITY_ALLOWED_PROXYUSERS` if you want to use secure impersonation for hue.

Debugging Failed Sentry Authorization Requests

Sentry logs all facts that lead up to authorization decisions at the debug level. If you do not understand why Sentry is denying access, the best way to debug is to temporarily turn on debug logging:

- In Cloudera Manager, add `log4j.logger.org.apache.sentry=DEBUG` to the logging settings for your service through the corresponding **Logging Safety Valve** field for the Impala, Hive Server 2, or Solr Server services.
- On systems not managed by Cloudera Manager, add `log4j.logger.org.apache.sentry=DEBUG` to the `log4j.properties` file on each host in the cluster, in the appropriate configuration directory for each service.

Specifically, look for exceptions and messages such as:

```
FilePermission server..., RequestPermission server...., result [true|false]
```

which indicate each evaluation Sentry makes. The `FilePermission` is from the policy file, while `RequestPermission` is the privilege required for the query. A `RequestPermission` will iterate over all appropriate `FilePermission` settings until a match is found. If no matching privilege is found, Sentry returns `false` indicating "Access Denied" .

Appendix: Authorization Privilege Model for Search

The tables below refer to the request handlers defined in the generated `solrconfig.xml.secure`. If you are not using this configuration file, the below may not apply.

`admin` is a special collection in sentry used to represent administrative actions. A non-administrative request may only require privileges on the collection on which the request is being performed. This is called `collection1` in this appendix. An administrative request may require privileges on both the `admin` collection and `collection1`. This is denoted as `admincollection1` in the tables below.

Table 21: Privilege table for non-administrative request handlers

Request Handler	Required Privilege	Collections that Require Privilege
select	QUERY	collection1
query	QUERY	collection1
get	QUERY	collection1
browse	QUERY	collection1
tvrh	QUERY	collection1
clustering	QUERY	collection1
terms	QUERY	collection1
elevate	QUERY	collection1
analysis/field	QUERY	collection1
analysis/document	QUERY	collection1
update	UPDATE	collection1
update/json	UPDATE	collection1
update/csv	UPDATE	collection1

Table 22: Privilege table for collections admin actions

Collection Action	Required Privilege	Collections that Require Privilege
create	UPDATE	admin, collection1
delete	UPDATE	admin, collection1
reload	UPDATE	admin, collection1

Collection Action	Required Privilege	Collections that Require Privilege
`createAlias`	`UPDATE`	`admin`, `collection1` **Note:** `collection1` here refers to the name of the alias, not the underlying collection(s). For example, `http://YOUR-HOST:8983/solr/admin/collections?action=CREATEALIAS&name=collection1&collections=underlyingCollection`
`deleteAlias`	`UPDATE`	`admin`, `collection1` **Note:** `collection1` here refers to the name of the alias, not the underlying collection(s). For example, `http://YOUR-HOST:8983/solr/admin/collections?action=DELETEALIAS&name=collection1`
`syncShard`	`UPDATE`	`admin`, `collection1`
`splitShard`	`UPDATE`	`admin`, `collection1`
`deleteShard`	`UPDATE`	`admin`, `collection1`

Table 23: Privilege table for core admin actions

Collection Action	Required Privilege	Collections that Require Privilege
`create`	`UPDATE`	`admin`, `collection1`
`rename`	`UPDATE`	`admin`, `collection1`
`load`	`UPDATE`	`admin`, `collection1`
`unload`	`UPDATE`	`admin`, `collection1`
`status`	`UPDATE`	`admin`, `collection1`
`persist`	`UPDATE`	`admin`
`reload`	`UPDATE`	`admin`, `collection1`
`swap`	`UPDATE`	`admin`, `collection1`
`mergeIndexes`	`UPDATE`	`admin`, `collection1`
`split`	`UPDATE`	`admin`, `collection1`
`prepRecover`	`UPDATE`	`admin`, `collection1`
`requestRecover`	`UPDATE`	`admin`, `collection1`
`requestSyncShard`	`UPDATE`	`admin`, `collection1`
`requestApplyUpdates`	`UPDATE`	`admin`, `collection1`

Table 24: Privilege table for Info and AdminHandlers

Request Handler	Required Privilege	Collections that Require Privilege
`LukeRequestHandler`	`QUERY`	`admin`
`SystemInfoHandler`	`QUERY`	`admin`
`SolrInfoMBeanHandler`	`QUERY`	`admin`
`PluginInfoHandler`	`QUERY`	`admin`
`ThreadDumpHandler`	`QUERY`	`admin`
`PropertiesRequestHandler`	`QUERY`	`admin`
`LogginHandler`	`QUERY`, `UPDATE` (or `*`)	`admin`
`ShowFileRequestHandler`	`QUERY`	`admin`

Configuring HBase Authorization

After you have configured HBase authentication as described in the previous section, you must establish authorization rules for the resources that a client is allowed to access. HBase currently allows you to establish authorization rules at the table, column and cell-level. Cell-level authorization is fully supported since CDH 5.2.

Understanding HBase Access Levels

HBase access levels are granted independently of each other and allow for different types of operations at a given scope.

- **Read (R)** - can read data at the given scope
- **Write (W)** - can write data at the given scope
- **Execute (X)** - can execute coprocessor endpoints at the given scope
- **Create (C)** - can create tables or drop tables (even those they did not create) at the given scope
- **Admin (A)** - can perform cluster operations such as balancing the cluster or assigning regions at the given scope

The possible scopes are:

- **Superuser** - superusers can perform any operation available in HBase, to any resource. The user who runs HBase on your cluster is a superuser, as are any principals assigned to the configuration property `hbase.superuser` in `hbase-site.xml` on the HMaster.
- **Global** - permissions granted at `global` scope allow the admin to operate on all tables of the cluster.
- **Namespace** - permissions granted at `namespace` scope apply to all tables within a given namespace.
- **Table** - permissions granted at `table` scope apply to data or metadata within a given table.
- **ColumnFamily** - permissions granted at `ColumnFamily` scope apply to cells within that ColumnFamily.
- **Cell** - permissions granted at Cell scope apply to that exact cell coordinate. This allows for policy evolution along with data. To change an ACL on a specific cell, write an updated cell with new ACL to the precise coordinates of the original. If you have a multi-versioned schema and want to update ACLs on all visible versions, you'll need to write new cells for all visible versions. The application has complete control over policy evolution. The exception is `append` and `increment` processing. `Appends` and `increments` can carry an ACL in the operation. If one is included in the operation, then it will be applied to the result of the `append` or `increment`. Otherwise, the ACL of the existing cell being appended to or incremented is preserved.

The combination of access levels and scopes creates a matrix of possible access levels that can be granted to a user. In a production environment, it is useful to think of access levels in terms of what is needed to do a specific job. The following list describes appropriate access levels for some common types of HBase users. It is important not to grant more access than is required for a given user to perform their required tasks.

- **Superusers** - In a production system, only the HBase user should have superuser access. In a development environment, an administrator may need superuser access in order to quickly control and manage the cluster. However, this type of administrator should usually be a `Global Admin` rather than a superuser.
- **Global Admins** - A `global admin` can perform tasks and access every table in HBase. In a typical production environment, an admin should not have `Read` or `Write` permissions to data within tables.
  - A global admin with `Admin` permissions can perform cluster-wide operations on the cluster, such as balancing, assigning or unassigning regions, or calling an explicit major compaction. This is an operations role.
  - A global admin with `Create` permissions can create or drop any table within HBase. This is more of a DBA-type role.

  In a production environment, it is likely that different users will have only one of `Admin` and `Create` permissions.

  > **Warning:**
  >
  > In the current implementation, a `Global Admin` with `Admin` permission can grant himself `Read` and `Write` permissions on a table and gain access to that table's data. For this reason, only grant `Global Admin` permissions to trusted user who actually need them.
  >
  > Also be aware that a `Global Admin` with `Create` permission can perform a `Put` operation on the ACL table, simulating a `grant` or `revoke` and circumventing the authorization check for `Global Admin` permissions. This issue (but not the first one) is fixed in CDH 5.3 and higher, as well as CDH 5.2.1. It is not fixed in CDH 4.x or CDH 5.1.x.
  >
  > Due to these issues, be cautious with granting Global Admin privileges.

- **Table Admins** - A table admin can perform administrative operations only on that table. A table admin with `Create` permissions can create snapshots from that table or restore that table from a snapshot. A table admin with `Admin` permissions can perform operations such as splits or major compactions on that table.
- **Users** - Users can read or write data, or both. Users can also execute coprocessor endpoints, if given `Executable` permissions.

> **Important:**
>
> If you are using Kerberos principal names when setting ACLs for users, note that Hadoop uses only the first part (short) of the Kerberos principal when converting it to the user name. Hence, for the principal `ann/fully.qualified.domain.name@YOUR-REALM.COM`, HBase ACLs should only be set for user `ann`.

Table 25: Real-World Example of Access Levels

This table shows some typical job descriptions at a hypothetical company and the permissions they might require in order to get their jobs done using HBase.

Job Title	Scope	Permissions	Description
Senior Administrator	Global	Admin, Create	Manages the cluster and gives access to Junior Administrators.
Junior Administrator	Global	Create	Creates tables and gives access to Table Administrators.
Table Administrator	Table	Admin	Maintains a table from an operations point of view.

Job Title	Scope	Permissions	Description
Data Analyst	Table	Read	Creates reports from HBase data.
Web Application	Table	Read, Write	Puts data into HBase and uses HBase data to perform operations.

Further Reading

- Access Control Matrix
- Security - Apache HBase Reference Guide

Enable HBase Authorization

HBase authorization is built on top of the Coprocessors framework, specifically `AccessController Coprocessor`.

To enable HBase authorization, add the following properties to the `hbase-site.xml` file *on every HBase server host (Master or RegionServer)*:

```
<property>
     <name>hbase.security.authorization</name>
     <value>true</value>
</property>
<property>
     <name>hbase.coprocessor.master.classes</name>
     <value>org.apache.hadoop.hbase.security.access.AccessController</value>
</property>
<property>
     <name>hbase.coprocessor.region.classes</name>

<value>org.apache.hadoop.hbase.security.token.TokenProvider,org.apache.hadoop.hbase.security.access.AccessController</value>
</property>
```

Note: Once the Access Controller coprocessor is enabled, any user who uses the HBase shell will be subject to access control. Access control will also be in effect for native (Java API) client access to HBase.

Configure Access Control Lists for Authorization

Now that HBase has the security coprocessor enabled, you can set ACLs using the HBase shell. Start the HBase shell as usual.

Important:

The host running the shell must be configured with a keytab file as described in Configuring Kerberos Authentication for HBase.

The commands that control ACLs take the following form. Group names are prefixed with the @ symbol.

```
hbase> grant <user> <permissions> [ @<namespace> [ <table>[ <column family>[ <column
qualifier> ] ] ] ]   # grants permissions

hbase> revoke <user> <permissions> [ @<namespace> [ <table> [ <column family> [ <column
 qualifier> ] ] ]  # revokes permissions

hbase> user_permission <table>
                    # displays existing permissions
```

In the above commands, fields encased in <> are variables, and fields in `[]` are optional. The `permissions` variable must consist of zero or more character from the set "RWCA".

- `R` denotes read permissions, which is required to perform `Get`, `Scan`, or `Exists` calls in a given scope.
- `W` denotes write permissions, which is required to perform `Put`, `Delete`, `LockRow`, `UnlockRow`, `IncrementColumnValue`, `CheckAndDelete`, `CheckAndPut`, `Flush`, or `Compact` in a given scope.
- `X` denotes execute permissions, which is required to execute coprocessor endpoints.
- `C` denotes create permissions, which is required to perform `Create`, `Alter`, or `Drop` in a given scope.
- `A` denotes admin permissions, which is required to perform `Enable`, `Disable`, `Snapshot`, `Restore`, `Clone`, `Split`, `MajorCompact`, `Grant`, `Revoke`, and `Shutdown` in a given scope.

For example:

```
grant 'user1', 'RWC'
grant 'user2', 'RW', 'tableA'
grant 'user3', 'C', '@my_namespace'
```

Be sure to review the information in Understanding HBase Access Levels on page 277 to understand the implications of the different access levels.

Overview of Impala Security

Impala includes a fine-grained authorization framework for Hadoop, based on the Sentry open source project. Sentry authorization was added in Impala 1.1.0. Together with the Kerberos authentication framework, Sentry takes Hadoop security to a new level needed for the requirements of highly regulated industries such as healthcare, financial services, and government. Impala also includes an auditing capability; Impala generates the audit data, the Cloudera Navigator product consolidates the audit data from all nodes in the cluster, and Cloudera Manager lets you filter, visualize, and produce reports. The auditing feature was added in Impala 1.1.1.

The Impala security features have several objectives. At the most basic level, security prevents accidents or mistakes that could disrupt application processing, delete or corrupt data, or reveal data to unauthorized users. More advanced security features and practices can harden the system against malicious users trying to gain unauthorized access or perform other disallowed operations. The auditing feature provides a way to confirm that no unauthorized access occurred, and detect whether any such attempts were made. This is a critical set of features for production deployments in large organizations that handle important or sensitive data. It sets the stage for multi-tenancy, where multiple applications run concurrently and are prevented from interfering with each other.

The material in this section presumes that you are already familiar with administering secure Linux systems. That is, you should know the general security practices for Linux and Hadoop, and their associated commands and configuration files. For example, you should know how to create Linux users and groups, manage Linux group membership, set Linux and HDFS file permissions and ownership, and designate the default permissions and ownership for new files. You should be familiar with the configuration of the nodes in your Hadoop cluster, and know how to apply configuration changes or run a set of commands across all the nodes.

The security features are divided into these broad categories:

authorization

Which users are allowed to access which resources, and what operations are they allowed to perform? Impala relies on the open source Sentry project for authorization. By default (when authorization is not enabled), Impala does all read and write operations with the privileges of the `impala` user, which is suitable for a development/test environment but not for a secure production environment. When authorization is enabled, Impala uses the OS user ID of the user who runs `impala-shell` or other client program, and associates various privileges with each user. See Enabling Sentry Authorization for Impala on page 257 for details about setting up and managing authorization.

authentication

How does Impala verify the identity of the user to confirm that they really are allowed to exercise the privileges assigned to that user? Impala relies on the Kerberos subsystem for authentication. See Enabling Kerberos Authentication for Impala on page 96 for details about setting up and managing authentication.

auditing

What operations were attempted, and did they succeed or not? This feature provides a way to look back and diagnose whether attempts were made to perform unauthorized operations. You use this information to track down suspicious activity, and to see where changes are needed in authorization policies. The audit data produced by this feature is collected by the Cloudera Manager product and then presented in a user-friendly form by the Cloudera Manager product. See Auditing Impala Operations for details about setting up and managing auditing.

These other topics in the *Security Guide* cover how Impala integrates with security frameworks such as Kerberos, LDAP, and Sentry:

- Impala Authentication on page 96
- Enabling Sentry Authorization for Impala on page 257

Security Guidelines for Impala

The following are the major steps to harden a cluster running Impala against accidents and mistakes, or malicious attackers trying to access sensitive data:

- Secure the `root` account. The `root` user can tamper with the `impalad` daemon, read and write the data files in HDFS, log into other user accounts, and access other system services that are beyond the control of Impala.
- Restrict membership in the `sudoers` list (in the `/etc/sudoers` file). The users who can run the `sudo` command can do many of the same things as the `root` user.
- Ensure the Hadoop ownership and permissions for Impala data files are restricted.
- Ensure the Hadoop ownership and permissions for Impala log files are restricted.
- Ensure that the Impala web UI (available by default on port 25000 on each Impala node) is password-protected. See Impala Web User Interface for Debugging for details.
- Create a policy file that specifies which Impala privileges are available to users in particular Hadoop groups (which by default map to Linux OS groups). Create the associated Linux groups using the `groupadd` command if necessary.
- The Impala authorization feature makes use of the HDFS file ownership and permissions mechanism; for background information, see the CDH HDFS Permissions Guide. Set up users and assign them to groups at the OS level, corresponding to the different categories of users with different access levels for various databases, tables, and HDFS locations (URIs). Create the associated Linux users using the `useradd` command if necessary, and add them to the appropriate groups with the `usermod` command.
- Design your databases, tables, and views with database and table structure to allow policy rules to specify simple, consistent rules. For example, if all tables related to an application are inside a single database, you can assign privileges for that database and use the `*` wildcard for the table name. If you are creating views with different privileges than the underlying base tables, you might put the views in a separate database so that you can use the `*` wildcard for the database containing the base tables, while specifying the precise names of the individual views. (For specifying table or database names, you either specify the exact name or `*` to mean all the databases on a server, or all the tables and views in a database.)
- Enable authorization by running the `impalad` daemons with the `-server_name` and `-authorization_policy_file` options on all nodes. (The authorization feature does not apply to the `statestored` daemon, which has no access to schema objects or data files.)
- Set up authentication using Kerberos, to make sure users really are who they say they are.

Securing Impala Data and Log Files

One aspect of security is to protect files from unauthorized access at the filesystem level. For example, if you store sensitive data in HDFS, you specify permissions on the associated files and directories in HDFS to restrict read and write permissions to the appropriate users and groups.

If you issue queries containing sensitive values in the `WHERE` clause, such as financial account numbers, those values are stored in Impala log files in the Linux filesystem and you must secure those files also. For the locations of Impala log files, see Using Impala Logging.

All Impala read and write operations are performed under the filesystem privileges of the `impala` user. The `impala` user must be able to read all directories and data files that you query, and write into all the directories and data files for `INSERT` and `LOAD DATA` statements. At a minimum, make sure the `impala` user is in the `hive` group so that it can access files and directories shared between Impala and Hive. See User Account Requirements for more details.

Setting file permissions is necessary for Impala to function correctly, but is not an effective security practice by itself:

- The way to ensure that only authorized users can submit requests for databases and tables they are allowed to access is to set up Sentry authorization, as explained in Enabling Sentry Authorization for Impala on page 257. With authorization enabled, the checking of the user ID and group is done by Impala, and unauthorized access is blocked by Impala itself. The actual low-level read and write requests are still done by the `impala` user, so you must have appropriate file and directory permissions for that user ID.

- You must also set up Kerberos authentication, as described in Enabling Kerberos Authentication for Impala on page 96, so that users can only connect from trusted hosts. With Kerberos enabled, if someone connects a new host to the network and creates user IDs that match your privileged IDs, they will be blocked from connecting to Impala at all from that host.

Installation Considerations for Impala Security

Impala 1.1 comes set up with all the software and settings needed to enable security when you run the `impalad` daemon with the new security-related options (`-server_name` and `-authorization_policy_file`). You do not need to change any environment variables or install any additional JAR files. In a cluster managed by Cloudera Manager, you do not need to change any settings in Cloudera Manager.

Securing the Hive Metastore Database

It is important to secure the Hive metastore, so that users cannot access the names or other information about databases and tables the through the Hive client or by querying the metastore database. Do this by turning on Hive metastore security, using the instructions in the CDH 5 Security Guide for securing different Hive components:

- Secure the Hive Metastore.
- In addition, allow access to the metastore only from the HiveServer2 server, and then disable local access to the HiveServer2 server.

Securing the Impala Web User Interface

The instructions in this section presume you are familiar with the `.htpasswd` mechanism commonly used to password-protect pages on web servers.

Password-protect the Impala web UI that listens on port 25000 by default. Set up a `.htpasswd` file in the `$IMPALA_HOME` directory, or start both the `impalad` and `statestored` daemons with the `--webserver_password_file` option to specify a different location (including the filename).

This file should only be readable by the Impala process and machine administrators, because it contains (hashed) versions of passwords. The username / password pairs are not derived from Unix usernames, Kerberos users, or any other system. The `domain` field in the password file must match the domain supplied to Impala by the new command-line option `--webserver_authentication_domain`. The default is `mydomain.com`.

Impala also supports using HTTPS for secure web traffic. To do so, set `--webserver_certificate_file` to refer to a valid `.pem` SSL certificate file. Impala will automatically start using HTTPS once the SSL certificate has been read and validated. A `.pem` file is basically a private key, followed by a signed SSL certificate; make sure to concatenate both parts when constructing the `.pem` file.

If Impala cannot find or parse the `.pem` file, it prints an error message and quits.

Note:

If the private key is encrypted using a passphrase, Impala will ask for that passphrase on startup, which is not useful for a large cluster. In that case, remove the passphrase and make the `.pem` file readable only by Impala and administrators.

When you turn on SSL for the Impala web UI, the associated URLs change from `http://` prefixes to `https://`. Adjust any bookmarks or application code that refers to those URLs.

Configuring Secure Access for Impala Web Servers

Cloudera Manager supports two methods of authentication for secure access to the Impala Catalog Server, Daemon, and StateStoreweb servers: password-based authentication and TLS/SSL certificate authentication.

Authentication for the three types of daemons can be configured independently.

Configuring Password Authentication

1. Navigate to **Clusters** > **Impala Service** > **Configuration**.
2. Search for "password" using the Search box in the **Configuration** tab. This should display the password-related properties (Username and Password properties) for the Impala Daemon, StateStore, and Catalog Server. If there are multiple role groups configured for Impala Daemon instances, the search should display all of them.
3. Enter a username and password into these fields.
4. Click **Save Changes**, and restart the Impala service.

Now when you access the Web UI for the Impala Daemon, StateStore, or Catalog Server, you are asked to log in before access is granted.

Configuring TLS/SSL Certificate Authentication

1. Create or obtain an TLS/SSL certificate.
2. Place the certificate, in `.pem` format, on the hosts where the Impala Catalog Server and StateStore are running, and on each host where an Impala Daemon is running. It can be placed in any location (path) you choose. If all the Impala Daemons are members of the same role group, then the `.pem` file must have the same path on every host.
3. Navigate to **Clusters** > **Impala Service** > **Configuration**.
4. Search for "certificate" using the Search box in the **Configuration** tab. This should display the certificate file location properties for the Impala Catalog Server, Impala Daemon, and StateStore. If there are multiple role groups configured for Impala Daemon instances, the search should display all of them.
5. In the property fields, enter the full path name to the certificate file.
6. Click **Save Changes**, and restart the Impala service.

> **Important:** If Cloudera Manager cannot find the `.pem` file on the host for a specific role instance, that role will fail to start.

When you access the Web UI for the Impala Catalog Server, Impala Daemon, and StateStore, `https` will be used.

Miscellaneous Topics

This section comprises miscellaneous security guide topics that you may find useful once you have secured your cluster with authentication, encryption and authorization techniques.

Jsvc, Task Controller and Container Executor Programs

This section contains information about the following Hadoop security programs:

MRv1 and YARN: The jsvc Program

The `jsvc` program is part of the `bigtop-jsvc` package and installed in either `/usr/lib/bigtop-utils/jsvc` or `/usr/libexec/bigtop-utils/jsvc` depending on the particular Linux flavor.

`jsvc` (more info) is used to start the DataNode listening on low port numbers. Its entry point is the `SecureDataNodeStarter` class, which implements the Daemon interface that `jsvc` expects. `jsvc` is run as root, and calls the `SecureDataNodeStarter.init(...)` method while running as root. Once the `SecureDataNodeStarter` class has finished initializing, `jsvc` sets the effective UID to be the `hdfs` user, and then calls `SecureDataNodeStarter.start(...)`. `SecureDataNodeStarter` then calls the regular DataNode entry point, passing in a reference to the privileged resources it previously obtained.

MRv1 Only: The Linux TaskController Program

A setuid binary called `task-controller` is part of the `hadoop-0.20-mapreduce` package and is installed in either `/usr/lib/hadoop-0.20-mapreduce/sbin/Linux-amd64-64/task-controller` or `/usr/lib/hadoop-0.20-mapreduce/sbin/Linux-i386-32/task-controller`.

This `task-controller` program, which is used on MRv1 only, allows the TaskTracker to run tasks under the Unix account of the user who submitted the job in the first place. It is a setuid binary that must have a very specific set of permissions and ownership in order to function correctly. In particular, it must:

1. Be owned by root
2. Be owned by a group that contains only the user running the MapReduce daemons
3. Be setuid
4. Be group readable and executable

This corresponds to the ownership `root:mapred` and the permissions `4754`.

Here is the output of `ls` on a correctly-configured Task-controller:

```
-rwsr-xr-- 1 root mapred 30888 Mar 18 13:03 task-controller
```

The TaskTracker will check for this configuration on start up, and fail to start if the Task-controller is not configured correctly.

YARN Only: The Linux Container Executor Program

A setuid binary called `container-executor` is part of the `hadoop-yarn` package and is installed in `/usr/lib/hadoop-yarn/bin/container-executor`.

This `container-executor` program, which is used on YARN only and supported on GNU/Linux only, runs the containers as the user who submitted the application. It requires all user accounts to be created on the cluster nodes where the containers are launched. It uses a setuid executable that is included in the Hadoop distribution. The NodeManager uses this executable to launch and kill containers. The setuid executable switches to the user who has submitted the application and launches or kills the containers. For maximum security, this executor sets up restricted permissions and user/group ownership of local files and directories used by the containers such as the shared objects, jars,

intermediate files, log files, and so on. As a result, only the application owner and NodeManager can access any of the local files/directories including those localized as part of the distributed cache.

The `container-executor` program must have a very specific set of permissions and ownership in order to function correctly. In particular, it must:

1. Be owned by root
2. Be owned by a group that contains only the user running the YARN daemons
3. Be setuid
4. Be group readable and executable

This corresponds to the ownership `root:yarn` and the permissions `6050`.

```
---Sr-s--- 1 root yarn 91886 2012-04-01 19:54 container-executor
```

Important: Configuration changes to the Linux container executor could result in local NodeManager directories (such as `usercache`) being left with incorrect permissions. To avoid this, when making changes using either Cloudera Manager or the command line, first manually remove the existing NodeManager local directories from all configured local directories (`yarn.nodemanager.local-dirs`), and let the NodeManager recreate the directory structure.

Task-controller and Container-executor Error Codes

When you set up a secure cluster for the first time and debug problems with it, the `task-controller` or `container-executor` may encounter errors. These programs communicate these errors to the TaskTracker or NodeManager daemon via numeric error codes which will appear in the TaskTracker or NodeManager logs respectively (`/var/log/hadoop-mapreduce or /var/log/hadoop-yarn`). The following sections list the possible numeric error codes with descriptions of what they mean:

- MRv1 ONLY: Task-controller Error Codes on page 286
- YARN ONLY: Container-executor Error Codes on page 288

MRv1 ONLY: Task-controller Error Codes

The following table applies to the task-controller in MRv1.

Numeric Code	Name	Description
1	INVALID_ARGUMENT_NUMBER	• Incorrect number of arguments provided for the given task-controller command • Failure to initialize the job localizer
2	INVALID_USER_NAME	The user passed to the task-controller does not exist.
3	INVALID_COMMAND_PROVIDED	The task-controller does not recognize the command it was asked to execute.
4	SUPER_USER_NOT_ALLOWED_TO_RUN_TASKS	The user passed to the task-controller was the super user.
5	INVALID_TT_ROOT	The passed TaskTracker root does not match the configured TaskTracker root (`mapred.local.dir`), or does not exist.
6	SETUID_OPER_FAILED	Either could not read the local groups database, or could not set UID or GID

Numeric Code	Name	Description
7	UNABLE_TO_EXECUTE_TASK_SCRIPT	The task-controller could not execute the task launcher script.
8	UNABLE_TO_KILL_TASK	The task-controller could not kill the task it was passed.
9	INVALID_TASK_PID	The PID passed to the task-controller was negative or 0.
10	ERROR_RESOLVING_FILE_PATH	The task-controller could not resolve the path of the task launcher script file.
11	RELATIVE_PATH_COMPONENTS_IN_FILE_PATH	The path to the task launcher script file contains relative components (for example, "...").
12	UNABLE_TO_STAT_FILE	The task-controller did not have permission to stat a file it needed to check the ownership of.
13	FILE_NOT_OWNED_BY_TASKTRACKER	A file which the task-controller must change the ownership of has the wrong the ownership.
14	PREPARE_ATTEMPT_DIRECTORIES_FAILED	The `mapred.local.dir` is not configured, could not be read by the task-controller, or could not have its ownership secured.
15	INITIALIZE_JOB_FAILED	The task-controller could not get, stat, or secure the job directory or job working working directory.
16	PREPARE_TASK_LOGS_FAILED	The task-controller could not find or could not change the ownership of the task log directory to the passed user.
17	INVALID_TT_LOG_DIR	The `hadoop.log.dir` is not configured.
18	OUT_OF_MEMORY	The task-controller could not determine the job directory path or the task launcher script path.
19	INITIALIZE_DISTCACHEFILE_FAILED	Could not get a unique value for, stat, or the local distributed cache directory.
20	INITIALIZE_USER_FAILED	Could not get, stat, or secure the per-user task tracker directory.
21	UNABLE_TO_BUILD_PATH	The task-controller could not concatenate two paths, most likely because it ran out of memory.
22	INVALID_TASKCONTROLLER_PERMISSIONS	The task-controller binary does not have the correct permissions set. See Information about Other Hadoop Security Programs.
23	PREPARE_JOB_LOGS_FAILED	The task-controller could not find or could not change the ownership of the job log directory to the passed user.
24	INVALID_CONFIG_FILE	The taskcontroller.cfg file is missing, malformed, or has incorrect permissions.

Numeric Code	Name	Description
255	Unknown Error	There are several causes for this error. Some common causes are: • There are user accounts on your cluster that have a user ID less than the value specified for the `min.user.id` property in the `taskcontroller.cfg` file. The default value is 1000 which is appropriate on Ubuntu systems, but may not be valid for your operating system. For information about setting `min.user.id` in the `taskcontroller.cfg` file, see this step. • Jobs won't run and the TaskTracker is unable to create a Hadoop logs directory. For more information, see (MRv1 Only) Jobs will not run and TaskTracker is unable to create a Hadoop logs directory. on page 145. • This error is often caused by previous errors; look earlier in the log file for possible causes.

YARN ONLY: Container-executor Error Codes

The codes in the table apply to the container-executor in YARN, but are used by the LinuxContainerExecutor only.

Numeric Code	Name	Description
1	INVALID_ARGUMENT_NUMBER	• Incorrect number of arguments provided for the given task-controller command • Failure to initialize the container localizer
2	INVALID_USER_NAME	The user passed to the task-controller does not exist.
3	INVALID_COMMAND_PROVIDED	The container-executor does not recognize the command it was asked to run.
5	INVALID_NM_ROOT	The passed NodeManager root does not match the configured NodeManager root (`yarn.nodemanager.local-dirs`), or does not exist.
6	SETUID_OPER_FAILED	Either could not read the local groups database, or could not set UID or GID
7	UNABLE_TO_EXECUTE_CONTAINER_SCRIPT	The container-executor could not run the container launcher script.
8	UNABLE_TO_SIGNAL_CONTAINER	The container-executor could not signal the container it was passed.
9	INVALID_CONTAINER_PID	The PID passed to the container-executor was negative or 0.

Numeric Code	Name	Description
18	OUT_OF_MEMORY	The container-executor couldn't allocate enough memory while reading the container-executor.cfg file, or while getting the paths for the container launcher script or credentials files.
20	INITIALIZE_USER_FAILED	Couldn't get, stat, or secure the per-user NodeManager directory.
21	UNABLE_TO_BUILD_PATH	The container-executor couldn't concatenate two paths, most likely because it ran out of memory.
22	INVALID_CONTAINER_EXEC_PERMISSIONS	The container-executor binary does not have the correct permissions set. See Information about Other Hadoop Security Programs.
24	INVALID_CONFIG_FILE	The container-executor.cfg file is missing, malformed, or has incorrect permissions.
25	SETSID_OPER_FAILED	Could not set the session ID of the forked container.
26	WRITE_PIDFILE_FAILED	Failed to write the value of the PID of the launched container to the PID file of the container.
255	Unknown Error	This error has several possible causes. Some common causes are: • User accounts on your cluster have a user ID less than the value specified for the `min.user.id` property in the `container-executor.cfg` file. The default value is 1000 which is appropriate on Ubuntu systems, but may not be valid for your operating system. For information about setting `min.user.id` in the `container-executor.cfg` file, see this step. • This error is often caused by previous errors; look earlier in the log file for possible causes.

Sqoop, Pig, and Whirr Security Support Status

Here is a summary of the status of security in the other CDH 5 components:

- Sqoop 1 and Pig support security with no configuration required.
- Sqoop 2 and Whirr do not support security in CDH 5.

Setting Up a Gateway Node to Restrict Cluster Access

Use the instructions that follow to set up and use a Hadoop cluster that is entirely firewalled off from outside access; the only exception will be one node which will act as a gateway. Client machines can access the cluster through the gateway using the REST API.

HttpFS will be used to allow REST access to HDFS, and Oozie will allow REST access for submitting and monitoring jobs.

Installing and Configuring the Firewall and Gateway

Follow these steps:

1. Choose a cluster node to be the gateway machine.
2. Install and configure the Oozie server by following the standard directions starting here: Installing Oozie.
3. Install HttpFS.
4. Start the Oozie server:

```
$ sudo service oozie start
```

5. Start the HttpFS server:

```
$ sudo service hadoop-httpfs start
```

6. Configure firewalls.

   Block all access from outside the cluster.

   - The gateway node should have ports 11000 (oozie) and 14000 (hadoop-httpfs) open.
   - Optionally, to maintain access to the Web UIs for the cluster's JobTrackers, NameNode, and so on, open their HTTP ports: see Ports Used by Components of CDH 5.

7. Optionally configure authentication in simple mode (default) or using Kerberos. See HttpFS Authentication on page 83 to configure Kerberos for HttpFS and Oozie Authentication on page 102 to configure Kerberos for Oozie.
8. Optionally encrypt communication using HTTPS for Oozie by following these directions.

Accessing HDFS

With the Hadoop client:

All of the standard `hadoop fs` commands will work; just make sure to specify `-fs webhdfs://HOSTNAME:14000`. For example (where *GATEWAYHOST* is the hostname of the gateway machine):

```
$ hadoop fs -fs webhdfs://GATEWAYHOST:14000 -cat /user/me/myfile.txt
Hello World!
```

Without the Hadoop client:

You can run all of the standard `hadoop fs` commands by using the WebHDFS REST API and any program that can do `GET`, `PUT`, `POST`, and `DELETE` requests; for example:

```
$ curl "http://GATEWAYHOST:14000/webhdfs/v1/user/me/myfile.txt?op=OPEN&user.name=me"
Hello World!
```

> **Important:** The user.name parameter is valid only if security is disabled. In a secure cluster, you must a initiate a valid Kerberos session.

In general, the command will look like this:

```
$ curl "http://GATEWAYHOST/webhdfs/v1/PATH?[user.name=USER&]op=…"
```

You can find a full explanation of the commands in the WebHDFS REST API documentation.

Submitting and Monitoring Jobs

The Oozie REST API supports the direct submission of jobs for MapReduce, Pig, and Hive; Oozie automatically creates a workflow with a single action. For any other action types, or to execute anything more complicated than a single job, you must create an actual workflow. Required files (JAR files, input data, and so on.) must already exist on HDFS; if they do not, you can use HttpFS to upload the files.

With the Oozie client:

All of the standard Oozie commands will work. You can find a full explanation of the commands in the documentation for the command-line utilities.

Without the Oozie client:

You can run all of the standard Oozie commands by using the REST API and any program that can do GET, PUT, and POST requests. You can find a full explanation of the commands in the Oozie Web Services API documentation.

Logging a Security Support Case

Before you log a support case, ensure you have either part or all of the following information to help Support investigate your case:

- If possible, provide a diagnostic data bundle following the instructions in Collecting and Sending Diagnostic Data to Cloudera.
- Provide details about the issue such as what was observed and what the impact was.
- Provide any error messages that were seen, using screen capture if necessary & attach to the case.
- If you were running a command or performing a series of steps, provide the commands and the results, captured to a file if possible.
- Specify whether the issue took place in a new install or a previously-working cluster.
- Mention any configuration changes made in the follow-up to the issue being seen.
- Specify the type of release environment the issue is taking place in, such as sandbox, development, or production.
- The severity of the impact and whether it is causing outage.

Kerberos Issues

- For Kerberos issues, your `krb5.conf` and `kdc.conf` files are valuable for support to be able to understand your configuration.
- If you are having trouble with client access to the cluster, provide the output for `klist -ef` after kiniting as the user account on the client host in question. Additionally, confirm that your ticket is renewable by running `kinit -R` after successfully kiniting.
- Specify if you are authenticating (kiniting) with a user outside of the Hadoop cluster's realm (such as Active Directory, or another MIT Kerberos realm).
- If using AES-256 encryption, ensure you have the Unlimited Strength JCE Policy Files deployed on all cluster and client nodes.

SSL/TLS Issues

- Specify whether you are using a private/commercial CA for your certificates, or if they are self-signed.
- Clarify what services you are attempting to setup SSL/TLS for in your description.
- When troubleshooting SSL/TLS trust issues, provide the output of the following `openssl` command:

```
openssl s_client -connect host.fqdn.name:port
```

LDAP Issues

- Specify the LDAP service in use (Active Directory, OpenLDAP, one of Oracle Directory Server offerings, OpenDJ, etc)
- Provide a screenshot of the LDAP configuration screen you are working with if you are troubleshooting setup issues.
- Be prepared to troubleshoot using the `ldapsearch` command (requires the `openldap-clients` package) on the host where LDAP authentication or authorization issues are being seen.

Using Antivirus Software on CDH Hosts

If you use antivirus software on your servers, consider configuring it to skip scans on certain types of Hadoop-specific resources. It can take a long time to scan large files or directories with a large number of files. In addition, if your antivirus software locks files or directories as it scans them, those resources will be unavailable to your Hadoop processes during the scan, and can cause latency or unavailability of resources in your cluster. Consider skipping scans on the following types of resources:

- Scratch directories used by services such as Impala
- Log directories used by various Hadoop services
- Data directories which can grow to petabytes in size

The specific directory names and locations depend on the services your cluster uses and your configuration. In general, avoid scanning very large directories and filesystems. Instead, limit write access to these locations using security mechanisms such as access controls at the level of the operating system, HDFS, or at the service level.

Appendix: Apache License, Version 2.0

SPDX short identifier: Apache-2.0

Apache License
Version 2.0, January 2004
http://www.apache.org/licenses/

TERMS AND CONDITIONS FOR USE, REPRODUCTION, AND DISTRIBUTION

1. Definitions.

"License" shall mean the terms and conditions for use, reproduction, and distribution as defined by Sections 1 through 9 of this document.

"Licensor" shall mean the copyright owner or entity authorized by the copyright owner that is granting the License.

"Legal Entity" shall mean the union of the acting entity and all other entities that control, are controlled by, or are under common control with that entity. For the purposes of this definition, "control" means (i) the power, direct or indirect, to cause the direction or management of such entity, whether by contract or otherwise, or (ii) ownership of fifty percent (50%) or more of the outstanding shares, or (iii) beneficial ownership of such entity.

"You" (or "Your") shall mean an individual or Legal Entity exercising permissions granted by this License.

"Source" form shall mean the preferred form for making modifications, including but not limited to software source code, documentation source, and configuration files.

"Object" form shall mean any form resulting from mechanical transformation or translation of a Source form, including but not limited to compiled object code, generated documentation, and conversions to other media types.

"Work" shall mean the work of authorship, whether in Source or Object form, made available under the License, as indicated by a copyright notice that is included in or attached to the work (an example is provided in the Appendix below).

"Derivative Works" shall mean any work, whether in Source or Object form, that is based on (or derived from) the Work and for which the editorial revisions, annotations, elaborations, or other modifications represent, as a whole, an original work of authorship. For the purposes of this License, Derivative Works shall not include works that remain separable from, or merely link (or bind by name) to the interfaces of, the Work and Derivative Works thereof.

"Contribution" shall mean any work of authorship, including the original version of the Work and any modifications or additions to that Work or Derivative Works thereof, that is intentionally submitted to Licensor for inclusion in the Work by the copyright owner or by an individual or Legal Entity authorized to submit on behalf of the copyright owner. For the purposes of this definition, "submitted" means any form of electronic, verbal, or written communication sent to the Licensor or its representatives, including but not limited to communication on electronic mailing lists, source code control systems, and issue tracking systems that are managed by, or on behalf of, the Licensor for the purpose of discussing and improving the Work, but excluding communication that is conspicuously marked or otherwise designated in writing by the copyright owner as "Not a Contribution."

"Contributor" shall mean Licensor and any individual or Legal Entity on behalf of whom a Contribution has been received by Licensor and subsequently incorporated within the Work.

2. Grant of Copyright License.

Subject to the terms and conditions of this License, each Contributor hereby grants to You a perpetual, worldwide, non-exclusive, no-charge, royalty-free, irrevocable copyright license to reproduce, prepare Derivative Works of, publicly display, publicly perform, sublicense, and distribute the Work and such Derivative Works in Source or Object form.

3. Grant of Patent License.

Subject to the terms and conditions of this License, each Contributor hereby grants to You a perpetual, worldwide, non-exclusive, no-charge, royalty-free, irrevocable (except as stated in this section) patent license to make, have made, use, offer to sell, sell, import, and otherwise transfer the Work, where such license applies only to those patent claims

licensable by such Contributor that are necessarily infringed by their Contribution(s) alone or by combination of their Contribution(s) with the Work to which such Contribution(s) was submitted. If You institute patent litigation against any entity (including a cross-claim or counterclaim in a lawsuit) alleging that the Work or a Contribution incorporated within the Work constitutes direct or contributory patent infringement, then any patent licenses granted to You under this License for that Work shall terminate as of the date such litigation is filed.

4. Redistribution.

You may reproduce and distribute copies of the Work or Derivative Works thereof in any medium, with or without modifications, and in Source or Object form, provided that You meet the following conditions:

1. You must give any other recipients of the Work or Derivative Works a copy of this License; and
2. You must cause any modified files to carry prominent notices stating that You changed the files; and
3. You must retain, in the Source form of any Derivative Works that You distribute, all copyright, patent, trademark, and attribution notices from the Source form of the Work, excluding those notices that do not pertain to any part of the Derivative Works; and
4. If the Work includes a "NOTICE" text file as part of its distribution, then any Derivative Works that You distribute must include a readable copy of the attribution notices contained within such NOTICE file, excluding those notices that do not pertain to any part of the Derivative Works, in at least one of the following places: within a NOTICE text file distributed as part of the Derivative Works; within the Source form or documentation, if provided along with the Derivative Works; or, within a display generated by the Derivative Works, if and wherever such third-party notices normally appear. The contents of the NOTICE file are for informational purposes only and do not modify the License. You may add Your own attribution notices within Derivative Works that You distribute, alongside or as an addendum to the NOTICE text from the Work, provided that such additional attribution notices cannot be construed as modifying the License.

You may add Your own copyright statement to Your modifications and may provide additional or different license terms and conditions for use, reproduction, or distribution of Your modifications, or for any such Derivative Works as a whole, provided Your use, reproduction, and distribution of the Work otherwise complies with the conditions stated in this License.

5. Submission of Contributions.

Unless You explicitly state otherwise, any Contribution intentionally submitted for inclusion in the Work by You to the Licensor shall be under the terms and conditions of this License, without any additional terms or conditions. Notwithstanding the above, nothing herein shall supersede or modify the terms of any separate license agreement you may have executed with Licensor regarding such Contributions.

6. Trademarks.

This License does not grant permission to use the trade names, trademarks, service marks, or product names of the Licensor, except as required for reasonable and customary use in describing the origin of the Work and reproducing the content of the NOTICE file.

7. Disclaimer of Warranty.

Unless required by applicable law or agreed to in writing, Licensor provides the Work (and each Contributor provides its Contributions) on an "AS IS" BASIS, WITHOUT WARRANTIES OR CONDITIONS OF ANY KIND, either express or implied, including, without limitation, any warranties or conditions of TITLE, NON-INFRINGEMENT, MERCHANTABILITY, or FITNESS FOR A PARTICULAR PURPOSE. You are solely responsible for determining the appropriateness of using or redistributing the Work and assume any risks associated with Your exercise of permissions under this License.

8. Limitation of Liability.

In no event and under no legal theory, whether in tort (including negligence), contract, or otherwise, unless required by applicable law (such as deliberate and grossly negligent acts) or agreed to in writing, shall any Contributor be liable to You for damages, including any direct, indirect, special, incidental, or consequential damages of any character arising as a result of this License or out of the use or inability to use the Work (including but not limited to damages for loss of goodwill, work stoppage, computer failure or malfunction, or any and all other commercial damages or losses), even if such Contributor has been advised of the possibility of such damages.

9. Accepting Warranty or Additional Liability.

While redistributing the Work or Derivative Works thereof, You may choose to offer, and charge a fee for, acceptance of support, warranty, indemnity, or other liability obligations and/or rights consistent with this License. However, in accepting such obligations, You may act only on Your own behalf and on Your sole responsibility, not on behalf of any other Contributor, and only if You agree to indemnify, defend, and hold each Contributor harmless for any liability incurred by, or claims asserted against, such Contributor by reason of your accepting any such warranty or additional liability.

END OF TERMS AND CONDITIONS

APPENDIX: How to apply the Apache License to your work

To apply the Apache License to your work, attach the following boilerplate notice, with the fields enclosed by brackets "[]" replaced with your own identifying information. (Don't include the brackets!) The text should be enclosed in the appropriate comment syntax for the file format. We also recommend that a file or class name and description of purpose be included on the same "printed page" as the copyright notice for easier identification within third-party archives.

```
Copyright [yyyy] [name of copyright owner]

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

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

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