Cloudera Manager 7.13.2

Cloudera Authorization

Date published: 2020-11-30
Date modified: 2026-02-27

https://docs.cloudera.com/

Legal Notice

Contents

Overview

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 Ranger permissions apply to Hive (HiveServer2) and Impala.

Cloudera 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.
- Apache HDFS ACLs 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.
- Access control with Apache Ranger.

Configuring LDAP Group Mappings

Each host that comprises a node in a Cloudera cluster runs an operating system, such as CentOS or Oracle Linux. At the OS-level, there are user:group accounts created during installation that map to the services running on that specific node of the cluster. The default shell-based group mapping provider, org.apache.hadoop.security.ShellBase dUnixGroupsMapping, handles the mapping from the local host system (the OS) to the specific cluster service, such as HDFS. The hosts authenticate using these local OS accounts before processes are allowed to run on the node.

For clusters integrated with Kerberos for authentication, the hosts must also provide Kerberos tickets before processes can run on the node. The cluster can use the organization's LDAP directory service to provide the login credentials, including Kerberos tickets, to authenticate transparently while the system runs. That means that the local user:group accounts on each host must be mapped to LDAP accounts. To map local user:group accounts to an LDAP service:

- Use tools such as SSSD (Systems Security Services Daemon) or Centrify Server Suite (see Identity and Access management for Cloudera).
- The Hadoop LdapGroupsMapping group mapping mechanism. The LdapGroupsMapping library may not be as robust a solution needed for large organizations in terms of scalability and manageability, especially for organizations managing identity across multiple systems and not exclusively for Hadoop clusters. Support for the LdapGroupsMapping library is not consistent across all operating systems.
- Do not use Winbind to map Linux user:group accounts to Active Directory. It cannot scale, impedes cluster performance, and is not supported.
- Use the same user:group mappings across all cluster nodes, for ease of management.
- Use either Cloudera Manager or the command-line process detailed below.

The local user:group accounts must be mapped to LDAP for group mappings in Hadoop. You must create the users and groups for your Hadoop services in LDAP.

To integrate the cluster with an LDAP service, the user:group relationships must be contained in the LDAP directory. The admin must create the user accounts and define groups for user:group relationships on each host.

The user and group names listed in the table are the default user:group values for Cloudera services.

Note: If the defaults have been changed for any service, use the custom values to create the users and configure the group for that service in the LDAP server, rather than the defaults listed in the table below. For example, you changed the defaults in the Cloudera Manager Admin Console to customize the System User or System Group setting for the service.

Cloudera Product or Component	User	Group
Cloudera Manager	cloudera-scm	cloudera-scm
Apache Accumulo	accumulo	accumulo
Apache Avro	(No default)	(No default)
Apache HBase (Master, RegionServer processes)	hbase	hbase
Apache HCatalog	hive	hive
Apache Hive (HiveServer2, Hive Metastore)	hive	hive
Apache Kafka	kafka	kafka
Apache Oozie	oozie	oozie
Apache Spark	spark	spark
Apache Sqoop1	sqoop	sqoop
Apache Whirr	(No default)	(No default)
Apache ZooKeeper	zookeeper	zookeeper
Impala	impala	impala, hive
Cloudera Search	solr	solr
HDFS (NameNode, DataNodes)	hdfs	hdfs, hadoop
HttpFS	httpfs	httpfs
Cloudera Data Explorer (Hue)	hue	hue
Cloudera Data Explorer (Hue) Load Balancer (needs apache2 package)	apache	apache
Kudu	kudu	kudu
Llama	llama	llama
MapReduce (JobTracker, TaskTracker)	mapred	mapred, hadoop
Parquet	(No default)	(No default)
YARN	yarn	yarn, hadoop

Using Cloudera Manager

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

Make the following changes to the Core Settings service's security configuration:

1. Open the Cloudera Manager Admin Console and go to the Core Settings service.
2. Click the Configuration tab.
3. Select Category Security .
4. Modify the following configuration properties using values from the table below:

Configuration Property	Value
Hadoop User Group Mapping Implementation	org.apache.hadoop.security.LdapGroupsMapping
Hadoop User Group Mapping LDAP URL	ldap://<server>

Configuration Property	Value
Hadoop User Group Mapping LDAP Bind User Distinguished Name	Administrator@example.com
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.

Emitting the LDAP Bind password in core-site.xml for client configurations

If the Cloudera cluster has LDAP group to OS group mapping enabled, then applications running in Spark or YARN would fail to authenticate to the LDAP server when trying to use the LDAP bind account during the LDAP group search. This is because the LDAP bind password was not passed to the /etc/hadoop/conf/core-site.xml file. This was intended behavior to prevent leaking the LDAP bind password in a clear text field.

Starting with Cloudera Manager 7.7.1 CHF 17 and Cloudera Manager 7.11.3 CHF 2, the cluster administrator can enable Cloudera Manager to pass the LDAP bind password to client applications in clear text through the /etc/hadoop/conf/core-site.xml file.

1. To do this, modify the Cloudera Manager server configuration file /etc/default/cloudera-scm-server by adding the flag -Dcom.cloudera.cmf.service.config.emitLdapBindPasswordInClientConfig=true to the variable CMF_JAVA_OPTS flag such as:

```
CMF_JAVA_OPTS="-Xmx4G -XX:+HeapDumpOnOutOfMemoryError
-XX:HeapDumpPath=/tmp
-Dcom.sun.management.jmxremote.ssl.enabled.protocols=TLSv1.2
-Dcom.cloudera.cmf.service.config.emitLdapBindPasswordInClientConfig=true"
```

   Important: The previous contents of the CMF_JAVA_OPTS variable might be slightly different than the example above, but those contents should be preserved.

2. To have the setting take effect, restart the Cloudera Manager server by running the following command:

```
sudo systemctl restart cloudera-scm-server
```

3. Then deploy client configuration files to have Cloudera Manager redistribute the Hadoop core-site.xml client configuration file. See Manually Redeploying Client Configuration Files.

Using Ranger to Provide Authorization in Cloudera

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

Once a user has been authenticated, their access rights must be determined. Authorization defines user access rights to resources. For example, a user may be allowed to create a policy and view reports, but not allowed to edit users and groups. You can use Ranger to set up and manage access to Hadoop services.

Ranger enables you to create services for specific resources (HDFS, HBase, Hive, etc.) and add access policies to those services. Ranger security zones enable you to organize service resources into multiple security zones. You can also create tag-based services and add access policies to those services. Using tag-based policies enables you to control access to resources across multiple components without creating separate services and policies in each

component. You can also use Ranger TagSync to synchronize the Ranger tag store with an external metadata service such as Apache Atlas.

Note:

You can configure authorization using the Ranger UI, REST APIs, or client libraries. For more information about:

- Ranger REST APIs, see see https://ranger.apache.org/apidocs/index.html.
- Ranger client libraries, see Using Ranger client libraries.

Related Information

Apache Ranger Authorization

Apache Ranger REST API: Resources

Using Ranger client libraries