Cloudera Runtime 7.3.2

Configuring and Using Ranger RMS (Hive-S3 ACL Sync)

Date published: 2020-07-28
Date modified: 2026-03-31

https://docs.cloudera.com/

Legal Notice

Contents

Introduction to Ranger RMS 4
Ranger RMS - HIVE-HDFS ACL Sync Overview 4

About Ranger RMS for Ozone 8
Ranger RMS Assumptions and Limitations 8
Installing/Verifying RMS for Ozone configuration 9
Enabling RMS for Ozone authorization 10

About Ranger RMS for S3 (HIVE-S3 ACL Sync) 11

Understanding Ranger policies with RMS 14

How to full sync the Ranger RMS database 16

Configuring High Availability for Ranger RMS (Hive-HDFS ACL-Sync) 17

Configuring Ranger RMS (Hive-HDFS/Hive-OZONE/Hive-S3 ACL Sync) 22

Configuring HDFS plugin to view permissions through getfacl interface 26

Migrating Ranger RMS server role instance to a new host 27

Ranger RMS (Hive-HDFS ACL-Sync) Use Cases 28

Introduction to Ranger RMS

Ranger Resource Mapping Server (RMS) enables automatic translation of access policies from HIVE to HDFS.

Ranger RMS - HIVE-HDFS ACL Sync Overview

Ranger Resource Mapping Server (RMS) enables automatic translation of access policies from HIVE to HDFS.

About HIVE-HDFS ACL Sync

It is common to have different workloads use the same data – some require authorizations at the table level (Apache Hive queries) and others at the underlying files (Apache Spark jobs). Unfortunately, in such instances you would have to create and maintain separate Ranger policies for both Hive and HDFS, that correspond to each other.

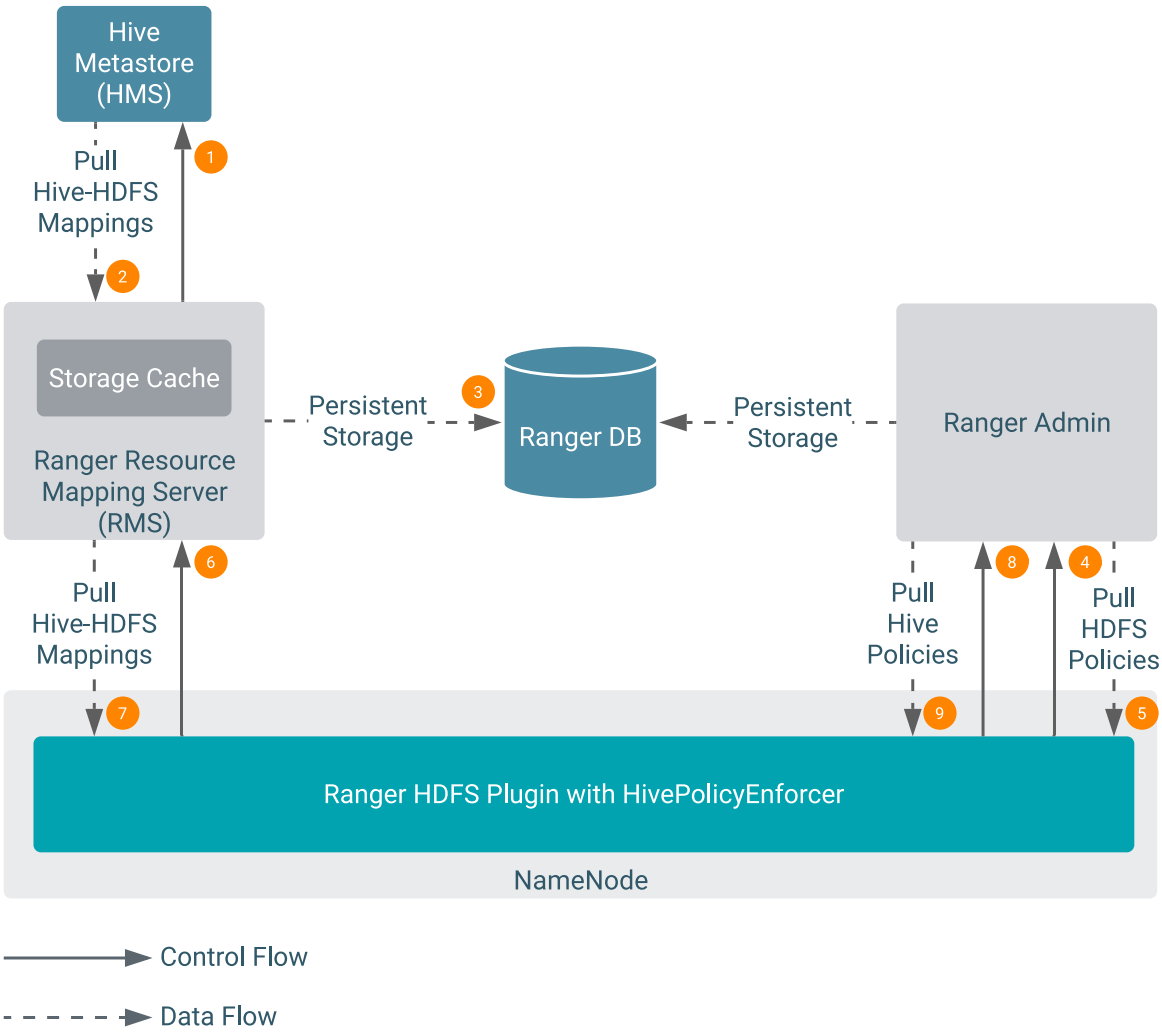
As a result, whenever a change is made on a Hive table policy, the data admin should make a consistent change in the corresponding HDFS policy. Failure to do so could result in security and/or data exposure issues. Ideally the data admin would set a single table policy, and the corresponding file access policies would automatically be kept in sync along with access audits, referring to the table policy that enforced it.

Legacy CDH users had a feature called the Hive-HDFS ACL sync which had Hive policies in Apache Sentry that automatically linked Hive permissions with HDFS ACLs. This was especially convenient for external table data used by Spark or Hive.

Prior to Cloudera Runtime 7.1.6, Ranger only supported manually managing Hive and HDFS policies separately. Ranger RMS (Resource Mapping Server) allows you to authorize access to HDFS directories and files using policies defined for Hive tables. RMS is the service that enables Hive-HDFS Policy Sync.

RMS periodically connects to the Hive Metastore and pulls Hive metadata (database-name, table-name) to HDFS file-name mapping. The Ranger HDFS Plugin (running in the NameNode) has been extended with an additional HivePolicyEnforcer module. The HDFS plugin downloads Hive policies from Ranger Admin, along with the mappings from Ranger RMS. HDFS access is determined by both HDFS policies and Hive policies.

Figure 1: HIVE-HDFS ACL Sync using Ranger RMS



Phase I (items 1-3 above)

Ranger RMS periodically connects to the HIVE Metastore and pulls HIVE metadata (database-name, table-name) to HDFS file-name mapping.

Phase II (items 4-9 above)

The Ranger HDFS Plugin (running in the NameNode) periodically pulls HDFS policies from Ranger Admin. With the introduction of Ranger RMS, the Ranger HDFS Plugin (running in the NameNode) that has been extended with an additional HIVEPolicyEnforcer module. It now pulls down the HIVE-HDFS mappings from RMS and HIVE Policies from Ranger Admin.

After phase II completes, the requested HDFS access is determined in the NameNode by the HDFS and HIVE policies defined by the Ranger Administrator.

Currently, Ranger performs authorization of the HDFS commands which require access to the hierarchy of files/ directory rooted at the argument passed to the HDFS command as described below. Some examples of such commands are :

```
hdfs dfs -count -q -h -v <directory>; hdfs dfs -R <directory>
```

HDFS Authorization Interface

When these commands are invoked, HDFS Namenode builds a tree of i-nodes corresponding to <directory>, and passes it to the authorizer with a flag indicating that subAccess (access to the directory hierarchy rooted at <directory>) is to be checked.

About database-level grants feature

Legacy CDH users used HIVE policies in Apache Sentry that automatically linked HIVE permissions with HDFS ACLs. This was especially convenient for external table data used by Spark or HIVE. Specifically, using Sentry, you could make grants at the HIVE database level and HDFS permissions would propagate to the database directory, and to all tables and partitions under it.

Previously, Ranger only supported managing HIVE and HDFS policies separately. Ranger Resource Mapping Server (RMS) now allows you to create a database level policy in HIVE and have these permissions propagate to the HDFS locations and all tables under it. RMS is the service that enables HIVE-HDFS ACL Sync.

RMS captures database metadata from the HIVE Meta Store (HMS). After the first, full-synchronization run, RMS downloads mappings for tables and databases present in the HMS.

Whenever you create a new database, RMS synchronizes metadata information from HMS and uses it to update the resource mapping file linking HIVE database resources to their corresponding HDFS location. Any user with access permissions on a HIVE database automatically receives similar HDFS file-level access permissions on the database's data files. Select/ Read access for any user in the database location is allowed through default HIVE policy for all-databases. This behavior is treated as _any access, which is similar to the HIVE command `show tables`. If a user has no HIVE policy which allows access on the database, then the user is denied access to the corresponding HDFS location of that database. Previously, users were not allowed to access the HDFS location of a database even if the user had permission to access the database through a HIVE policy. The HDFS to HIVE access type mappings follow:

Access Type mapping for HDFS to HIVE for Database:

- _any=[_any]
- read=[_any]
- write=[drop, alter]
- execute=[_any]

Access Type mapping for HDFS to HIVE for Table:

- _any=[_any]
- read=[select]
- write=[update, alter]
- execute=[_any]

If you create tables under a database but the HDFS location of the corresponding table does not reside under the HDFS location of that database (for example: table locations are external locations), the HIVE policies (database-name, table = *, column= *) translate into HDFS access rules and allow the HDFS NameNode to enforce them. If the policy is created only for the database resource, the same access translates to the HDFS location of that database only; not for the tables residing under that database.

Ranger RMS Assumptions and Limitations

- All partitions of a table are assumed to be under the location specified for the table. Therefore, table permissions will not authorize access to partitions that store data outside the location specified for the table. For example, if a table is located in a /warehouse/foo HDFS directory, all partitions of the table must have locations that are under the /warehouse/foo directory.
- The Ranger RMS service is not set up automatically when a Cloudera Base on premises cluster is deployed. You must install and configure Ranger RMS separately.
- Ranger policies should be configured (with rangerrms user access) before RMS is started and runs the first sync from the HIVE Metastore (HMS).
- The Ranger RMS ACL-sync feature supports a single logical HMS, to evaluate HDFS access via HIVE permissions. This is aligned with the Sentry implementation in CDH.

- Permissions granted on views (traditional and materialized) do not extend to HDFS access. This is aligned with the Sentry implementation in CDH.
- Default services (also known as Repos) for Hive, HDFS, Ozone, and S3 must be present in the Ranger Admin DB for RMS to function correctly. Check *Configuring Ranger RMS* for the default values of these services.
- If a Cloudera Base on premises deployment supports multiple logical HMS with a single Ranger, Ranger RMS (Hive-HDFS ACL-Sync) works with only one logical HMS. Permissions granted on databases/tables in other logical HMS instances will not be considered to authorize HDFS access.
- Namenode memory requirements must be increased, based on the number of HIVE table mappings downloaded to HDFS Ranger plugin. Additionally, maintaining HIVE policies in memory cache will also require additional memory.
- Expect Namenode CPU load to increase, due to additional access evaluation performed to enforce HIVE policies and periodic downloading and processing of the HIVE table mappings. The latter increase is proportional to the number of table mappings downloaded to HDFS Ranger plugin.
- When multiple databases are mapped to single HDFS location, and if a HIVE policy allows a user to access one database. Then, users will be able to access its HDFS location and all other files & directories under it. This may include table or database directories of other databases and tables. However, users will not be able to access other databases or tables under it through Hive queries.

  For example,

  music_a, music_b, music_c are created at HDFS path'/data'.

  Policy-A to allow 'sam' user 'all' access on resource = {database=music_a; table= * ; column= * ; }

  Now, 'sam' user will get all access on hdfs path /data and files, directories under it. Therefore, 'sam' user will be able to access hdfs location of tables under music_b and music_c databases as long as those locations reside under /data directory.

  However, 'sam' user will not be able to access music_b and music_c databases or any tables under these databases through Hive queries.

  Note: This is the expected behaviour when the hdfs location for different database/tables are designed in such a way that these entities share the same hdfs path or HIVE database/table locations fall under another database hdfs location.

Comparison with Sentry HDFS ACL sync

The Ranger RMS (Hive-HDFS ACL-Sync) feature resembles the Sentry HDFS ACL Sync feature in the way it downloads and keeps track of the HIVE table to HDFS location mapping.

It differs from Sentry in the way it completely and transparently supports all features that Ranger policies express. Therefore, support for tag-based policies, security-zones, masking and row-filtering and audit logging is included with this implementation.

Also, the feature is enabled or disabled by a simple configuration on the HDFS side, allowing each installation the option of turning this feature on or off.

Considerations when upgrading Ranger RMS

Prior to Cloudera Runtime 7.1.7, Ranger RMS synchronizes only table metadata. Database metadata mappings were not downloaded from HMS. After migrating from Cloudera Runtime versions <7.1.7 to CDP 7.1.8+, only pre-existing table mappings will be available. Any newly created tables or database mappings will be synchronized in RMS. Any pre-existing database mappings will not be present. To ensure the pre-existing databases are mapped correctly, you must perform a full-sync after completing the upgrade.

If you configured RMS for HDFS/Ozone before Cloudera Runtime version 7.3.2.0 and also stored external table data in S3, then you need a full synchronization after upgrading. This ensures that any databases and tables created prior to the upgrade, and whose data resides in S3, are correctly mapped. Any newly created tables or database mappings will be synchronized in RMS post upgrade. For specific steps, see How to full sync the Ranger RMS database.

Related Information
Installing Ranger RMS
What's New in Apache Ranger
Configuring Ranger RMS

About Ranger RMS for Ozone

Ranger RMS supports authorization for Ozone storage locations. RMS for Ozone coexists with Hive-HDFS ACL sync and provide authorization for both HDFS and Ozone file systems.

RMS periodically connects to the Hive Metastore (HMS), pulls the following Hive metadata

- database-name
- table-name
- location

and stores it into the Ranger database. The Ranger Ozone plugin (running in the OzoneManager) has been extended with an additional RangerOzoneHiveChainedPlugin module. After enabling RMS for Ozone authorization, it downloads Hive policies, Tags and Roles from Ranger Admin, along with the resource-mappings from Ranger RMS. Both Ozone policies and Hive policies determine Ozone access.

Previously, Ranger only supported managing Hive and Ozone policies separately. Ranger Resource Mapping Server (RMS) now allows you to create a database/table level policy in HIVE and have these permissions propagate to the Ozone storage locations of these databases/tables and all files and directories under it. Users can use Ozone as a storage technology to create databases/tables along with the existing HDFS filesystem and RMS is the service that enables HIVE-OZONE ACL Sync along with the HIVE-HDFS ACL Sync. (Prior to Cloudera Runtime 7.1.9 only HIVE-HDFS ACL sync was supported).

After the first, full-synchronization run, RMS downloads mappings for tables and databases present in the HMS. These tables/databases could be located in both OZONE and HDFS file systems. RMS will map the tables and databases with their respective storage locations and store them into the Ranger database with its associated service-id (cm_ozone or cm_hdfs). When ranger-ozone-plugin or ranger-hdfs-plugin requests mappings, it only fetches the requested service mappings. In other words, ranger-ozone-plugin will download mappings only for tables/databases whose storage location is Ozone. Similarly, ranger-hdfs-plugin will download mappings only for tables/databases whose storage location is HDFS. Read access for any user in the database location is allowed through the default HIVE policy for all-databases. This behavior is treated as _any access, which is similar to the HIVE command `show tables`. If a user has no HIVE policy which allows access to the database, then the access is denied to the corresponding Ozone location of that database. This access evaluation aligns with the HDFS db-level grants feature.

Ranger RMS Assumptions and Limitations

Learn about the assumptions and limitations for Ranger RMS.

- RMS for Ozone does not support semantics enforced by Sentry for HDFS ACLs. This implies that even when ranger.plugin.ozone.mapping.hive.authorize.with.only.chained.policies is set to true, to access ozone locations when RMS is enabled; READ access must be given for the user on bucket and volume where Hive Tables and Databases are located.
- In addition to Ozone policies, Ranger RMS will authorize access only for "key" resources which are the database/table locations in ozone. To access volumes and buckets, user should have appropriate access through policies of Ozone service configured Ranger Admin.
- The Ranger RMS ACL-sync feature supports a single logical HMS, to evaluate OZONE/HDFS access through HIVE permissions.
- RMS ACL sync is designed to work on a specific pair of HDFS<->Hive and OZONE<->Hive Ranger services. It will support only one pair of “HDFS and HIVE” services and one pair of “OZONE and HIVE” services.

- By default, all external tables are stored into HDFS. To create an external table in Ozone, please specify the location clause by providing the Ozone location at the time of creating a table or configure Hive External Warehouse Directory in HMS.
- The default Hive Warehouse directory is configured in HiveMetaStore (HMS) to store managed tables into HDFS. To store managed tables into Ozone, please configure the Hive Warehouse directory or update the database managed location as an Ozone storage location. Please check with the Ozone/Hive support members to know more about creating Hive Tables/Databases into Ozone.

Installing/Verifying RMS for Ozone configuration

Learn how to verify Ozone configurations for Ranger RMS.

About this task

Manual steps to verify RMS for Ozone setup:

Procedure

1. Verify if Ranger RMS Url is configured in Ozone ranger-ozone-security.xml file.
   a) Go to Cloudera Manager Ozone Configuration Ozone Manager Advanced Configuration Snippet (Safety Valve) for ozone-conf/ranger-ozone-security.xml .
   b) Add the Ranger RMS url to the following property:

   ranger.plugin.ozone.mapping.source.url =

   <rms-url>

   After configuring this property with appropriate Ranger RMS URL, ranger-ozone-plugin will be able to communicate with RMS (Resource Mapping Server) for downloading files.
2. Verify if the Ozone service name is correctly configured.
   a) Go to Cloudera Manager Ranger RMS Configuration , and search for the property ranger_rms_hms_source_service_name_ozone .
   b) Verify the following property and value:

   ranger-rms.HMS.source.service.name.ozone =

   cm_ozone

   Note: The value cm_ozone is the name of the OZONE service configured in the Ranger admin. If this configuration is changed after Ranger RMS is started and has synchronized with Hive Metastore, the only way to have Ranger RMS use a new configuration is by performing a RMS Full Sync. Please refer to the related topic, "How to full sync the Ranger RMS database" for specific steps.
3. Verify whether ozone schema's are configured.
   a) Go to Cloudera Manager Ranger RMS Configuration Ranger RMS Supported Uri Scheme .
   b) Verify the following property and value:

   ranger-rms.supported.uri.scheme =

   hdfs,o3fs,ofs
4. Log in to Ranger Admin Web UI.
5. On Service Manager, in Hadoop SQLservice, click Edit.
6. Verify that "om" user has been added to the Policy Download Users and Tag Download Users configurations.

   Otherwise, add the following values to the cm_hive service configuration:
   a) Policy Download Users = hive,hdfs,impala,om
   b) Tag Download Users = hive,hdfs,impala,om

   Note: This ozone service user can also be found in the ranger-admin-site.xml file with the following property key: ranger.plugins.ozone.serviceuser=om or in the service-usernames.csv file generated by Cloudera Manager.

Enabling RMS for Ozone authorization

You must grant necessary permissions and update policies to access Ozone.

About this task

When Ranger is enabled in the cluster, any user other than the default admin user, om requires the necessary Ranger permissions and policy updates to access the Ozone file system. To create a Hive external table that points to the Ozone file system, the hive user should have the required permissions in Ranger. Give all accesses to hive user by updating default policy all - volume, bucket, key in cm_ozone service.

Procedure

1. Login to Ranger Admin Web UI.
   a) Go to Cloudera Manager Clusters Ranger Ranger Admin Web UI .
   b) Type your username and password.
   c) Click Sign In.

   The Service Manager for Resource-based Policies page displays.
2. In Service Manager cm_ozone service, click Edit.
3. In cm_ozone policies all - volume, bucket, key policy, click Edit.
4. In Allow Conditions, add the hive user, choose necessary permissions, then click Save, as shown in:

   Figure 2: Editing Ozone plugin permissions for hive user

5. Grant Read access to users only on volumes and buckets where Hive tables and databases are located. (Recommended)
   a) create a new policy in Ozone service (cm_ozone) for volumes and buckets where Hive tables/databases are located.
   b) In Allow Conditions, add the users, groups, and roles and give Read permission.

   or else, to grant Read access to everyone on all volumes and buckets, in Ozone service (cm_ozone) policies page,
   a) edit the all - volume, bucket policy
   b) add the public group to the group list
   c) give Read permission

   Note: For all databases and tables whose storage location is in Ozone; these locations will be mapped to ozone resource type Key and after enabling RMS for OZONE authorization; ranger-ozone-plugin will evaluate access on a Key based on Hive policies. If no applicable Hive policy exists, then the Ozone policies will determine access. Otherwise, access is denied.

6. Add chained properties to the ranger-ozone-security.xml file.
   a) Go to Cloudera Manager Ozone Configuration Ozone Manager Advanced Configuration Snippet (Safety Valve) for ozone-conf/ranger-ozone-security.xml .
   b) Click +Add to add the following properties and values:

   ranger.plugin.ozone.chained.services =

   cm_hive

   ranger.plugin.ozone.chained.services.cm_hive.impl =

   org.apache.ranger.chainedplugin.ozone.hive.RangerOzoneHiveChainedPlugin

   Note: The value cm_hive is the name of the Hive/Hadoop SQL service configured in the Ranger admin.

7. Restart the Ozone service.

Results

If everything is correctly configured as explained above, the ranger-ozone-plugin automatically communicates with RMS, downloads Hive-Ozone mappings and stores them into policy-cache directory as a ozone_cm_hive_resource_mapping.json file. It will also communicate with the Ranger admin and download Hive policies, Hive tags and Hive roles.

The following files download into the policy-cache directory

(default configured value: /var/lib/ranger/ozone/policy-cache/)

after enabling RMS for Ozone authorization:

- ozone_cm_hive_resource_mapping.json
- ozone_cm_hive.json
- ozone_cm_hive_tag.json
- ozone_cm_hive_roles.json

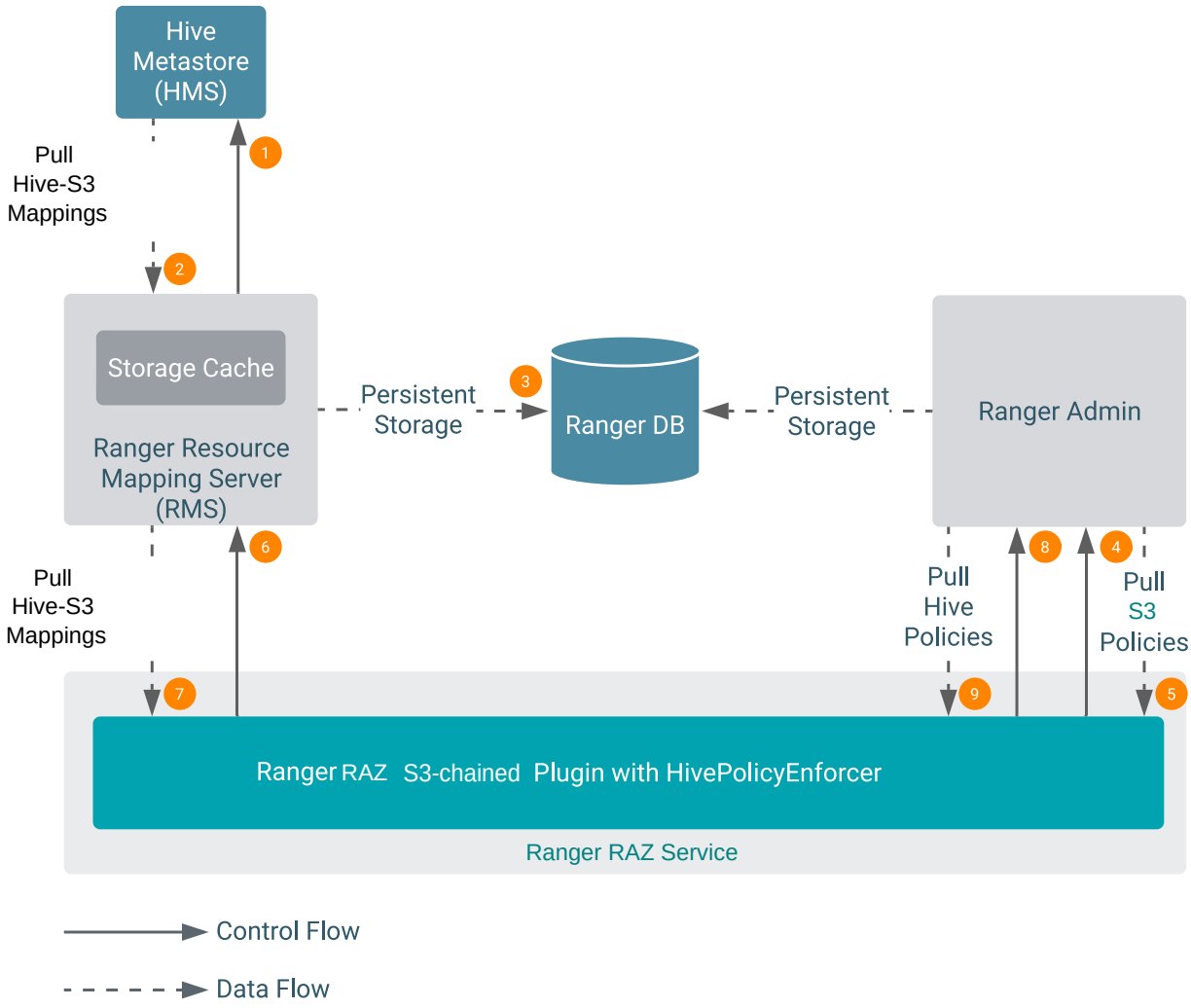
About Ranger RMS for S3 (HIVE-S3 ACL Sync)

Ranger Resource Mapping Server (RMS) enables automatic translation of access policies from HIVE to S3. This feature is available on Cloudera Base on premises from the 7.3.2.0 release onwards, where RAZ on premises for Amazon S3 compatible object store installation is supported. This feature co-exists with RMS support for HDFS and Ozone file systems on Cloudera Base on premises.

Ranger RMS allows you to authorize access to S3 locations using policies defined for Hive tables and database resources. The permissions defined in these hive policies are propagated to the S3 locations of those databases/tables and all files and directories under them.

RMS periodically connects to the Hive Metastore and pulls Hive metadata (database name, table name) to S3 file-name mapping. This is done by a RAZ-chained plugin (running in the Ranger RAZ service) which has an additional HivePolicyEnforcer module. After enabling RMS for S3 authorization, the RAZ-chained plugin downloads Hive policies, tags, and roles from Ranger Admin, along with the mappings from Ranger RMS. S3 access is determined by both S3 policies and Hive policies.

Figure 3: HIVE-S3 ACL Sync using Ranger RMS



Phase I (items 1-3 above)

Ranger RMS periodically connects to the HIVE Metastore and pulls HIVE metadata (database name, table name) to S3 file-name mapping.

Phase II (items 4-9 above)

The Ranger RAZ S3 Chained Plugin (running in the RAZ service) periodically pulls S3 policies from Ranger Admin. With the introduction of Ranger RMS, the Ranger RAZ S3 Chained Plugin (running in the RAZ service) has been extended with an additional HIVEPolicyEnforcer module. It now pulls down the HIVE-S3 mappings from RMS and HIVE policies from Ranger Admin.

After phase II completes, the requested S3 access is determined in the RAZ service by the S3 and HIVE policies defined by the Ranger Administrator.

The S3 to HIVE access type mappings follow:

- Access Type mapping for S3 to HIVE for Database:
  - _any=[_any]
  - read=[_any]
  - write=[create, drop, alter]

- Access Type mapping for S3 to HIVE for Table:
  - _any=[_any]
  - read=[select]
  - write=[update, alter]

If you create tables under a database but the S3 location of the corresponding table does not reside under the S3 location of that database (for example, table locations are external locations), the HIVE policies (database name, table = *, column= *) translate into S3 access rules and allow the RAZ S3 chained plugin to enforce them. If the policy is created only for the database resource, the same access translates to the S3 location of that database only, not for the tables residing under that database.

Read access for any user in the database location is allowed through the default HIVE policy for all databases. This behavior is treated as _any access, which is similar to the HIVE command show tables. If a user has no HIVE policy which allows access to the database, then the access is denied to the corresponding Ozone location of that database. This access evaluation aligns with the HDFS db-level grants feature.

Ranger RMS assumptions and limitations

- All partitions of a table are assumed to be under the location specified for the table. Therefore, table permissions will not authorize access to partitions that store data outside the location specified for the table. For example, if a table is located in a /warehouse/foo S3 directory, all partitions of the table must have locations that are under the /warehouse/foo directory.
- The Ranger RMS ACL-sync feature supports a single logical HMS to evaluate S3 access through HIVE permissions. This is aligned with the Sentry implementation in CDH.
- When Ranger RAZ and RMS services are installed in Cloudera Manager, the RMS HIVE-S3 ACL Sync feature to authorize S3 locations is enabled by default, as it is pre-configured with chained plugin settings.
- Permissions granted on views (traditional and materialized) do not extend to S3 access. This is aligned with the Sentry implementation in CDH.
- RMS ACL-sync is designed to work on a specific pair of S3<->Hive Ranger services. Ranger RMS supports only one pair of Hive and S3 services. By default, cm_s3 is configured as the source service and cm_hive as the target service.
- If a Cloudera Base on premises deployment supports multiple logical HMS with a single Ranger, Ranger RMS (Hive-S3 ACL-Sync) works with only one logical HMS. Permissions granted on databases/tables in other logical HMS instances will not be considered to authorize S3 access.
- Ranger RAZ memory requirements must be increased based on the number of HIVE table mappings downloaded to the S3 Ranger plugin. Additionally, maintaining HIVE policies in the memory cache will also require additional memory.
- Ranger RMS service will use the same database as Ranger Admin to store mappings downloaded from HMS.
- Ranger RAZ service will have an S3 chained plugin, and it will perform authorization based on the policies and mappings downloaded and stored in the policy-cache directory of the RAZ service. Even if the RMS service is stopped, authorization will continue to work based on the files available in the policy-cache directory.
- Expect Ranger RAZ CPU load to increase due to additional access evaluation performed to enforce HIVE policies and periodic downloading and processing of the HIVE table mappings. The latter increase is proportional to the number of table mappings downloaded to the RAZ S3 chained plugin.
- When multiple databases are mapped to a single S3 location, and if a HIVE policy allows a user to access one database, then users will be able to access its S3 location and all other files and directories under it. This may include table or database directories of other databases and tables. However, users will not be able to access other databases or tables under it through Hive queries.

  For example,

  music_a, music_b, and music_c are created at S3 path '/data'. Policy-A allows the 'sam' user 'all' access on resource = {database=music_a; table= * ; column= * ; }. Now, the 'sam' user will get all access to the S3 path '/data' and files and directories under it. Therefore, the 'sam' user will be able to access S3 location of tables under the music_b and music_c databases as long as those locations reside under the '/data' directory. However, the 'sam'

user will not be able to access the music_b and music_c databases or any tables under these databases through Hive queries.

Note: This is the expected behavior when the S3 location for different database/tables is designed in such a way that these entities share the same S3 path or HIVE database/table locations fall under another database's S3 location.

Comparison with Sentry HDFS ACL sync

The Ranger RMS (Hive-S3 ACL-Sync) feature resembles the Sentry HDFS ACL-Sync feature in the way it downloads and keeps track of the HIVE table-to-S3 location mapping.

It differs from Sentry in the way it completely and transparently supports all features that Ranger policies express. Therefore, support for tag-based policies, security zones, masking, row-filtering, and audit logging is included with this implementation.

Also, the feature is enabled or disabled by a simple configuration on the Ranger RAZ side, allowing each installation the option of turning this feature on or off.

Understanding Ranger policies with RMS

Ranger RMS for HDFS access evaluation workflow

At a high level, the Ranger RMS for HDFS access evaluation workflow is as follows:

- Ranger policies for the HDFS service are evaluated. If any policy explicitly denies access, access is denied.
- Ranger checks to see if the accessed location maps to a Hive table.
- If it does, Hive policies are evaluated for the mapped Hive table. If there is an HDFS policy allowing access, access is allowed. Otherwise, the default HDFS ACLs determine the access.
  Requested HDFS permission is mapped to Hive permissions as follows:

  HDFS ‘read’ ==> Hive ‘select’

  HDFS ‘write’ ==> Hive ‘update’ or ‘alter’

  HDFS ‘execute’ ==> Any Hive permission
- If there is no Hive policy that explicitly allows access to the mapped table, access is denied, otherwise access is allowed.

Appropriate tag policies are considered both during HDFS access evaluation and if needed, during Hive access evaluation phases. Also, one or more log records are generated to indicate which policy, if any, made the access decision.

The following scenarios illustrate how the access permissions are determined. All scenarios assume that the HDFS location is NOT explicitly denied access by a Ranger HDFS policy.

Location does not correspond to a Hive table.

In this case, access will be granted only if a Ranger HDFS policy allows access or HDFS native ACLs allow access. The audit log will show which policy (or Hadoop-acl) made the decision.

Location corresponds to a Hive table.

A Ranger Hive policy explicitly denied access to the mapped table for any of the accesses derived from the original HDFS request.

- Access will be denied by Hive policy.

There is no matching Ranger Hive policy.

- Access will be denied. Audit log will not specify the policy.

Ranger policy masks some columns in the mapped table.

- Access will be denied. Audit log will show Hive masking policy.

Mapped Hive table has a row-filter policy

- Access will be denied. Audit log will show Hive Row-filter policy.

A Ranger Hive policy allows access to the mapped table for the access derived from the original HDFS access request.

- Access will be granted. If the access was originally granted by HDFS policy, the audit log will show HDFS policy.

Ranger RMS for Ozone access evaluation workflow

At a high level, the Ranger RMS for Ozone access evaluation workflow is as follows:

- For every “key” access request made to ranger-ozone-plugin, there are three separate requests are received for authorization as follows:
  - To check READ access on volume
  - To check READ access on bucket
  - To check access made originally for a key
- First two accesses on volume and buckets are determined by Ozone policies in Ranger Admin. RMS will only evaluate access for the key resource.
- Further for the key access evaluation, Ranger policies for the Ozone service are evaluated. If any policy explicitly denies access, access is denied.
- Ranger checks to see if the accessed location (key) maps to a Hive table.
- If it does, Hive policies are evaluated for the mapped Hive table. If there is an OZONE policy allowing access, access is allowed. Otherwise, the access will be denied.
- Requested OZONE permission is mapped to Hive permissions as follows:

Ozone ACL	Should translate into Hive ACL For Table	Should translate into Hive ACL For Database
read	select	_any
write	update	update
create	create	create
list	select	select
delete	drop	drop
read_acl	select	select
write_acl	update	update

- If there is no Hive policy that explicitly allows access to the mapped table, access is denied, otherwise access is allowed.
- Appropriate tag policies are considered both during OZONE access evaluation and if needed, during Hive access evaluation phases.
- One or more log records are generated to indicate which policy, if any, made the access decision.

Ranger RMS for S3 access evaluation workflow

At a high level, the Ranger RMS for S3 access evaluation workflow is as follows:

- Ranger policies for the S3 service are evaluated. If any policy explicitly denies access, access is denied.
- Ranger checks to see if the accessed location maps to a Hive table.

- If it does, Hive policies are evaluated for the mapped Hive table. Otherwise, if there is an S3 policy allowing access, then the access is allowed.
  Requested S3 permission is mapped to Hive permissions as follows:

  S3 ‘read’ ==> Hive ‘select’

  S3 ‘write’ ==> Hive ‘update’ or ‘alter’
- If there is no Hive policy that explicitly allows access to the mapped table, access is denied; otherwise, access is allowed.

Appropriate tag policies are considered both during S3 access evaluation and, if needed, during Hive access evaluation phases. Also, one or more log records are generated to indicate which policy, if any, made the access decision.

The following scenarios illustrate how the access permissions are determined. All scenarios assume that the S3 location is NOT explicitly denied access by a Ranger S3 policy.

Location does not correspond to a Hive table

In this case, access will be granted only if a Ranger S3 policy allows access. The audit log will show which policy made the decision.

Location corresponds to a Hive table

A Ranger Hive policy explicitly denied access to the mapped table for any of the accesses derived from the original S3 request.

- Access will be denied by Hive policy.

There is no matching Ranger Hive policy.

- Access will be denied. Audit log will not specify the policy.

Ranger policy masks some columns in the mapped table.

- Access will be denied. Audit log will show Hive masking policy.

Mapped Hive table has a row-filter policy.

- Access will be denied. Audit log will show Hive Row-filter policy.

A Ranger Hive policy allows access to the mapped table for the access derived from the original RAZ-S3 access request.

- Access will be granted. If the access was originally granted by S3 policy, the audit log will show Hive policy.

How to full sync the Ranger RMS database

You must perform a full-sync of the Ranger RMS database after upgrading RMS to Cloudera Runtime 7.1.8 or above.

About this task

Performing a full-sync in Ranger RMS truncates all the existing data from RMS tables in the Ranger database. The sync process initiates a fresh synchronization from the Hive Metastore (HMS). All available tables and database metadata download from HMS and persist into the Ranger database.

Prior to Cloudera Runtime 7.1.7, Ranger RMS synchronizes only table metadata. Database metadata mappings were not downloaded from HMS. After migrating from Cloudera versions before Cloudera Runtime 7.1.7 to Cloudera Runtime 7.1.8+, only pre-existing table mappings will be available. Any newly created tables or database mappings will be synchronized in RMS. Any pre-existing database mappings will not be present.

Important: To ensure the pre-existing databases are mapped correctly, you must perform a full-sync after upgrading RMS to Cloudera Runtime 7.1.8 or above versions.

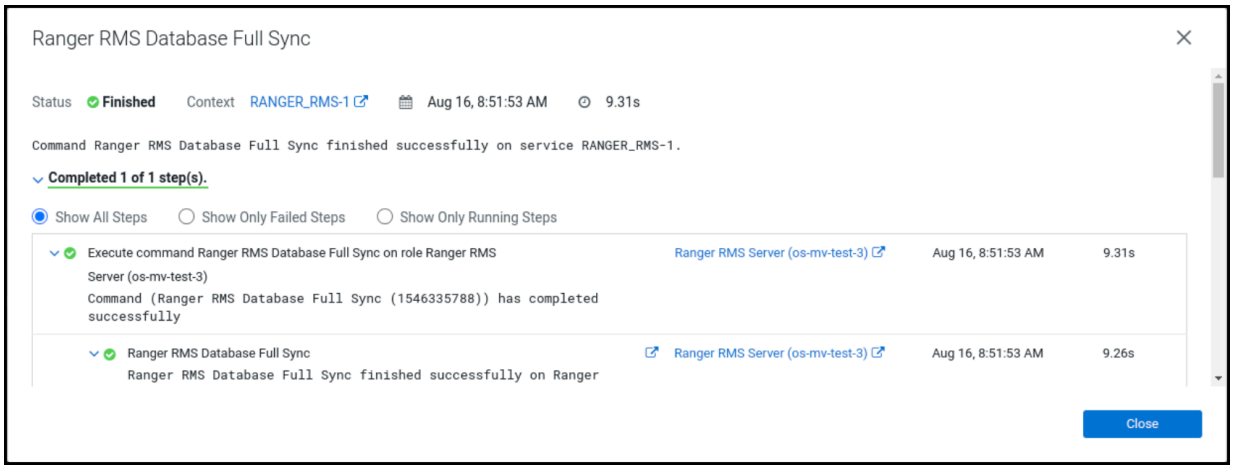
Before you begin

Have a cluster with Ranger and Ranger RMS service up and running.

Procedure

1. In Cloudera Manager Ranger_RMS Actions , click Stop.
2. Ensure that all Ranger RMS roles are in a stopped state before proceeding with the next steps.
3. In Ranger_RMS Actions , click Ranger RMS Database Full Sync.
4. Confirm that the command completed successfully, as shown in the following example:

Figure 4: Confirming successful Ranger RMS database full sync



Note: You can also confirm the text - '[I] Truncate operation executed successfully on Ranger RMS tables' from the stderr log of the command from Cloudera Manager UI.

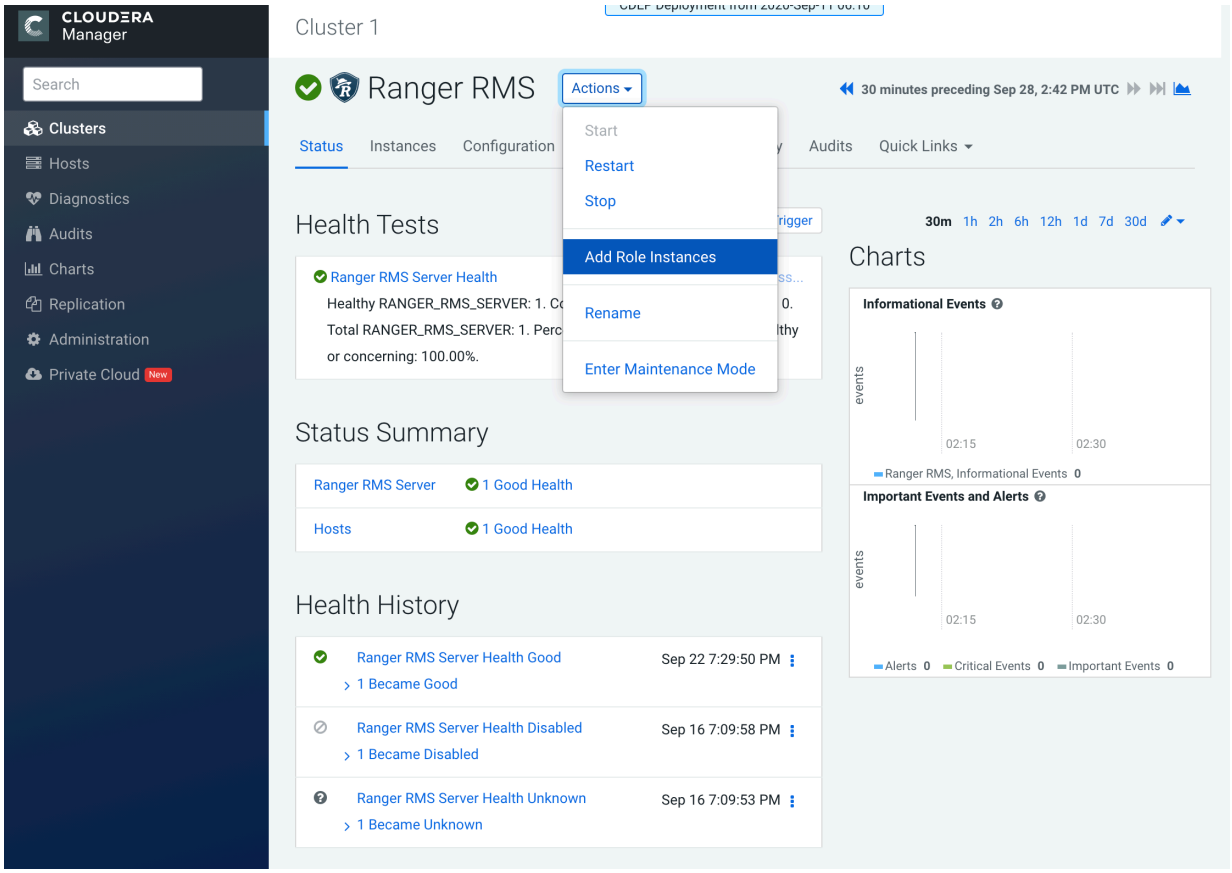
5. In Cloudera Manager Ranger_RMS , click Start.

Configuring High Availability for Ranger RMS (Hive-HDFS ACL-Sync)

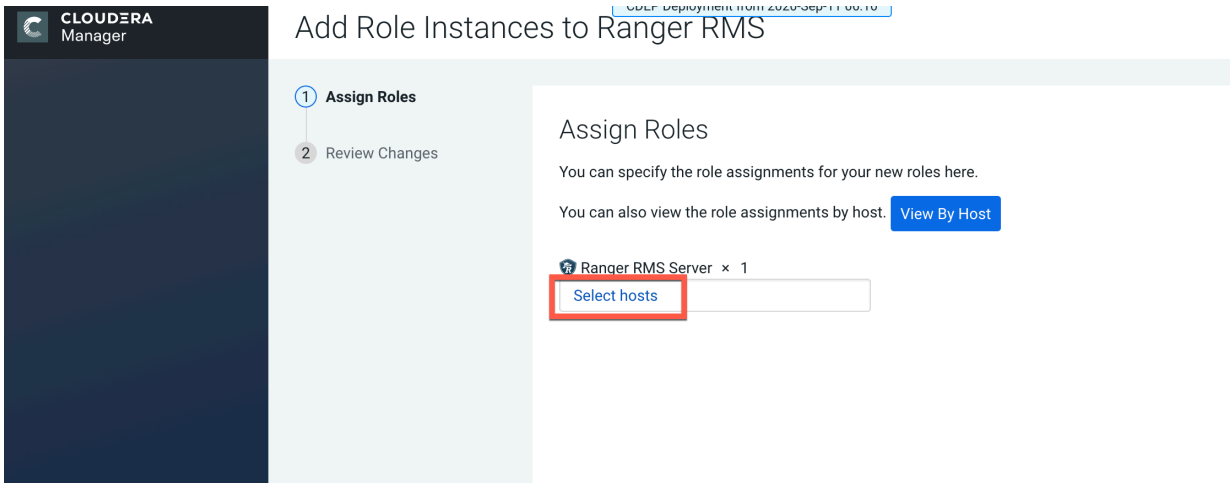
Use the following steps to configure high availability for the Ranger Resource Mapping Server (RMS) and Hive-HDFS ACL Sync.

Procedure

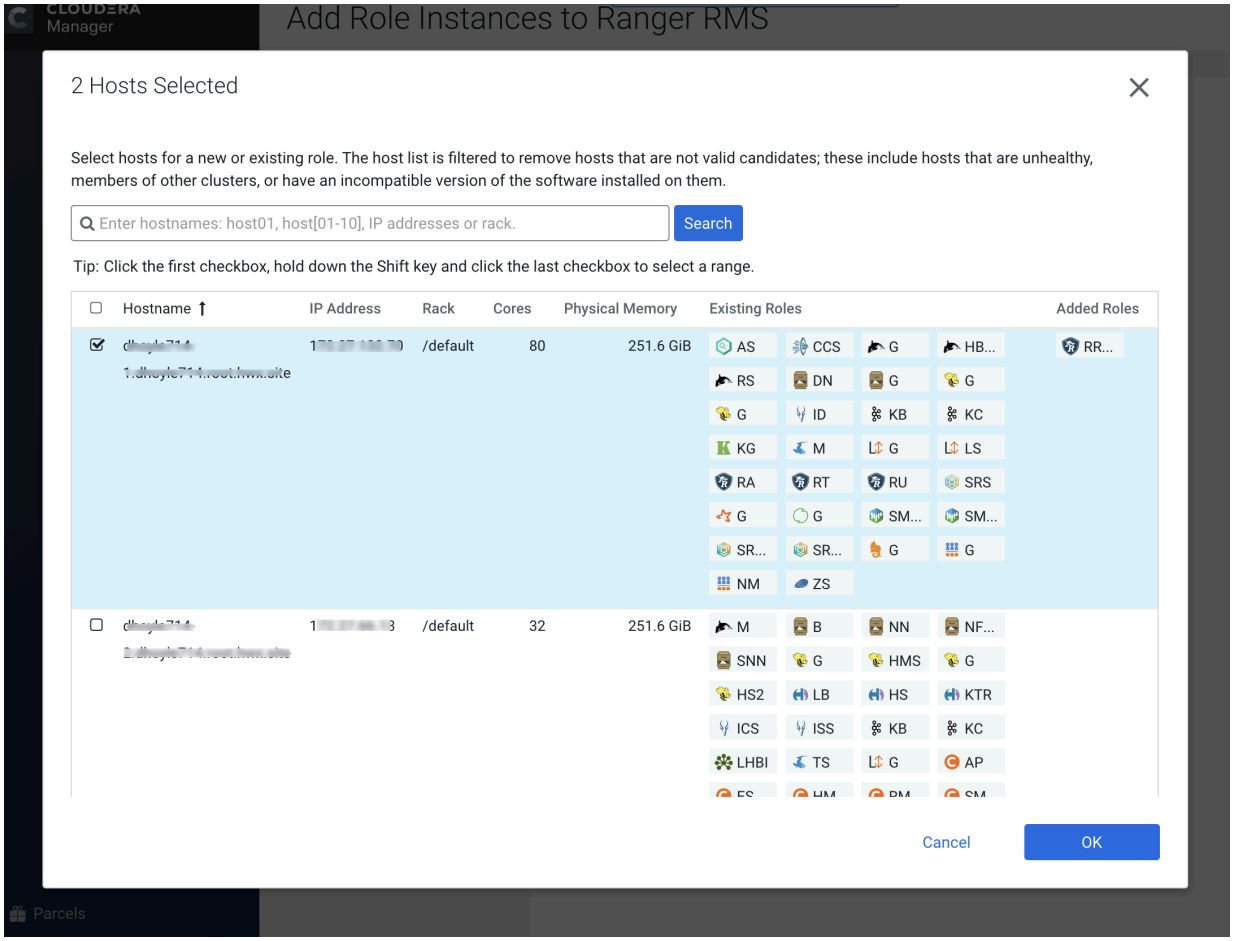
1. In Cloudera Manager, select Ranger RMS, then select Actions > Add Role Instances.



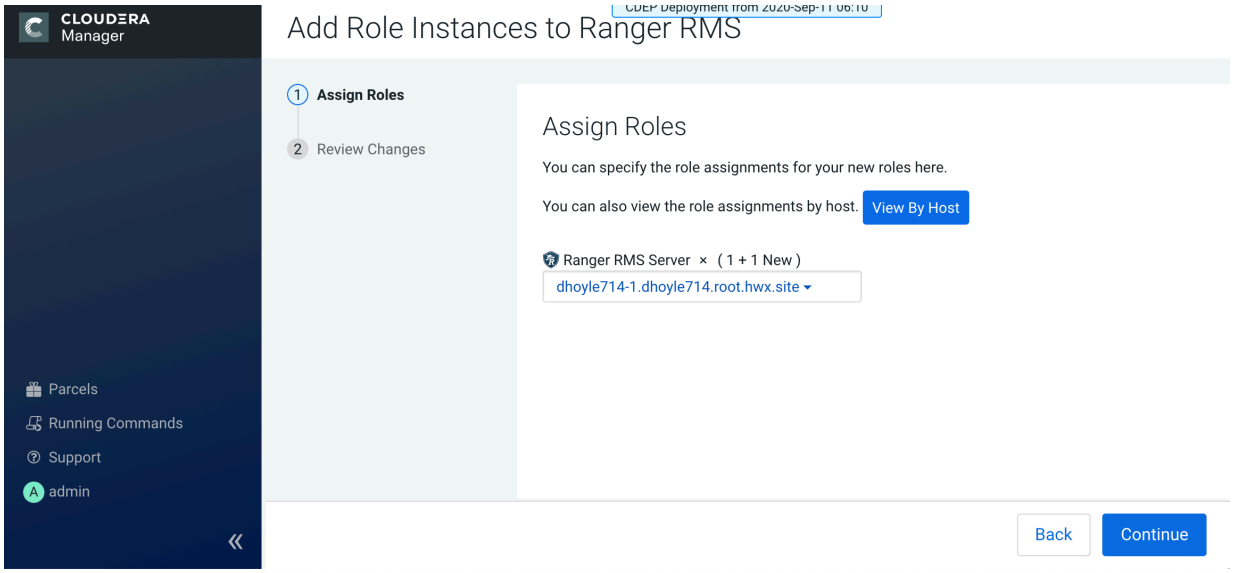
2. On the Add Role Instances page, click Select hosts.



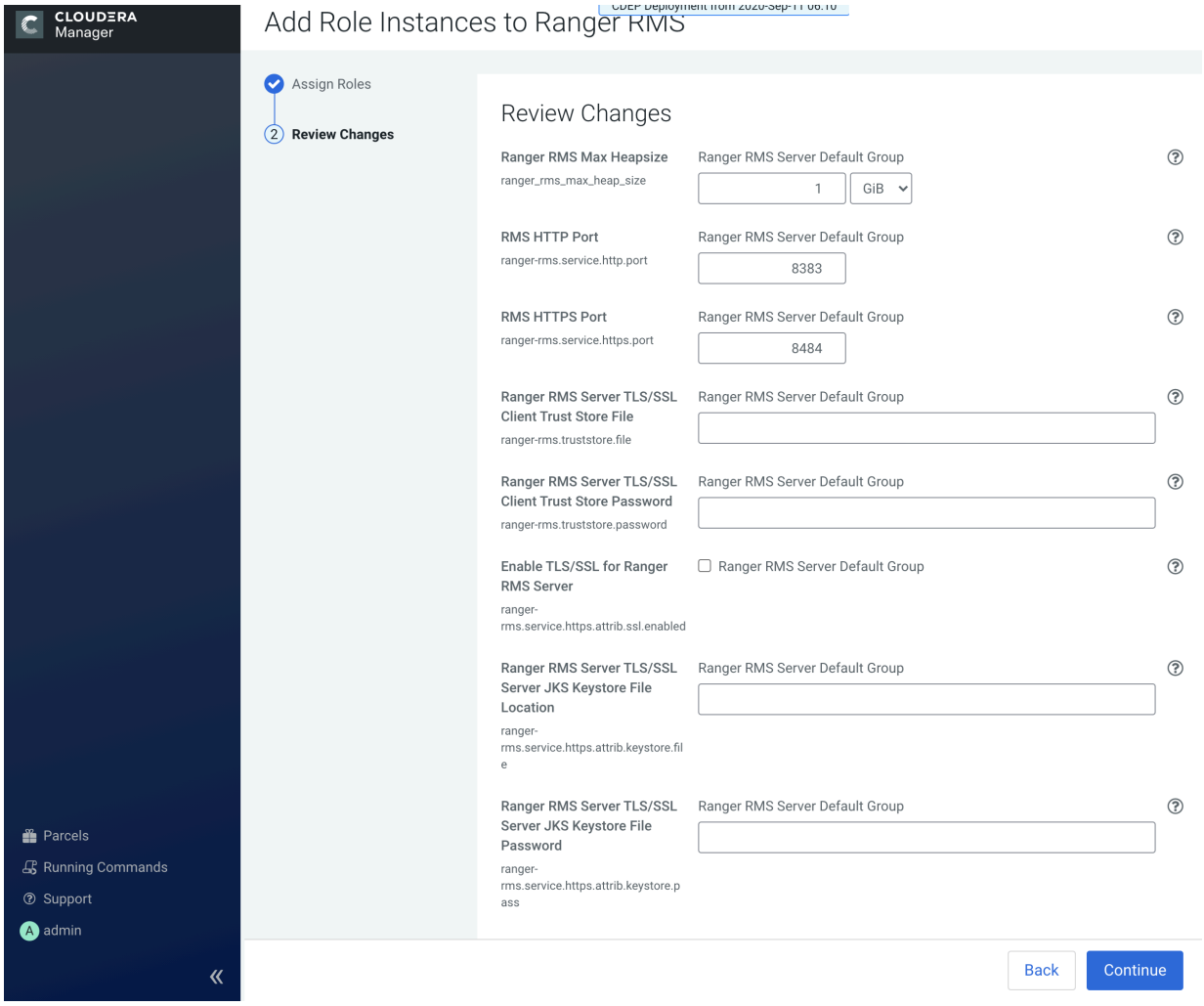
3. On the selected hosts page, select a backup Ranger RMS host. A Ranger RM (RR) icon appears in the Added Roles column for the selected host. Click OK to continue.



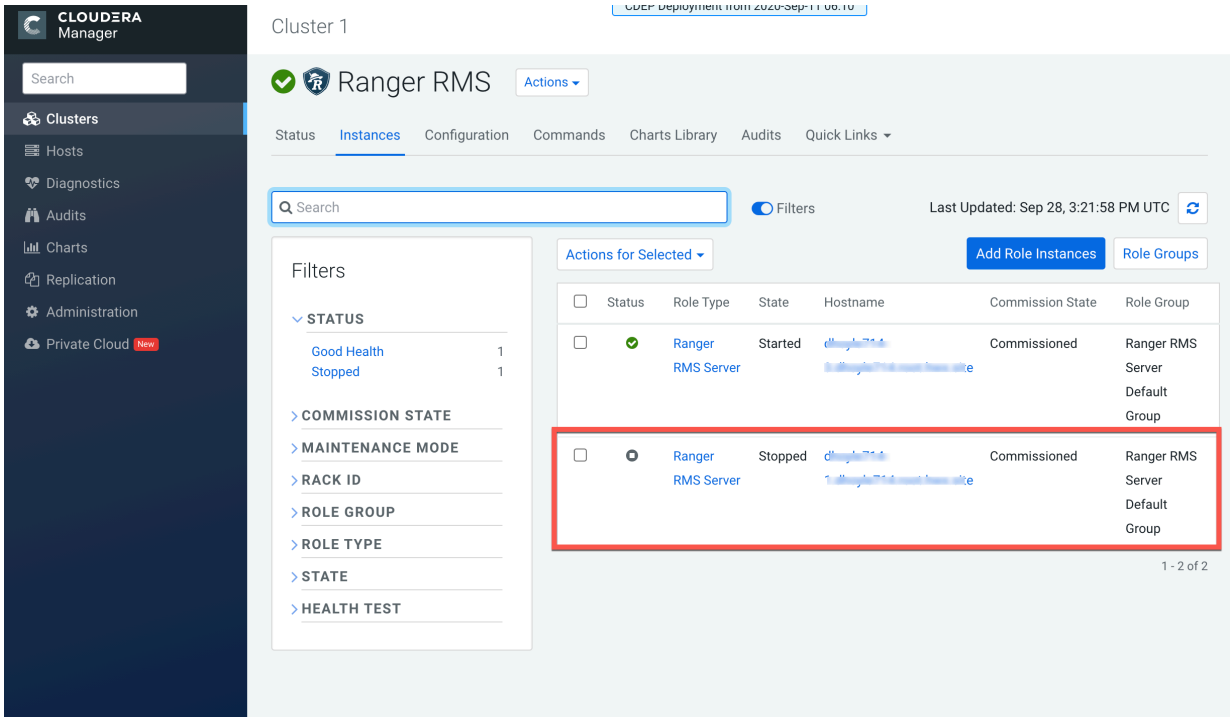
4. The Add Role Instances page is redisplayed with the new backup host. Click Continue.



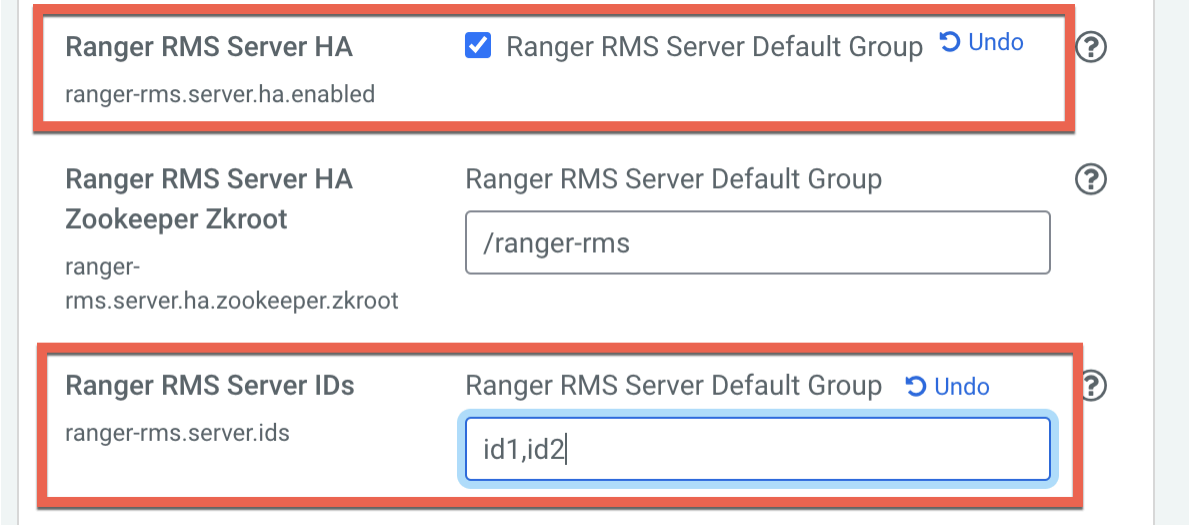
5. Review the settings on the Review Changes page, then click Continue.



6. The new role instance appears on the Ranger RMS page.



7. In Cloudera Manager, select Ranger RMS, then click Configuration.
   a) Select the Ranger RMS Server HA checkbox. In the Ranger RMS Server IDs box, add a comma-separated list of IDs for each RMS server.



8. Click Save Changes, then Click the Restart icon.
9. On the Stale Configurations page, click Restart Stale Services.
10. On the Restart Stale Services page, select the Re-deploy client configuration checkbox, then click Restart Now.
11. A progress indicator page appears while the services are being restarted. When the services have restarted, click Finish.

Configuring Ranger RMS (Hive-HDFS/Hive-OZONE/Hive-S3 ACL Sync)

Ranger Resource Mapping Server (RMS) should be fully configured after installation. This topic provides further information about RMS configuration settings and workflows.

Important configuration information

- Ranger RMS enables OZONE/HDFS/S3 access through Ranger Hive policies. Ranger RMS must be configured with the names of OZONE, HDFS, S3 and Hive services (AKA Repos). In your installation, there may be multiple Ranger services created for OZONE, HDFS, S3, and Hive. These can be seen from the Ranger Admin Web UI. RMS ACL sync is designed to work on a specific pair of HDFS<->Hive, OZONE<->Hive, and S3<->Hive Ranger services. Therefore, it is important to identify those service names before Ranger RMS is installed. These names should be configured during the installation of Ranger RMS. The default value for Ranger HDFS service name is cm_hdfs, for Ranger OZONE service name is cm_ozone, for Ranger S3 service name is cm_s3, and for the Ranger Hive service the default name is cm_hive.

  Note: Default services (also known as Repos) for Hive, HDFS, Ozone, and S3 must be present in the Ranger Admin DB for RMS to function correctly. If any service is missing in Ranger Admin, then navigate to Cloudera Manager Ranger Actions and run the `Setup Ranger Plugin Service` command. This will create the default services that are missing or have been deleted.
- Before starting the Ranger RMS installation, ensure that the Hive service identified in the installation above allows the rangerrms user select access to all tables in all databases in default (no-zone), as well as in all security-zones for the Hive service.
- In case of custom kerberos principal/user, ensure that the Hive service identified in the installation above allows the rangerrmsfoo0 (custom) user select access to all tables in all databases in default (no-zone), as well as in all security-zones for the Hive service.
- By default, Ranger RMS tracks only external tables in Hive. To configure Ranger RMS to also track managed Hive tables, add the following configuration setting to Ranger RMS:

  ```
  ranger-rms.HMS.map.managed.tables=true
  ```

  Note:

  Locations behind managed tables are granted Read access only, even if the users have Write access at the Hive table level. However, this restriction is not enforced for the hive and impala users.
- To enable RMS for HDFS authorization, go to Cloudera Manager, select HDFS > Configuration > HDFS Service Advanced Configuration Snippet (Safety Valve) for ranger-hdfs-security.xml, then confirm the following settings:

  ```
  ranger.plugin.hdfs.chained.services = cm_hive
  ranger.plugin.hdfs.chained.services.cm_hive.impl = org.apache.ranger.chain
  edplugin.hdfs.hive.RangerHdfsHiveChainedPlugin
  ```
- To enable RMS for OZONE authorization, go to Cloudera Manager, select OZONE > Configuration > OZONE Manager Advanced Configuration Snippet (Safety Valve) for ozone-conf/ranger-ozone-security.xml, then use +Add to add the following properites:

  ```
  ranger.plugin.ozone.chained.services = cm_hive
  ranger.plugin.ozone.chained.services.cm_hive.impl = org.apache.ranger.cha
  inedplugin.ozone.hive.RangerOzoneHiveChainedPlugin
  ```

- To disable RMS for S3 authorization, go to Cloudera Manager Ranger-Ranger RAZ Configuration Advanced Configuration Snippet (Safety Valve) for ranger-raz-conf/ranger-raz-site.xml , then **add empty values** to the following settings:
  - ranger.raz.service-type.S3.chained.services =

    Default value is cm_hive.
  - ranger.raz.service-type.S3.chained.services.cm_hive.impl =

    Default value is org.apache.ranger.raz.chainedplugin.RazS3HiveChainedPlugin.

Note: If any of these configurations are changed after Ranger RMS is started and has synchronized with Hive Metastore, the only way to have Ranger RMS use a new configuration is by performing an “RMS Full Sync”. Please refer to the related topic, "How to full sync the Ranger RMS database" for specific steps.

Advanced configurations

HDFS plugin side configurations

- ranger.plugin.hdfs.mapping.hive.authorize.with.only.chained.policies
  - true: Enforce strict Sentry semantics.
  - false: If there is no applicable Hive policy, let HDFS determine access.
  - Default setting: true
- ranger.plugin.hdfs.accesstype.mapping.read
  - A comma-separated list of hive access types that HDFS "read" maps to.
  - Default setting: select
- ranger.plugin.hdfs.accesstype.mapping.write
  - A comma-separated list of hive access types that HDFS "write" maps to.
  - Default setting: update,alter
- ranger.plugin.hdfs.accesstype.mapping.execute
  - A comma-separated list of hive access types that HDFS "execute" maps to.
  - Default setting: _any
- ranger.plugin.hdfs.mapping.source.download.interval
  - The time in milliseconds between mappings download requests from the HDFS Ranger plugin to RMS.
  - Default setting: 30 seconds

    By default, the RMS plugin checks for new mapping downloads every 30 seconds, based on this configuration. If you have mapping data (found in the hdfs_cm_hive_resource_mapping.json file) of approximately 360MB file size; then performing this operation every 30 seconds could cause an excessive load on the NameNode. After enabling performance logs, we can observe that saveToCache takes 11 seconds and loadFromCache operations take 7 seconds to complete. The cacheing process takes approximately 18~19 seconds to complete, as shown in the following example performance logs:

    ```
    DEBUG org.apache.ranger.perf.resourcemapping.init: [PERF] RangerMappingR
    efresher.loadFromCache(serviceName=cm_hive): 7449
    DEBUG org.apache.ranger.perf.resourcemapping.init: [PERF] RangerMappin
    gRefresher.saveToCache(serviceName=cm_hive): 11787
    ```

    In this case, you should adjust the frequency of download RMS mappings to at least 18*2= 36 seconds. A more conservative value = 45 seconds. In this way, you can tune RMS configurations to optimize performance in the NameNode plugin.

- ranger.plugin.hdfs.bypass.chained.plugin.evaluation.if.access.is.determined
  - boolean valued configuration variable, default value = false
  - When it is set to true, the chained-plugin (in our case, RMS code for looking up and enforcing Hive policies) is NOT executed if a matching Ranger HDFS policy was found for the accessed HDFS location and that policy determined the access result (either ALLOW or DENY).

Hive service configuration

- ranger.plugin.audit.excluder.users
  - This configuration, added in the Hive service-configs, lists the users whose access to Hive or Hive Metastore does not generate audit records. There may be a large number of audit records created when "rangerrms" makes requests to the Hive Metastore when downloading Hive table data. By adding the "rangerrms" user to the comma-separated list of users in this configuration, such audit records will not be generated.

OZONE plugin side configurations

- ranger.plugin.ozone.mapping.source.download.interval
  - The time in milliseconds between mappings download requests from the OZONE Ranger plugin to RMS.
  - Default setting: 30 seconds
- ranger.plugin.ozone.privileged.user.names
  - Default setting : admin,dpprofiler,hue,beacon,hive,impala
- ranger.plugin.ozone.service.names
  - Default setting : hive,impala

Note: The comma-separated lists that you define for ozone privileged user names and service names are users that, based on default hive policies, have all access for all Hive resources. Therefore, for these users, checking Hive policies when they access storage locations which map to Hive resources is unnecessary, and may cause access violations if masking/row-filtering policies are configured for public group.

OZONE to HIVE access type mapping for Table and Database

Ozone ACL	HIVE ACL For Table	HIVE ACL For Database
read	select	_any
write	update	update
create	create	create
list	select	select
delete	drop	drop
read_acl	select	select
write_acl	update	update

- ranger.plugin.ozone.accesstype.mapping.<ozone_acl>
  - This config is used to map Ozone access types to Hive Table access types. Modify the config <ozone_acl> as per the ozone access mentioned “OZONE to HIVE access type mapping for Table and Database”
  - Please refer to the “HIVE ACL for Table” column for default configuration with respect to Ozone ACL.
  - For example :

    ranger.plugin.ozone.accesstype.mapping.read = select

- ranger.plugin.ozone.db.accesstype.mapping.<ozone_acl>
  - This config is used to map Ozone access types to Hive database access types. Modify the config <ozone_acl> as per the Ozone access mentioned “OZONE to HIVE access type mapping for Table and Database”.
  - Please refer to the “HIVE ACL for Database” column for default configuration with respect to Ozone ACL.
  - For example :

    ranger.plugin.ozone.db.accesstype.mapping.read = _any

S3 plugin side configurations

- ranger.raz.service-type.s3.mapping.hive.authorize.with.only.chained.policies
  - true: Enforce strict Sentry semantics.
  - false: If there is no applicable Hive policy, let S3 determine access.
  - Default setting: false
- ranger.raz.service-type.s3.accesstype.mapping.read
  - A comma-separated list of Hive access types that S3 "read" maps to.
  - Default setting: select
- ranger.raz.service-type.s3.accesstype.mapping.write
  - A comma-separated list of Hive access types that S3 "write" maps to.
  - Default setting: update,alter
- ranger.raz.service-type.s3.privileged.user.names
  - Default setting: admin,dpprofiler,hue,beacon,hive,impala

    Note: The comma-separated lists that you define for S3 privileged user names and service names are users that, based on default Hive policies, have all access for all Hive resources. Therefore, for these users, checking Hive policies when they access storage locations that map to Hive resources is unnecessary and may cause access violations if masking/row-filtering policies are configured for public group.
- ranger.raz.service-type.s3.mapping.source.download.interval
  - The time in milliseconds between mapping download requests from the S3 Ranger plugin to RMS.
  - Default setting: 30 seconds

    By default, the RMS plugin checks for new mapping downloads every 30 seconds, based on this configuration. If you have mapping data (found in the raz_cm_hive_resource_mapping.json file) of approximately 360MB file size, then performing this operation every 30 seconds could cause an excessive load on the NameNode. After enabling performance logs, you can observe that saveToCache takes 11 seconds and loadFromCache operations take 7 seconds to complete. The cacheing process takes approximately 18-19 seconds to complete, as shown in the following example performance logs:

    ```
    DEBUG org.apache.ranger.perf.resourcemapping.init: [PERF] RangerMappingR
    efresher.loadFromCache(serviceName=cm_hive): 7449
    DEBUG org.apache.ranger.perf.resourcemapping.init: [PERF] RangerMappin
    gRefresher.saveToCache(serviceName=cm_hive): 11787
    ```

    In this case, you should adjust the frequency of downloading RMS mappings to at least 18*2= 36 seconds. A more conservative value = 45 seconds. In this way, you can tune RMS configurations to optimize performance in the RAZ S3 chained plugin.
- ranger.plugin.hdfs.bypass.chained.plugin.evaluation.if.access.is.determined
  - boolean valued configuration variable, default value = false
  - When it is set to true, the chained-plugin (in our case, RMS code for looking up and enforcing Hive policies) is NOT executed if a matching Ranger HDFS policy is found for the accessed HDFS location and that policy determines the access result (either ALLOW or DENY).

RMS side configurations

Note: Changes to any of these requires that RMS is stopped, all rows are deleted from RMS database table x_rms_mapping_provider and RMS is restarted. On restart, RMS downloads complete table data from RMS, which may take a significant amount of time depending on the number of tables in HMS.

- ranger-rms.HMS.source.service.name
  - The Ranger HDFS service name (default: cm_hdfs).
- ranger-rms.HMS.target.service.name
  - The Ranger Hive service name (default: cm_hive).
- ranger-rms.HMS.map.managed.tables
  - true – Track managed and external tables.
  - false – Track only external tables.
  - Default setting: false
- ranger-rms.polling.notifications.frequency.ms
  - The time in milliseconds between polls from RMS to HMS for changes to tables.
  - Default setting: 30 seconds
- ranger-rms.HMS.source.service.name.ozone
  - The Ranger OZONE service name
  - Default setting : cm_ozone
- ranger-rms.HMS.source.service.name.s3
  - The Ranger S3 service name (default: cm_s3)
- ranger-rms.supported.uri.scheme
  - A comma-separated list of uri schemes supported by RMS
  - Default setting: hdfs,s3a,o3fs,ofs

Note: Ranger RMS supports RMS Hive-S3 Acl-Sync feature on Cloudera Base on premises from 7.3.2.0 release onwards. Therefore, s3a file scheme is now supported by RMS.

Configuring HDFS plugin to view permissions through getfacl interface

You can configure the HDFS plugin to view user accees permissions in a manner similar to the HDFS `getfacl` command after migratng from CDH to Cloudera cluster.

Introduction

Ranger RMS (Hive-HDFS ACL Sync) feature replaces a similar feature previously supported in Sentry. Ranger RMS (Hive-HDFS ACL Sync) is implemented through RMS module which consists of RMS server and chained-plugin and corresponding storage plugin - HDFS/Ozone. Ranger RMS (Hive-HDFS ACL Sync) supports viewing access permissions on storage locations across:

- Ranger policies for storage locations
- Ranger policies for Hive entities
- any native storage ACLs

Ranger RMS (Hive-HDFS ACL Sync) is considered essential for those customers migrating from CDH to Cloudera clusters who also used Sentry ACL-sync feature.

Ranger RMS (Hive-HDFS ACL Sync) addresses the following customer requirements to support:

- an operational need for end-users with no access to Ranger-admin to check the permissions on a HDFS resource

- a tool-set that can verify if the Sentry access permissions are correctly migrated

Sentry implemented an HDFS command `getfacl` that provided access permissions on a given resource to the end-user in a structured, easy-to-understand form. Ranger RMS (Hive-HDFS ACL Sync) is designed to offer the same user experience to the end-users.

Design

The Ranger policy-engine supports an API that, for a resource, returns a static view of the permissions allowed for all users, groups and roles. The view of the access permissions is specific to the component's access model. In other words, for Hive resource the permissions as defined in the Hive service definition, and for HDFS resource, the view consists of read/write/execute permissions. This functionality is leveraged further, by extending it to chained plugins.

Ranger RMS (Hive-HDFS ACL Sync) internally chains HDFS plugin with Hive plugin to provide composite evaluation of access for an HDFS location that maps to a Hive entity. With this extension, the static determination of accesses for a resource considers any chained plugins (and corresponding policy-engine) in its computation. This also requires conversion between Hive access types and HDFS access types, and combining of accesses by multiple (in this case HDFS and Hive) plugins.

The front-end for accessing this static permission map is the `getfacl` command supported by HDFS. Using this front-end obviates the need for any additional configuration set-up (other than one that enables/disables this feature). It also gives Sentry users (almost) identical experience when interrogating accesses supported for a resource.

Configuration

There are two new configuration parameters introduced for this feature. Both are specified using the safety valve mechanism on HDFS (ranger-hdfs-security.xml).

1. Go to Cloudera Manager HDFS Configuration .
2. In Search, type ranger-hdfs-security.xml.
3. In HDFS Service Advanced Configuration Snippet (Safety Valve) for ranger-hdfs-security.xml, add the following properties/values:
   is.facl.interception.enabled=true

   (Default:false): If true, this feature is activated.

   ranger.plugin.hdfs.policyengine.option.disable.role.resolution=true

   (Default:true) : If false, the roles are resolved to their constituent users and groups for reporting the access permissions.
4. Click Save Changes (CTRL+S).
5. Restart services with stale configurations.

Limitations

Ranger evaluates access based on static and dynamic (time-based, accessing IP-based, condition-based) policies. The static view of the access permissions given only a resource, is not a complete and always accurate picture. However, for the customers moving from Sentry to Ranger, this may not be an issue.

Migrating Ranger RMS server role instance to a new host

You can move the Ranger Resource Mapping Server (RMS) server role instance for an existing Ranger RMS service from one host to another using Cloudera Manager.

About this task

In some cases, for example, after upgrading your servers, you may want to migrate a Ranger RMS server role instance to a new host. This procedure describes how to move a Ranger RMS server role instance from an existing cluster host to another cluster host.

Note: If you enabled manual SSL on this cluster, you must update the SSL configurations when adding a new role.

Procedure

1. Go to Cloudera Manager.
2. Back up your configurations using the host templates.
   a) Go to Cloudera Manager Hosts Host Templates .
   b) Click Create in the **Host Templates** page.
   c) Enter Template Name, and then select the Ranger RMS Server under the Ranger_RMS service.
   d) Click Create.

   After a template is created for the Ranger RMS server, all the user configurations will be saved in that template.

   Note: Cloudera recommends saving the actual configuration files used on the host for the Ranger RMS server. It is better to verify the configurations on the newly added role on the new host with the saved old configuration files (for example, ranger-rms-site.xml)
3. Go to Cloudera Manager Ranger_RMS service.
4. Go to the Instances tab.
5. Select the role that you want to migrate to a new host.
6. Select Stop from the Actions for Selected dropdown.
7. After the selected role stops, select Delete from the same Actions for Selected dropdown.
8. Click Add Role Instances.

   The **Assign Roles** page appears.
9. Select a new host for the Ranger RMS server, and click OK to add the role.
10. Click Continue.
11. Review the settings on the **Review Changes** page, and click Finish.

    The new role instance appears on the Ranger RMS page.
12. Select the newly added role instance, and select Start from the same Actions for Selected dropdown.
13. Restart the cluster from the Actions dropdown.

    Restarting the cluster removes the stale changes. Cloudera generally recommends running the Ranger RMS Database Full Sync command. To do this, go to the Ranger RMS service page and select Ranger RMS Database Full Sync. However, running this command is not always necessary.

    If you want to add multiple roles to the Ranger RMS server, then see Configuring High Availability for Ranger RMS.

Ranger RMS (Hive-HDFS ACL-Sync) Use Cases

This topic presents a few common use cases for Ranger RMS (Hive-HDFS ACL-Sync).

Use Case 1: RMS Hive policies control access to a table's HDFS directories

Prerequisites:

1. Create a "Customer" Hive table under the default database.
2. Create a "unixuser1" user.
3. User "unixuser1" does not have any policy to allow it access to table "Customer".
4. User "unixuser1" tries to access the Hive data through the hdfs command.

Before setting up RMS:

If HDFS ACLs allow access to the location for Customer table, access will be granted to "unixuser1". The audit log will have "hadoop-acl" as the access enforcer.

After setting up RMS:

Access will not be granted to user "unixuser1". The audit log will not specify denying policy.

Use Case 2: RMS Hive policies propagate tag-based access control on tables to HDFS directories

Prerequisites:

1. Create a "Customer" Hive table under the default database.
2. Create a "unixuser1" user.
3. The tag "SPECIAL_ACCESS" is associated with the "Customer" table.
4. A policy for the tag "SPECIAL_ACCESS" provides Hive select access to "unixuser1".
5. User "unixuser1" tries to read the Hive data through the hdfs command.

Before setting up RMS:

If HDFS ACLs allow access to the location for "Customer" table, access will be granted to ‘"unixuser1". The audit log will have "hadoop-acl" as the access enforcer.

After setting up RMS:

Access will be granted by tag-based policy for "SPECIAL_ACCESS".

Use Case 3: RMS Hive policies propagate tag-based masking on tables and denies access to HDFS directories

Prerequisites:

1. Create a "Customer" Hive table under the default database.
2. Create a "unixuser1" user.
3. The tag "SPECIAL_ACCESS" is associated with the "Customer" table.
4. A policy for the tag "SPECIAL_ACCESS" provides Hive select access to "unixuser1".
5. A masking policy for the "Customer" table is set up so that for "unixuser1" a column "SSN" is redacted.
6. User "unixuser1" tries to read the Hive data through the hdfs command.

Before setting up RMS:

If HDFS ACLs allow access to the location for Customer table, access will be granted to "unixuser1". The audit log will have "hadoop-acl" as the access enforcer.

After setting up RMS:

Access will be denied by the masking policy.

Use Case 4: RMS Hive policies take precedence over HDFS policies

Prerequisites:

1. Create a "Customer" Hive table under the default database.
2. Create a "unixuser1" user.
3. User "unixuser1" has a HDFS policy allowing read access.
4. User "unixuser1" does not have any policy to allow it access to the "Customer" table.

5. User "unixuser1" tries to access the Hive data through the hdfs command.

Before setting up RMS:

Access will be granted by the Ranger HDFS policy.

After setting up RMS:

Access will not be granted to the "unixuser1" user. The audit log will not specify a denying policy.