Cloudera Runtime 7.2.13

Configuring and Using Hive-HDFS ACL Sync

Date published: 2020-07-28
Date modified: 2021-12-13

Legal Notice

Contents

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.

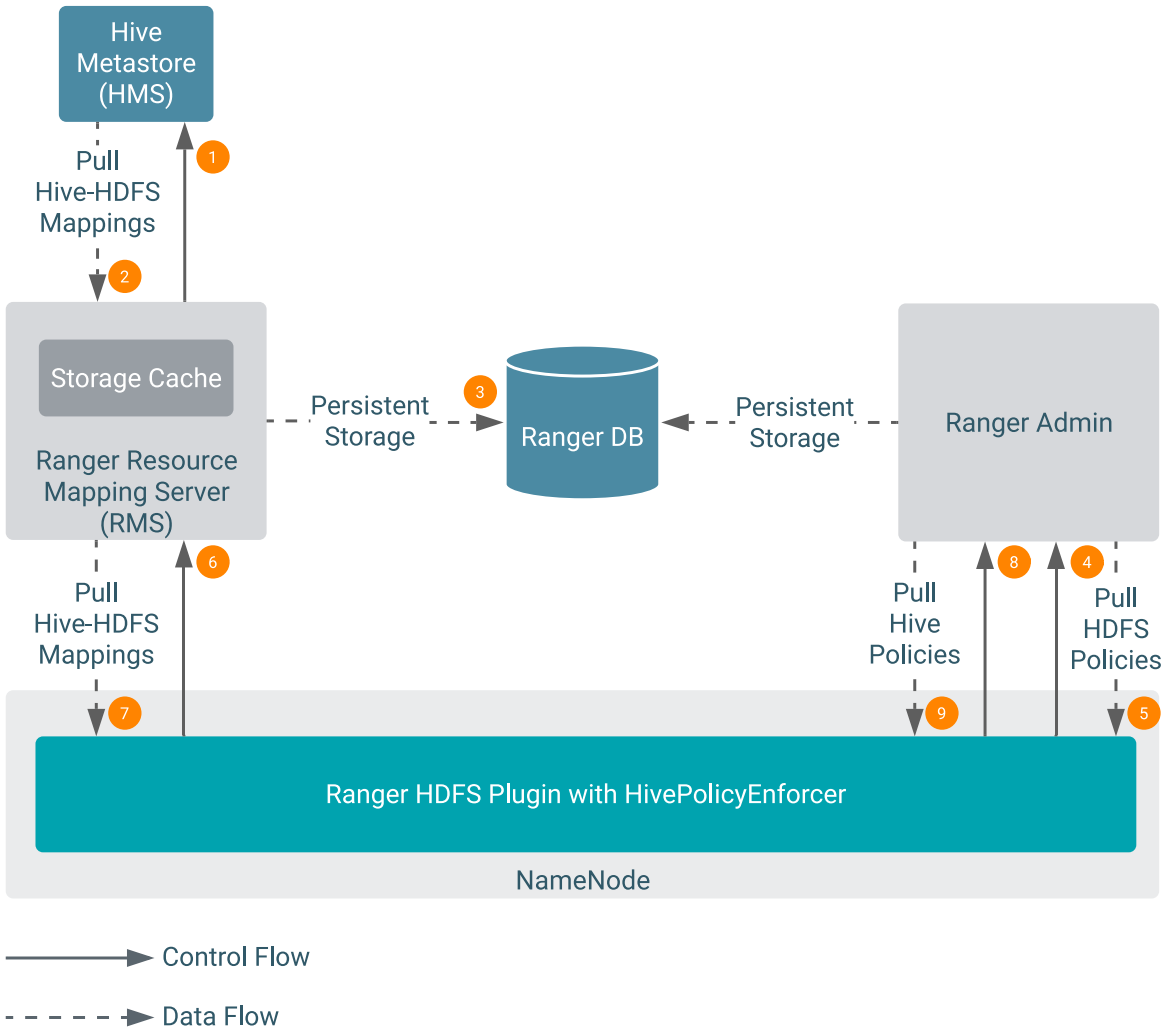
As a result, whenever a change is made on a Hive table policy, the data admin should make a consistent change in the corresponding HDFS policy. Failure to do so could result in security and/or data exposure issues. Ideally the data admin would set a single table policy, and the corresponding file access 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 CDP 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=[create, 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 CDP Private Cloud Base 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.
- If a Private Cloud Base deployment supports multiple logical HMS with a single Ranger, Ranger 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 RMS 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 CDP-7.1.7, Ranger RMS synchronizes only table metadata. Database metadata mappings were not downloaded from HMS. After migrating from CDP 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.

Related Information

Installing Ranger RMS

Analyzing Ranger RMS resources

You can use the tools described in this topic to analyze Ranger RMS issues.

About this task

In CDP 7.1.7 Ranger RMS installations, we have observed numerous identical resource-signatures with different IDs for a single Hive resource due to which evaluation of access policies can be slow. This topic provides a tool set you can use to analyze the database as well as the resource-mapping cache (.json) file to help identify resource data if such a problem exists. This tool set can also help determine whether the source of such duplicate entries is:

- incorrect entries in the Ranger database tables,
- incorrect in-memory cache in Ranger RMS server, or
- incorrect processing of resource-mapping deltas on the plugin side.

Procedure

1. The resource_mapping.json file can be located in the active NameNode host at /var/lib/ranger/hdfs/policy-cache/hdfs_cm_hive_resource_mapping.json path. To find duplicate resource-signatures in resource-mapping.json file:
   a) Update the json file in readable format

   ```
   python -m json.tool hdfs_cm_hive_resource_mapping.json > pretty_resource
   _mapping.json
   ```

   b) To find duplicate entries, use the following grep command:

   ```
   grep -o -E 'resourceSignature.*,' pretty_resource_mapping.json | sort |
   uniq -c | grep -v '1 resourceSignature' > outputFile.txt
   ```

2. To find duplicate resource-signature entries in x_rms_service_resource table,
   a) Connect to Ranger database.

   To check configured database; go to Cloudera Manager Ranger Configuration Database .

   use either of the following queries:

   ```
   select resource_signature, COUNT(resource_signature) from
   x_rms_service_resource
   group by resource_signature HAVING COUNT(resource_signature) > 1;
   ```

   or

   ```
   SELECT a.id, a.service_resource_elements_text,
   a.resource_signature from x_rms_service_resource a where
   a.resource_signature in
   (select resource_signature from x_rms_service_resource
   group by resource_signature HAVING COUNT(resource_signature) >
   1) ORDER BY a.resource_signature;
   ```

3. To find total count of duplicate entries,
   a) use the following query:

   ```
   select sum(occurrences) as total_occurrences from (select
   resource_signature, COUNT(resource_signature) as occurrences
   from x_rms_service_resource
   group by resource_signature HAVING COUNT(resource_signature) > 1) tbl1;
   ```

How to full sync the Ranger RMS database

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

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 CDP-7.1.7, Ranger RMS synchronizes only table metadata. Database metadata mappings were not downloaded from HMS. After migrating from CDP versions before CDP-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.

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

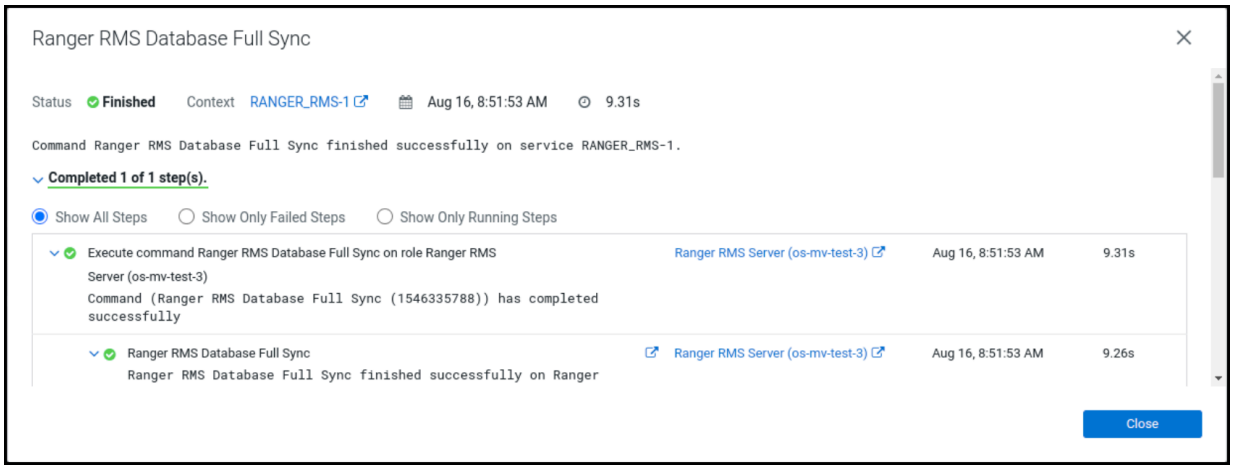
Before you begin

Have a (7.1.8) cluster with Ranger and Ranger RMS service up and running.

Procedure

1. In Cloudera Manager Ranger_RMS Actions , click Stop.
2. In Ranger_RMS Actions , click Ranger RMS Database Full Sync.
3. Confirm that the command completed successfully, as shown in the following example:

Figure 2: 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.

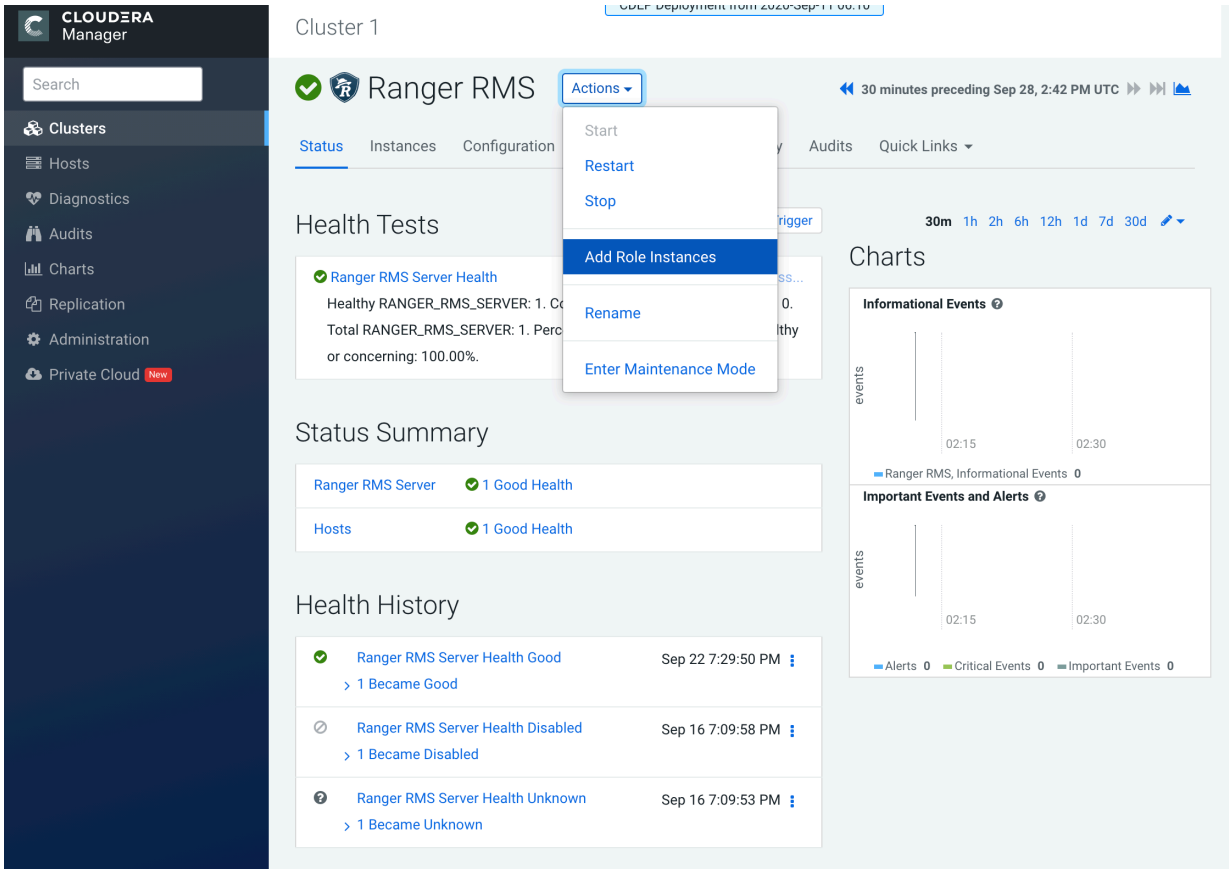
4. In Cloudera Manager Ranger_RMS , click Start.

Configure High Availability for Hive-HDFS ACL Sync

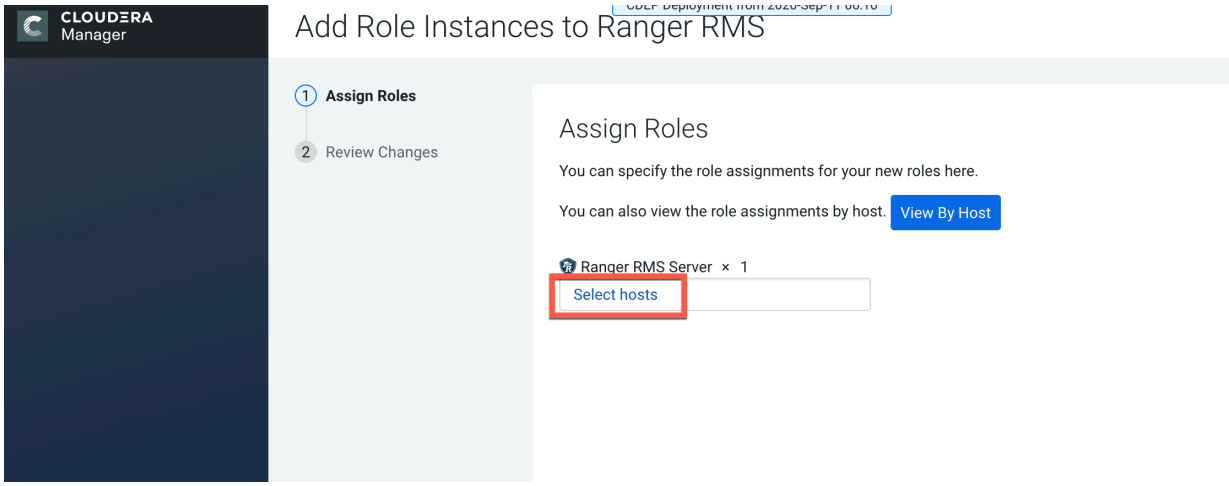
Use the following steps to configure high availability for the Ranger Resource Mapping Server (RMS) and Hive-HDFS ACL Sync.

Procedure

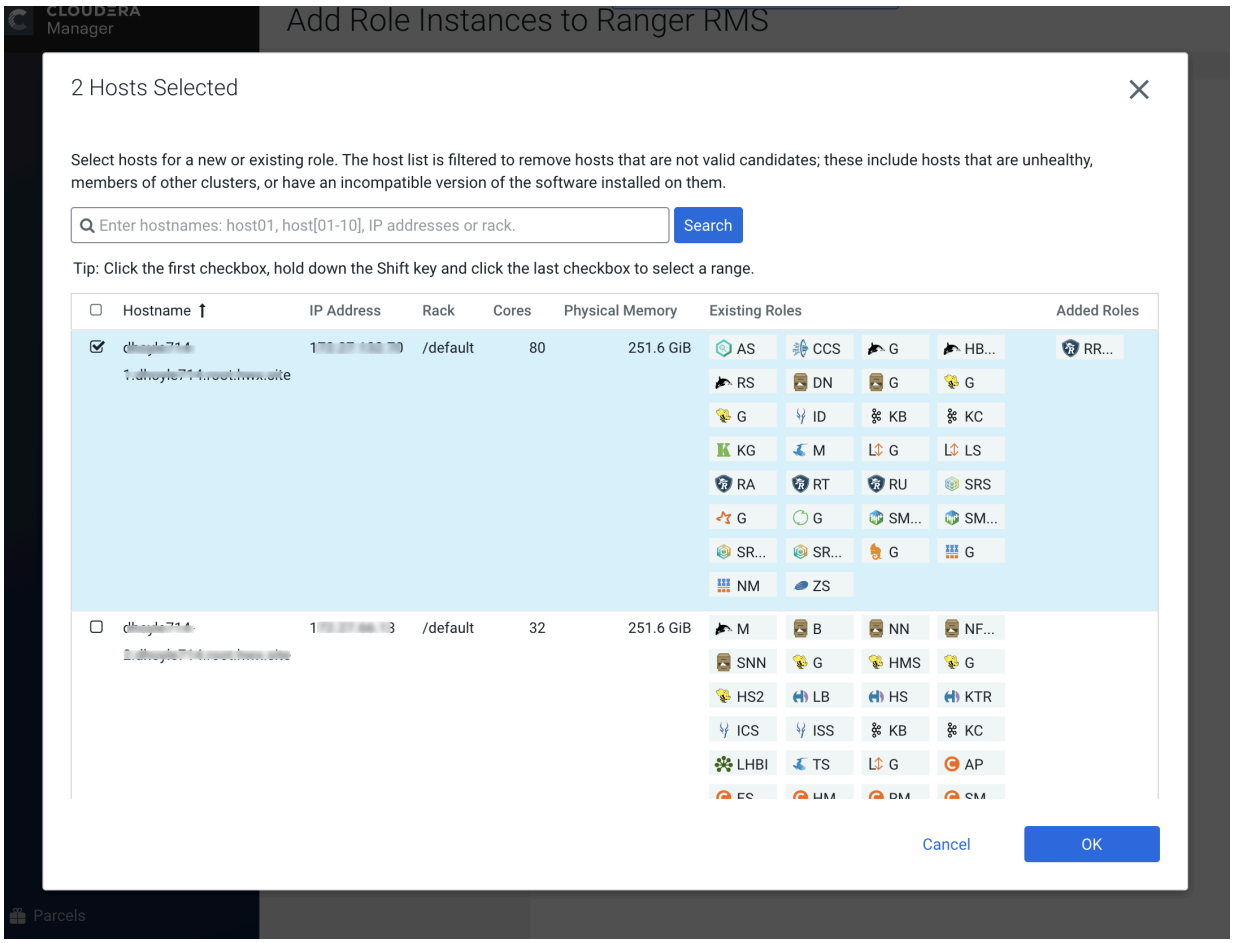
1. In Cloudera Manager, select Ranger RMS, then select Actions > Add Role Instances.



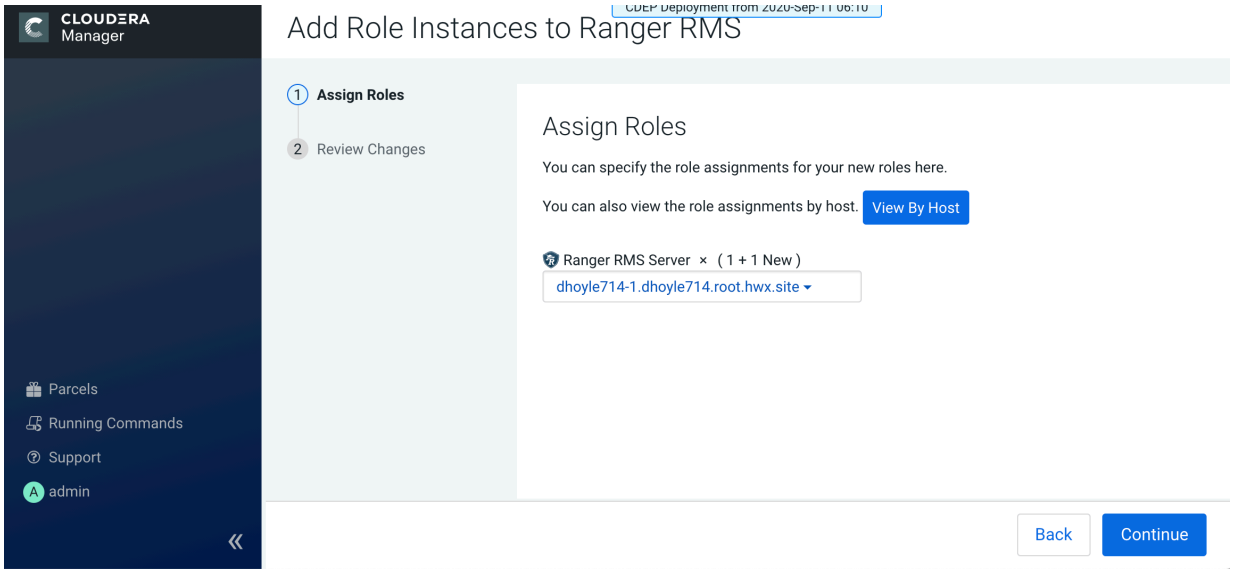
2. On the Add Role Instances page, click Select hosts.



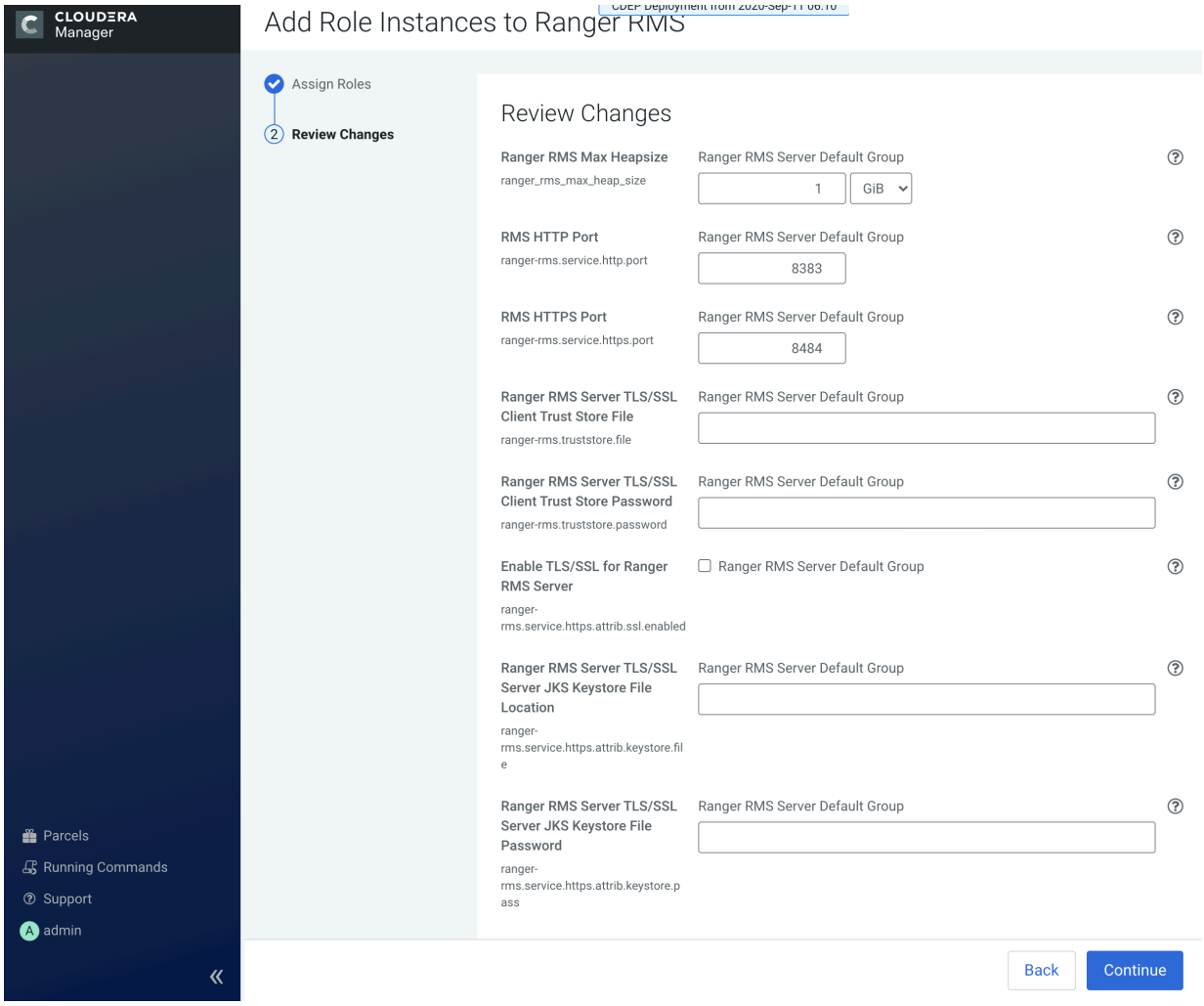
3. On the selected hosts page, select a backup Ranger RMS host. A Ranger RM (RR) icon appears in the Added Roles column for the selected host. Click OK to continue.



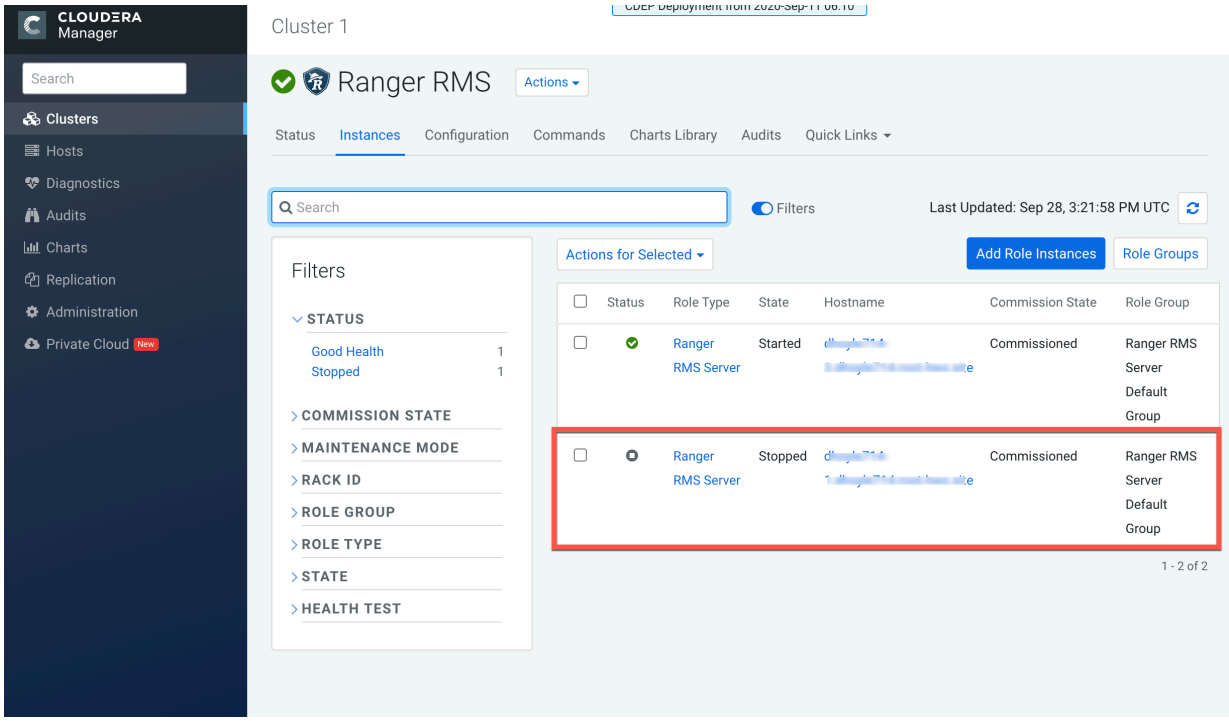
4. The Add Role Instances page is redisplayed with the new backup host. Click Continue.



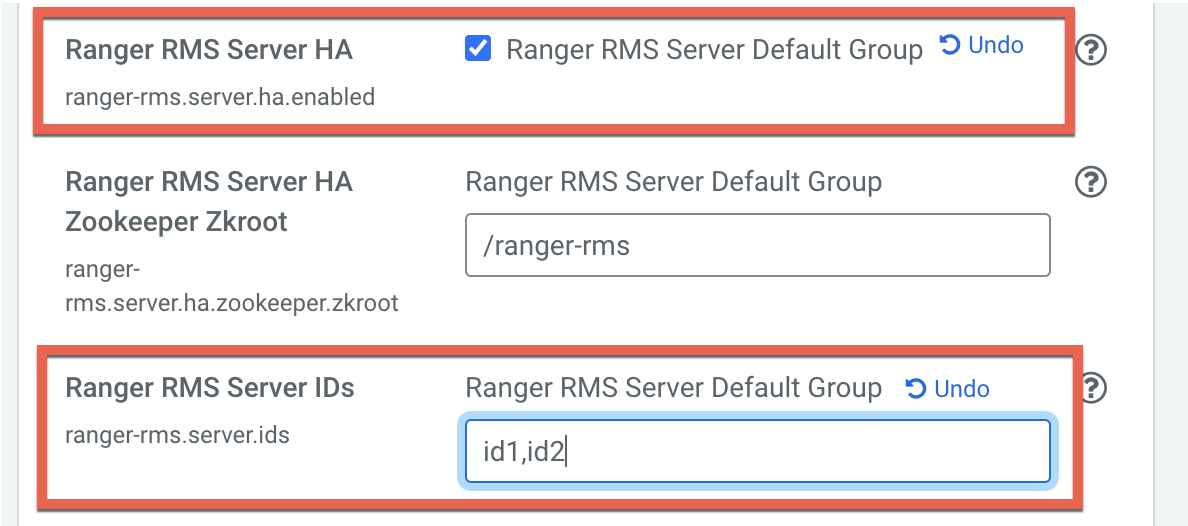
5. Review the settings on the Review Changes page, then click Continue.



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

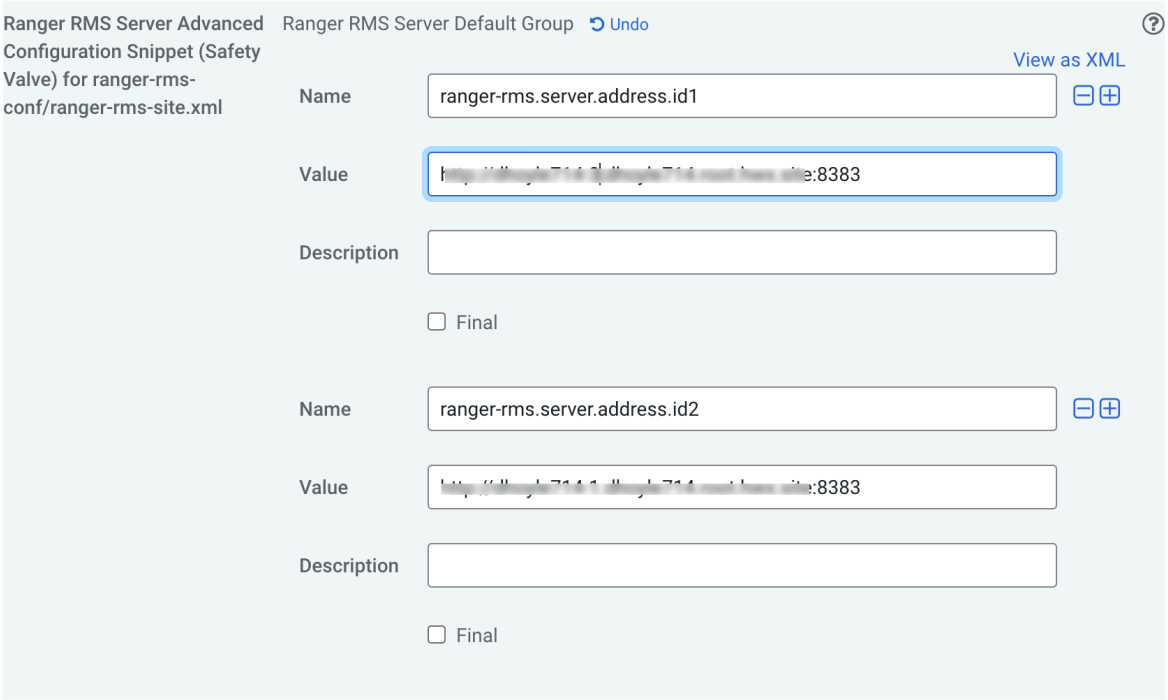


7. In Cloudera Manager, select Ranger RMS, then click Configuration.
   a) Select the Ranger RMS Server HA checkbox. In the Ranger RMS Server IDs box, add a comma-separated list of IDs for each RMS server.



   b) Use the Add (+) icons for the Ranger RMS Server Advanced Configuration Snippet (Safety Valve) for ranger-rms-conf/ranger-rms-site.xml property to add entries for each RMS host with its corresponding server ID.
      - ranger-rms.server.address.id1=<hostname of RMS server for id1>:8383
      - ranger-rms.server.address.id2=<hostname of RMS server for id2>:8383

      Note: If SSL is enabled, use port 8484.



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.

Related Information

Installing Ranger RMS

Configure Hive-HDFS 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 HDFS access via Ranger Hive policies. Ranger RMS must be configured with the names of HDFS and Hive services (AKA Repos). In your installation, there may be multiple Ranger services created for HDFS 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 and 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, and for the Ranger Hive service the default name is cm_hive.
- 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, 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.
- In 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
  ```

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 following these steps:

1. Stop Ranger RMS.
2. Log in to the Ranger RMS database and run delete from x_rms_mapping_provider; to remove the only row from this table.
3. Start Ranger RMS.

On restart, Ranger RMS will resynchronize all data from the Hive Metastore. This may take a significant amount of time depending on the number of Hive tables in the Hive Metastore.

Understanding Ranger policies with RMS

At a high level, the Ranger RMS 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.

Related Information

Installing Ranger RMS

Hive-HDFS ACL Sync Use Cases

This topic presents a few common use cases for 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.

Hive-HDFS ACL Sync Reference

This topic provides reference information for Hive-HDFS ACL Sync.

Debugging common issues

- To reload Ranger RMS mapping from scratch, perform the following steps.
  1. Stop Ranger RMS.
  2. Log in to the Ranger RMS database and run delete from x_rms_mapping_provider; to remove the only row from this table.
  3. Start Ranger RMS.

RMS tables

The following tables are used by Ranger RMS to persist mappings:

- X_rms_mapping_provider
- X_rms_notification
- X_rms_resource_mapping
- X_rms_service_resource

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.

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.

RMS side configurations

Note that 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