Cloudera Runtime 7.3.2

Securing Apache Atlas

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

https://docs.cloudera.com/

Legal Notice

Contents

Securing Atlas

Cloudera Manager manages Apache Atlas as a service, automatically ensuring Atlas can communicate securely with its clients and the services it depends on. Use Cloudera Manager to manage additional Atlas settings; use Apache Ranger to control user access in Atlas.

When you include Atlas as a service in a cluster, Cloudera Manager automatically configures the following settings:

- Ranger plugin. Atlas uses Ranger to determine which users have access to perform actions in Atlas.
- TagSync with Ranger. Atlas passes entity metadata for classifications to Ranger using a Kafka topic.
- Access Policies in Ranger. Default policies are configured for the following users:
  - admin: the initial Atlas administrator user has full access to all Atlas actions, including full access to entity metadata, classifications, business metadata attributes, labels and relationship creation, the ability to create new entity, enumeration, structure, and relationship types, the ability to import, export, and purge metadata from Atlas, and the ability to save searches.
  - dpprofiler: the Data Plane service user has the same extensive privileges as the admin user. These privileges allow integration between Cloudera Data Catalog (Data Steward Studio) and Atlas.
  - beacon: the Cloudera Replication Manager service user has the same extensive privileges as the admin user. These privileges allow Atlas to participate in cluster-level disaster recovery operations.
  - rangertagsync: the TagSync service user has read access to entity metadata, specifically to entity classifications, business metadata, and labels to be used in Ranger tag-based policies.
  - rangerlookup: the Ranger lookup service user has read access to entity metadata, specifically to entity classifications, business metadata, and labels to be used in enforcing Ranger policies.
  - public: all users are granted access to read Atlas entity metadata, classifications, labels, and relationships (such as lineage).
  - {USER}: any user who successfully logs in to Atlas can save searches so they are available in subsequent Atlas sessions.

  You will probably want to update and add to these policies to include users and groups in your organization who will need access to Atlas actions.
- TLS-enabled clusters. Cloudera Manager configures:
  - The option to enable TLS for Atlas (atlas.enableTLS)
  - Keystore file locations and passwords for encrypting client-server communication
  - Trust store location and password for the Atlas server to communicate as a client to other services such as Apache HBase and Apache Solr
  - Trust store location and password for the Atlas gateway role that passes information through Kafka topics.

  Note: In Cloudera on cloud, communication among services within the cluster is configured to use TLS by default. There is no need to manually configure TLS for Atlas in an SDX cluster.
- Kerberos-enabled clusters. Cloudera Manager configures:
  - Principals for Atlas service users
  - Ranger policies to support authentication for Atlas server and hook communication to Apache Kafka
  - Ranger policies to support authentication for the Atlas server to communicate with Solr and HBase

  Note: In Cloudera on cloud, user authentication is managed using Free IPA and Kerberos. There is no need to manually configure authentication settings for Atlas.

Note: Governance through Apache Atlas is just one element of a secure production cluster: Cloudera supports Atlas when it runs on a cluster where enabling Kerberos is optional to authenticate users. Atlas works even without Kerberos enabled.

Setting Atlas session inactivity timeout

For Atlas, you can configure the session inactivity timeout.

In Cloudera Manager for Atlas service, search for the property atlas.session.timeout.secs. By default, the value is -1. You can set the value of this property to 900 (15 seconds).

For example: atlas.session.timeout.secs=900

Related Information

FreeIPA identity management

Configuring TLS/SSL for Apache Atlas

How to configure TLS/SSL for Apache Atlas if you don't choose to use Cloudera Manager's Auto-TLS.

About this task

Typically you would configure TLS for Atlas by using Cloudera Manager's Auto-TLS option. If you need to change these settings or want to understand more about the values set by Cloudera Manager, follow these steps.

Note: In Cloudera on cloud, communication among services within the cluster is configured to use TLS by default. There is no need to manually configure TLS for Atlas in an SDX cluster.

Procedure

1. In Cloudera Manager, select the Atlas service, then click the Configuration tab.
2. Under Category, select Security.
3. Set or update the following properties.

Table 1: Apache Atlas TLS/SSL Settings

Grouping	Configuration Property	Description
Turn on TLS	Enable TLS/SSL for Atlas Server atlas.enableTLS	Select this check box to encrypt Atlas server communication using Transport Layer Security (TLS) (formerly known as Secure Socket Layer (SSL)). This option enables TLS for communication between clients and Atlas Server, between the Atlas Server as a client and services it depends on, such as HBase, and between the Atlas Gateway server and Kafka messaging topics. **Attention:** You must set the value of atlas.ssl.exclude.protocols configuration as TLSv1,TLSv1.1
Private Key Management	Atlas Server TLS/SSL Server JKS Keystore File Location keystore.file	The path to the TLS/SSL keystore file containing the server certificate and private key Atlas uses to prove its own identity when it receives communication from other applications using the public key identified in Atlas Server TLS/SSL Client Trust Store File. The keystore must be in JKS format.
	Atlas Server TLS/SSL Server JKS Keystore File Password keystore.password	The password that allows access to the Atlas Server JKS keystore file.

Grouping	Configuration Property	Description
	Atlas Server TLS/SSL Server JKS Keystore File Password password	The password that protects the Atlas Server private key contained in the JKS keystore.
Public Key Management	Atlas Server TLS/SSL Client Trust Store File truststore.file	The path to a TLS/SSL public key trust store file, in .jks format, that contains Atlas server public key. This trust store must contain the certificate(s) used to sign the connected service(s). If this parameter is not provided, the default list of well-known certificate authorities is used instead.
	Atlas Server TLS/SSL Client Trust Store Password truststore.password	(Optional) The password for the Atlas Server TLS/SSL Certificate trust store file. This password is not required to access the trust store; this field can be left blank. This password provides optional integrity checking of the file. The contents of trust stores are certificates, and certificates are public information.
	Gateway TLS/SSL Client Trust Store File atlas.kafka.ssl.truststore.location	The path to the trust store file, in .jks format, used to confirm the authenticity of TLS/SSL servers that the Atlas Gateway might connect to. This trust store must contain the certificate(s) used to sign the service(s) connected to. If this parameter is not provided, the default list of well-known certificate authorities is used instead. Typically, this file is the same file listed in the truststore.file.
	Gateway TLS/SSL Client Trust Store Password atlas.kafka.ssl.truststore.password	The password for the Gateway TLS/SSL Certificate trust store file. This password is not required to access the trust store; this field can be left blank. This password provides optional integrity checking of the file. The contents of trust stores are certificates, and certificates are public information.

4. Click Save Changes.
5. Restart the Atlas service.

Configuring Atlas Authentication

This section describes how to configure the authentication methods that determine who is allowed to log in to the Apache Atlas web UI. The authentication options are Kerberos, LDAP—including AD, PAM, or file-based.

Atlas allows more than one authentication method to be enabled at one time. If more than one authentication method is enabled, users failing the first method are authenticated against the second method. The priority order of the methods is Kerberos, LDAP, then file-based authentication. For example if both Kerberos and LDAP authentication are enabled, a request without a Kerberos principal and keytab are authenticated using LDAP.

Specifying more than one authentication method allows you to setup useful production and development scenarios:

- In a Production environment, you might configure Kerberos for service account access to the Atlas server while also supporting LDAP authentication for users logging in through the UI.
- In a Development environment, you might configure Kerberos for service account access while leaving file-based authentication enabled to allow a limited number of administrator to access the Atlas UI.

Note: By default, Cloudera Manager installs Atlas with PAM authentication with full Atlas access given to the username you configure for this authentication method. This configuration ensures a smooth end-to-end installation experience. Be sure to disable PAM authentication when you configure your production authentication method.

Configure Kerberos authentication for Apache Atlas

How to configure Kerberos Authentication for Apache Atlas

Note: In Cloudera on cloud, authentication is configured through Free IPA, which uses Kerberos to secure user identities in the environment. There is no need to manually configure authentication settings for Atlas. For more information, see "Managing user access and authorization" in the Cloudera Management Console documentation.

Kerberos authentication for Apache Atlas is automatically configured when you use Cloudera Manager to enable Kerberos authentication for the cluster (typically using the Cloudera Manager Kerberos Wizard).

A prerequisite for using Kerberos authentication is that your cluster is configured for TLS/SSL encryption. There are Atlas-specific steps to complete to enable TLS for Atlas.

Enabling Atlas log in via browser using kerberos keytab

To enable Atlas to log in using the kerberos keytab, you need to set the value of the property atlas.authentication.method.kerberos.support.keytab.browser.login=true in the Safety Valve on Cloudera Manager.

Note: It is not recommended to use this authentication to log in through the browser. Instead use the Knox trusted proxy to access Atlas UI with Kerberos identity.

Related Information

Configuring TLS/SSL for Apache Atlas

FreeIPA identity management

Configure Atlas authentication for AD

How to configure Apache Atlas to use AD for user authentication when using AD cluster-wide.

About this task

Note: In Cloudera on cloud, identity management is provided by FreeIPA and configured using the Cloudera Management Console. There is no need to manually configure authentication settings for Atlas. For more information on FreeIPA, see the Cloudera Management Console documentation.

The settings indicated in these steps apply to Atlas authentication and it is likely that the values will be the same as you use to configure other services on the cluster.

Procedure

1. In Cloudera Manager, select the Atlas service, then open the Configuration tab.
2. To display the authentication settings, type "authentication" in the Search box. You may need to scroll down to see all of the AD settings.
3. Configure the following settings for AD authentication:

Property	Description	Sample values
Enable LDAP Authentication atlas.authentication.method.ldap	Determines whether LDAP is used for authentication.	true
LDAP Authentication Type atlas.authentication.method.ldap.type	The LDAP type (ldap, ad, or none).	ad

Property	Description	Sample values
AD URL atlas.authentication.method.ldap.ad.url	The AD server URL.	
AD Bind DN Username atlas.authentication.method.ldap.ad.bind.dn	Full distinguished name (DN), including common name (CN), of an AD user account that has privileges to search.	cn=admin,dc=example,dc=com
AD Bind DN Password atlas.authentication.method.ldap.ad.bind.password	Password for the account that can search in AD.	Secret123!
AD Domain Name (Only for AD) atlas.authentication.method.ldap.ad.domain	AD domain, only used if Authentication method is AD.	example.com
AD User Search Filter atlas.authentication.method.ldap.ad.user.searchfilter	The AD user search filter.	(&(sAMAccountName={0})(objectClass=user)(memberOf=cn=PTTOR-Contract-Big data Admin,OU=OR Data Platform,OU=PTTOR Distribution Manual Group,DC=pttor,DC=corp)) **Note:** The {0} place holder is required in configuration to substitute to fill in with a valid user name to build the search filter.
AD Base DN atlas.authentication.method.ldap.ad.base.dn	The Distinguished Name (DN) of the starting point for directory server searches.	dc=example,dc=com
AD User Default Role atlas.authentication.method.ldap.ad.default.role	AD User default Role.	
AD Referral atlas.authentication.method.ldap.ad.referral*	See below. Defaults to ignore.	follow

* There are three possible values for atlas.authentication.method.ldap.ad.referral: follow, throw, and ignore. The recommended setting is follow.

When searching a directory, the server might return several search results, along with a few continuation references that show where to obtain further results. These results and references might be interleaved at the protocol level.

- When this property is set to follow, the AD service provider processes all of the normal entries first, and then follows the continuation references.
- When this property is set to throw, all of the normal entries are returned in the enumeration first, before the ReferralException is thrown. By contrast, a "referral" error response is processed immediately when this property is set to follow or throw.
- When this property is set to ignore, it indicates that the server should return referral entries as ordinary entries (or plain text). This might return partial results for the search. In the case of AD, a PartialResultException is returned when referrals are encountered while search results are processed.

4. Click Save Changes.
5. Restart the Atlas service.

Related Information

Cloudera Management Console on cloud: Managing user access and authorization

Configure Atlas authentication for LDAP

How to configure Apache Atlas to use LDAP for user authentication.

About this task

Note: In Cloudera on cloud, identity management is provided by FreeIPA and configured using the Cloudera Management Console. There is no need to manually configure authentication settings for Atlas. For more information on FreeIPA, see the Cloudera Management Console documentation.

The settings indicated in these steps apply to Atlas authentication and it is likely that the values will be the same as you use to configure other services on the cluster.

Procedure

1. In Cloudera Manager, select the Atlas service, then open the Configuration tab.
2. To display the authentication settings, type "authentication" in the Search box. You may need to scroll down to see all of the LDAP settings.
3. **Note:** Hadoop groups were not included during user/group lookup, because Atlas queries the UserGroupInformation for that specific user at the Operating System level (org.apache.hadoop.security.ShellBasedUnixGroupsMapping package). At the time of logging into the Atlas UI, it was seen that "no such user as Gen.tleman" exception was thrown. When user selects the LDAP configuration fromCloudera Manager, they must configure (org.apache.hadoop.security.LdapGroupsMapping package) in core-site.xml file and also add atlas.authentication.ugi-groups.include-hadoop-groups=true property.

   Later, the user can sync the Hadoop group and create entity and classification.

   Include the property atlas.authentication.ugi-groups.include-hadoop-groups=true in Atlas safety valve (atlas-application.properties file), as it takes groups from Hadoop LDAP-based group mapping.

Configure the following settings for LDAP authentication:

Grouping	Property	Description	Sample values
Enable LDAP Authentication	Enable LDAP Authentication atlas.authentication.method.ldap	Determines whether LDAP is used for authentication.	true
	LDAP Authentication Type atlas.authentication.method.ldap.type	The LDAP type (ldap, ad, or none).	ldap
LDAP Server Location	LDAP Server URL atlas.authentication.method.ldap.url	The LDAP server URL.	ldap://localhost:389 or ldaps://localhost:636
Bind Credentials	LDAP Bind Username atlas.authentication.method.ldap.bind.dn	Full distinguished name (DN), including common name (CN), of an LDAP user account that has privileges to query the LDAP database of user accounts on behalf of Atlas. This could be a read-only LDAP user.	cn=admin,ou=people,dc=example,dc=com
	LDAP Bind DN Password atlas.authentication.method.ldap.bind.password	Password for the account that can search for users.	Secret123!

Grouping	Property	Description	Sample values
Group Lookup	LDAP Group-Search Base atlas.authentication.method.ldap.groupSearchBase	The organizational unit (OU) and domain component (DC) properties for the LDAP search tree where Atlas searches for groups.	(|(CN=Hdp_users)(CN=Hdp_admins))
	LDAP Group-Search Filter atlas.authentication.method.ldap.groupSearchFilter	(Optional) Refine the scope of LDAP group search. The Groups-Search Filter is combined with the Group-Search Base to define the group lookup.	
Role Assignment	LDAP Group-Role Attribute atlas.authentication.method.ldap.groupRoleAttribute	The attribute stored in the LDAP Group object to use to map LDAP groups to Atlas roles.	cn
	LDAP User Default Role atlas.authentication.method.ldap.default.role	Atlas role to assign to LDAP users.	
LDAP Search-Bind Authentication Mode	LDAP DN atlas.authentication.method.ldap.base.dn	The Distinguished Name (DN) of the starting point of the LDAP search tree for directory server searches. You can also specify a User Search Filter to further reduce the scope of the search.	dc=example,dc=com
	LDAP User Search Filter atlas.authentication.method.ldap.user.searchfilter	The LDAP user search filter. Used with the User Search Base to further limit the scope of the search for a directory entry that matches the credentials of the user logging into Atlas. Use a user search filter along with a DN pattern so that the search filter provides a fallback if the DN pattern search fails.	
LDAP Direct-Bind Authentication Mode	LDAP User DN Pattern atlas.authentication.method.ldap.userDNpattern	Direct-bind authentication can be used if search is not required to determine the DN needed to bind to the LDAP server. Leave this property blank if LDAP DN is set. To use this authentication mode, all users must be under a single branch in the LDAP directory.	To search for a distinguished name where the uid attribute is the username at login, you might provide a pattern such as: uid={0},ou=users,dc=xasecure,dc=net where {0} indicates the username of the authenticating user. If a user provides the username "foo" at the login page, Atlas searches for the DN: uid=foo,ou=People,dc=corp,dc=com

Grouping	Property	Description	Sample values
LDAP Referral	LDAP Referral atlas.authentication.method.ldap.referral*	See below. Defaults to ignore.	follow

* There are three possible values for atlas.authentication.method.ldap.referral: follow, throw, and ignore. The recommended setting is follow.

When searching a directory, the server might return several search results, along with a few continuation references that show where to obtain further results. These results and references might be interleaved at the protocol level.

- When this property is set to follow, the LDAP service provider processes all of the normal entries first, and then follows the continuation references.
- When this property is set to throw, all of the normal entries are returned in the enumeration first, before the ReferralException is thrown. By contrast, a "referral" error response is processed immediately when this property is set to follow or throw.
- When this property is set to ignore, it indicates that the server should return referral entries as ordinary entries (or plain text). This might return partial results for the search. In the case of LDAP, a PartialResultException is returned when referrals are encountered while search results are processed.

4. Click Save Changes.
5. Restart the Atlas service.

Related Information

Cloudera Management Console on cloud: Managing user access and authorization

Configure Atlas PAM authentication

By default in Data Center installations, Apache Atlas uses PAM authentication, which means valid Atlas users correspond to the users configured for the operating system on the host where Atlas runs. Cloud installations do not use PAM authentication.

About this task

Note: In Cloudera on cloud, authentication is configured through Free IPA, which uses Kerberos to secure user identities in the environment. There is no need to manually configure authentication settings for Atlas. For more information, see "Managing user access and authorization" in the Cloudera Management Console documentation.

Before you begin

Minimum Required Role in Cloudera Manager: Full Administrator.

Procedure

1. In Cloudera Manager, select the Atlas service, then open the Configuration tab.
2. To display the appropriate property, type "safety" in the Search box.
3. Find the Atlas Server Advanced Configuration Snippet (Safety Valve) for conf/atlas-application.properties.
4. In the safety valve, set the following properties:

```
atlas.authentication.method.pam=true
```

```
atlas.authentication.method.pam.service=LOGIN SERVICE
```

where *LOGIN SERVICE* indicates the desired PAM login service. For example, set atlas.authentication.method.pam.service=login to use /etc/pam.d/login.

5. Click Save Changes.
6. Restart the Atlas service.

Once the PAM authentication is enabled for Atlas on a Cloudera Base on premises cluster, to access Atlas, note the following:

- If Atlas was configured using LDAP or Active Directory, the same user role which was set-up can be authenticated and used for accessing Atlas.
- For an Operating System user, you can add the user using useradd or adduser commands and later access Atlas.

Note: If Ranger is enabled, you must have appropriate permissions for the user (both the scenarios above) in Ranger policies. Alternatively, if Ranger is not enabled, the user must manually update the JSON file - atlas-simple-authz-policy.json

Related Information

Cloudera Management Console on on cloud: Managing user access and authorization

Configure Atlas file-based authentication

How to manage Apache Atlas user authentication when using user credentials from a file.

Note: In Cloudera Cloud, identity management is provided by FreeIPA and configured using the Cloudera Management Console. Therefore for Cloudera on cloud, you should leave the Admin Authentication Method set to Kerberos authentication settings. For more information, see "Managing user access and authorization" in the Cloudera Management Console documentation.

Warning: Atlas file-based authentication is intended as a convenience for managing authentication in a development environment. Do not use file-based authentication in a production environment.

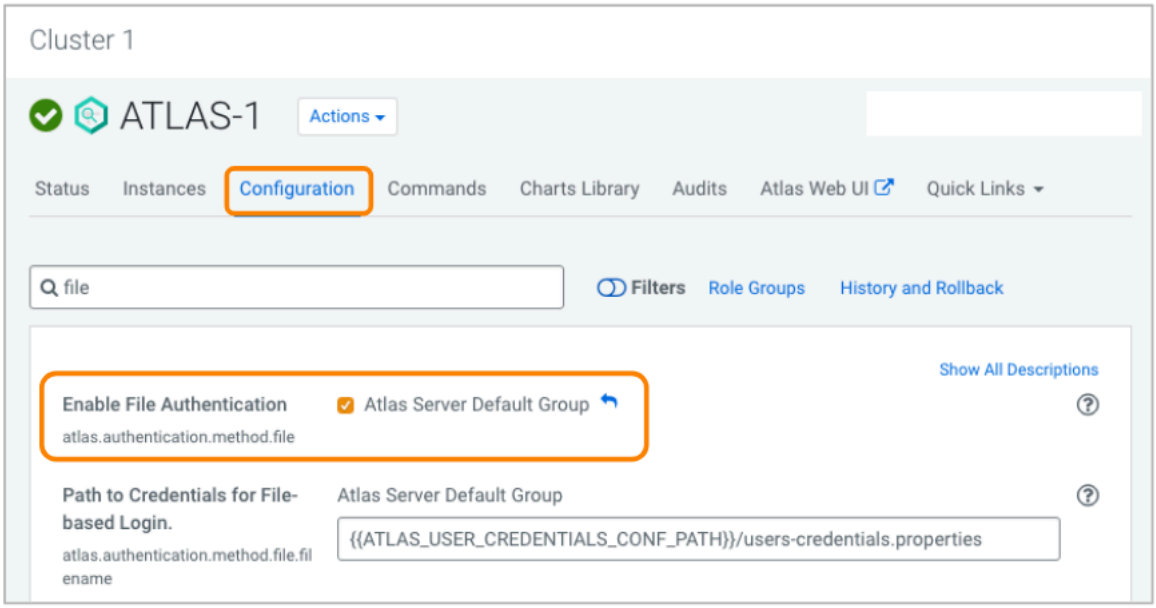
Two of the most likely tasks you would perform related to file-based authentication for Atlas:

- Enable or disable file-based authorization when transitioning among authentication methods
- Add to the list of authorized users for use in a non-production environment

To enable or disable file-based authentication

1. In Cloudera Manager, select the Atlas service, then open the Configuration tab.
2. Display the authentication settings by typing "file" in the Search box.

3. Check or uncheck the option Enable File Authentication.



4. If you are enabling file-based authentication, review the location of the file that contains user credentials as specified in Path to Credentials for File-based Login.

   The default directory location indicated by ATLAS_USER_CREDENTIALS_CONF_PATH is the Atlas configuration directory in the Cloudera Runtime installation location, typically

   ```
   /opt/cloudera/parcels/CDH-VERSION/etc/atlas/conf.dist
   ```

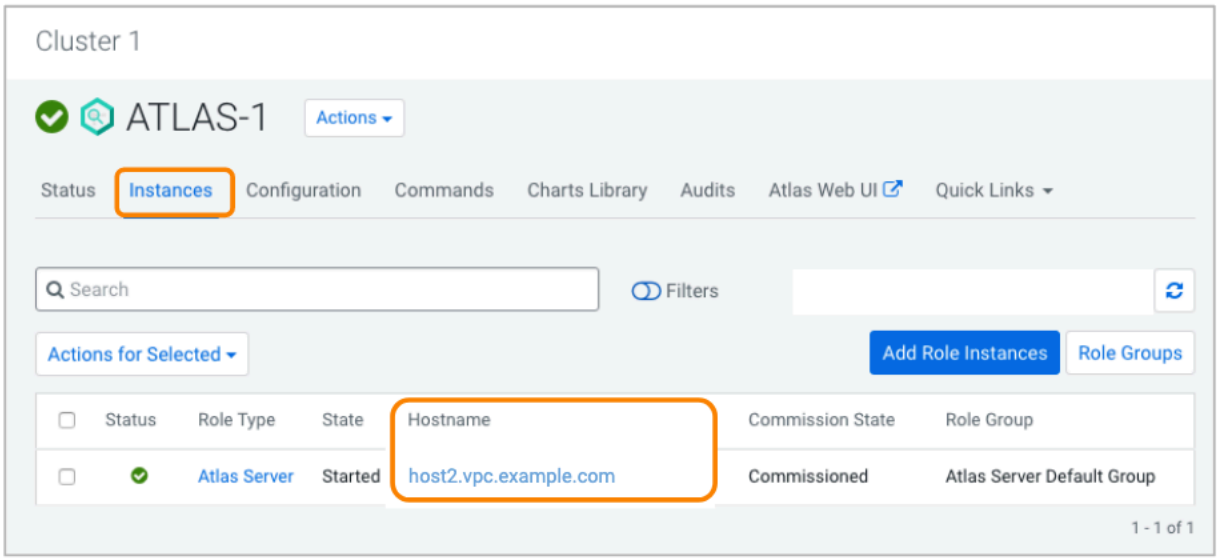
   If you need to change the location, specify an absolute path on the Atlas host. Alternatively you can reset the location for users-credentials.properties by setting the atlas.authentication.method.file.filename property in the Atlas Server Advanced Configuration Snippet (Safety Valve) for conf/atlas-application.properties. You can find the hostname where the Atlas service is running on the Instances tab in Cloudera Manager.
5. Configure the list of users as described in step 6 on page 14 in To add to the users list: on page 13.
6. Click Save Changes.
7. Restart the Atlas service.

To add to the users list:

This procedure requires that you restart the Atlas service.

1. In Cloudera Manager, select the Atlas service, then open the Instances tab.

2. Note the Hostname where the Atlas service is running. You'll need to be able to SSH into that host to update the user list.



3. Open the Configuration tab.
4. Type "file" in the search box to filter the property list.
5. Find the user file.

   Look for the location of the users-credentials.properties file as set in the Path to Credentials for File-based Login property.

   The default directory location indicated by ATLAS_USER_CREDENTIALS_CONF_PATH is the Atlas configuration directory in the Cloudera Runtime installation location, typically

   ```
   /opt/cloudera/parcels/CDH-VERSION/etc/atlas/conf.dist
   ```

   If you need to reset the location of the users-credentials.properties file, see Moving the Atlas user credentials file on page 15.
6. Update the user list.

   SSH into the Atlas host. Edit the users file to include additional users in the following format:

   ```
   USERNAME=GROUP::PASSWORD-HASH
   ```

   where the *USERNAME* is the string used in the login page, the *GROUP* is one of ADMIN, DATA_STEWARD, or DATA_SCIENTIST. The *PASSWORD-HASH* is the password encoded using salting.

   To generate an encoded password, use the python cputil.py script provided in the Atlas installation, typically:

   ```
   /opt/cloudera/parcels/CDH-version/lib/atlas/bin
   ```

   Run the command as follows, making sure that variables used in the script are available:

   ```
   $ export ATLAS_HOME=/opt/cloudera/parcels/CDH-VERSION/lib/atlas
   $ export ATLAS_CONF=<Atlas installation location>
   $ /opt/cloudera/parcels/CDH-version/lib/atlas/bin/cputil.py -g -u <user
   name> -p <new secure password> -s
   ```

   The -s option ensures that the output includes only the hash value. The Atlas installation location is typically /opt/cloudera/parcels/CDH-*VERSION*/etc/atlas/.
7. Restart the Atlas service.

Moving the Atlas user credentials file

If you find you need to point Atlas to a different location for the users-credentials.properties file, you can reset the location using a property in an Atlas advanced configuration snippet.

1. In Cloudera Manager, select the Atlas service, then open the Configuration tab.
2. Display the advanced configuration settings by typing "safety" in the Search box.
3. In the Atlas Server Advanced Configuration Snippet (Safety Valve) for conf/atlas-application.properties property, add the following property and set it to the absolute path to the users-credentials.properties file on the Atlas host. This path must be accessible to the Atlas service user (defaults to atlas).

```
atlas.authentication.method.file.filename=/some/secure/location
```

4. Restart the Atlas service.

Changing the Atlas admin password

Do the following steps, to update your password:

1. Go to Cloudera Manager Clusters Atlas Configuration .
2. Type Admin Password into the search bar.
3. Enter your new password.
4. Click Save Changes.
5. Go to Cloudera Manager Clusters Atlas Actions
6. Click Restart to apply your changes.

By file-based authentication, do the following to update the password stored in the authentication file:

1. Go to Cloudera Manager Clusters Atlas Configuration .
2. Type Path to Credentials for File-based Login into the search bar.
3. Check the path for the users-credentials.properties file.
4. Update the password and save your file.
5. Go to Cloudera Manager Clusters Atlas Actions
6. Click Restart to apply your changes.

Related Information

Cloudera Management Console on on cloud: Managing user access and authorization

Configuring Atlas Authorization

Apache Atlas in Cloudera uses Apache Ranger policies to control access to metadata that are managed by Atlas. Ranger policies also control access to Atlas administrative tasks.

Ranger provides authorization to access the following metadata and operations:

Types

Atlas "types" are the entity model definitions, whether provided in Atlas or added in your environment. Types include these "categories":

- Entity
- Classification
- Relationship
- Business Metadata
- Struct
- Enum

Ranger authorization allows you to configure access for users and groups to perform the following operations on types:

- Create
- Update
- Delete
- Read

The policies can be configured to apply to one or more types or all types. For example, the Atlas administrator user has access to create, update, and delete all type categories (type-category *).

Entities

Atlas "entities" are instances of entity types: entities represent assets and processes on your cluster. Ranger authorization allows you to configure access to users and groups to perform the following operations on entities:

- Read
- Create
- Update
- Delete
- Read classification
- Add classification
- Update classification
- Remove classification
- Add label
- Remove label
- Update Business Metadata

Note that the classification operations are those that involve associating a classification to an entity; operations on a classification definition are controlled by authorization on the classification category of type described previously. Use the entity authorization to give a user the ability to associate an existing classification with any entity (entity-type *); use the type authorization to give a user the ability to create new classifications (type-category classification).

Policies for labels and business metadata work similarly to classifications: you can control whether users can add labels or business metadata to specific entity types, individual entities, or entities marked with specific classifications. For example, a default policy allows any authenticated user to update all business metadata for any entity types with any classifications and on any instances of entities (entity-type *, entity-classification *, entity-id *, entity-business-metadata *).

Some Atlas features, such as saved searches, are modeled as entities. You can control access to these features using entity policies. For example, a default policy allows any authenticated user to save Atlas searches (entity-type __AtlasUserProfile, __AtlasUserSavedSearch).

Relationships

Atlas "relationships" describe connections between two entities, including, but not limited to, the input and output relationships that are used to build lineage graphs. Ranger authorization allows you to configure access to users and groups to perform the following operations on relationships:

- Add relationship
- Update relationship
- Remove relationship

These operations are required to build rich models among entities and are granted to administrative users and system users. Relationships cannot be updated by users through the Atlas UI.

Admin operations

Atlas administrative operations include:

- Import entities

- Export entities

These operations encompass all the privileges needed to create new and update existing entities. Typically, this access is granted to administrative users and system users such as RangerLookup and the Data Plane profiler user (DPProfiler).

Restricting classifications based on user permission

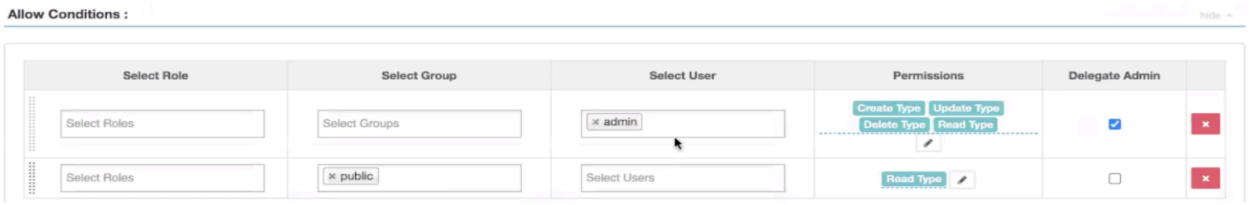
The Apache Atlas authorisation model has been updated with a new permission, type-read, to enable restricting visibility of types to specific users, groups, and roles.

Altas authorization model supports permissions to create, update, delete types, which enables setting up policies to restrict who can perform create, update, delete on types (like classifications, business metadata). Note that all authenticated users were allowed to perform read operation on types.

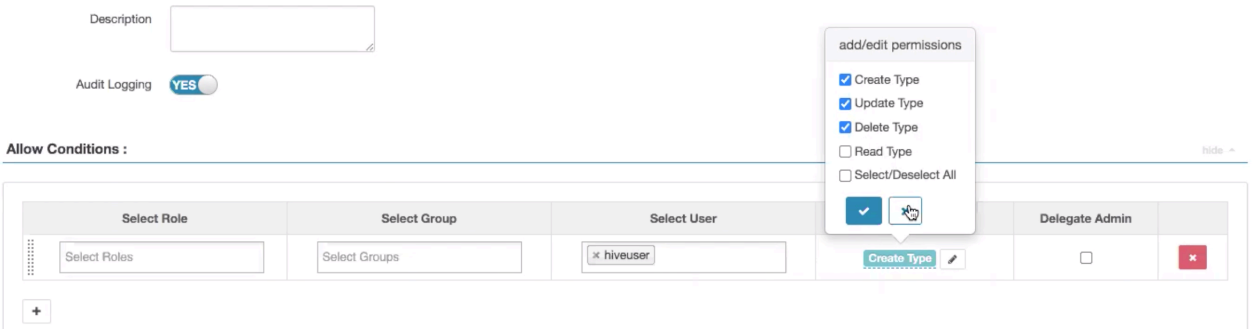
In the updated model, users with type-create or type-update or type-delete permission will implicitly be granted type-read permission. Also, to be compliant with earlier user experience, a default policy is created to allow type-read operation to all authenticated users. Security administrators can update the default policy and add new policies to restrict type-read operation to specific users.

Prior to this update, new permission, the type-read access was not available and read access for typedefs was public. Now the objective for Read Type permission is to have the ability to set access control permission on visibility of classifications and other typedefs (entitydef, enumdef, structdef, and business_metadatadef) in Atlas.

Previously, policy permission types only included Create Type, Update Type, and Delete Type.



Note: With the introduction of “Read Type” permission, when you select any one of the types - Create Type or Update Type or Delete Type, the Read Type permission is implicitly enabled.



Note: Upgrade scenario: The new Read Type permission is available as part of the upgrade scenario.

For example: If you are upgrading from Cloudera Private Cloud Base 7.1.x to 7.1.4, you must be able to make use of the new read type permission.

Configuring Ranger Authorization for Atlas

Apache Atlas is configured to use Apache Ranger for authorization by default. You might need to change configuration settings to disable Ranger as the source of authorization in a development environment; Ranger authorization is highly recommended in a production environment. In addition, there are some configuration values

that you might need to change should you make significant changes to how Atlas and Ranger are installed in your cluster.

About this task

Atlas behaves like any other service when it comes to integrating with Ranger for access control: turn on Ranger authorization from the Cloudera Manager configuration page for the Atlas service. This integration allows Atlas to use Ranger policies to determine authorization for user actions in Atlas; Atlas also reports success or failure against the policies back to Ranger. In addition to this standard integration for authorization, Atlas integrates with Ranger to send metadata updates to Ranger using an Apache Kafka topic. The configuration properties that support the two integration paths include specifying HDFS locations for caching Ranger policies, storing Atlas audits, and storing other metadata exchanged between the two services. In rare circumstances, you may need to relocate these storage locations.

Important: Native Hadoop Library must be added to the ATLAS classpath for Ranger group authorization to work. If this is missing from the Atlas class path, for users affected who need Hadoop Group based authorization to work, the below details must be added to the Environment Safety Valve for Atlas Server:

Key : ATLAS_CUSTOM_OPTS

Value : -Djava.library.path=${CDH_HADOOP_HOME}/lib/native

Before you begin

Minimum Required Role in Cloudera Manager: Full Administrator.

Procedure

1. In Cloudera Manager, select the Atlas service, then open the **Configuration** tab.
2. To display the Ranger configuration settings, type "ranger" in the Search box.
3. To modify the behavior of the Atlas to Ranger integration, update the following properties:

   To disable Atlas authentication through Ranger

   Uncheck the RANGER Service property. Leave the supporting properties as is so you can re-instate Ranger easily if needed.

   To set where Atlas stores intermediate data for Ranger audits and cached authorization policies

   Review and potentially change the path value for these properties:

   - Ranger Atlas Plugin Hdfs Audit Directory (HDFS location)
   - Ranger Atlas Plugin Audit Hdfs Spool Directory Path (local file system location)
   - Ranger Atlas Plugin Audit Solr Spool Directory Path (local file system location)
   - Ranger Atlas Plugin Policy Cache Directory Path (local file system location)

   To re-instate Ranger after disabling it

   Enable the RANGER Service property. The supporting properties should still be available.
4. Restart the Atlas service.

Configuring Atlas Authorization using Ranger

Use the Ranger Admin Web UI to add or update policies to control Apache Atlas access.

About this task

You can use Apache Ranger policies to control user-access to Atlas metadata and to actions that users can perform in Atlas.

Attention: To manage the Atlas Web UI authorisation using Ranger, **LDAP UGI Groups** (atlas.authentication.method.ldap.ugi-groups) property must be enabled.

The following policies are defined by default:

- admin: the initial Atlas administrator user has full access to all Atlas actions, including full access to entity metadata, classifications, business metadata attributes, labels and relationship creation, the ability to create new entity, enumeration, structure, and relationship types, the ability to import, export, and purge metadata from Atlas, and the ability to save searches.
- dpprofiler: the Data Plane service user has the same extensive privileges as the admin user. These privileges allow integration between Cloudera Data Catalog (Data Steward Studio) and Atlas.
- beacon: the Cloudera Replication Manager service user has the same extensive privileges as the admin user. These privileges allow Atlas to participate in cluster-level disaster recovery operations.
- rangertagsync: the TagSync service user has read access to entity metadata, specifically to entity classifications, business metadata, and labels to be used in Ranger tag-based policies.
- rangerlookup: the Ranger lookup service user has read access to entity metadata, specifically to entity classifications, business metadata, and labels to be used in enforcing Ranger policies.
- public: all users are granted access to read Atlas entity metadata, classifications, labels, and relationships (such as lineage).
- {USER}: any user who successfully logs in to Atlas can save searches so they are available in subsequent Atlas sessions.

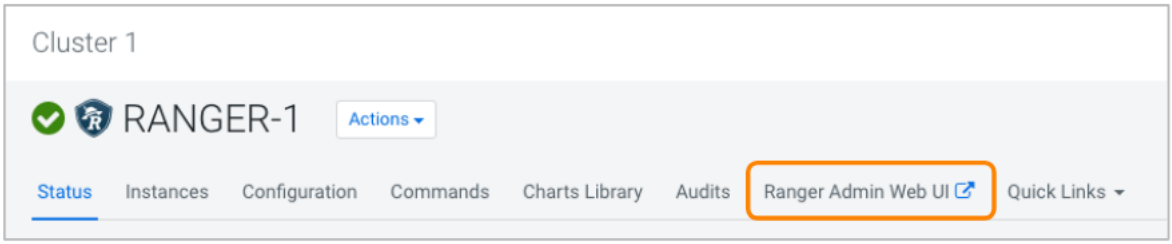
Before you begin

To change Ranger policies for Atlas, your user needs privileges in Ranger to change Resource Based Policies.

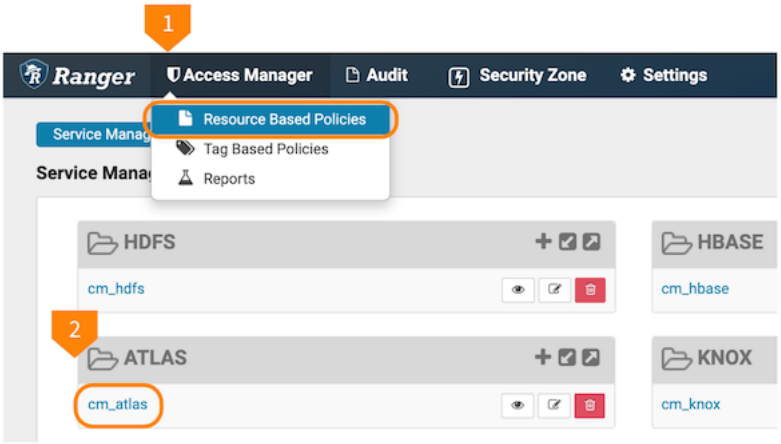
Procedure

1. Open the Ranger service that is running in the same cluster as Atlas.

   One way to do this is to open the Ranger Admin Web UI from Cloudera Manager.



2. Open Access Manager Resource Based Policies and select Atlas policies (cm_atlas).



3. On the List of Policies page, click Add New Policy.

4. Use the Create Policy page to specify the Atlas authorization policy.

   Ranger fields support ? and * wildcards for single and multiple character replacement. To apply the policy to all types of a given entry, use *.

   These selections can be set to "include" the selected resources or "exclude" the selected resources. An include policy applies only to the named types. An exclude policy for the same type would apply to all metadata types other than named types.

   Ranger Authorization with classifications

   Currently in Atlas we have the following authorization options for using classification:

   - Authorization on types: Where user can create a new classification def (not the association)

     For example: Only admin_user is allowed to create tags - PII, PHI..etc.
   - Authorization on Entity: For entities with typeName hive_table with qualifiedName (abc@cl1) and having classification (already associated) can add, update, and remove classifications.

     For example, if any Hive table with qualifiedName 'emp@cl1 and having the tag PII, the admin_user can add the SENSITIVE tag.

   Authorization Enhancement

   The enhancement is provided to authorize as to who can add, remove, and update classification for an entity. Even if the entities on which classification have to be applied, that do not have classifications already tagged to it, provided the entity-type, Entity-ID and classification on it matches the specified policy.

   For example, any users belonging to the user group - finance_group can add classification - FINANCE_PII to any entities (these entities may or may not have any classifications associated) if the policy has provided access to the user or user-group.

   To achieve this, a new field named classification is introduced in the Ranger UI in which tags to be added can be specified.

   Entity Classification denotes the classifications already associated with the entity.

   For example:

   Entity Classification - Finance_*

   Classification: Finance_PII

   Access: Group: finance_group, Permission: Add/Update/Delete Classifications.

Policy Type	Access. There are no other policy types available for an Atlas service.
Policy Name	255 character name that appears in the list of policies. Roles, users, and groups also show up in the list, so it helps if your name includes the operations or metadata that the policy controls.
Policy Label	Metadata you can include in the policy definition to help organize the policies for a given service. The same label can be added to any number of policies for the service. There is no limit to the number of characters in a label, but only 28 characters display in the policy list.
type-category menu	The metadata or operation type ("resources" in Ranger terms) that the policy applies to, including: • type-category • entity-type • atlas-service • relationship-type
type-category option	Choose this option to authorize actions generally against Atlas resource types, including business metadata, classifications, enumerations, entities, relationships, structures. With type-category selected, options include:

Type Name	Refine the authorization to specific types within the named type category. For example, to give users authorization to create Atlas Business Metadata, choose type-category and the category Business Metadata; then set the Type Name to *. For example, to authorize users to add values to an existing enum, such as AtlasGlossaryTermRelationshipStatus, add this enum to the Type Name and include the permission for "Update Type" in the Allow Condition. To allow users to update any types within the type category, use *. To determine the supported values, use the Atlas UI or API to show the defined types.
entity-type option	Authorizes actions against specific entity types, individual entities, entities identified by associated classifications, or entities identified by associated metadata. For example, to authorize users to add classifications or metadata to any Hive table entities, set the entity-type to hive_table and set additional options to *. With entity-type selected, options include:
Entity Classification	Refines the list of entities in entity-type to those associated with a specified classification. For example, to restrict authorization to Hive tables that were marked with some classification that indicates their readiness for use, set entity-type to hive_table and include the identifying classification name (e.g., Available) in Entity Classification.
Entity ID	Refines the list of entities in entity-type to those associated with a specified ID. When the detail page for an entity is open in the Atlas UI, the last element of the browser URL indicates the entity ID.
classification	Provides the option to authorize as to who can add, remove, and update classification for an entity, even if the entities on which classification have to be applied, which do not have classifications already tagged to it, provided the entity-type, Entity-ID and classification on it matches the specified policy.
Metadata types selection	Refines the list of entities in entity-type to those associated with specific user-defined metadata, including: • entity-label • entity-business-metadata • classification • none Set label names in the type entity-label to limit the authorization policy to entities marked with any of those labels. Use * to indicate any label. Set business metadata collection names in the type entity-business-metadata to limit the authorization policy to entities marked with metadata attributes from that business metadata collection. Use * to indicate any business metadata collections.
atlas-service option	Authorizes the import and export Atlas entities and purge deleted entities through the API. This privilege overrides specific privileges for entity-types. Typically the users with this privilege are service users creating entities in Atlas.
relationship-type option	Authorizes the creation and update of Atlas relationships. You can identify specific relationship types or use * to indicate any relationship type. Typically the users with this privilege are service users creating entities in Atlas.
End1 Entity Type End1 Entity Classification End1 Entity ID End2 Entity Type End2 Entity Classification End2 Entity ID	Refines the relationship authorization to specific attributes of relationships. "End1" and "End2" indicate the entities on each side of the relationship. For example, you could use the End1 and End2 Entity Type options to allow modification of relationships when one side of the relationship are Hive tables and the other side Hive columns.
Description	Information that you add to help you remember the value of this policy. The description can be up to 1000 characters.
Audit Logging	Enables Ranger's audit logging for this policy. There are other options in Ranger's configuration that can conflict with this option, but generally if you turn off this setting, Ranger enforces the policy but does not audit success or failed actions against the policy.

Allow Conditions	Choose the roles, users, and/or groups and the permissions they can access for the resources defined in the policy. If you need to include parts of overlapping groups, add an exclude condition in addition to the allow condition. For more information, see Ranger access conditions.
Deny Conditions	Choose the roles, users, and/or groups and the permissions they cannot access for the resources defined in the policy.

5. Click Add.

Results

You should be able to validate the policy almost immediately after saving a valid policy.

Related Information

Configure a resource-based service: Atlas

Add or edit permissions in Ranger

Configuring Simple Authorization in Atlas

Apache Atlas in Cloudera uses the simple authorization, which is the default authorizer to manage different users, user groups, and group roles.

This type of authorization does not involve creating policies in Ranger UI. This is NOT a recommended authorization method to grant permissions in Atlas but can still be used to manage user authorization. The simple authorizer employs policies that are defined in a JSON file.

The following properties must be configured in Atlas properties. When you uncheck Ranger authorizer, these properties are automatically added by the Cloudera Manager.

- atlas.authorizer.impl=simple
- atlas.authorizer.simple.authz.policy.file=atlas-simple-authz-policy.json

Simple Authorization provides certain roles for managing Atlas metadata and related operations. Under each role, you could define the permission for each resource, based on the entity type for which you want to grant access.

There are three supported roles:

- ROLE_ADMIN - Has all permissions to perform any action.
- DATA_SCIENTIST - Has permission for reading and classifying entities.
- DATA_STEWARD - Has permission to read, classify, create, update, add, and remove classifications.

An example JSON file that displays the above scenarios:

```
{

 "roles": {
   "ROLE_ADMIN": {
     "adminPermissions": [
       {
         "privileges": [ ".*" ]
       }
     ],

     "typePermissions": [

       {
         "privileges":     [ ".*" ],
         "typeCategories": [ ".*" ],
         "typeNames":      [ ".*" ]
       }
     ],

     "entityPermissions": [
```

```
      {

         "privileges":      [ ".*" ],
         "entityTypes":     [ ".*" ],
         "entityIds":       [ ".*" ],
         "classifications": [ ".*" ],
         "labels" :         [ ".*" ],
         "businessMetadata" : [ ".*" ],
         "attributes" :     [ ".*" ]

      }
    ],

    "relationshipPermissions": [

      {
        "privileges":     [ ".*" ],
        "relationshipTypes": [ ".*" ],
        "end1EntityType":           [ ".*" ],
        "end1EntityId":             [ ".*" ],
        "end1EntityClassification": [ ".*" ],
        "end2EntityType":           [ ".*" ],
        "end2EntityId":             [ ".*" ],
        "end2EntityClassification": [ ".*" ]

      }
    ]
  },

  "DATA_SCIENTIST": {

    "entityPermissions": [

      {
        "privileges":      [ "entity-read", "entity-read-classification" ],
        "entityTypes":     [ ".*" ],
        "entityIds":       [ ".*" ],
        "classifications": [ ".*" ]
      }
    ]
  },

  "DATA_STEWARD": {

    "entityPermissions": [
      {
        "privileges":      [ "entity-read", "entity-create", "entity-upda
te", "entity-read-classification", "entity-add-classification", "entity-upda
te-classification", "entity-remove-classification" ],
        "entityTypes":     [ ".*" ],
        "entityIds":       [ ".*" ],
        "classifications": [ ".*" ]=
      }
    ],

    "relationshipPermissions": [
      {
        "privileges":               [ "add-relationship", "update-relatio
nship", "remove-relationship" ],
        "relationshipTypes":        [ ".*" ],
        "end1EntityType":           [ ".*" ],
        "end1EntityId":             [ ".*" ],
        "end1EntityClassification": [ ".*" ],
        "end2EntityType":           [ ".*" ],
```

```
        "end2EntityId":             [ ".*" ],
        "end2EntityClassification": [ ".*" ]

      }
    ]
  }
 },
```

The simple authorization roles that are defined as explained earlier can be assigned to users:

```
"userRoles": {
"admin":[ "ROLE_ADMIN" ],
"rangertagsync":[ "DATA_SCIENTIST" ]
```

Note: The rangertagsync user role is pre-defined in the simple authorizer JSON.

Roles can be granted to user-groups. An user can belong to multiple groups; roles assigned to all groups the user belongs to will be used to authorize the access.

```
"groupRoles": {

   "ROLE_ADMIN":      [ "ROLE_ADMIN"],
   "hadoop":          [ "DATA_STEWARD" ],
   "DATA_STEWARD":    [ "DATA_STEWARD" ],
   "RANGER_TAG_SYNC": [ "DATA_SCIENTIST" ]
 }

}
```

Note: The RANGER_TAG_SYNC group role is pre-defined in the simple authorizer JSON.