Cloudera Runtime 7.1.6

Ranger Auditing

Date published: 2019-11-01
Date modified:

https://docs.cloudera.com/

Legal Notice

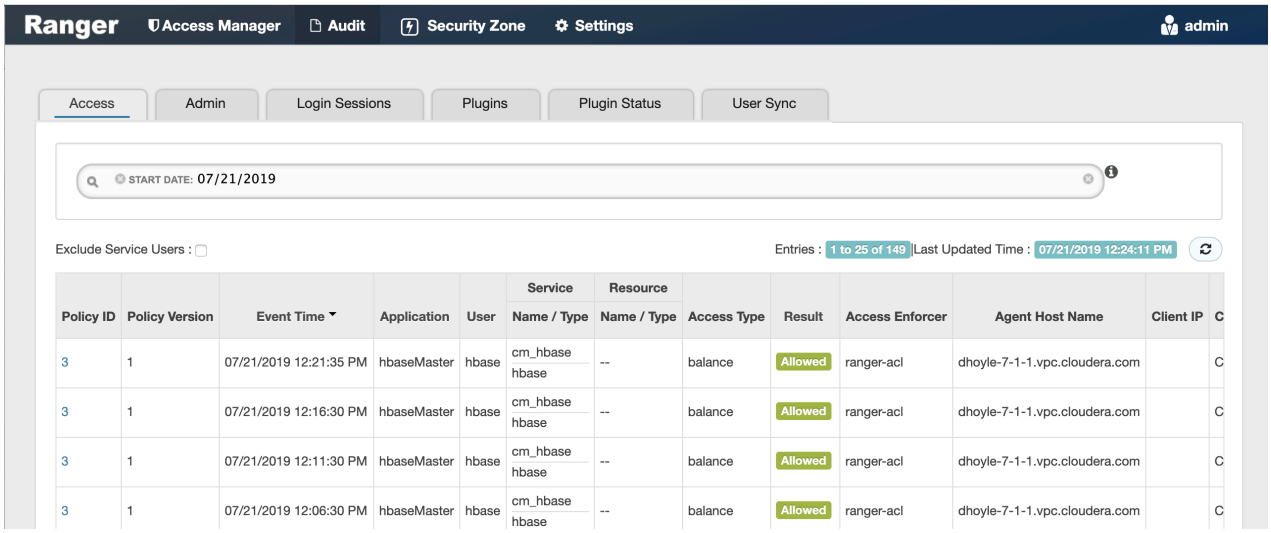
Contents

Audit Overview

Apache Ranger provides a centralized framework for collecting access audit history and reporting data, including filtering on various parameters. Ranger enhances audit information obtained from Hadoop components and provides insights through this centralized reporting capability.

Managing Auditing with Ranger

To explore options for auditing policies in Ranger, click Audit in the top menu.



There are six tabs on the Audit page:

- Access
- Admin
- Login sessions
- Plugins
- Plugin Status
- User Sync

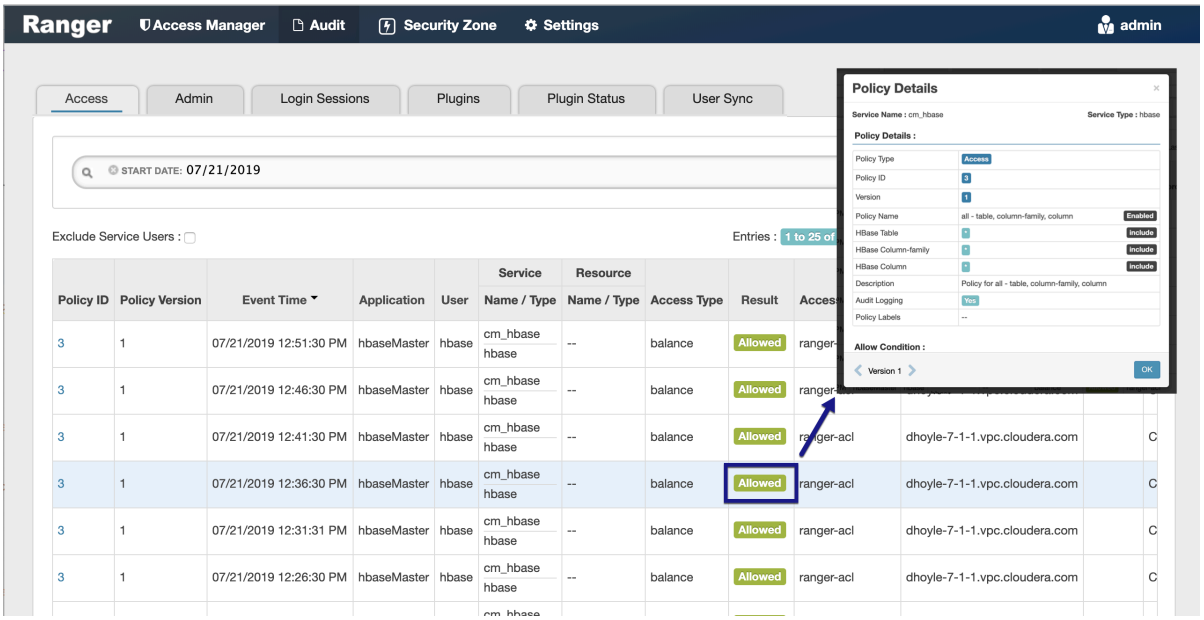
View audit details

How to view operation details in Ranger audits.

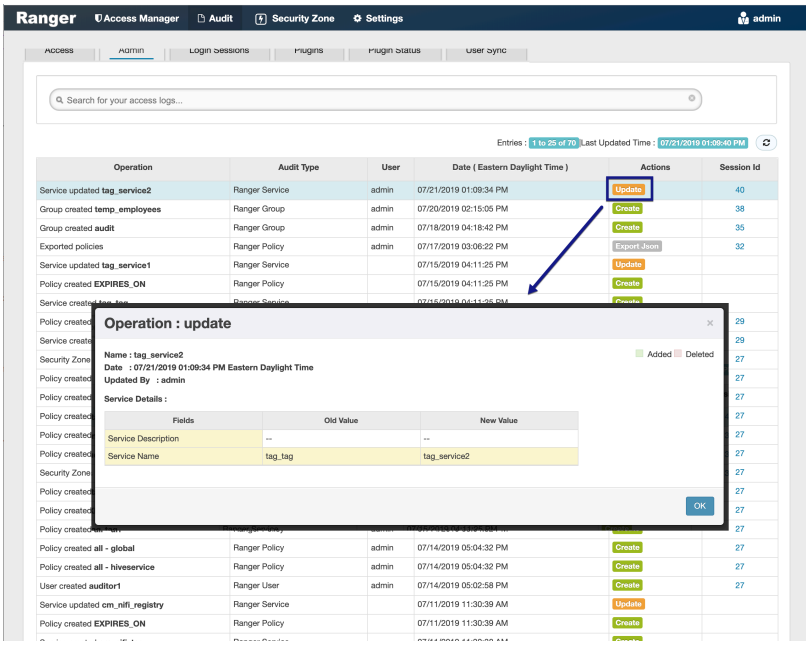
Procedure

To view details for a particular operation, click any tab, then Policy ID, Operation name, or Session ID.

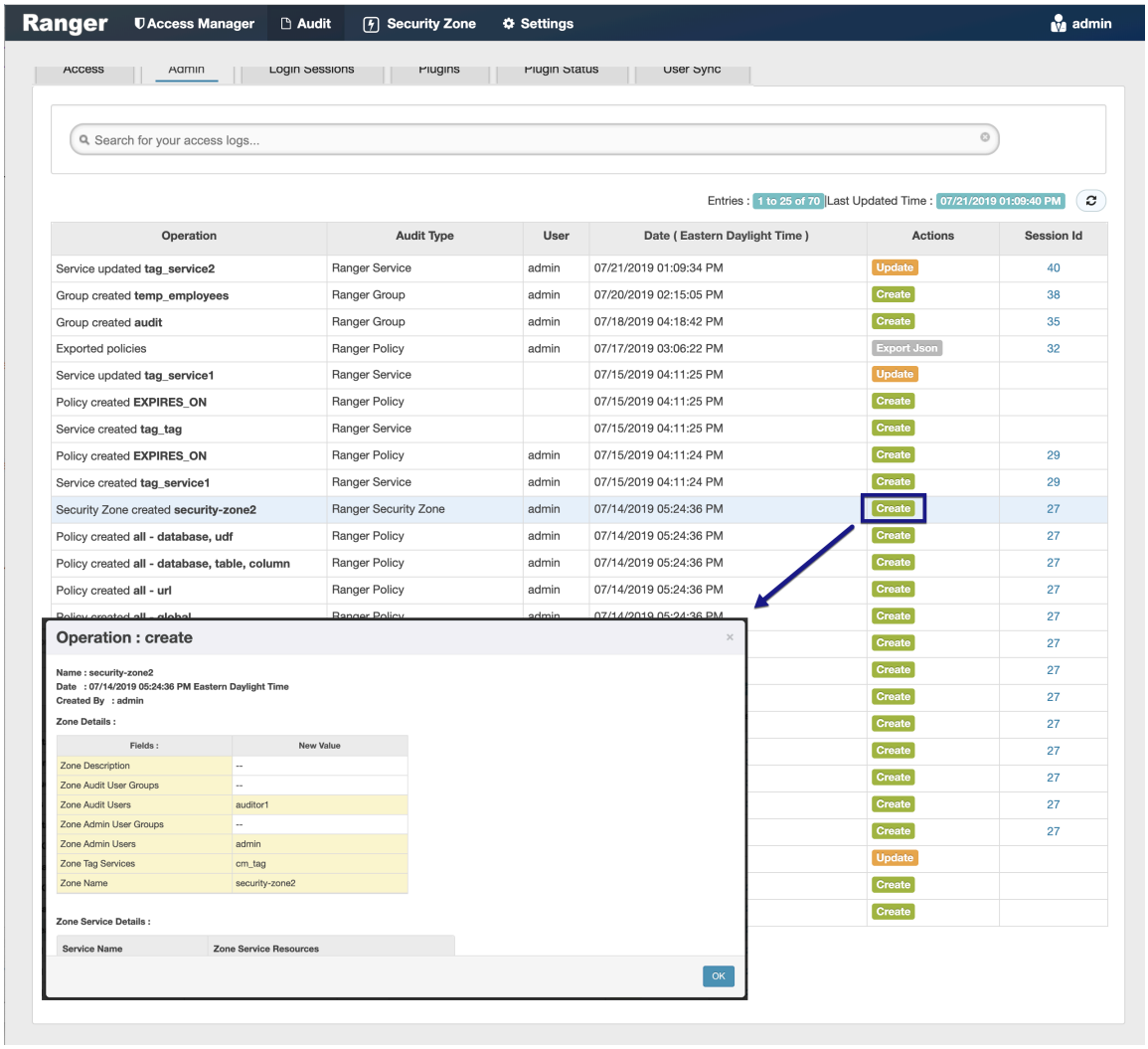
Audit > Access: HBase Table



Audit > Admin: Update

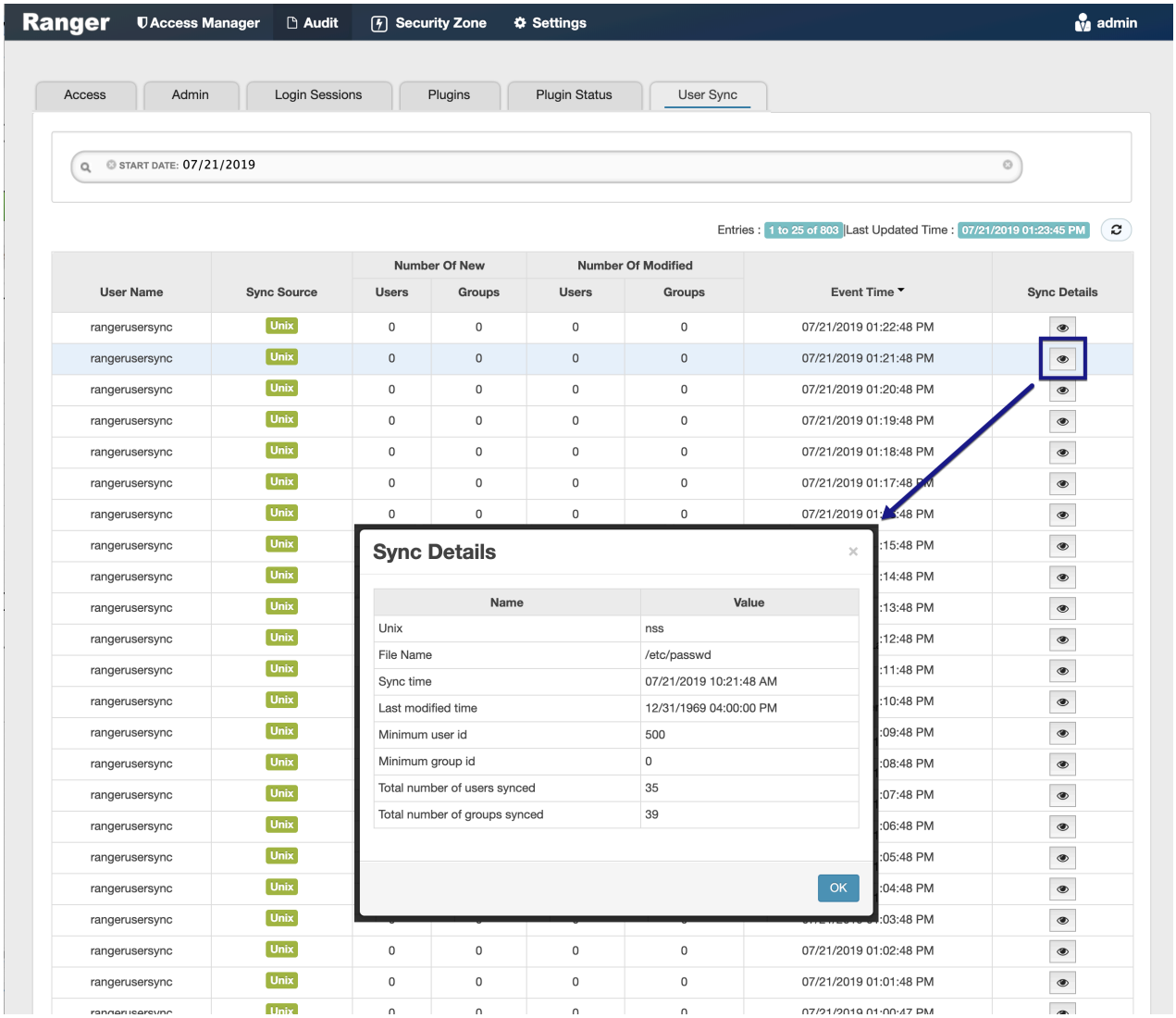


Audit > Admin: Create



Operation	Audit Type	User	Date (Eastern Daylight Time)	Actions	Session Id
Service updated **tag_service2**	Ranger Service	admin	07/21/2019 01:09:34 PM	Update	40
Group created **temp_employees**	Ranger Group	admin	07/20/2019 02:15:05 PM	Create	38
Group created **audit**	Ranger Group	admin	07/18/2019 04:18:42 PM	Create	35
Exported policies	Ranger Policy	admin	07/17/2019 03:06:22 PM	Export Json	32
Service updated **tag_service1**	Ranger Service		07/15/2019 04:11:25 PM	Update	
Policy created **EXPIRES_ON**	Ranger Policy		07/15/2019 04:11:25 PM	Create	
Service created **tag_tag**	Ranger Service		07/15/2019 04:11:25 PM	Create	
Policy created **EXPIRES_ON**	Ranger Policy	admin	07/15/2019 04:11:24 PM	Create	29
Service created **tag_service1**	Ranger Service	admin	07/15/2019 04:11:24 PM	Create	29
Security Zone created **security-zone2**	Ranger Security Zone	admin	07/14/2019 05:24:36 PM	Create	27
Policy created **all - database, udf**	Ranger Policy	admin	07/14/2019 05:24:36 PM	Create	27
Policy created **all - database, table, column**	Ranger Policy	admin	07/14/2019 05:24:36 PM	Create	27
Policy created **all - url**	Ranger Policy	admin	07/14/2019 05:24:36 PM	Create	27
Policy created **all - global**	Ranger Policy	admin	07/14/2019 05:24:36 PM	Create	27
				Create	27
				Create	27
				Create	27
				Create	27
				Create	27
				Create	27
				Create	27
				Create	27
				Update	
				Create	
				Create	

Audit > User Sync: Sync details



Create a read-only Admin user (Auditor)

Creating a read-only Admin user (Auditor) enables compliance activities because this user can monitor policies and audit events, but cannot make changes.

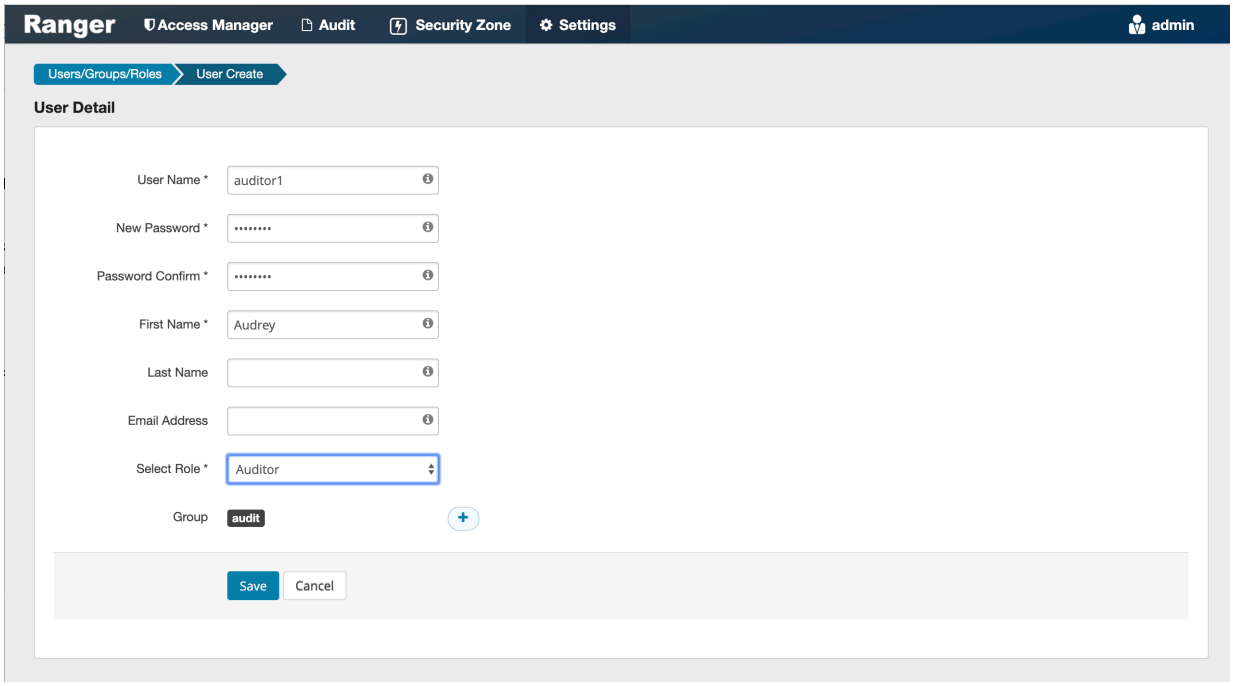
About this task

When a user with the Auditor role logs in, they see a read-only view of Ranger policies and audit events. An Auditor can search and filter on access audit events, and access and view all tabs under Audit to understand access events. They cannot edit users or groups, export/import policies, or make changes of any kind.

Procedure

1. Select Settings > Users/Groups/Roles.
2. Click Add New User.

3. Complete the **User Detail** section, selecting Auditor as the role:



4. Click Save.

Changing Ranger audit storage location and migrating data

How to change the location of existing and future Ranger audit data collected by Solr from HDFS to local or from local to HDFS.

Before you begin

- Stop Atlas from Cloudera Manager.
- If using Kerberos, set the SOLR_PROCESS_DIR environment variable.

```
# export SOLR_PROCESS_DIR=$(ls -1dtr /var/run/cloudera-scm-agent/process/
*SOLR_SERVER | tail -1)
```

About this task

Starting with Cloudera Runtime version 7.1.4 / 7.2.2, the storage location for ranger audit data collected by Solr changed to local file system from HDFS, as was true for previous versions. The default storage location Ranger audit data storage location for Cloudera Runtime-7.1.4+ and Cloudera Runtime-7.2.2+ installations is local file system. After upgrading from an earlier Cloudera platform version, follow these steps to backup and migrate your Ranger audit data and change the location where Solr stores your future Ranger audit records.

- The default value of the index storage in the local file system is /var/lib/solr-infra. You can configure this, using Cloudera Manager Solr Configuration "Solr Data Directory" .
- The default value of the index storage in HDFS is /solr-infra. You can configure this, using Cloudera Manager Solr Configuration "HDFS Data Directory" .

Procedure

1. Create HDFS Directory to store the collection backups.

   As an HDFS super user, run the following commands to create the backup directory:

   ```
   # hdfs dfs -mkdir /solr-backups
   # hdfs dfs -chown solr:solr /solr-backups
   ```

2. Obtain valid kerberos ticket for Solr user.

   ```
   # kinit -kt solr.keytab solr/$(hostname -f)
   ```

3. Download the configs for the collection.

   ```
   # solrctl instancedir --get ranger_audits /tmp/ranger_audits
   # solrctl instancedir --get atlas_configs /tmp/atlas_configs
   ```

4. Modify the solrconfig.xml for each of the configs for which data needs to be stored in HDFS.

   In /tmp/<config_name>/conf created during Step 3., edit properties in the solrconfig.xml file as follows:

   - When migrating your data storage location from local file system to HDFS, replace these two lines:

     ```
     <directoryFactory name="DirectoryFactory"
      class="${solr.directoryFactory:solr.NRTCachingDirectoryFactory}">

     <lockType>${solr.lock.type:native}</lockType>
     ```

     with

     ```
     <directoryFactory name="DirectoryFactory"
      class="${solr.directoryFactory:org.apache.solr.core.HdfsDirectoryFactory}">

     <lockType>${solr.lock.type:hdfs}</lockType>
     ```

   - When migrating your data storage location from HDFS to local file system, replace these two lines:

     ```
     <directoryFactory name="DirectoryFactory"
      class="${solr.directoryFactory:org.apache.solr.core.HdfsDirectoryFactory}">

     <lockType>${solr.lock.type:hdfs}</lockType>
     ```

     with

     ```
     <directoryFactory name="DirectoryFactory"
      class="${solr.directoryFactory:solr.NRTCachingDirectoryFactory}">

     <lockType>${solr.lock.type:native}</lockType>
     ```

5. Update the modified configs in Zookeeper.

   ```
   # solrctl --jaas $SOLR_PROCESS_DIR/jaas.conf instancedir --update
    atlas_configs /tmp/atlas_configs

   # solrctl --jaas $SOLR_PROCESS_DIR/jaas.conf instancedir --update
    ranger_audits /tmp/ranger_audits
   ```

6. Backup the Solr collections.

   - When migrating your data storage location from local file system to HDFS, run:

     ```
     # curl -k --negotiate -u : "https://$(hostname
      -f):8995/solr/admin/collections?action=BACKUP&name=vertex_backup&col
     lection=vertex_index&
     ```

```
location=hdfs://<Namenode_Hostname>:8020/solr-backups"
```

In the preceding command, the important points are name, collection, and location:

name

specifies the name of the backup. It should be unique per collection

collection

specifies the collection name for which the backup will be performed

location

specifies the HDFS path, where the backup will be stored

Repeat the curl command for different collections, modifying the parameters as necessary for each collection.

The expected output would be -

```
"responseHeader":{
 "status":0,
 "QTime":10567},
"success":{
 "Solr_Server_Hostname:8995_solr":{
   "responseHeader":{
   "status":0,
   "QTime":8959}}}}
```

- When migrating your data storage location from HDFS to local file system:

  Refer to Back up a Solr collection for specific steps, and make the following adjustments:

  - If TLS is enabled for the Solr service, specify the trust store and password by using the ZKCLI_JVM_FLAGS environment variable before you begin the procedure.

    ```
    # export ZKCLI_JVM_FLAGS="-Djavax.net.ssl.trustStore=/path/to/
    truststore.jks -Djavax.net.ssl.trustStorePassword="
    ```

  - Create Snapshot

    ```
    # solrctl --jaas $SOLR_PROCESS_DIR/jaas.conf collection --create-
    snapshot <snapshot_name> -c <collection_name>
    ```

  - or use the Solr API to take the backup:

    ```
    curl -i -k --negotiate -u : "https://(hostname -f):8995/solr/admin/
    collections?
    action=BACKUP&name=ranger_audits_bkp&collection=ranger_audits&location=/
    path/to/solr-backups"
    ```

  - Export Snapshot

    ```
    # solrctl --jaas $SOLR_PROCESS_DIR/jaas.conf collection
     --export-snapshot <snapshot_name> -c <collection_name> -d
     <destination_directory>
    ```

    Note: The <destination_directory> is a HDFS path. The ownership of this directory should be solr:solr.

7. Delete the collections from the original location.

   All instances of Solr service should be up, running, and healthy before deleting the collections. Use Cloudera Manager to check for any alerts or warnings for any of the instances. If alerts or warnings exist, fix those before deleting the collection.

   ```
   # solrctl collection --delete edge_index
   # solrctl collection --delete vertex_index
   # solrctl collection --delete fulltext_index
   ```

```
# solrctl collection --delete ranger_audits
```

8. Verify that the collections are deleted from the original location.

```
# solrctl collection --list
```

This will give an empty result.

9. Verify that no leftover directories for any of the collections have been deleted.
   - When migrating your data storage location from local file system to HDFS:

     ```
     # cd /var/lib/solr-infra
     ```

     Get the value of "Solr Data Directory, using Cloudera Manager Solr Configuration .

     ```
     # ls -ltr
     ```
   - When migrating your data storage location from HDFS to local file system, replace these two lines:

     ```
     # hdfs dfs -ls /solr/<collection_name>
     ```

     Note: If any directory name which starts with the collection name deleted in Step 7. exists, delete/move the directory to another path.

10. Restore the collection from backup to the new location.

Refer to Restore a Solr collection, for more specific steps.

```
# curl -k --negotiate -u : "https://$(hostname
 -f):8995/solr/admin/collections?
action=RESTORE&name=<Name_of_backup>&location=hdfs:/
<<Namenode_Hostname>:8020/solr-backups&collection=<Collection_Name>"
```

```
# solrctl collection --restore ranger_audits
 -l hdfs://<Namenode_Hostname>:8020/solr-backups
 -b ranger_backup -i ranger1
```

The request id must be unique for each restore operation, as well as for each retry.

To check the status of restore operation:

```
# solrctl collection --request-status <requestId>
```

Note: If the Atlas Collections (vertex_index, fulltext_index and edge_index) restore operations fail, restart the solr service and rerun the restore command. Now, the restart operations should complete successfully.

11. Verify the Atlas & Ranger functionality.

Verify that both Atlas and Ranger audits functions properly, and that you can see the latest audits in Ranger Web UI and latest lineage in Atlas.

- To verify Atlas audits, create a test table in Hive, and then query the collections to see if you are able to view the data.
- You can also query the collections every 20-30 seconds (depending on how other services utilize Atlas/Ranger), and verify if the "numDocs" value increases at every query.

```
# curl -k --negotiate -u : "https://$(hostname -f):8995/solr/edge_index/
select?q=*%3A*&wt=json&ident=true&rows=0"
# curl -k --negotiate -u : "https://$(hostname -f):8995/solr/vertex_index/
select?q=*%3A*&wt=json&ident=true&rows=0"
# curl -k --negotiate -u : "https://$(hostname -f):8995/solr/
fulltext_index/select?q=*%3A*&wt=json&ident=true&rows=0"
# curl -k --negotiate -u : "https://$(hostname -f):8995/solr/
ranger_audits/select?q=*%3A*&wt=json&ident=true&rows=0"
```