Post-Transition Steps

Date published: 2020-04-30
Date modified:

https://docs.cloudera.com/

Legal Notice

Contents

Post transition steps

You must complete the post transition steps to start the services in Cloudera Private Cloud Base.

The AM2CM tool transitions the component configurations. However, you must configure and perform additional steps to start the services in Cloudera Private Cloud Base.

Note:

- Review configuration warnings for all the services.
- Review JVM parameters and configurations for all services as some of the JVM parameters and configurations are not transitioned. Most of the heap configurations are transitioned.
- Review the Log4j configurations. Log4j configurations such as logs dir, size, and backup index are transitioned to Cloudera Manager as it uses the default log4j settings of Cloudera Manager.

Note: If the cluster is kerberized, you can generate keytabs in Cloudera Manager under Security section ->Kerberos Credentials -> Generate Missing Credentials

Enable Auto Start setting

Ensure that you enable Auto Start Settings in Cloudera Manager for all the components.

About this task

If you enable Auto Start Settings in Ambari, the Ambari services are configured to start automatically on system boot. However, post migration to Cloudera Manager, this Auto Start Settings field is enabled only for Zookeeper-Server. For other components, enable this setting manually in Cloudera Manager.

Procedure

1. In the Search box, search for Automatically Restart Process.
   Results are displayed for all the components.
2. Select the component from the results displayed.
3. Click the Automatically Restart Process checkbox.

ZooKeeper

You must complete the following steps to start the ZooKeeper service.

Procedure

1. Remove or move the myid file from the ZooKeeper server hosts. The path is ${dataDir}/myid. For example,
   # mv /hadoop/zookeeper/myid /hadoop/zookeeper/myid.bak
2. Start ZooKeeper service from Cloudera Manager

Delete ZNODES

Remove the ZNode for HDFS, YARN, Hive, HBase, and Oozie

About this task

If the cluster is configured for Namenode HA, the Failover controllers cannot determine the Active Namenode after the migration. This is because the naming conventions used to identify Namenodes for the HA service are different

for Ambari and Cloudera Manager. You must format the ZooKeeper ZNode for the Failover controller to continue further.

If the cluster is configured for Yarn HA, the Resource Managers cannot elect an Active Resource Manager after the migration. You must clear the previous Znode from ZooKeeper so that YARN replaces it.

If the cluster is configured for Ranger KMS, you must delete the Ranger KMS Znode with superuser privileges. When you restart Ranger KMS, the znode gets created with correct privileges.

You must format the ZooKeeper Znode for the failover controller to continue further.

Procedure

1. Start ZooKeeper client CLI session from a master node. (Assuming that the skipACL is set to Yes to avoid authentication issues). zookeeper-client -server <zk_hostname>
2. Remove the HDFS HA Failover controller Znode
   deleteall /hadoop-ha
3. Remove the YARN ZNode
   deleteall /yarn-leader-election
4. Remove the Hive ZNode
   deleteall /hive
   deleteall /hiveserver2
5. Remove the HBase ZNode
   deleteall /hbase-secure
6. Remove the Oozie ZNode
   deleteall /oozie
7. Remove the Ranger KMS ZNode
   deleteall /zktsm

Ranger

To add the Ranger Service to your cluster, use the available database information.

While setting up the service, use the database configured from the Ambari instance. Once Ranger is set up, it connects to the services available on the cluster. The connection creates a NEW Ranger repository for each of these services. Each repository contains default configurations. All the Service Repositories that were defined in Ranger from the Ambari managed cluster are visible. This contains all the custom policies.

To migrate policies from the old repositories to the new repositories use a simple python REST API application. For more information consult the Cloudera account team for this utility.

Ranger Policy Migration Tool

There are multiple service repositories present in each service after the migration from Ambari to Cloudera Manager. The new repository name appears as cm_* and contains the basic defaults installed for new deployments, in addition to being properly integrated with the other services under Cloudera Manager management.

About this task

Use the Ranger policy migration tool to migrate the old policies to the new repository created by the Ranger Service. To download the ranger policy migration tool, see Software download matrix

Procedure

- Usage

```
Usage: ranger_policy_migration.py [options]
```

- Options:

```
 -h, --help            show this help message and exit
    -l LOG, --log=LOG     Activity Log
```

- Connection:

```
   URL, User and password information needed to connect to Ranger
    -u HOST_URL, --ranger-host-url=HOST_URL
    Ranger Host Base URL
    -n USER, --username=USER
    Username
    -p PASSWORD, --password=PASSWORD
    Password
    -c CREDENTIALS, --credentials=CREDENTIALS
    Credentials File
```

- Repo Details:

```
   Repositories details for migration
    -f REPO_FROM, --from-repo=REPO_FROM
    Ranger Repository "from" Name
    -t REPO_TO, --to-repo=REPO_TO
    Ranger Repository "to" Name
    -m METHOD, --migration-method=METHOD
    Migration Method: m(move) or u(upsert). u is default
```

- Migration Control:

```
    Optional, define include/exclude policy "id's" to handle

    -e EXCLUDE_POLICIES, --exclude-policies=EXCLUDE_POLICIES
    Comma separated list of policy id's to exclude
    -i INCLUDE_POLICIES, --include-policies=INCLUDE_POLICIES
    Comma separated list of policy id's to include
```

Example

This example explains Ranger policy migration tool.

About this task

Example

Procedure

1. Upsert policies from repo hdp_hdfs to repo cm_hdfs (Recommended). The `-t` and `-f` repositories must be of the same type. ./ranger_policy_migration.py -u http://my_ranger.server.org:6080 -n admin -p admin -f hdp_hdfs -t cm_hdfs This matches and updates the policies in the to repository with the ones in the from repository. Matching is done based on the resource path. Whenever a matching policy is not found, it creates a new policy.
2. Move policies between repositories with the `-m` option. The `-t` and `-f` repositories must be of the same type. ./ranger_policy_migration.py -u http://my_ranger.server.org:6080 -n admin -p admin -f hdp_hdfs -t cm_hdfs -m m

3. Restrict or include policies between repositories with the `-i` and `-e` option. The `-t` and `-f` repositories must be of the same type../ranger_policy_migration.py -u http://my_ranger.server.org:6080 -n admin -p admin -f hdp_hdfs -t cm_hdfs -i 207
4. The `-e` option is the Policy Id in the *from* repository. Only non-matching policies are considered. ./ranger_policy_migration.py -u http://my_ranger.server.org:6080 -n admin -p admin -f hdp_hdfs -t cm_hdfs -e 210

   You can use these two options together.

What to do next

Run process

Gather the matching *from* and *to* repositories for each service type and run this tool for each set. It can include, but not limited to: HDFS, Hive, YARN, HBase, Kafka, KNOX, Solr, and Atlas.

Migrating Policy Activities

There are two administration logins for policy management. One for KMS, which uses the keyadmin login credentials and another for the remaining services. These include HDFS, HBase, Hive, YARN, Knox, Kafka, and Atlas.

Using the above examples and the admin login that matches the service you want to migrate, issue the migration for each service.

- HDFS - ./ranger_policy_migration.py -u http://my_ranger.server.org:6080 -n admin -p <admin_pw> -f < existing_service> -t cm_hdfs
- HBase - ./ranger_policy_migration.py -u http://my_ranger.server.org:6080 -n admin -p <admin_pw> -f < existing_service> -t cm_hbase
- Hive - ./ranger_policy_migration.py -u http://my_ranger.server.org:6080 -n admin -p <admin_pw> -f < existing_service> -t cm_hive
- YARN - ./ranger_policy_migration.py -u http://my_ranger.server.org:6080 -n admin -p <admin_pw> -f < existing_service> -t cm_yarn
- Knox - ./ranger_policy_migration.py -u http://my_ranger.server.org:6080 -n admin -p <admin_pw> -f < existing_service> -t cm_knox
- Kafka - ./ranger_policy_migration.py -u http://my_ranger.server.org:6080 -n admin -p <admin_pw> -f < existing_service> -t cm_kafka
- Atlas - ./ranger_policy_migration.py -u http://my_ranger.server.org:6080 -n admin -p <admin_pw> -f < existing_service> -t cm_atlas
- KMS - ./ranger_policy_migration.py -u http://my_ranger.server.org:6080 -n keyadmin -p <keyadmin_pw> -f <existing_service> -t cm_kms

Note: Change log in for KMS.

HDFS

Perform the following post migration steps.

Configure the ports, check the HA NameNode, custom topology, and review other configurations to ensure that HDFS service is set up.

Ports

Ensure that you review service configuration warnings and address inconsistencies.

Start all role instances in HDFS service.

HDFS HA

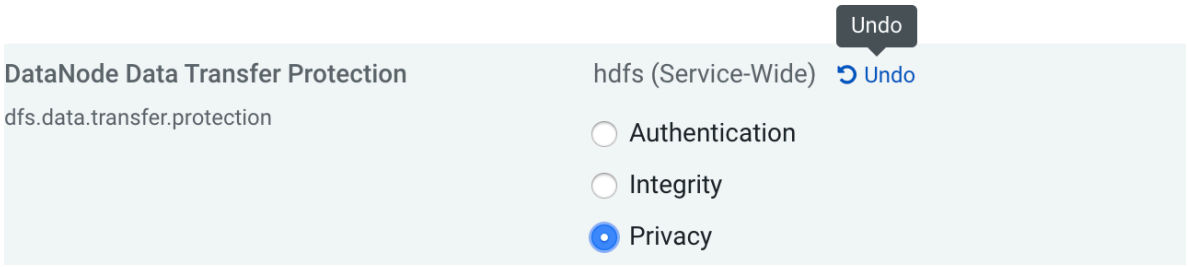
You must check HDFS HA and set Failover controller.

About this task
Perform the task.

Procedure

1. Cloudera Manager does not support the comma separated entries. Review dfs.data.transfer.protection and hadoop.rpc.protection parameters. If SSL is not enabled, then click Undo. This will clear the selections.

   Note: Privacy option requires SSL.

2. Start HDFS service.
   Failover Controller fails to start. You must continue with step 3 or 4.
3. Format the ZooKeeper Failover controller ZnNode. SSH to the Failover controller host And perform the following using the example:# cd /var/run/cloudera-scm-agent/process/<xx>-hdfs-FAILOVERCONTROLLER /opt/cloudera/parcels/CDH/bin/hdfs --config /var/run/cloudera-scm-agent/process/<xx>-hdfs-FAILOVERCONTROLLER/ zkfc -formatZK
   a) Check the output in the log for success. If the error is: `ERROR tools.DFSZKFailoverController: DFSZKFailOverController exiting due to earlier exception java.io.IOException: Running in secure mode, but config doesn't have a keytab`.
   b) The ZooKeeper nodes are not created. You must manually create the ZNodes for the Failover controllers to start and allow an Active Namenode to be elected.
   c) With the zookeeper-client, open a ZooKeeper Shell `zookeeper-client -server <a_zk_server>`
   d) Create the required ZNodes create /hadoop-ha create /hadoop-ha/<namenode_namespace>.
4. This is an alternative step to the above step 3.
   a) Log in to Cloudera Manager.
   b) Navigate to Clusters
   c) Click HDFS
   d) Go to Configuration tab
   e) Under Filters > Scope, click Failover Controller
   f) Navigate to the Failover Controller instance
   g) Initialize Automatic Failover Znode.
5. Start HDFS service again.

Custom Topology

Custom topology configuration in Cloudera Manager.

In the Ambari HDFS service, if net.topology.script.file.name is configured with custom topology file, then, in Cloudera Manager > HDFS Configuration page, you must configure net.topology.script.file.name with the same file.

Add Balancer Role to HDFS

Use HDFS Balancer for balancing the data. This section is not applicable if you are upgrading to CDP Private Cloud Base 7.1.7.

The HDFS Balancer is a tool for balancing the data across the storage devices of a HDFS cluster. For more information on HDFS Balancer, see *Balancing data across an HDFS cluster*.

Related Information

Balancing data across an HDFS cluster

Other review configurations for HDFS

1. Review dfs.client.failover.proxy.provider.mycluster value. If this parameter has a non-default value, then configure this parameter in Cloudera Manager. This configuration is not migrated from the Ambari managed HDP cluster to Cloudera Manager.
2. Review and add the cloud service related configurations as those are not transitioned to Cloudera Manager.

Solr

You must create a Ranger Plug-in audit directory and HDFS Home directory and start the Solr service.

On the Cloudera Manager UI:

1. Initialize Solr
2. Create Ranger Plug-in audit directory
3. Create HDFS Home directory
4. Start Solr service

Kafka

You must cleanup metadata on broker hosts after migrating the Ambari-managed HDP cluster to Cloudera Private Cloud Base. If you are upgrading to CDP Private Cloud Base 7.1.7, you can ignore step 3, step 4, and step 5.

About this task

Cleanup Metadata on Broker Hosts

Procedure

1. On each Broker host, remove $log.dirs/meta.properties file from Kafka broker hosts. For example,
   `#mv /grid/0/kafka-logs/meta.properties /tmp`.

   Remove ${log.dirs}/meta.properties file from Kafka broker.

   If the Kafka log.dirs property points to /kafka-logs, then the command is

   ```
   #mv /kafka-logs/meta.properties /tmp
   ```
2. Set the Kafka port value. For more information, see Change Kafka Port Value.
3. Enable kerberos.auth.enable. For more information, see Kafka cluster Kerberos.
4. In the Ambari kafka configuration, get the zookeeper.connect configuration value. Find the path defined at the end after port number and update the path at Cloudera Manager Kafka zookeeper.chroot configuration. For example, if the path is zookeeper.connect=hostname1:port1,hostname2:port2,hostname3:port3/chroot/path, then the path is /chroot/path. Another example, if the path is zookeeper.connect=hostname1:port1,hostname2:port2,hostname3:port3, then the path is /. Also, the default path is /.

5. Update the broker IDs:
   a. Log in to Cloudera Manager
   b. Navigate to Clusters
   c. Select the Kafka service
   d. Navigate to the Configurations tab
   e. Search for broker.id and update the IDs for each hostname using the output of the third step of Extract broker ID Procedure 2
6. Start the Kafka service.

YARN

Perform the following post migration steps.

Start job history

You must start job history after migration.

About this task

To start job history after migration

Procedure

1. Log in to Cloudera Manager
2. Navigate to Clusters
3. Go to YARN
4. Under the Actions drop-down, click Create Job History Dir
5. Create Job History Dir dialog box appears, click Create Job History Dir.
   Successfully created HDFS directory. message appears

Yarn Mapreduce framework jars

About this task

Perform the following steps

Procedure

1. Go to Cloudera Manager UI.
2. Navigate to YARN
3. Under Actions drop-down, click Install Yarn Mapreduce framework jars

YARN NodeManager

Set the file permission for Cloudera Manager deployment.

If yarn.nodemanager.linux-container-executor.group setting is set to hadoop in your HDP cluster, you must reset yarn.nodemanager.linux-container-executor.group setting to yarn in Cloudera Manager.

If this parameter is not set correctly,

```
Issue Yarn NodeManagers fail to start with error
```

error appears.

1. Log into Cloudera Manager UI.
2. Navigate to Clusters

3. Select the YARN service
4. Go to Configurations
5. Search for yarn.nodemanager.linux-container-executor.group and reset this to yarn

Spark

You must initialize a few directories for the Spark service.

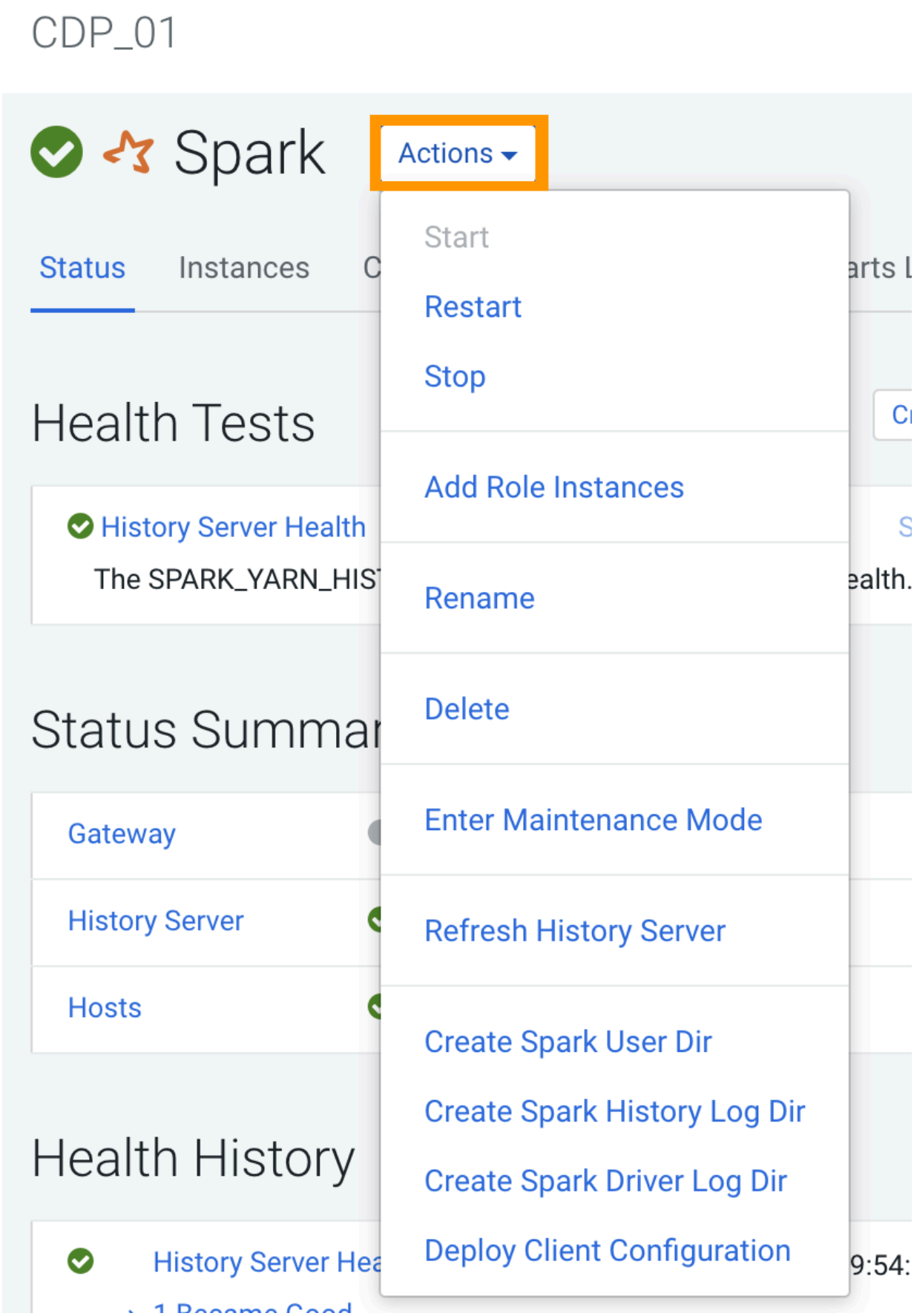
Before you begin

Run the following directory options:

Procedure

- Create Spark User Dir
- Create Spark History Log Dir

- Create Spark Driver Log Dir


CDP_01
Spark
Actions
Start
Restart
Stop
Add Role Instances
Rename
Delete
Enter Maintenance Mode
Refresh History Server
Create Spark User Dir
Create Spark History Log Dir
Create Spark Driver Log Dir
Deploy Client Configuration
Status
Instances
Health Tests
History Server Health
Status Summar
Gateway
History Server
Hosts
Health History

Livy2

Review and configure Livy2 parameters.

Livy server heap/memory parameters are not migrated automatically when migrating Ambari-managed HDP cluster to Cloudera Private Cloud Base. You must review and configure these parameters in Cloudera Manager.

Tez

Install the Tez tar files on HDFS.

About this task

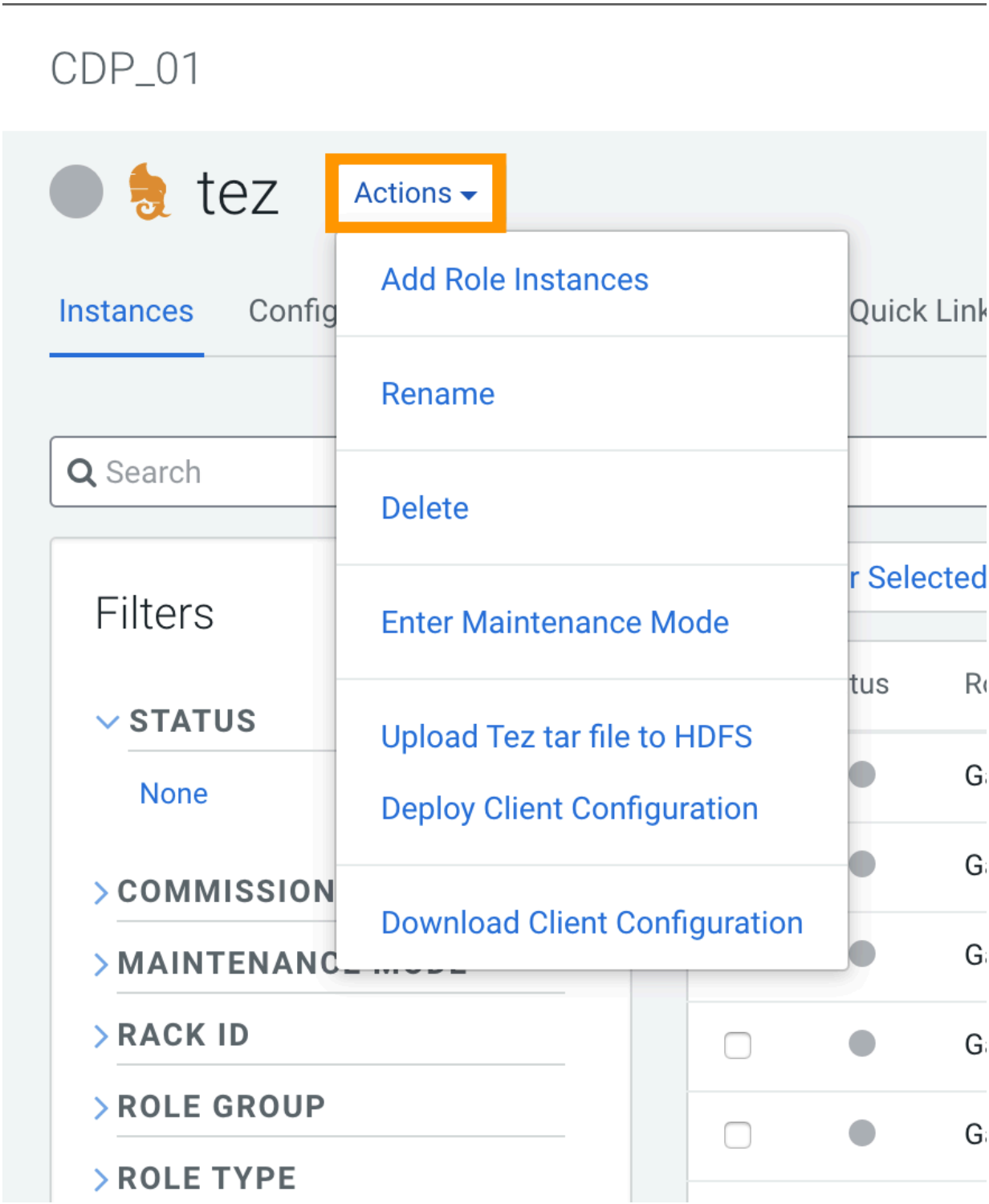
From the Tez cluster service, install Tez tar files on HDFS used by Hive use and then deploy Client Configuration.

Procedure

1. Run Upload Tez tar file to HDFS

2. Deploy Client Configuration.

Note: You must provide sufficient permissions to HDFS or Ranger to install Tez and deploy configuration.



Hive

Perform the post migration steps. If you are expediting the Hive upgrade process and modified the upgrade process to skip materializing every table in the metastore, you need to modify the Hive Strict Metastore Migration (HSMM) process by running the Hive Upgrade Check tool and provided scripts.

Related Information

Hive Post-Upgrade Tasks

Hive Metastore

Identifying and fixing invalid Hive schema versions

As Administrator, after upgrading from Ambari-managed HDP to Cloudera Private Cloud Base, you need to identify Hive metastore operations that might fail due to Hive schema version incompatibility.

About this task

Incompatibility might exist if the upgrade process failed to make schema updates. You need to turn on the Hive Metastore Schema validation process for the metastore during the migration of your workloads to CDP. The Hive metastore captures any schema updates that occur during the upgrade, and displays issues in the Hive metastore logs. With this information, you can use the Apache Hive Schema tool to fix any problems.

Procedure

1. In Cloudera Manager, click Clusters HIVE Configuration .
2. Check the hive.metastore.server.max.message.size.

3. Set hive.metastore.server.max.message.size to the recommended value: 10% of the value of your Java heap size for Hive Metastore Server in bytes, but no more than 21478364. Recommended value: 214748364
4. Click Clusters HIVE Configuration , and search for schema.
5. Check Strict Hive Metastore Schema Validation to set hive.metastore.schema.verification to true.
6. Check the Hive metastore logs and set a compatible metastore schema for the current Hive version using the Apache Hive Schema Tool.

Safety Valve Entries

Depending on the complexities of the previous Hive environment, there can be several safety valve Entries that were taken during the transition. Review and verify them carefully.

Oozie

Perform the following post migration tasks by validating the database URL, installing the new shared libraries, accessing Oozie Loadbalancer URL, and configuring load balancer.

Validate Database URL

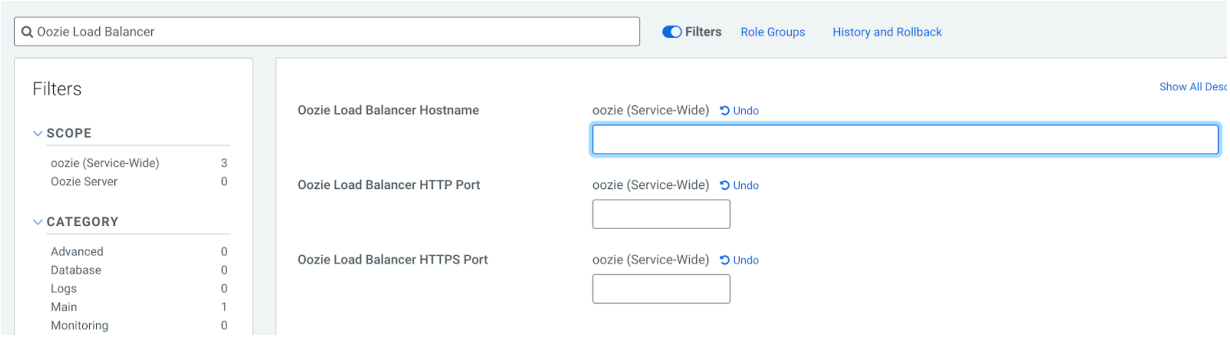
Validate the database URL.

Before you begin

Validate the configured Oozie JDBC URL. Before starting the Oozie service, check the value.

Procedure

1. Verify JDBC Database settings in Oozie service.
2. Configure Oozie Load Balancer Hostname, Oozie Load Balancer HTTP Port and Oozie Load Balancer HTTPS Port if OOZIE HA is enabled.



3. In Oozie service configurations under Oozie Server Advanced Configuration Snippet (Safety Valve) for oozie-site.xml add oozie.service.AuthorizationService.admin.users configuration with value of oozie, oozie-admin.

Note: Replace Oozie with the service account username if it is different from default one.

Installing the new Shared Libraries

Install new Shared Libraries

About this task

Follow the steps:

Procedure

1. Start Oozie services.
2. Set ShareLib Root Directory=/user/<oozie user> in Oozie configs
3. From Cloudera Manager, navigate to Cloudera Manager > Oozie > Install Oozie Shared Lib

Note: Tez libraries are removed from Oozie ShareLib. If you are using Tez on Oozie, then you must manually copy the Tez jar files and execute a ShareLib update.

Phoenix

Add the Apache Phoenix service. If you are using Phoenix Query Server (PQS) in your source HDP deployment, you must manually add the Apache Phoenix service using Cloudera Manager to complete the Phoenix upgrade.

About this task

You must add the Apache Phoenix service using Cloudera Manager. To add the Apache Phoenix service:

Procedure

1. In Cloudera Manager, click Home 

.
2. Click  to the right of the cluster name and select Add Service. A list of service types display. Select Phoenix from the list and click Continue. Follow the wizard to add the Phoenix service.

Map Phoenix schemas to HBase namespaces

HBase namespaces enable tighter control of where a particular data set is stored on the HBase RegionServers. If you have mapped Phoenix schemas to HBase namespaces in your source HDP cluster, you must enable namespace mapping by configuring a set of properties using Cloudera Manager.

You can check if you have enabled namespace mapping in source HDP deployment Ambari's HBase service. Check the following HBase configurations in your source HDP deployment to see if you have mapped Phoenix schemas to HBase namespaces.

HBase Config	Value
phoenix.schema.isNamespaceMappingEnabled	true
phoenix.schema.mapSystemTablesToNamespace	true

If you have enabled namespace mapping in HDP, and want to use in CDP, you must enable namespace mapping in CDP using Cloudera Manager. For more information, see *Enable namespace mapping* using the link in the related information section.

Related Information

Enable namespace mapping

Starting all services

You must now start all the services.

Start all the services and ensure that all the services of Cloudera Private Cloud Base are up and running.

Enabling the Ranger and Knox services

Manually enable Ranger and Knox services.

The components like Ranger and Knox are not handled in the initial transition. You must migrate them manually in the Cloudera Manager cluster configuration.

YARN Queue Manager

Perform the following post migration step to add YARN Queue Manager.

Add YARN Queue Manager Service in Cloudera Manager to manage Yarn Queues.

Note: You must explicitly enable (kerberos.auth.enabled) Kerberos and restart the service before launching the UI.

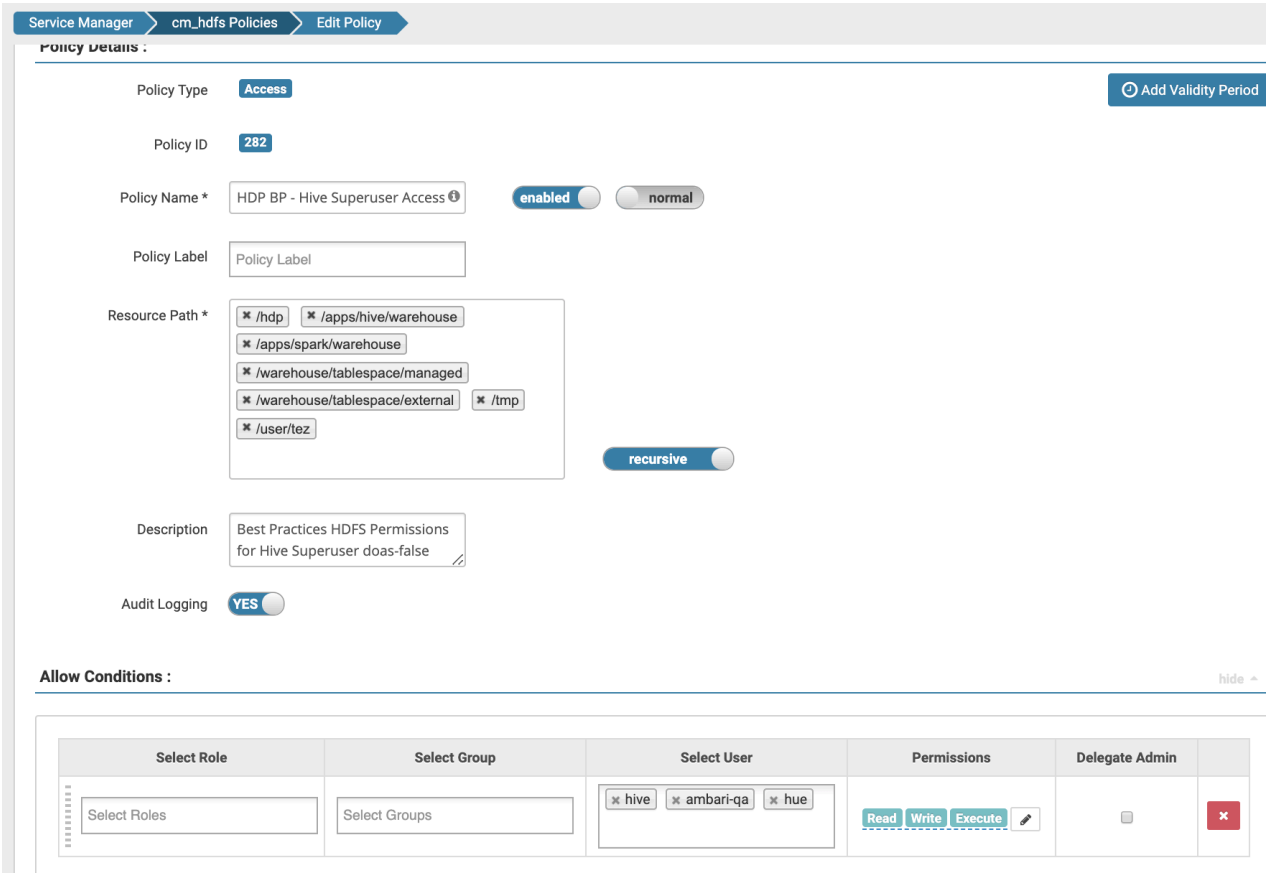
Enable Queue Manager Service under Yarn service configurations.

Hive Policy Additions

Hive Metastore Canary test might fail on clusters migrated from HDP because the service account used to perform these tests is different between the two management consoles.

The new service account is Hue. You must add the Hue user to the policies defined that hold Hive as the superuser. If there is no policy in Ranger defining Hive as a superuser across key directories in HDFS, then you are using impersonation. Impersonation is no longer the standard or recommended. Consult the Cloudera team for details.

Sample Hive or Hue Privileged User Policy



Activating Ranger for Services

Adding the Ranger Service to a cluster does not enable Ranger for each of the services automatically, even though policy repositories are configured for each. The services must enable Ranger integration independently.

About this task

After activating and restarting each of the services, test the successful integration with Ranger by exercising (with good and bad credentials), each of the services. Review the logs of each service to validate there are no Ranger credential or connectivity issues.

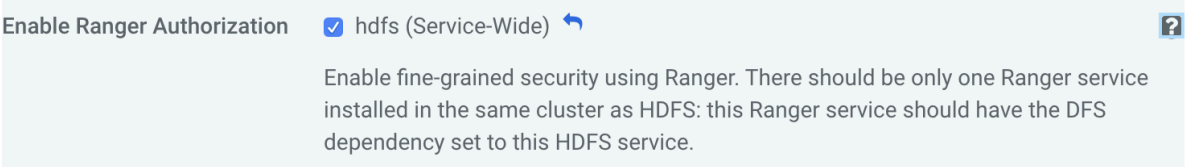
If the service has issues connecting or authorizing with Ranger, then it downloads the policies and does not function correctly. Each service attempts to authenticate with the Ranger service based on the way it is configured. For secure kerberized environments, it is the service keytab. For unsecured environments, it is the service startup user.

In both methods, the user must be available in the Ranger users list. If the service use is not available, the service is not authorized. You can view this in the Ranger access logs as well. Add the user to the Users list. Contact Cloudera support for any assistance.

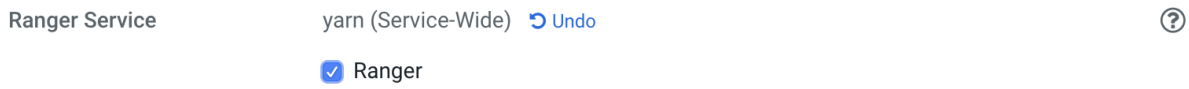
Procedure

- On your new cluster, navigate to Ranger service.
- Click Actions drop-down and click Setup Ranger Plugin Service.

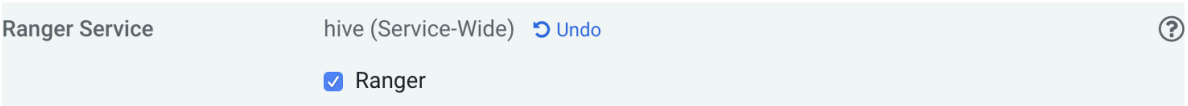
- HDFS: In the HDFS service configuration, select Enable Ranger Authorization.



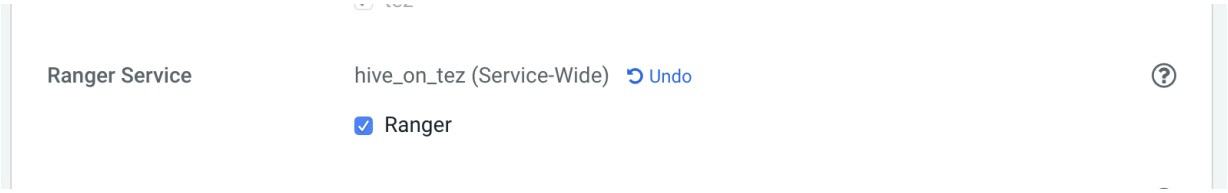
- YARN: In the Yarn service configuration, select Ranger Service



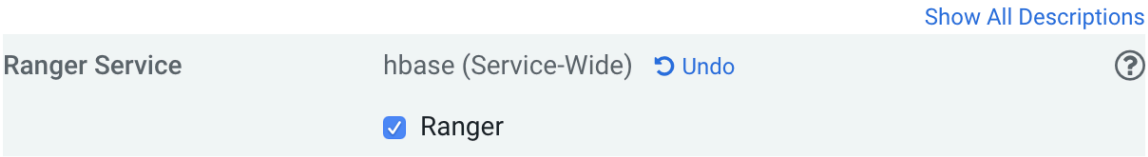
- Hive (Hive Metastore Integration): To enable the Hive Metastore integration with Ranger Hive Policies, select Ranger Service.



- Hive On Tez: In the Hive On Tez service configuration, select Ranger Service.



- HBase: In the HBase service configuration, select Ranger Service.



- Kafka: In the Kafka service configuration, select Ranger Service.

Knox

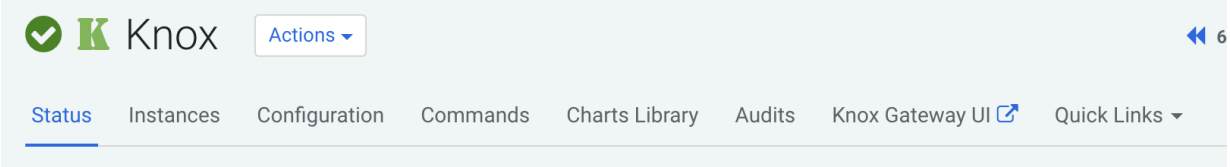
Perform the following post migration steps.

About this task

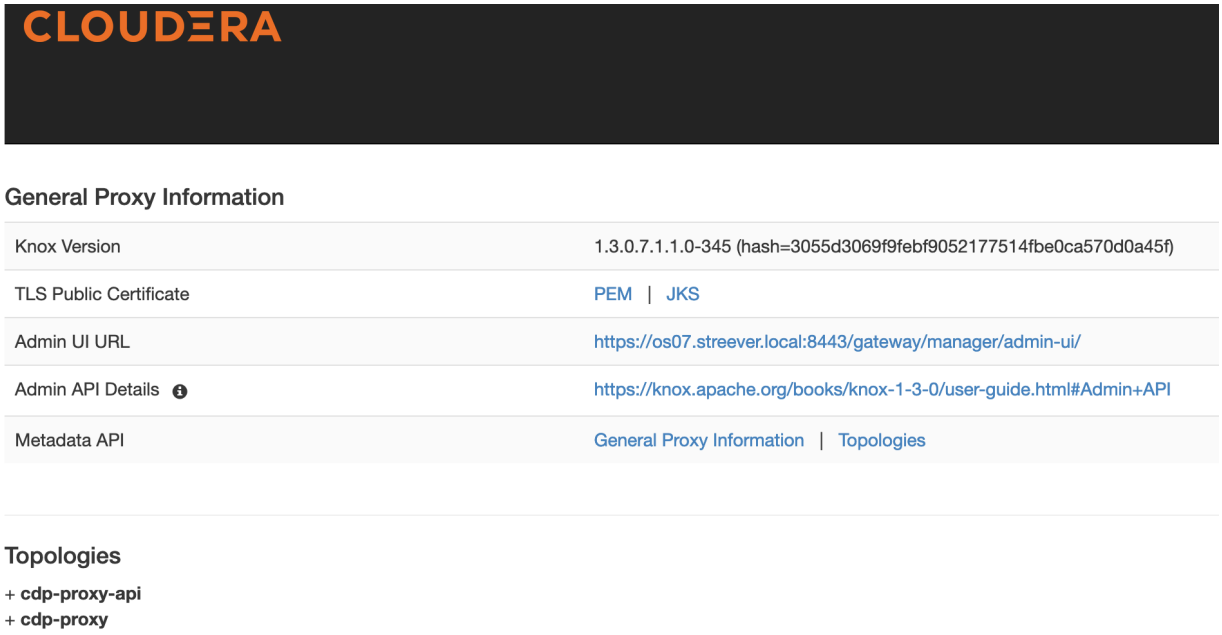
After the other services are configured, install the Knox service for the cluster. Start the Knox service.

Procedure

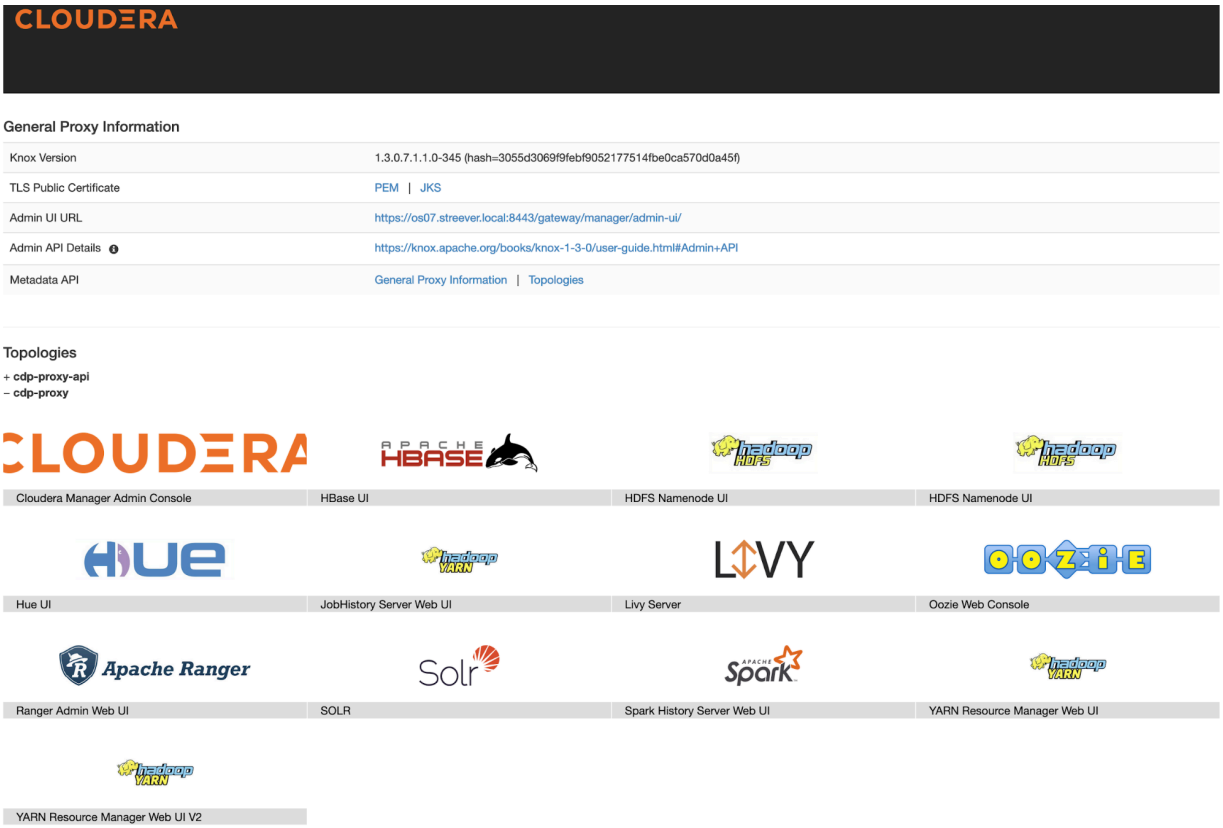
- Monitor the gateway.log during the first startup to view the list of discovery entries for Cloudera Manager. After the startup you can navigate to the KNOX Gateway user interface from the Knox service window in Cloudera Manager.



- In the Knox Gateway user interface, at least two topologies are auto-created through the discovery.

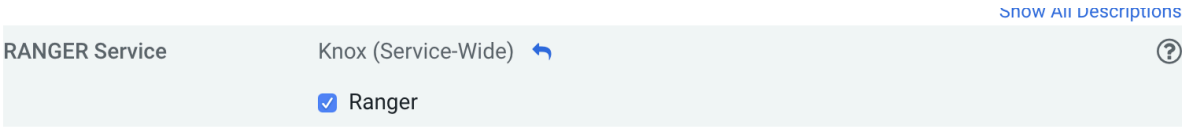


- Open one of the topologies to discover the proxy links for each of the discovered services.



What to do next

- If the web UIs associated with these services were kerberized, then the KNOX proxy URLs allow you to access them without SPNEGO.
- Make sure that each user interface for the services is kerberized and has users listed in each of the respective services' user lists to secure them.
- Configure TLS for the cluster and the services if you have not already done so.
- By default, KNOX leverages PAM integration for authentication. This is different from most KNOX configurations that were set up with direct LDAP integration through the shimo topology configurations. PAM integration requires hosts to have a valid sssd configuration with the controlling IDM for cluster identity.
- Ensure that the Ranger plugin is activated for KNOX and follow up on the policies in Ranger for Knox to control the access to the endpoints. Select Ranger Service.



- Enable SSO from the discovered services using either of these methods:
  - Enabling Kerberos
  - Ensuring a user is defined in the service user list for the proxied web UIs for each service.
  - Restart each service after making the required changes , if they are not already enabled.

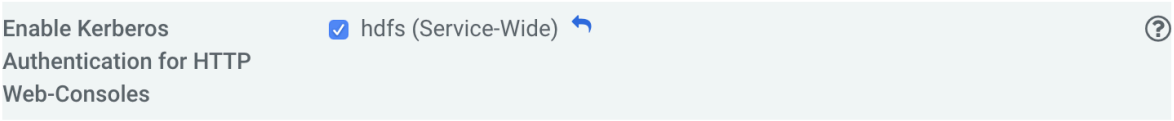
Enabling Knox for Various Services

Enable Knox for various services.

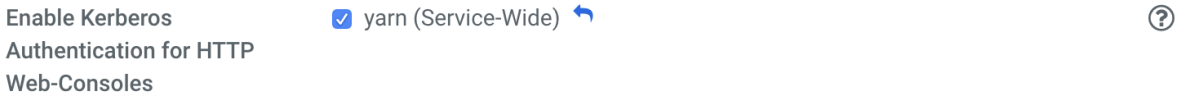
About this task

Procedure

- HDFS: Enable Kerberos HDFS Web UI by selecting Enable Kerberos Authentication for HTTP Web-Consoles.



- YARN: Enable Kerberos for YARN Web UI by selecting Enable Kerberos Authentication for HTTP Web-Consoles.



- Oozie: Enable Kerberos for Oozie Web UI.

  Note: Activate the Oozie Web UI by installing the Extjs library. Cloudera recommends using the Hue interface.

- Hue (Not working - redirect strips endpoint after login attempt to Hue UI): Ensure the target user is defined in the Hue User List.
- Spark: Select Enable User Authentication.

- Ranger: Ensure the SSO user is registered in the Ranger users list.

  Note: Knox creates a self-signed certificate for the user interface if TLS is not active in Cloudera Manager. The self-signed certificate generated is compatible with Mozilla Firefox. The self-signed certificate generated is not compatible with Google Chrome or Microsoft Edge browsers.

Client Configurations

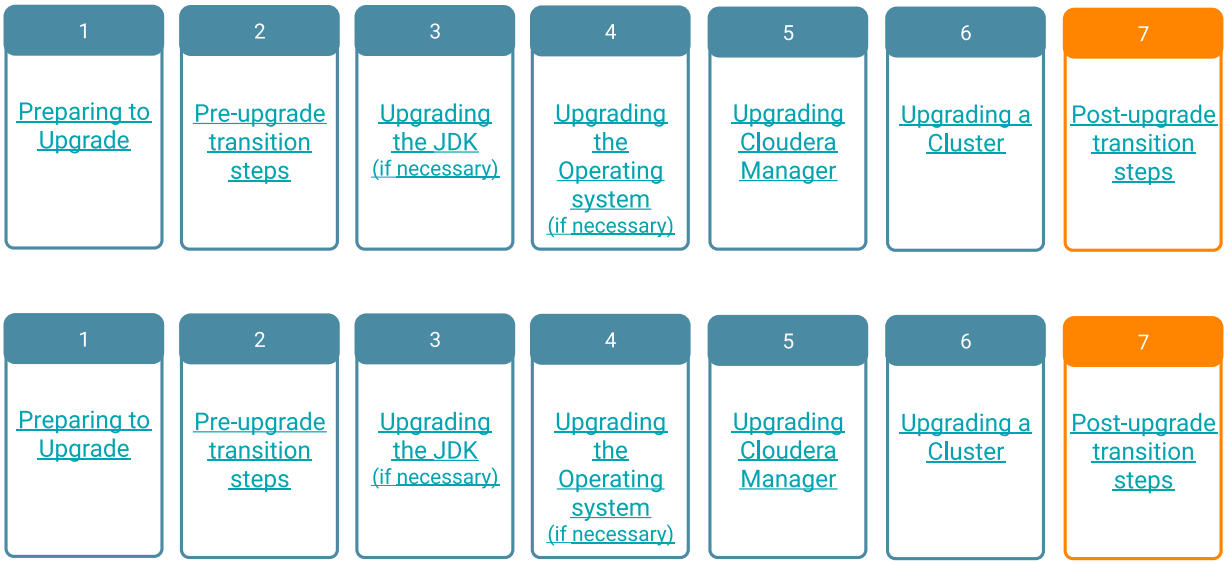
Migrating from Ambari to Cloudera Manager can leave the Ambari-managed HDP artifacts and links that may not have changed. After everything has been configured, deploy Client Configuration [Cloudera Manager > Clusters > Actions > Deploy Client Configurations]. This fixes any missing configuration references in the cluster.

Securing ZooKeeper

By default, the AM2CM tool adds the -Dzookeeper.skipACL=yes configuration to assist with the migration. You must remove the -Dzookeeper.skipACL=yes configuration under Java Configuration Options for Zookeeper Service to secure ZooKeeper and restart the service.

Apache Hive Changes in CDP

You need to know where your tables are located and the property changes that the upgrade process makes. You need to perform some post-migration tasks before using Hive tables and handle semantic changes.



1
Preparing to Upgrade
2
Pre-upgrade transition steps
3
Upgrading the JDK (if necessary)
4
Upgrading the Operating system (if necessary)
5
Upgrading Cloudera Manager
6
Upgrading a Cluster
7
Post-upgrade transition steps

Understanding Apache Hive 3 major design features, such as default ACID transaction processing, can help you use Hive to address the growing needs of enterprise data warehouse systems.

If you are expediting the Hive upgrade process and modified the upgrade process to skip materializing every table in the metastore, you need to modify the Hive Strict Metastore Migration (HSMM) process by running the Hive Upgrade Check tool and provided scripts. Scripts are not included to address legacy Kudu storage handler classes.

Hive Configuration Property Changes

You need to know the property value changes made by the upgrade process as the change might impact your work. You might need to consider reconfiguring property value defaults that the upgrade changes.

Hive Configuration Property Values

The upgrade process changes the default values of some Hive configuration properties and adds new properties. The following list describes those changes that occur after upgrading from CDH or HDP to .

datanucleus.connectionPool.maxPoolSize

Before upgrade: 30

After upgrade: 10

datanucleus.connectionPoolingType

Before upgrade: BONECP

After upgrade: HikariCP

hive.auto.convert.join.noconditionaltask.size

Before upgrade: 20971520

After upgrade: 52428800

Exception: Preserves pre-upgrade value if old default is overridden; otherwise, uses new default.

hive.auto.convert.sortmerge.join

Before upgrade: FALSE in the old CDH; TRUE in the old HDP.

After upgrade: TRUE

hive.auto.convert.sortmerge.join.to.mapjoin

Before upgrade: FALSE

After upgrade: TRUE

hive.cbo.enable

Before upgrade: FALSE

After upgrade: TRUE

hive.cbo.show.warnings

Before upgrade: FALSE

After upgrade: TRUE

hive.compactor.worker.threads

Before upgrade: 0

After upgrade: 5

hive.compute.query.using.stats

Before upgrade: FALSE

After upgrade: TRUE

hive.conf.hidden.list

Before upgrade:

```
javax.jdo.option.ConnectionPassword,hive.server2.keystore.passwo
rd,hive.metastore.dbaccess.ssl.truststore.password,fs.s3.awsAcce
ssKeyId,fs.s3.awsSecretAccessKey,fs.s3n.awsAccessKeyId,fs.s3n.aw
sSecretAccessKey,fs.s3a.access.key,fs.s3a.secret.key,fs.s3a.prox
y.password,dfs.adls.oauth2.credential,fs.adl.oauth2.credential,f
s.azure.account.oauth2.client.secret
```

After upgrade:

```
javax.jdo.option.ConnectionPassword,hive.server2.keystore.passwo
rd,hive.druid.metadata.password,hive.driver.parallel.compilation
.global.limit
```

hive.conf.restricted.list

Before upgrade:

```
hive.security.authenticator.manager,hive.security.authorization.
manager,hive.users.in.admin.role,hive.server2.xsrf.filter.enable
d,hive.spark.client.connect.timeout,hive.spark.client.server.con
nect.timeout,hive.spark.client.channel.log.level,hive.spark.clie
nt.rpc.max.size,hive.spark.client.rpc.threads,hive.spark.client.
secret.bits,hive.spark.client.rpc.server.address,hive.spark.clie
nt.rpc.server.port,hive.spark.client.rpc.sasl.mechanisms,hadoop.
bin.path,yarn.bin.path,spark.home,bonecp.,hikaricp.,hive.driver.
parallel.compilation.global.limit,_hive.local.session.path,_hive
.hdfs.session.path,_hive.tmp_table_space,_hive.local.session.pat
h,_hive.hdfs.session.path,_hive.tmp_table_space
```

After upgrade:

```
hive.security.authenticator.manager,hive.security.authorization.
manager,hive.security.metastore.authorization.manager,hive.secur
ity.metastore.authenticator.manager,hive.users.in.admin.role,hiv
e.server2.xsrf.filter.enabled,hive.security.authorization.enable
d,hive.distcp.privileged.doAs,hive.server2.authentication.ldap.b
aseDN,hive.server2.authentication.ldap.url,hive.server2.authenti
```

```
cation.ldap.Domain,hive.server2.authentication.ldap.groupDNPatte
rn,hive.server2.authentication.ldap.groupFilter,hive.server2.aut
hentication.ldap.userDNPattern,hive.server2.authentication.ldap.
userFilter,hive.server2.authentication.ldap.groupMembershipKey,h
ive.server2.authentication.ldap.userMembershipKey,hive.server2.a
uthentication.ldap.groupClassKey,hive.server2.authentication.lda
p.customLDAPQuery,hive.privilege.synchronizer.interval,hive.spar
k.client.connect.timeout,hive.spark.client.server.connect.timeou
t,hive.spark.client.channel.log.level,hive.spark.client.rpc.max.
size,hive.spark.client.rpc.threads,hive.spark.client.secret.bits
,hive.spark.client.rpc.server.address,hive.spark.client.rpc.serv
er.port,hive.spark.client.rpc.sasl.mechanisms,bonecp.,hive.druid
.broker.address.default,hive.druid.coordinator.address.default,h
ikaricp.,hadoop.bin.path,yarn.bin.path,spark.home,hive.driver.pa
rallel.compilation.global.limit,_hive.local.session.path,_hive.h
dfs.session.path,_hive.tmp_table_space,_hive.local.session.path,
_hive.hdfs.session.path,_hive.tmp_table_space
```

hive.default.fileformat.managed

Before upgrade: None

After upgrade: ORC

hive.default.rcfile.serde

Before upgrade: org.apache.hadoop.hive.serde2.columnar.ColumnarSerDe

After upgrade: org.apache.hadoop.hive.serde2.columnar.LazyBinaryColumnarSerDe

Not supported in Impala. Impala cannot read Hive-created RC tables.

hive.driver.parallel.compilation

Before upgrade: FALSE

After upgrade: TRUE

hive.exec.dynamic.partition.mode

Before upgrade: strict

After upgrade: nonstrict

In , accidental use of dynamic partitioning feature is not prevented by default.

hive.exec.max.dynamic.partitions

Before upgrade: 1000

After upgrade: 5000

In , fewer restrictions on dynamic paritioning occur than in the pre-upgrade CDH or HDP cluster.

hive.exec.max.dynamic.partitions.pernode

Before upgrade: 100

After upgrade: 2000

In , fewer restrictions on dynamic paritioning occur than in the pre-upgrade CDH or HDP cluster.

hive.exec.post.hooks

Before upgrade:

```
com.cloudera.navigator.audit.hive.HiveExecHookContext,org.apache
.hadoop.hive.ql.hooks.LineageLogger
```

After upgrade: org.apache.hadoop.hive.ql.hooks.HiveProtoLoggingHook

A prime number is recommended.

hive.exec.reducers.max

Before upgrade: 1099

After upgrade: 1009

Exception: Preserves pre-upgrade value if old default is overridden; otherwise, uses new default

hive.execution.engine

Before upgrade: mr

After upgrade: tez

Tez is now the only supported execution engine, existing queries that change execution mode to Spark or MapReduce within a session, for example, fail.

hive.fetch.task.conversion

Before upgrade: minimal

After upgrade: more

hive.fetch.task.conversion.threshold

Before upgrade: 256MB

After upgrade: 1GB

Exception: Preserves pre-upgrade value if old default is overridden; otherwise, uses new default.

hive.hashtable.key.count.adjustment

Before upgrade: 1

After upgrade: 0.99

Exception: Preserves pre-upgrade value if old default is overridden; otherwise, uses new default.

hive.limit.optimize.enable

Before upgrade: FALSE

After upgrade: TRUE

hive.limit.pushdown.memory.usage

Before upgrade: 0.1

After upgrade: 0.04

Exception: Preserves pre-upgrade value if old default is overridden; otherwise, uses new default.

hive.mapjoin.hybridgrace.hashtable

Before upgrade: TRUE

After upgrade: FALSE

hive.mapred.reduce.tasks.speculative.execution

Before upgrade: TRUE

After upgrade: FALSE

hive.metastore.aggregate.stats.cache.enabled

Before upgrade: TRUE

After upgrade: FALSE

hive.metastore.disallow.incompatible.col.type.changes

Before upgrade: FALSE

After upgrade: TRUE

Schema evolution is more restrictive in than in CDH to avoid data corruption. The new default disallows column type changes if the old and new types are incompatible.

hive.metastore.dml.events

Before upgrade: FALSE

After upgrade: TRUE

hive.metastore.event.message.factory

Before upgrade: org.apache.hadoop.hive.metastore.messaging.json.ExtendedJSONMessageFactory

After upgrade: org.apache.hadoop.hive.metastore.messaging.json.gzip.GzipJSONMessageEncoder

hive.metastore.uri.selection

Before upgrade: SEQUENTIAL

After upgrade: RANDOM

hive.metastore.warehouse.dir

Before upgrade from CDH: /user/hive/warehouse

Before upgrade from HDP: /apps/hive/warehouse

After upgrade from CDH: /warehouse/tablespace/managed/hive

After upgrade from HDP: /warehouse/tablespace/managed/hive

For information about the location of old tables and new tables, which you create after the upgrade, see Changes to CDH Hive Tables or Changes to HDP Hive tables.

hive.optimize.metadataonly

Before upgrade: FALSE

After upgrade: TRUE

hive.optimize.point.lookup.min

Before upgrade: 31

After upgrade: 2

hive.prewarm.numcontainers

Before upgrade: 10

After upgrade: 3

hive.script.operator.env.blacklist

Before upgrade: hive.txn.valid.txns,hive.script.operator.env.blacklist

After upgrade: hive.txn.valid.txns,hive.txn.tables.valid.writeids,hive.txn.valid.writeids,hive.script.operator.env.blacklist

hive.security.authorization.sqlstd.confwhitelist

Before upgrade:

```
hive\.auto\..*hive\.cbo\..*hive\.convert\..*hive\.exec\.dynamic\
.partition.*hive\.exec\..*\.dynamic\.partitions\..*hive\.exec\.c
ompress\..*hive\.exec\.infer\..*hive\.exec\.mode.local\..*hive\.
exec\.orc\..*hive\.exec\.parallel.*hive\.explain\..*hive\.fetch.
task\..*hive\.groupby\..*hive\.hbase\..*hive\.index\..*hive\.ind
ex\..*hive\.intermediate\..*hive\.join\..*hive\.limit\..*hive\.l
og\..*hive\.mapjoin\..*hive\.merge\..*hive\.optimize\..*hive\.or
c\..*hive\.outerjoin\..*hive\.parquet\..*hive\.ppd\..*hive\.prew
arm\..*hive\.server2\.proxy\.userhive\.skewjoin\..*hive\.smbjoin
\..*hive\.stats\..*hive\.strict\..*hive\.tez\..*hive\.vectorized
\..*mapred\.map\..*mapred\.reduce\..*mapred\.output\.compression
\.codecmapred\.job\.queuenamemapred\.output\.compression\.typema
```

```
pred\.min\.split\.sizemapreduce\.job\.reduce\.slowstart\.complet
edmapsmapreduce\.job\.queuenamemapreduce\.job\.tagsmapreduce\.in
put\.fileinputformat\.split\.minsizemapreduce\.map\..*mapreduce\
.reduce\..*mapreduce\.output\.fileoutputformat\.compress\.codecm
apreduce\.output\.fileoutputformat\.compress\.typeoozie\..*tez\.
am\..*tez\.task\..*tez\.runtime\..*tez\.queue\.namehive\.transpo
se\.aggr\.joinhive\.exec\.reducers\.bytes\.per\.reducerhive\.cli
ent\.stats\.countershive\.exec\.default\.partition\.namehive\.ex
ec\.drop\.ignorenonexistenthive\.counters\.group\.namehive\.defa
ult\.fileformat\.managedhive\.enforce\.bucketmapjoinhive\.enforc
e\.sortmergebucketmapjoinhive\.cache\.expr\.evaluationhive\.quer
y\.result\.fileformathive\.hashtable\.loadfactorhive\.hashtable\
.initialCapacityhive\.ignore\.mapjoin\.hinthive\.limit\.row\.max
\.sizehive\.mapred\.modehive\.map\.aggrhive\.compute\.query\.usi
ng\.statshive\.exec\.rowoffsethive\.variable\.substitutehive\.va
riable\.substitute\.depthhive\.autogen\.columnalias\.prefix\.inc
ludefuncnamehive\.autogen\.columnalias\.prefix\.labelhive\.exec\
.check\.crossproductshive\.cli\.tez\.session\.asynchive\.compath
ive\.exec\.concatenate\.check\.indexhive\.display\.partition\.co
ls\.separatelyhive\.error\.on\.empty\.partitionhive\.execution\.
enginehive\.exec\.copyfile\.maxsizehive\.exim\.uri\.scheme\.whit
elisthive\.file\.max\.footerhive\.insert\.into\.multilevel\.dirs
hive\.localize\.resource\.num\.wait\.attemptshive\.multi\.insert
\.move\.tasks\.share\.dependencieshive\.support\.quoted\.identif
iershive\.resultset\.use\.unique\.column\.nameshive\.analyze\.st
mt\.collect\.partlevel\.statshive\.exec\.schema\.evolutionhive\.
server2\.logging\.operation\.levelhive\.server2\.thrift\.results
et\.serialize\.in\.taskshive\.support\.special\.characters\.tabl
enamehive\.exec\.job\.debug\.capture\.stacktraceshive\.exec\.job
\.debug\.timeouthive\.llap\.io\.enabledhive\.llap\.io\.use\.file
id\.pathhive\.llap\.daemon\.service\.hostshive\.llap\.execution\
.modehive\.llap\.auto\.allow\.uberhive\.llap\.auto\.enforce\.tre
ehive\.llap\.auto\.enforce\.vectorizedhive\.llap\.auto\.enforce\
.statshive\.llap\.auto\.max\.input\.sizehive\.llap\.auto\.max\.o
utput\.sizehive\.llap\.skip\.compile\.udf\.checkhive\.llap\.clie
nt\.consistent\.splitshive\.llap\.enable\.grace\.join\.in\.llaph
ive\.llap\.allow\.permanent\.fnshive\.exec\.max\.created\.filesh
ive\.exec\.reducers\.maxhive\.reorder\.nway\.joinshive\.output\.
file\.extensionhive\.exec\.show\.job\.failure\.debug\.infohive\.
exec\.tasklog\.debug\.timeouthive\.query\.id
```

After upgrade:

```
hive\.auto\..*hive\.cbo\..*hive\.convert\..*hive\.druid\..*hive\
.exec\.dynamic\.partition.*hive\.exec\.max\.dynamic\.partitions.
*hive\.exec\.compress\..*hive\.exec\.infer\..*hive\.exec\.mode.l
ocal\..*hive\.exec\.orc\..*hive\.exec\.parallel.*hive\.exec\.que
ry\.redactor\..*hive\.explain\..*hive\.fetch.task\..*hive\.group
by\..*hive\.hbase\..*hive\.index\..*hive\.index\..*hive\.interme
diate\..*hive\.jdbc\..*hive\.join\..*hive\.limit\..*hive\.log\..
*hive\.mapjoin\..*hive\.merge\..*hive\.optimize\..*hive\.materia
lizedview\..*hive\.orc\..*hive\.outerjoin\..*hive\.parquet\..*hi
ve\.ppd\..*hive\.prewarm\..*hive\.query\.redaction\..*hive\.serv
er2\.thrift\.resultset\.default\.fetch\.sizehive\.server2\.proxy
\.userhive\.skewjoin\..*hive\.smbjoin\..*hive\.stats\..*hive\.st
rict\..*hive\.tez\..*hive\.vectorized\..*hive\.query\.reexecutio
n\..*reexec\.overlay\..*fs\.defaultFSssl\.client\.truststore\.lo
cationdistcp\.atomicdistcp\.ignore\.failuresdistcp\.preserve\.st
atusdistcp\.preserve\.rawxattrsdistcp\.sync\.foldersdistcp\.dele
te\.missing\.sourcedistcp\.keystore\.resourcedistcp\.liststatus\
.threadsdistcp\.max\.mapsdistcp\.copy\.strategydistcp\.skip\.crc
distcp\.copy\.overwritedistcp\.copy\.appenddistcp\.map\.bandwidt
h\.mbdistcp\.dynamic\..*distcp\.meta\.folderdistcp\.copy\.listin
```

```
g\.classdistcp\.filters\.classdistcp\.options\.skipcrccheckdistc
p\.options\.mdistcp\.options\.numListstatusThreadsdistcp\.option
s\.mapredSslConfdistcp\.options\.bandwidthdistcp\.options\.overw
ritedistcp\.options\.strategydistcp\.options\.idistcp\.options\.
p.*distcp\.options\.updatedistcp\.options\.deletemapred\.map\..*
mapred\.reduce\..*mapred\.output\.compression\.codecmapred\.job\
.queue\.namemapred\.output\.compression\.typemapred\.min\.split\
.sizemapreduce\.job\.reduce\.slowstart\.completedmapsmapreduce\.
job\.queuenamemapreduce\.job\.tagsmapreduce\.input\.fileinputfor
mat\.split\.minsizemapreduce\.map\..*mapreduce\.reduce\..*mapred
uce\.output\.fileoutputformat\.compress\.codecmapreduce\.output\
.fileoutputformat\.compress\.typeoozie\..*tez\.am\..*tez\.task\.
.*tez\.runtime\..*tez\.queue\.namehive\.transpose\.aggr\.joinhiv
e\.exec\.reducers\.bytes\.per\.reducerhive\.client\.stats\.count
ershive\.exec\.default\.partition\.namehive\.exec\.drop\.ignoren
onexistenthive\.counters\.group\.namehive\.default\.fileformat\.
managedhive\.enforce\.bucketmapjoinhive\.enforce\.sortmergebucke
tmapjoinhive\.cache\.expr\.evaluationhive\.query\.result\.filefo
rmathive\.hashtable\.loadfactorhive\.hashtable\.initialCapacityh
ive\.ignore\.mapjoin\.hinthive\.limit\.row\.max\.sizehive\.mapre
d\.modehive\.map\.aggrhive\.compute\.query\.using\.statshive\.ex
ec\.rowoffsethive\.variable\.substitutehive\.variable\.substitut
e\.depthhive\.autogen\.columnalias\.prefix\.includefuncnamehive\
.autogen\.columnalias\.prefix\.labelhive\.exec\.check\.crossprod
uctshive\.cli\.tez\.session\.asynchive\.compathive\.display\.par
tition\.cols\.separatelyhive\.error\.on\.empty\.partitionhive\.e
xecution\.enginehive\.exec\.copyfile\.maxsizehive\.exim\.uri\.sc
heme\.whitelisthive\.file\.max\.footerhive\.insert\.into\.multil
evel\.dirshive\.localize\.resource\.num\.wait\.attemptshive\.mul
ti\.insert\.move\.tasks\.share\.dependencieshive\.query\.results
\.cache\.enabledhive\.query\.results\.cache\.wait\.for\.pending\
.resultshive\.support\.quoted\.identifiershive\.resultset\.use\.
unique\.column\.nameshive\.analyze\.stmt\.collect\.partlevel\.st
atshive\.exec\.schema\.evolutionhive\.server2\.logging\.operatio
n\.levelhive\.server2\.thrift\.resultset\.serialize\.in\.taskshi
ve\.support\.special\.characters\.tablenamehive\.exec\.job\.debu
g\.capture\.stacktraceshive\.exec\.job\.debug\.timeouthive\.llap
\.io\.enabledhive\.llap\.io\.use\.fileid\.pathhive\.llap\.daemon
\.service\.hostshive\.llap\.execution\.modehive\.llap\.auto\.all
ow\.uberhive\.llap\.auto\.enforce\.treehive\.llap\.auto\.enforce
\.vectorizedhive\.llap\.auto\.enforce\.statshive\.llap\.auto\.ma
x\.input\.sizehive\.llap\.auto\.max\.output\.sizehive\.llap\.ski
p\.compile\.udf\.checkhive\.llap\.client\.consistent\.splitshive
\.llap\.enable\.grace\.join\.in\.llaphive\.llap\.allow\.permanen
t\.fnshive\.exec\.max\.created\.fileshive\.exec\.reducers\.maxhi
ve\.reorder\.nway\.joinshive\.output\.file\.extensionhive\.exec\
.show\.job\.failure\.debug\.infohive\.exec\.tasklog\.debug\.time
outhive\.query\.idhive\.query\.tag
```

hive.security.command.whitelist

Before upgrade: set,reset,dfs,add,list,delete,reload,compile

After upgrade: set,reset,dfs,add,list,delete,reload,compile,llap

hive.server2.enable.doAs

Before upgrade: TRUE (in case of an insecure cluster only)

After upgrade: FALSE (in all cases)

Affects only insecure clusters by turning off impersonation. Permission issues are expected to arise for affected clusters.

hive.server2.idle.session.timeout

Before upgrade: 12 hours

After upgrade: 24 hours

Exception:Preserves pre-upgrade value if old default is overridden; otherwise, uses new default.

hive.server2.max.start.attempts

Before upgrade: 30

After upgrade: 5

hive.server2.parallel.ops.in.session

Before upgrade: TRUE

After upgrade: FALSE

A Tez limitation requires disabling this property; otherwise, queries submitted concurrently on a single JDBC connection fail or execute slower.

hive.server2.support.dynamic.service.discovery

Before upgrade: FALSE

After upgrade: TRUE

hive.server2.tez.initialize.default.sessions

Before upgrade: FALSE

After upgrade: TRUE

hive.server2.thrift.max.worker.threads

Before upgrade: 100

After upgrade: 500

Exception: Preserves pre-upgrade value if the old default is overridden; otherwise, uses new default.

hive.server2.thrift.resultset.max.fetch.size

Before upgrade: 1000

After upgrade: 10000

hive.service.metrics.file.location

Before upgrade: /var/log/hive/metrics-hiveserver2/metrics.log

After upgrade: /var/log/hive/metrics-hiveserver2-hiveontez/metrics.log

This location change is due to a service name change.

hive.stats.column.autogather

Before upgrade: FALSE

After upgrade: TRUE

hive.stats.deserialization.factor

Before upgrade: 1

After upgrade: 10

hive.support.special.characters.tablename

Before upgrade: FALSE

After upgrade: TRUE

hive.tez.auto.reducer.parallelism

Before upgrade: FALSE

After upgrade: TRUE

hive.tez.bucket.pruning

Before upgrade: FALSE

After upgrade: TRUE

hive.tez.container.size

Before upgrade: -1

After upgrade: 4096

hive.tez.exec.print.summary

Before upgrade: FALSE

After upgrade: TRUE

hive.txn.manager

Before upgrade: org.apache.hadoop.hive.ql.lockmgr.DummyTxnManager

After upgrade: org.apache.hadoop.hive.ql.lockmgr.DbTxnManager

hive.vectorized.execution.mapjoin.minmax.enabled

Before upgrade: FALSE

After upgrade: TRUE

hive.vectorized.execution.mapjoin.native.fast.hashtable.enabled

Before upgrade: FALSE

After upgrade: TRUE

hive.vectorized.use.row.serde.deserialize

Before upgrade: FALSE

After upgrade: TRUE

Related Information

Changes to CDH Hive Tables

Changes to HDP Hive Tables

Apache Hive Post-Upgrade Tasks

A successful upgrade requires performing a number of procedures that you can follow using step-by-step instructions. Important configuration tasks set up security on your cluster. You learn about semantic changes that might affect your applications, and see how to find your tables or move them. You find out about the Hive Warehouse Connector (HWC) to access files from Spark.

Customizing critical Hive configurations

As Administrator, you need property configuration guidelines. You need to know which properties you need to reconfigure after upgrading. You must understand which the upgrade process carries over from the old cluster to the new cluster.

The upgrade process tries to preserve your Hive configuration property overrides. These overrides are the custom values you set to configure Hive in the old CDH or HDP cluster. The upgrade process does not perserve all overrides. For example, a custom value you set for hive.exec.max.dynamic.partitions.pernode is preserved. In the case of other properties, for example hive.cbo.enable, the upgrade ignores any override and just sets the -recommended value.

The upgrade process does not preserve overrides to the configuration values of the following properties that you likely need to reconfigure to meet your needs:

- hive.conf.hidden.list
- hive.conf.restricted.list

- hive.exec.post.hooks
- hive.script.operator.env.blacklist
- hive.security.authorization.sqlstd.confwhitelist
- hive.security.command.whitelist

The Apache Hive Wiki describes these properties. The values of these properties are lists.

The upgrade process ignores your old list and sets a new generic list. For example, the hive.security.command.whitelist value is a list of security commands you consider trustworthy and want to keep. Any overrides of this list that you set in the old cluster are not preserved. The new default is probably a shorter (more restrictive) list than the original default you were using in the old cluster. You need to customize this to meet your needs.

Check and change each property listed above after upgrading as described in the next topic.

Consider reconfiguring more property values than the six listed above. Even if you did not override the default value in the old cluster, the default might have changed in a way that impacts your work.

Setting Hive Configuration Overrides

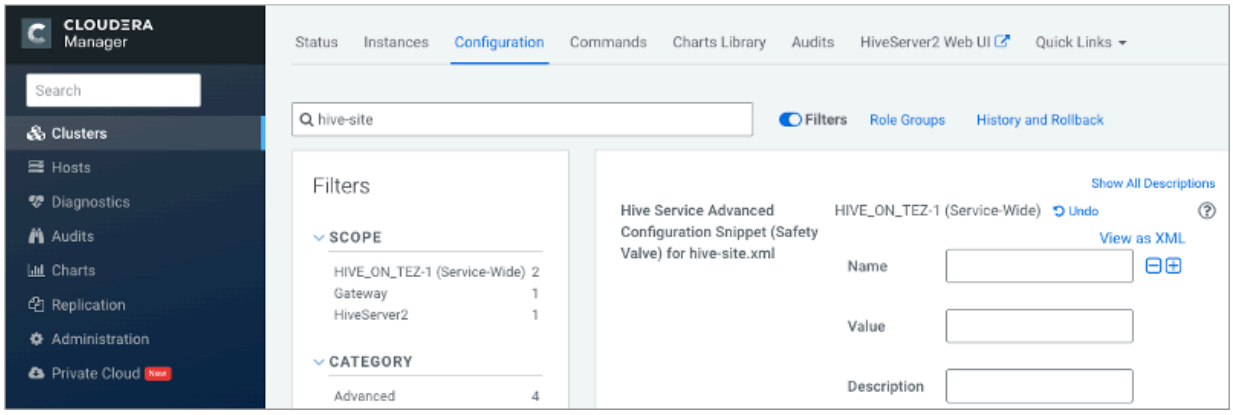
You need to know how to configure the critical customizations that the upgrade process does not preserve from your old Hive cluster. Referring to your records about your old configuration, you follow steps to set at least six critical property values.

About this task

By design, the six critical properties that you need to customize are not visible in , as you can see from the Visible in column of Configurations Requirements and Recommendations. You use the Safety Valve to add these properties to hive-site.xml as shown in this task.

Procedure

1. In Clusters select the Hive on Tez service. Click Configuration, and search for hive-site.xml.
2. In Hive Service Advanced Configuration Snippet (Safety Valve) for hive-site.xml, click +.



3. In Name, add the hive.conf.hidden.list property.
4. In Value, add your custom list.
5. Customize the other critical properties: hive.conf.restricted.list, hive.exec.post.hooks, hive.script.operator.env.blacklist, hive.security.authorization.sqlstd.confwhitelist, hive.security.command.whitelist.

   Use hive.security.authorization.sqlstd.confwhitelist.append, for example, to set up the list.
6. Save the changes and restart the Hive service.
7. Look at the Configurations Requirements and Recommendations to understand which overrides were preserved or not.

Hive Configuration Requirements and Recommendations

You need to set certain Hive and HiveServer (HS2) configuration properties after upgrading. You review recommendations for setting up for your needs, and understand which configurations remain unchanged after upgrading, which impact performance, and default values.

Requirements and Recommendations

The following table includes the Hive service and HiveServer properties that the upgrade process changes. Other property values (not shown) are carried over unchanged from CDH or HDP to

- Set After Upgrade column: properties you need to manually configure after the upgrade to . Pre-existing customized values are not preserved after the upgrade.
- Default Recommended column: properties that the upgrade process changes to a new value that you are strongly advised to use.
- Impacts Performance column: properties changed by the upgrade process that you set to tune performance.
- Safety Value Overrides column: How the upgrade process handles Safety Valve overrides.
  - Disregards: the upgrade process removes any old CDH Safety Valve configuration snippets from the new CDP configuration.
  - Preserves means the upgrade process carries over any old CDH snippets to the new CDP configuration.
  - Not applicable means the value of the old parameter is preserved.
- Visible in CM column: property is visible in Cloudera Manager after upgrading. after upgrading.

  If a property is not visible, and you want to configure it, use the Safety Valve to safely add the parameter to the correct file, for example to a cluster-wide, hive-site.xml file.

Table 1:

Property	Set After Upgrade	Default Recomme	Impacts Performa	New Feature	Safety Valve Overrides	Visible in CM
datanucleus.connectionPool.maxPoolSize			#		Preserve	
datanucleus.connectionPoolingType			#		Disregard	
hive.async.log.enabled					Disregard	#
hive.auto.convert.join.noconditionaltask.size					Not applicable	#
hive.auto.convert.sortmerge.join					Preserve	
hive.auto.convert.sortmerge.join.to.mapjoin					Preserve	
hive.cbo.enable					Disregard	#
hive.cbo.show.warnings					Disregard	
hive.compactor.worker.threads				#	Disregard	#
hive.compute.query.using.stats			#		Disregard	#
hive.conf.hidden.list	#				Disregard	
hive.conf.restricted.list	#				Disregard	
hive.default.fileformat.managed					Disregard	#
hive.default.rcfile.serde		#			Preserve	
hive.driver.parallel.compilation					Disregard	#
hive.exec.dynamic.partition.mode					Disregard	
hive.exec.max.dynamic.partitions					Preserve	
hive.exec.max.dynamic.partitions.pernode					Preserve	
hive.exec.post.hooks	#				Disregard	

Property	Set After Upgrade	Default Recomme	Impacts Performa	New Feature	Safety Valve Overrides	Visible in CM
hive.exec.reducers.max		# or other prime number			Not applicable	#
hive.execution.engine					Disregard	
hive.fetch.task.conversion			#		Not applicable	#
hive.fetch.task.conversion.threshold			#		Not appliable	#
hive.hashtable.key.count.adjustment			#		Preserve	
hive.limit.optimize.enable		#			Disregard	
hive.limit.pushdown.memory.usage			#		Not Applicable	#
hive.mapjoin.hybridgrace.hashtable		#	#		Disregard	
hive.mapred.reduce.tasks.speculative.execution		#			Disregard	
hive.metastore.aggregate.stats.cache.enabled		#	#		Disregard	
hive.metastore.disallow.incompatible.col.type.changes					Disregard	
hive.metastore.dml.events					Disregard	#
hive.metastore.event.message.factory		#			Disregard	
hive.metastore.uri.selection		#			Disregard	
hive.metastore.warehouse.dir					Preserve	#
hive.optimize.metadataonly		#			Disregard	
hive.optimize.point.lookup.min					Disregard	
hive.prewarm.numcontainers					Disregard	
hive.script.operator.env.blacklist	#				Disregard	
hive.security.authorization.sqlstd.confwhitelist	#				Disregard	
hive.security.command.whitelist	#				Disregard	
hive.server2.enable.doAs					Disregard	#
hive.server2.idle.session.timeout					Not applicable	#
hive.server2.max.start.attempts					Preserve	
hive.server2.parallel.ops.in.session					Preserve	
hive.server2.support.dynamic.service.discovery				#	Disregard	#
hive.server2.tez.initialize.default.sessions				#	Disregard	
hive.server2.thrift.max.worker.threads					Not Applicable	#
hive.server2.thrift.resultset.max.fetch.size					Preserve	
hive.service.metrics.file.location					Disregard	#
hive.stats.column.autogather		#			Disregard	
hive.stats.deserialization.factor		#			Disregard	
hive.support.special.characters.tablename		#			Disregard	
hive.tez.auto.reducer.parallelism				#	Disregard	#
hive.tez.bucket.pruning				#	Disregard	#
hive.tez.container.size				#	Disregard	#
hive.tez.exec.print.summary				#	Disregard	#

Property	Set After Upgrade	Default Recomme	Impacts Performa	New Feature	Safety Valve Overrides	Visible in CM
hive.txn.manager				#	Disregard	#
hive.vectorized.execution.mapjoin.minmax.enabled		#			Disregard	
hive.vectorized.execution.mapjoin.native.fast.hashtable.enabled		#			Disregard	
hive.vectorized.use.row.serde.deserialize		#			Disregard	

Configuring HiveServer for ETL using YARN queues

You need to set several configuration properties to allow placement of the Hive workload on the Yarn queue manager, which is common for running an ETL job. You need to set several parameters that effectively disable the reuse of containers. Each new query gets new containers routed to the appropriate queue.

About this task

Hive configuration properties affect mapping users and groups to YARN queues. You set these properties to use with YARN Placement Rules.

To set Hive properties for YARN queues:

Procedure

1. In Cloudera Manager, click Clusters Hive-on-Tez Configuration .
2. Search for the Hive Service Advanced Configuration Snippet (Safety Valve) for hive-site.xml setting.
3. In the Hive Service Advanced Configuration Snippet (Safety Valve) for hive-site.xml setting, click +.
4. In Name enter the property hive.server2.tez.initialize.default.sessions and in value enter false.
5. In Name enter the property hive.server2.tez.queue.access.check and in value enter true.
6. In Name enter the property hive.server2.tez.sessions.custom.queue.allowed and in value enter true.

Removing Hive on Spark Configurations

Your scripts, or queries, include the Hive on Spark configuration, which is no longer supported, and you must know how to recognize and remove these configurations.

In , there is no Hive-Spark dependency. The Spark site and libs are not in the classpath. This execution engine has been replaced by Apache Tez.

Before Upgrade to

CDH supported Hive on Spark and the following configuration to enable Hive on Spark: set hive.execution.engine=spark

After Upgrade to

does not support Hive on Spark. Scripts that enable Hive on Spark do not work.

Action Required

Remove set hive.execution.engine=spark from your scripts.

Configuring authorization to tables

Although the upgrade process makes no change to the location of external tables, you need to set up access to external tables in HDFS. If you choose the recommended Ranger security model for authorization, you need to set up policies and configure Hive metastore (HMS).

About this task

Set up access to external tables in HDFS using one of the following methods.

- Set up a Hive HDFS policy in Ranger (recommended) to include the paths to external table data.

- Put an HDFS ACL in place. Store the external text file, for example a comma-separated values (CSV) file, in HDFS that will serve as the data source for the external table.

If you want to use Ranger to authorize access to your tables, you must configure a few HMS properties for authorization in addition to setting up Ranger policies. If you have not configured HMS, attempting to create a table using Spark SQL, Beeline, or Hue results in the following error:

```
org.apache.hadoop.hive.ql.ddl.DDLTask. MetaException(message:No privilege 'C
reate' found for outputs { database:DATABASE_NAME, table:TABLE_NAME})
```

Setting up access control lists

Several sources of information about setting up HDFS ACLS plus a brief Ranger overview and pointer to Ranger information prepare you to set up Hive authorization.

In , HDFS supports POSIX ACLs (Access Control Lists) to assign permissions to users and groups. In lieu of Ranger policies, you use HDFS ACLs to check and make any necessary changes in HDFS permission changes. For more information, see HDFS ACLs, Apache Software Foundation HDFS Permissions Guide, and HDFS ACL Permissions.

In Ranger, you give multiple groups and users specific permissions based on your use case. You apply permissions to a directory tree instead of dealing with individual files. For more information, see Authorizing Apache Hive Access.

If possible, you should use Ranger policies over HDFS ACLs to control HDFS access. Controlling HDFS access through Ranger provides a single, unified interface for understanding and managing your overall governance framework and policy design. If you need to mimic the legacy Sentry HDFS ACL Sync behavior for Hive and Impala tables, consider using Ranger RMS.

Configure encryption zone security

Under certain conditions, you as Administrator, need to perform a security-related task to allow users to access to tables stored in encryption zones. You find out how to prevent access problems to these tables.

About this task

Hive on Tez cannot run some queries on tables stored in encryption zones under certain conditions. Perform the following procedure only when the cluster uses self-signed certificates.

Important: Skip this task for clusters where TLS certificates are properly signed by a Certificate Authority (CA), and the CA is in the truststore files.

Procedure

1. Copy the ssl-client.xml file to a directory that is available on all hosts.
2. In , click Clusters Hive on Tez Configuration .
3. Search for the Hive Service Advanced Configuration Snippet (Safety Valve) for hive-site.xml setting.
4. In the Hive Service Advanced Configuration Snippet (Safety Valve) for hive-site.xml setting, click + .
5. In Name enter the property tez.aux.uris and in value enter path-to-ssl-client.xml.

   Ensure that you include the file URI scheme in the path. For example:

   ```
   tez.aux.uris=file:///etc/hadoof/conf
   ```

Configure edge nodes as gateways

If you use command-line clients, such as Sqoop, to access Hive, you must configure these gateways to use defaults for your service. You can accomplish this task in a few steps.

About this task

By default, the HS2 instances configured in the migration already have the default beeline-site.xml file defined for the service. Other hosts do not. Configure these hosts as a gateway for that service.

Procedure

1. Find the notes you made before the upgrade about edge nodes and default, connected endpoints.
2. In , configure hosts other than HiveServer (HS2) hosts that you want to be Hive Gateway nodes as gateways for the default beeline-site.xml file for the gateway service.

Spark integration with Hive

You need to know a little about Hive Warehouse Connector (HWC) and how to find more information because to access Hive from Spark, you need to use HWC implicitly or explicitly.

You can use the Hive Warehouse Connector (HWC) to access Hive managed tables from Spark. HWC is specifically designed to access managed ACID v2 Hive tables, and supports writing to tables in Parquet, ORC, Avro, or Textfile formats. HWC is a Spark library/plugin that is launched with the Spark app.

You do not need HWC to read from or write to Hive external tables. Spark uses native Spark to access external tables.

Use the Spark Direct Reader and HWC for ETL jobs. For other jobs, consider using Apache Ranger and the HiveWarehouseConnector library to provide row and column, fine-grained access to the data.

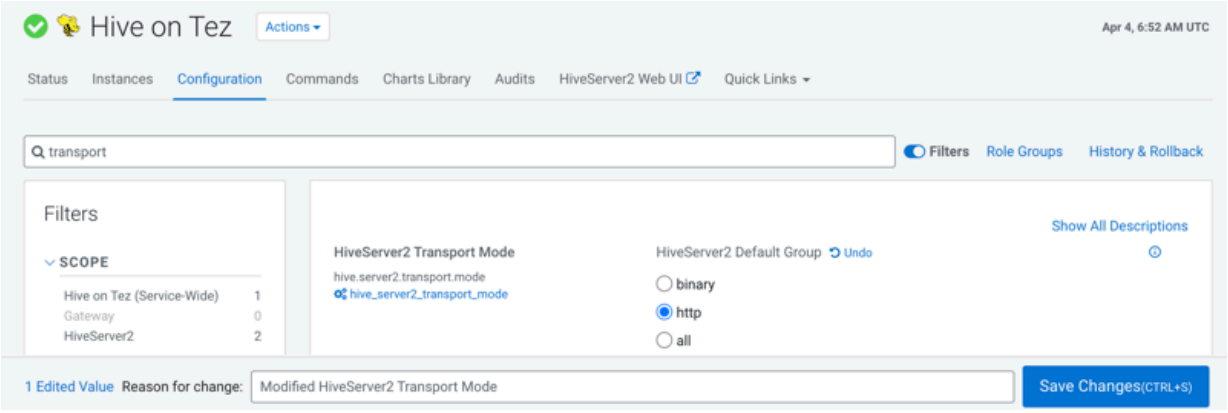
HWC supports spark-submit and pyspark. The spark thrift server is not supported.

Configure HiveServer HTTP mode

If you use Knox, you might need to change the HTTP mode configuration. If you installed Knox on and want to proxy HiveServer with Knox, you need to change the default HiveServer transport mode (hive.server2.transport.mode).

Procedure

1. Click Clusters HIVE_ON_TEZ Configuration
2. In Search, type transport.
3. In HiveServer2 Transport Mode, select http.



4. Save and restart Hive on Tez.

Configuring HMS for high availability

To provide failover to a secondary Hive metastore if your primary instance goes down, you need to know how to add a Metastore role in Cloudera Manager and configure a property.

About this task

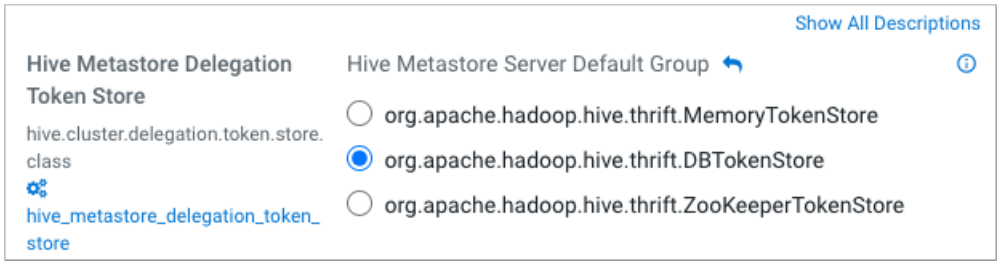
Multiple HMS instances run in active/active mode. No load balancing occurs. An HMS client always reaches the first instance unless it is down. In this case, the client scans the hive.metastore.uris property that lists the HMS instances for a replacement HMS. The second HMS is the designated replacement if hive.metastore.uri.selection is set to SEQUENTIAL (recommended and the default); otherwise, the replacement is selected randomly from the list if hive .metastore.uri.selection is set to RANDOM.

Before you begin

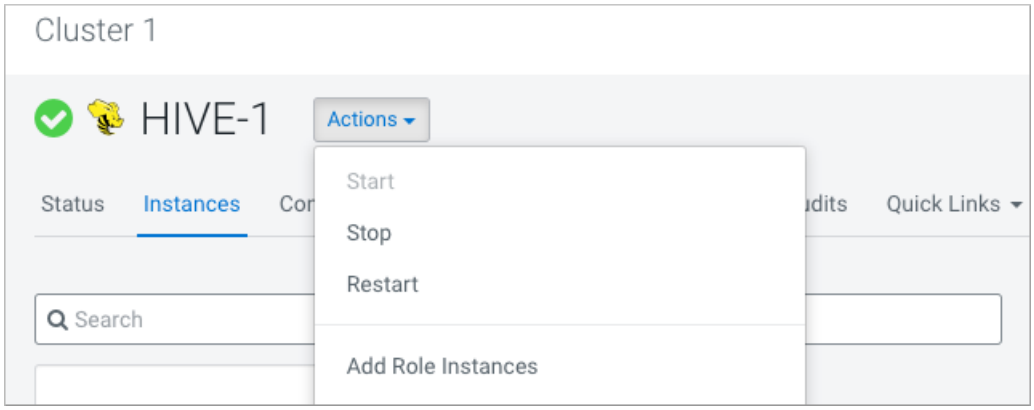
Minimum Required Role: Configurator (also provided by Cluster Administrator, Full Administrator)

Procedure

1. In Cloudera Manager, click Clusters Hive Configuration .
2. Take one of the following actions:
   - If you have a cluster secured by Kerberos, search for Hive Delegation Token Store, which specifies storage for the Kerberos token as described below.
   - If you have an unsecured cluster, skip the next step.
3. Select org.apache.hadoop.hive.thrift.DBTokenStore, and save the change.



   Storage for the Kerberos delegation token is defined by the hive.cluster.delegation.token.store.class property. The available choices are Zookeeper, the Metastore, and memory. Cloudera recommends using the database by setting the org.apache.hadoop.hive.thrift.DBTokenStore property.
4. Click Instances Actions Add Role Instances



5. In Assign Roles, in Metastore Server, click Select Hosts.
6. In Hosts Selected, scroll and select the host that you want to serve as the backup Metastore, and click OK.
7. Click Continue until you exit the wizard.
8. Start the Metastore role on the host from the Actions menu.

   The hive.metastore.uris property is updated automatically.
9. To check or to change the hive.metastore.uri.selection property, go to Clusters Hive Configurations , and search for Hive Service Advanced Configuration Snippet (Safety Valve) for hive-site.xml.
10. Add the property and value (SEQUENTIAL or RANDOM).

Installing Hive on Tez and adding a HiveServer role

services include Hive on Tez and Hive Metastore (HMS). Hive on Tez is a SQL query engine using Apache Tez that performs the HiveServer (HS2) role in a cluster. You must install the Hive service (HMS) before Hive on Tez and HMS in the correct order; otherwise, HiveServer will fail. You must add the HiveServer role additionally to Hive on Tez, not the Hive service; otherwise, HiveServer will fail.

Procedure

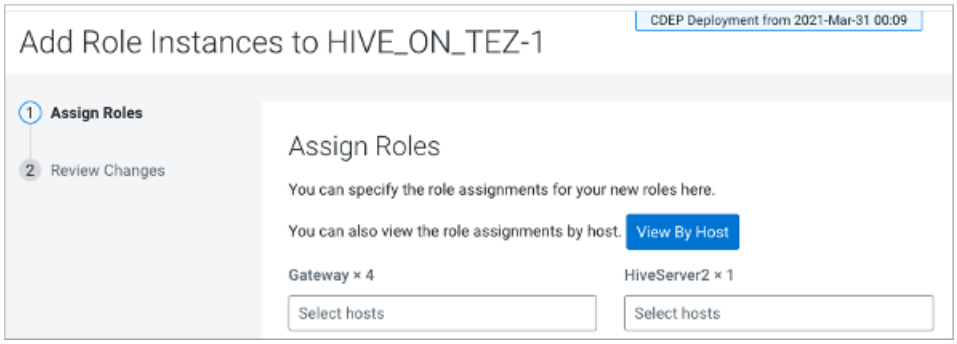
1. Install the HMS, designated Hive service in .
2. Install Hive on Tez
   HiveServer is installed automatically during this process.
3. Accept the default, or change the Hive warehouse location for managed and external tables as described below.

Adding a HiveServer role

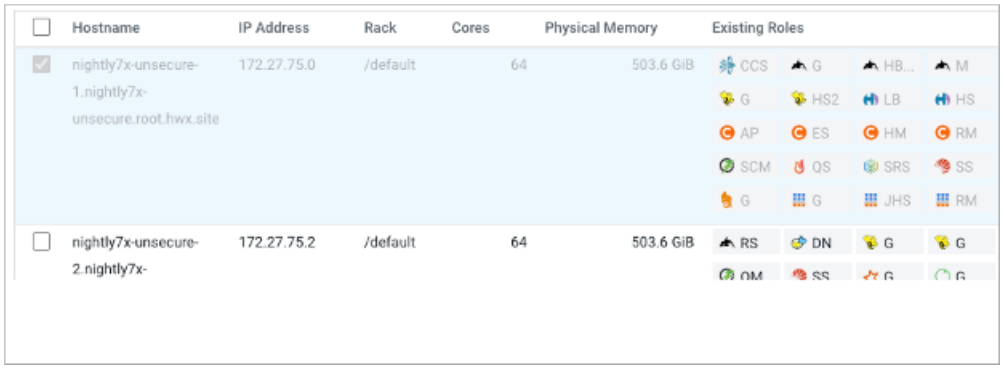
Procedure

1. In Cloudera Manager, click Clusters Hive on Tez .

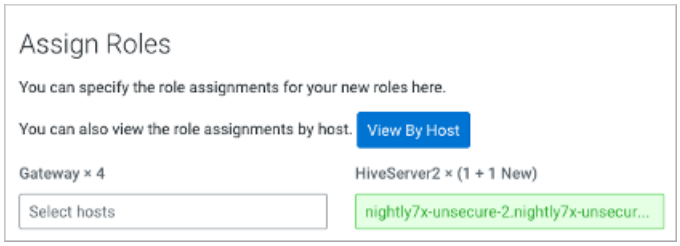
   Do not click Clusters Hive by mistake. This selects the Hive metastore and ultimately results in failure.
2. Click Actions Add Role Instances .



3. Click in the HiveServer2 box to select hosts.

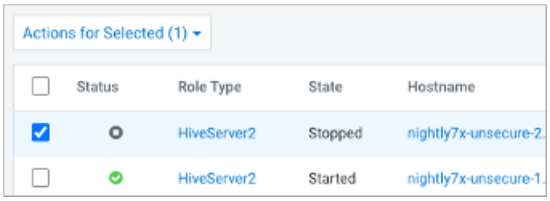


4. In the Host name column, select a host for the HiveServer2 role, and click OK.
   The selected host name you assigned the HiveServer2 role appears under HiveServer2.

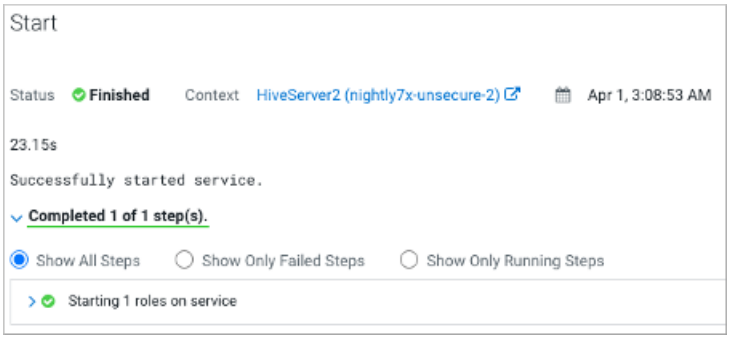


5. Click Continue.

   The new HiveServer2 role state is stopped.
6. Select the new HiveServer2 role.



7. In Actions for Selected, select Start, and then click Start to confirm.
   You see that the service successfully started.



Changing the Hive warehouse location

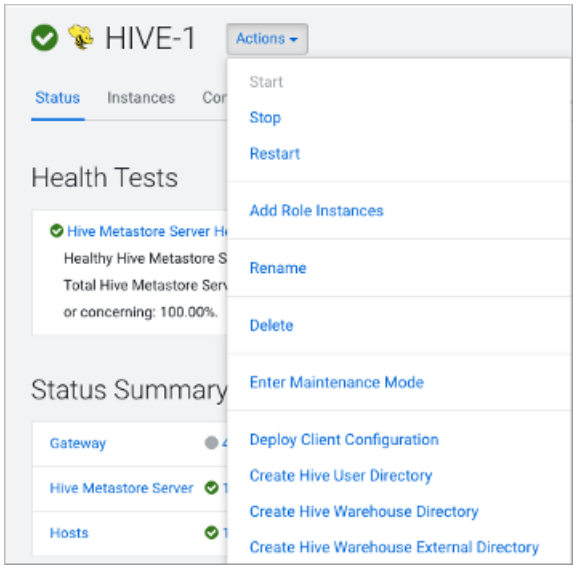
About this task

You use the Hive Metastore Action menu in Cloudera Manager, and navigate to one of the following menu items in the first step below.

- Hive Action Menu Create Hive Warehouse Directory
- Hive Action Menu Create Hive Warehouse External Directory

Procedure

1. Set up directories for the Hive warehouse directory and Hive warehouse external directory from Cloudera Manager Actions.



2. In Cloudera Manager, click Clusters Hive (the Hive Metastore service) Configuration , and change the hive.metastore.warehouse.dir property value to the path you specified for the new Hive warehouse directory.

3. Change the hive.metastore.warehouse.external.dir property value to the path you specified for the Hive warehouse external directory.
4. Configure Ranger policies or set up ACL permissions to access the directories.

Updating Hive and Impala JDBC/ODBC drivers

After upgrading, Cloudera recommends that you update your Hive and Impala JDBC and ODBC drivers. You follow a procedure to download a driver.

Before you begin

Configure authenticated users for running SQL queries through a JDBC or ODBC driver. For example, set up a Ranger policy.

Getting the JDBC driver

You learn how to download the Cloudera Hive and Impala JDBC drivers to give clients outside the cluster access to your SQL engines.

Procedure

1. Download the lastest Hive JDBC driver for CDP from the Hive JDBC driver download page.
2. Go to the Impala JDBC driver page, and download the latest Impala JDBC driver.
3. Follow JDBC driver installation instructions on the download page.

Getting the ODBC driver

You learn how to download the Cloudera ODBC drivers for Hive and Impala.

Procedure

1. Download the latest Hive ODBC driver for CDP from the Cloudera ODBC driver download page.
2. Go to the Impala ODBC driver page, and download the latest Impala ODBC driver.
3. Follow ODBC driver installation instructions on the download page.

Key syntax changes

You need to modify queries affected by changes to Hive syntax after upgrading to . Hive has changed the syntax related to `db.table` references, such as CREATE TABLE `mydb.mytable` … . Other syntax changes involve the LOCATION clause in CREATE TABLE. Hive in supports the enhancement to CREATE TABLE that adds the MANAGEDLOCATION clause.

Handling table reference syntax

For ANSI SQL compliance, Hive 3.x rejects `db.table` in SQL queries as described by the Hive-16907 bug fix. A dot (.) is not allowed in table names. As a Data Engineer, you need to ensure that Hive tables do not contain these references before migrating the tables to , that scripts are changed to comply with the SQL standard references, and that users are aware of the requirement.

About this task

To change queries that use such `db.table` references thereby preventing Hive from interpreting the entire db.table string incorrectly as the table name, you enclose the database name and the table name in backticks as follows:

A dot (.) is not allowed in table names.

Procedure

1. Find a table having the problematic table reference.
   For example, math.students appears in a CREATE TABLE statement.
2. Enclose the database name and the table name in backticks.

```
CREATE TABLE `math`.`students` (name VARCHAR(64), age INT, gpa DECIMAL(3
,2));
```

Add Backticks to Table References

CDP Private Cloud Base includes the Hive-16907 bug fix, which rejects `db.table` in SQL queries. A dot (.) is not allowed in table names. You need to change queries that use such references to prevent Hive from interpreting the entire db.table string as the table name.

Procedure

1. Find a table having the problematic table reference.

```
math.students
```

   appears in a CREATE TABLE statement.
2. Enclose the database name and the table name in backticks.

```
 CREATE TABLE `math`.`students` (name VARCHAR(64), age INT, gpa DECIMAL
(3,2));
```

LOCATION and MANAGEDLOCATION clauses

Before upgrading, your Hive version might have supported using the LOCATION clause in queries to create either managed or external tables or databases for managed and external tables. After upgrading, Hive stores managed and external tables in separate HDFS locations. CREATE TABLE limits the use of the LOCATION clause, and consequently requires a change to your queries. Hive in also supports a new location-related clause.

External table limitation for creating table locations

Hive assigns a default location in the warehouse for external tables—/warehouse/tablespace/external/hive. In , Hive does not allow the LOCATION clause in queries to create a managed table. Using this clause, you can specify a location only when creating external tables. For example:

```
CREATE EXTERNAL TABLE my_external_table (a string, b string)
ROW FORMAT SERDE 'com.mytables.MySerDe'
WITH SERDEPROPERTIES ( "input.regex" = "*.csv")
LOCATION '/warehouse/tablespace/external/hive/marketing';
```

Table MANAGEDLOCATION clause

In , Hive has been enhanced to include a MANAGEDLOCATION clause to specify the location of managed tables as shown in the following syntax:

```
CREATE (DATABASE|SCHEMA) [IF NOT EXISTS] database_name
  [COMMENT database_comment]
  [LOCATION external_table_path]
  [MANAGEDLOCATION managed_table_directory_path]
  [WITH DBPROPERTIES (property_name=property_value, ...)];
```

Hive assigns a default location in the warehouse for managed tables—/warehouse/tablespace/managed/hive. In the MANAGEDLOCATION clause, you specify a top level directory for managed tables when creating a Hive database.

Important: Do not use the LOCATION clause to specify the location of managed tables. This clause is only used to specify the location of external tables. Use the MANAGEDLOCATION clause if you are creating managed tables. You must also ensure that you do not set LOCATION and MANAGEDLOCATION to the same HDFS path.

Use DESCRIBE DATABASE db_name; to view the root location of the database on the filesystem.

Related Information

Create a default directory for managed tables

Key semantic changes and workarounds

As SQL Developer, Analyst, or other Hive user, you need to know potential problems with queries due to semantic changes. Some of the operations that changed were not widely used, so you might not encounter any of the problems associated with the changes.

Over the years, Apache Hive committers enhanced versions of Hive supported in legacy releases of CDH and HDP, with users in mind. Changes were designed to maintain compatibility with Hive applications. Consequently, few syntax changes occurred over the years. A number of semantic changes, described in this section did occur, however. Workarounds are described for these semantic changes.

Casting timestamps

Results of applications that cast numerics to timestamps differ from Hive 2 to Hive 3. Apache Hive changed the behavior of CAST to comply with the SQL Standard, which does not associate a time zone with the TIMESTAMP type.

Before Upgrade to

Casting a numeric type value into a timestamp could be used to produce a result that reflected the time zone of the cluster. For example, 1597217764557 is 2020-08-12 00:36:04 PDT. Running the following query casts the numeric to a timestamp in PDT:

```
> SELECT CAST(1597217764557 AS TIMESTAMP);
| 2020-08-12 00:36:04 |
```

After Upgrade to

Casting a numeric type value into a timestamp produces a result that reflects the UTC instead of the time zone of the cluster. Running the following query casts the numeric to a timestamp in UTC.

```
> SELECT CAST(1597217764557 AS TIMESTAMP);
| 2020-08-12 07:36:04.557  |
```

Action Required

Change applications. Do not cast from a numeral to obtain a local time zone. Built-in functions from_utc_timestamp and to_utc_timestamp can be used to mimic behavior before the upgrade.

Changing incompatible column types

A default configuration change can cause applications that change column types to fail.

Before Upgrade to

In HDP 2.x and CDH 5.x and CDH 6 hive.metastore.disallow.incompatible.col.type.changes is false by default to allow changes to incompatible column types. For example, you can change a STRING column to a column of an incompatible type, such as MAP<STRING, STRING>. No error occurs.

After Upgrade to

In , hive.metastore.disallow.incompatible.col.type.changes is true by default. Hive prevents changes to incompatible column types. Compatible column type changes, such as INT, STRING, BIGINT, are not blocked.

Action Required

Change applications to disallow incompatible column type changes to prevent possible data corruption. Check ALTER TABLE statements and change those that would fail due to incompatible column types.

Understanding CREATE TABLE behavior

Hive table creation has changed significantly since Hive 3 to improve useability and functionality. If you are upgrading from CDH or HDP, you must understand the changes affecting legacy table creation behavior.You must understand the default behavior of the CREATE TABLE statement in .

Hive has changed table creation in the following ways:

- Creates ACID-compliant table, which is the default in
- Supports simple writes and inserts
- Writes to multiple partitions
- Inserts multiple data updates in a single SELECT statement
- Eliminates the need for bucketing.

If you have an ETL pipeline that creates tables in Hive, the tables will be created as ACID. Hive now tightly controls access and performs compaction periodically on the tables. Using ACID-compliant, transactional tables causes no performance or operational overload. The way you access managed Hive tables from Spark and other clients changes. In , access to external tables requires you to set up security access permissions.

In , by default, the CREATE TABLE statement creates a full ACID transactional table in ORC format or insert-only transactional tables for all other table formats. You can choose to modify the default behavior by configuring certain properties. The order of preference for configuration is as follows:

You must understand the behavior of the CREATE TABLE statement in legacy platforms like CDH or HDP and how the behavior changes after you upgrade to .

Before upgrading to CDP Private Cloud Base

In CDH 5, CDH 6, and HDP 2, by default CREATE TABLE creates a non-ACID managed table in plain text format.

In HDP 3 and CDP 7.1.0 through 7.1.7.x, by default CREATE TABLE creates either a full ACID transactional table in ORC format or insert-only ACID transactional tables for all other table formats.

After upgrading to CDP Private Cloud Base

- If you are upgrading from HDP 2, CDH 5, or CDH 6 to CDP 7.1.0 through CDP 7.1.8, by default CREATE TABLE creates a full ACID transactional table in ORC format or insert-only ACID transactional tables for all other table formats.
- If you are upgrading from HDP 3 or CDP 7.1.0 through 7.1.7.x to CDP 7.1.8, the existing behavior persists and CREATE TABLE creates either a full ACID transactional table in ORC format or insert-only ACID transactional tables for all other table formats.

Now that you understand the behavior of the CREATE TABLE statement, you can choose to modify the default table behavior by configuring certain properties. The order of preference for configuration is as follows:

Modify the default CREATE TABLE behavior

Override default behavior when creating the table

Irrespective of the database, session, or site-level settings, you can override the default table behavior by using the MANAGED or EXTERNAL keyword in the CREATE TABLE statement.

```
CREATE [MANAGED][EXTERNAL] TABLE foo (id INT);
```

Set the default table type at a database level

You can use the database property, defaultTableType=EXTERNAL or ACID to specify the default table type to be created using the CREATE TABLE statement. You can specify this property when creating the database or at a later point using the ALTER DATABASE statement. For example:

```
CREATE DATABASE test_db WITH DBPROPERTIES ('defaultTableType'='E
XTERNAL');
```

In this example, tables created under the test_db database using the CREATE TABLE statement creates external tables with the purge fucntionality enabled (external.table.purge = 'true').

You can also choose to configure a database to allow only external tables to be created and prevent creation of ACID tables. While creating a database, you can set the database property, EXTER NAL_TABLES_ONLY=true to ensure that only external tables are created in the database. For example:

```
CREATE DATABASE test_db WITH DBPROPERTIES ('EXTERNAL_TABLES_ONLY
'='true');
```

Set the default table type at a session level

You can configure the CREATE TABLE behavior within an existing beeline session by setting hive.create.as.external.legacy to true or false. Setting the value to true results in configuring the CREATE TABLE statement to create external tables by default.

When the session ends, the default CREATE TABLE behavior also ends.

Set the default table type at a site level

You can configure the CREATE TABLE behavior at the site level by configuring the hive.create.as.insert.only and hive.create.as.acid properties in under Hive configuration. When configured at the site level, the behavior persists from session to session. For more information, see Configuring CREATE TABLE behavior.

If you are a Spark user, switching to legacy behavior is unnecessary. Calling ‘create table’ from SparkSQL, for example, creates an external table after upgrading to as it did before the upgrade. You can connect to Hive using the Hive Warehouse Connector (HWC) to read Hive ACID tables from Spark. To write ACID tables to Hive from Spark, you use the HWC and HWC API. Spark creates an external table with the purge property when you do not use the HWC API. For more information, see Hive Warehouse Connector for accessing Spark data.

Disabling Partition Type Checking

An enhancement in Hive 3 checks the types of partitions. This feature can be disabled by setting a property. For more information, see the ASF Apache Hive Language Manual.

Before Upgrade to

In CDH 5.x, partition values are not type checked.

After Upgrade to

Partition values specified in the partition specification are type checked, converted, and normalized to conform to their column types if the property hive.typecheck.on.insert is set to true (default). The values can be numbers.

Action Required

If type checking of partitions causes problems, disable the feature. To disable partition type checking, set hive.typ echeck.on.insert to false. For example:

```
SET hive.typecheck.on.insert=false;
```

Dropping partitions

The OFFLINE and NO_DROP keywords in the CASCADE clause for dropping partitions causes performance problems and is no longer supported.

Before Upgrade to CDP Private Cloud Base

You could use OFFLINE and NO_DROP keywords in the DROP CASCADE clause to prevent partitions from being read or dropped.

After Upgrade to CDP Private Cloud Base

OFFLINE and NO_DROP are not supported in the DROP CASCADE clause.

Action Required

Change applications to remove OFFLINE and NO_DROP from the DROP CASCADE clause. Use an authorization scheme, such as Ranger, to prevent partitions from being dropped or read.

Handling output of greatest and least functions

To calculate the greatest (or least) value in a column, you need to work around a problem that occurs when the column has a NULL value.

Before Upgrade to

The greatest function returned the highest value of the list of values. The least function returned the lowest value of the list of values.

After Upgrade to

Returns NULL when one or more arguments are NULL.

Action Required

Use NULL filters or the nvl function on the columns you use as arguments to the greatest or least functions.

```
SELECT greatest(nvl(col1,default value incase of NULL),nvl(col2,default valu
e incase of NULL));
```

Renaming tables

To harden the system, Hive data can be stored in HDFS encryption zones. RENAME has been changed to prevent moving a table outside the same encryption zone or into a no-encryption zone.

Before Upgrade to

In CDH and HDP, renaming a managed table moves its HDFS location.

After Upgrade to

Renaming a managed table moves its location only if the table is created without a LOCATION clause and is under its database directory.

Action Required

None

Hive unsupported interfaces and features

You need to understand the interfaces that are not supported.

Unsupported Interfaces and features

The following interfaces are not supported in :

- Druid

- Hcat CLI (however HCatalog is supported)
- Hive CLI (replaced by Beeline)
- Hive View UI feature in Ambari
- Apache Hive Standalone driver
- Renaming Hive databases
- Multiple insert overwrite queries that read data from a source table.
- LLAP
- MapReduce execution engine (replaced by Tez)
- Pig
- S3 for storing tables (available in only)
- Spark execution engine (replaced by Tez)
- Spark thrift server

  Spark and Hive tables interoperate using the Hive Warehouse Connector.
- SQL Standard Authorization
- Storage Based Authorization
- Tez View UI feature in Ambari
- WebHCat

You can use Hue in lieu of Hive View.

Storage Based Authorization

Storage Based Authorization (SBA) is no longer supported in . Ranger integration with Hive metastore provides consistency in Ranger authorization enabled in HiveServer (HS2). SBA did not provide authorization support for metadata that does not have a file/directory associated with it. Ranger-based authorization has no such limitation.

Hive-Kudu integration

does not support the integration of HiveServer (HS2) with Kudu tables. You cannot run queries against Kudu tables from HS2.

Partially unsupported interfaces

Apache Hadoop Distributed Copy (DistCP) is not supported for copying Hive ACID tables.

Unsupported Features

does not support the following features that were available in HDP and CDH platforms:

- CREATE TABLE that specifies a managed table location

  Do not use the LOCATION clause to create a managed table. Hive assigns a default location in the warehouse to managed tables. That default location is configured in Hive using the hive.metastore.warehouse.dir configuration property, but can be overridden for the database by setting the CREATE DATABASE MANAGEDLOCATION parameter.
- CREATE INDEX and related index commands were removed in Hive 3, and consequently are not supported in .

  In , you use the Hive 3 default ORC columnar file formats to achieve the performance benefits of indexing. Materialized Views with automatic query rewriting also improves performance. Indexes migrated to are preserved but render any Hive tables with an undroppable index. To drop the index, google the Known Issue for CDPD-23041.
- Hive metastore (HMS) high availablility (HA) load balancing in CDH

  You need to set up HMS HA as described in the documentation.
- Local or Embedded Hive metastore server

  does not support the use of a local or embedded Hive metastore setup.

Unsupported Connector Use

does not support the Sqoop exports using the Hadoop jar command (the Java API) that Teradata documents. For more information, see Migrating data using Sqoop.

Related Information

Configuring HMS for high availability

Changes to CDH Hive Tables

As a Data Scientist, Architect, Analyst, or other Hive user you need to locate and use your Apache Hive 3 tables after an upgrade. You also need to understand the changes that occur during the upgrade process. The location of existing tables after a CDH to upgrade does not change. Upgrading CDH to CDP Private Cloud Base converts Hive managed tables to external tables in Hive 3.

About this task

When the upgrade process converts a managed table to external, it sets the table property external.table.purge to true. The table is equivalent to a managed table having purge set to true in your old CDH cluster.

Managed tables on the HDFS in /user/hive/warehouse before the upgrade remain there after the conversion to external. Tables that were external before the upgrade are not relocated. You need to set HDFS policies to access external tables in Ranger, or set up HDFS ACLs.

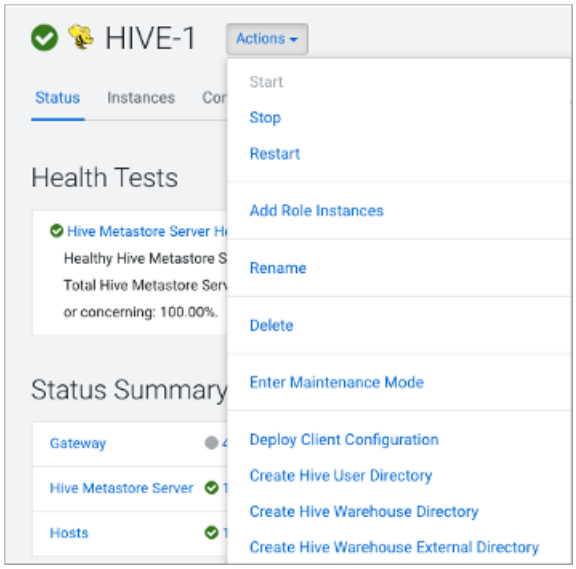
The upgrade process sets the hive.metastore.warehouse.dir property to /warehouse/tablespace/managed/hive, designating it the Hive warehouse location for managed tables. New managed tables that you create in are stored in the Hive warehouse. New external tables are stored in the Hive external warehouse /warehouse/tablespace/external/hive.

To change the location of the Hive warehouses, you navigate to one of the following menu items in the first step below.

- Hive Action Menu Create Hive Warehouse Directory
- Hive Action Menu Create Hive Warehouse External Directory

Procedure

1. Set up directories for the Hive warehouse directory and Hive warehouse external directory from Actions.



2. In , click Clusters Hive (the Hive Metastore service) Configuration , and change the hive.metastore.warehouse.dir property value to the path you specified for the new Hive warehouse directory.

3. Change the hive.metastore.warehouse.external.dir property value to the path you specified for the Hive warehouse external directory.
4. Configure Ranger policies or set up ACL permissions to access the directories.

Changes to HDP Hive tables

As a Data Scientist, Architect, Analyst, or other Hive user you need to locate and use your Apache Hive 3 tables after an upgrade. You also need to understand the changes that occur during the upgrade process.

Managed, ACID tables that are not owned by the hive user remain managed tables after the upgrade, but hive becomes the owner.

After the upgrade, the format of a Hive table is the same as before the upgrade. For example, native or non-native tables remain native or non-native, respectively.

After the upgrade, the location of managed tables or partitions do not change under any one of the following conditions:

- The old table or partition directory was not in its default location /apps/hive/warehouse before the upgrade.
- The old table or partition is in a different file system than the new warehouse directory.
- The old table or partition directory is in a different encryption zone than the new warehouse directory.

Otherwise, the upgrade process from HDP to CDP Private Cloud Base moves managed files to the Hive warehouse / warehouse/tablespace/managed/hive. The upgrade process carries the external files over to CDP Private Cloud Base with no change in location. By default, Hive places any new external tables you create in /warehouse/tablespace/external/hive. The upgrade process sets the hive.metastore.warehouse.dir property to this location, designating it the Hive warehouse location.

Changes to table references using dot notation

Upgrading to includes the Hive-16907 bug fix, which rejects `db.table` in SQL queries. The dot (.) is not allowed in table names. To reference the database and table in a table name, both must be enclosed in backticks as follows: `db`.`table`.

Changes to ACID properties

Hive 3.x in supports transactional and non-transactional tables. Transactional tables have atomic, consistent, isolation, and durable (ACID) properties. In Hive 2.x, the initial version of ACID transaction processing was ACID v1. In Hive 3.x, the mature version of ACID is ACID v2, which is the default table type in CDP Private Cloud Base.

Native and non-native storage formats

Storage formats are a factor in upgrade changes to table types. Hive 2.x and 3.x support the following native and non-native storage formats:

- Native: Tables with built-in support in Hive, such as those in the following file formats:
  - Text
  - Sequence File
  - RC File
  - AVRO File
  - ORC File
  - Parquet File
- Non-native: Tables that use a storage handler, such as the DruidStorageHandler or HBaseStorageHandler

CDP Private Cloud Base upgrade changes to HDP table types

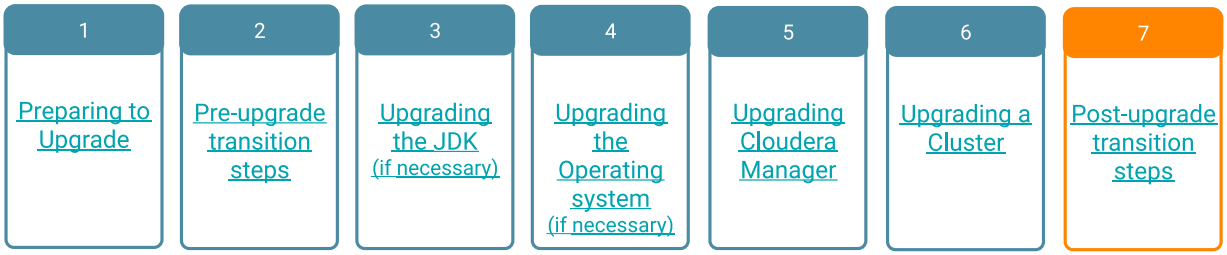
The following table compares Hive table types and ACID operations before an upgrade from HDP 2.x and after an upgrade to . The ownership of the Hive table file is a factor in determining table types and ACID operations after the upgrade.

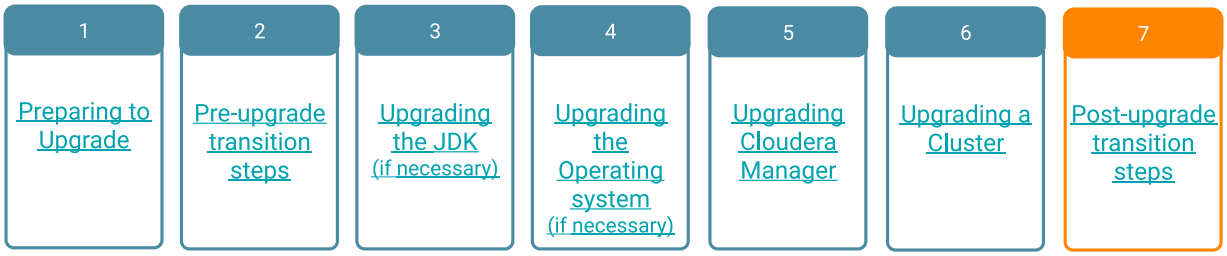
Table 2: HDP 2.x and Table Type Comparison

HDP 2.x				CDP	
Table Type	ACID v1	Format	Owner (user) of Hive Table File	Table Type	ACID v2
External	No	Native or non-native	hive or non-hive	External	No
Managed	Yes	ORC	hive or non-hive	Managed, updatable	Yes
Managed	No	ORC	hive	Managed, updatable	Yes
			non-hive	External, with data delete	No
Managed	No	Native (but non-ORC)	hive	Managed, insert only	Yes
			non-hive	External, with data delete	No
Managed	No	Non-native	hive or non-hive	External, with data delete	No

Apache Hadoop YARN default value changes

A list of default value changes when upgrading from CDH to CDP.





Note: This is not a complete list. It only contains the default value changes that cause behaviour changes.

- Scheduler: In CDP, Capacity Scheduler is the supported and default scheduler. For more information about scheduler migration and post-upgrade fine tuning, see Manual configuration of scheduler properties.
- YARN and MapReduce daemons: YARN daemons (ResourceManager, NodeManager) and the JobHistory Server runs with unix group hadoop instead of yarn:yarn and mapred:mapred.

- Cross-Origin Resource Sharing: CORS is enabled for every role by default.
- YARN admin commands: By default YARN admin commands can be run only as yarn. In 7.1.7 and higher a placeholder value ${yarn_user} is also supported. In such cases Cloudera Manager replace the placeholder value with the collected principal name.
- ResourceManager recovery: ResourceManager recovery is enabled by default.
- Filter entity list by use (filter-entity-list-by-user): Enabled by default, meaning that users can see only those applications on the UI which they have access to.
- Log aggregation: IFile is the default file controller.
- MapReduce shuffle connection keep-alive (mapreduce.shuffle.connection-keep-alive.enabled):: Set to true by default because Auto TLS requires it.
- YARN Admin ACL: If the yarn.admin.acl property is not configured before the upgrade, its default value is changed from * to yarn. In 7.1.7 and higher a placeholder value ${yarn_user} is also supported. In such cases Cloudera Manager replace the placeholder value with the collected principal name.