HMS-Mirror

"hms-mirror" is a utility used to bridge the gap between two clusters and migrate `hive` *metadata* **AND** *data*.

Get pdf version of this README.

Setup

Obtaining Pre-Built Binary

Ask your account rep or check the PS Asset Repo for links.

HMS-Mirror Setup from Binary Distribution

On the edgenode:

- Expand the tarball `tar zxvf hms-mirror-dist.tar.gz` .

  > This produces a child `hms-mirror` directory.

- As the root user (or `sudo`), run `hms-mirror/setup.sh` .

Optimizations

Moving metadata and data between two clusters is a pretty straight forward process, but depends entirely on the proper configurations in each cluster. Listed here are a few tips on some important configurations.

Make Backups before running `hms-mirror`

Take snapshots of areas you'll touch:

- The HMS database on the LEFT and RIGHT clusters
- A snapshot of the HDFS directories on BOTH the LEFT and RIGHT clusters that will be used/touched.

Isolate Migration Activities

The migration of schemas can put a heavy load on HS2 and the HMS server it's using. That impact can manifest itself as 'pauses' for other clients trying to run queries. Long schema/discovery operations have a 'blocking' tendency in HS2.

To prevent normal user operational impact, I suggest establishing an isolated HMS and HS2 environment for the migration process.
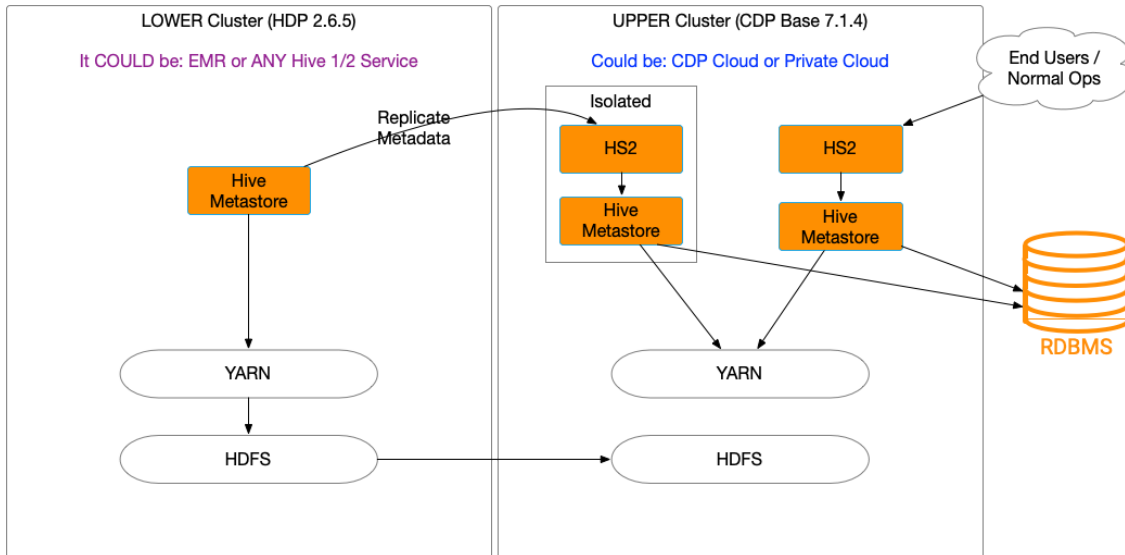
Speed up CREATE/ALTER Table Statements - with existing data

Set `ranger.plugin.hive.urlauth.filesystem.schemes=file` in the Hive Server 2(hive_on_tez) Ranger Plugin Safety Value, via Cloudera Manager.

Add this to the HS2 instance on the RIGHT cluster, when Ranger is used for Auth. This skips the check done against every directory at the table location (for CREATE or ALTER LOCATION). Allowing the process of CREATE/ALTER to run much faster.

The default (true) behavior works well for interactive use case, but bulk operations like this can take a really long time if this validation needs to happen for every new partition during creation or discovery.

I recommend turning this back after the migration is complete. This setting exposes permissions issues at the time of CREATE/ALTER. So by skipping this, future access issues may arise if the permissions aren't aligned, which isn't a Ranger/Hive issue, it's a permissions issue.

Turn ON HMS partition discovery

In CDP 7.1.4 and below, the house keeping threads in HMS used to discover partitions is NOT running. Add `metastore.housekeeping.threads.on=true` to the HMS Safety Value to activate the partition discovery thread. Once this has been set, the following parameters can be used to modify the default behavior.

```
metastore.partition.management.task.frequency
hive.exec.input.listing.max.threads
hive.load.dynamic.partitions.thread
hive.metastore.fshandler.threads
```

Source Reference

```
    METASTORE_HOUSEKEEPING_LEADER_HOSTNAME("metastore.housekeeping.leader.hostname",
            "hive.metastore.housekeeping.leader.hostname", "",
"If there are multiple Thrift metastore services running, the hostname of Thrift
metastore " +
        "service to run housekeeping tasks at. By default this values is empty, which
" +
        "means that the current metastore will run the housekeeping tasks. If
configuration" +
        "metastore.thrift.bind.host is set on the intended leader metastore, this
value should " +
        "match that configuration. Otherwise it should be same as the hostname
returned by " +
        "InetAddress#getLocalHost#getHostName(). Given the uncertainty in the later "
+
        "it is desirable to configure metastore.thrift.bind.host on the intended
leader HMS."),
    METASTORE_HOUSEKEEPING_THREADS_ON("metastore.housekeeping.threads.on",
        "hive.metastore.housekeeping.threads.on", false,
        "Whether to run the tasks under metastore.task.threads.remote on this
metastore instance or not.\n" +
            "Set this to true on one instance of the Thrift metastore service as part
of turning\n" +
            "on Hive transactions. For a complete list of parameters required for
turning on\n" +
            "transactions, see hive.txn.manager."),
```

The default batch size for partition discovery via `msck` is 3000. Adjustments to this can be made via the `hive.msck.repair.batch.size` property in HS2.

Pre-Requisites

Backups

DO NOT SKIP THIS!!!

The `hms-mirror` process DOES 'DROP' tables when asked to. If those tables *manage* data like a *legacy* managed, *ACID*, or *external.table.purge=true* scenario, we do our best to NOT DROP those and ERROR out. But, protect yourself and make backups of the areas you'll be working in.

HDFS Snapshots

Use HDFS Snapshots to make a quick backup of directories you'll be working on. Do this especially in the *LEFT* cluster. We only drop tables, so a snapshot of the database directory is good. BUT, if you are manually doing any `DROP DATABASE <x> CASCADE` operations, that will delete the snapshotted directory (and the snapshot). In this case, create the *snapshot* one level above the database directory.

Metastore Backups

Take a DB backup of your metastore and keep it in a safe place before starting.

Shared Authentication

The clusters must share a common authentication model to support cross cluster HDFS access, when HDFS is the underlying storage layer for the datasets. This means that a **kerberos** ticket used in the RIGHT cluster must be valid for the LEFT cluster.

For cloud storage, the two clusters must have rights to the target storage bucket.

It you are able to [distcp](#) [between the clusters](#), then you have the basic connectivity required to start working with `hms-mirror` .

Linking Clusters Storage Layers

For the `hms-mirror` process to work, it relies on the RIGHT clusters ability to *SEE* and *ACCESS* data in the LEFT clusters HDFS namespace. This is the same access/configuration required to support DISTCP.

Access is required by the RIGHT cluster HCFS namespace to the LEFT clusters HCFS namespace. RIGHT clusters with a greater HDFS version support **LIMITED** functionality for data access in the LEFT cluster.

NOTE: This isn't designed to be a permanent solution and should only be used for testing and migration purposes.

Goal

What does it take to support HDFS visibility between these two clusters?

Can that integration be used to support the Higher Clusters use of the Lower Clusters HDFS Layer for distcp AND Hive External Table support?

Scenario #1

HDP 2.6.5 (Hadoop 2.7.x)

Kerberized - sharing same KDC as CDP Base Cluster

Configuration Changes

The *namenode kerberos* principal MUST be changed from `nn` to `hdfs` to match the namenode principal of the CDP cluster.

Note: You may need to add/adjust the `auth_to_local` settings to match this change.

If this isn't done, `spark-shell` and `spark-submit` will fail to initialize. When changing this in Ambari on HDP, you will need to *reset* the HDFS zkfc `ha` zNode in Zookeeper and reinitialize the hdfs `zkfc` .

From a Zookeeper Client: `/usr/hdp/current/zookeeper-client/bin/zkCli.sh -server localhost`

```
rmr /hadoop-ha
```

Initialize zkfc

```
hdfs zkfc -formatZK
```

hdfs-site.xml

```
hadoop.rpc.protection=true
```

core-site.xml

```
dfs.encrypt.data.transfer=true
dfs.encrypt.data.transfer.algorithm=3des
dfs.encrypt.data.transfer.cipher.key.bitlength=256
```

CDP 7.1.4 (Hadoop 3.1.x)

Kerberized, TLS Enabled

Configuration Changes

Requirements that allow this (upper) cluster to negotiate and communicate with the lower environment.

Cluster Wide hdfs-site.xml Safety Value

```
ipc.client.fallback-to-simple-auth-allowed=true
```

HDFS Service Advanced Config hdfs-site.xml

```
# For this Clusters Name Service
dfs.internal.nameservices=HOME90

# For the target (lower) environment HA NN Services
dfs.ha.namenodes.HDP50=nn1,nn2
dfs.namenode.rpc-address.HDP50.nn1=k01.streever.local:8020
dfs.namenode.rpc-address.HDP50.nn2=k02.streever.local:8020
dfs.namenode.http-address.HDP50.nn1=k01.streever.local:50070
dfs.namenode.http-address.HDP50.nn2=k02.streever.local:50070
dfs.namenode.https
 address.HDP50.nn1=k01.streever.local:50471
dfs.namenode.https-address.HDP50.nn2=k02.streever.local:50470
dfs.client.failover.proxy.provider.HDP50=org.apache.hadoop.hdfs.server.namenode.ha.Confi


# For Available Name Services
dfs.nameservices=HOME90,HDP50
```

Running `distcp` from the RIGHT Cluster

NOTE: Running `distcp` from the **LEFT** cluster isn't supported since the `hcfs client` is not forward compatible.

Copy 'from' Lower Cluster

```
hadoop distcp hdfs://HDP50/user/dstreev/sstats/queues/2020-10.txt /user/dstreev/temp
```

Copy 'to' Lower Cluster

```
hadoop distcp /warehouse/tablespace/external/hive/cvs_hathi_workload.db/queue/2020-
10.txt hdfs://HDP50/user/dstreev/temp
```

Sourcing Data from Lower Cluster to Support Upper Cluster External Tables

Proxy Permissions

The lower cluster must allow the upper clusters *Hive Server 2* host as a 'hive' proxy. The setting in the lower clusters *custom* `core-site.xml` may limit this to that clusters (lower) HS2 hosts. Open it up to include the upper clusters HS2 host.

Custom core-site.xml in Lower Cluster

```
hadoop.proxyuser.hive.hosts=*
```

Credentials from the 'upper' cluster will be projected down to the 'lower' cluster. This means the `hive` user in the upper cluster, when running with 'non-impersonation' will require access to the datasets in the lower cluster HDFS.

For table creation in the 'upper' clusters Metastore, a permissions check will be done on the lower environments directory for the submitting user. So, both the service user AND `hive` will require access to the directory location specified in the lower cluster.

When the two clusters *share* accounts and the same accounts are used between environment for users and service accounts, then access should be simple.

When a different set of accounts are used, the 'principal' from the upper clusters service account for 'hive' and the 'user' principal will be used in the lower cluster. Which means additional HDFS policies in the lower cluster may be required to support this cross environment work.

Permissions

In both the METADATA and STORAGE phases of `hms-mirror` the RIGHT cluster will reach down into the LEFT clusters storage layer to either *use* or *copy* the data.

`hms-mirror` access each cluster via JDBC and use the RIGHT cluster for *storage* layer access.

When the RIGHT cluster is using 'non-impersonation' (hive `doas=false`), the *hive* service account on the **RIGHT** cluster (usually `hive`) needs access to the storage layer on the **LEFT** cluster in order to use this data to support sidecar testing, where we use the data of the LEFT cluster but *mirror* the metadata.

> *Having Ranger on both clusters helps considerably because you can create additional ACL's to provide the access required.*

OR

> *Checked permissions of '': Found that the '' user was NOT the owner of the files in these directories. The user running the process needs to be in 'dfs.permissions.superusergroup' for the LEFT clusters 'hdfs' service. Ambari 2.6 has issues setting this property:* https://jira.cloudera.com/browse/EAR-7805

> *Follow workaround above or add user to the 'hdfs' group. I had to use '/var/lib/ambari-server/resources/scripts/configs.py' to set it manually for Ambari.*

```
sudo ./configs.py --host=k01.streever.local --port=8080 -u admin -p admin -n hdp50 -
c hdfs-site -a set -k dfs.permissions.superusergroup -v hdfs_admin
```

Configuration

The configuration is done via a 'yaml' file, details below.

There are two ways to get started:

- The first time you run `hms-mirror` and it can't find a configuration, it will walk you through building one and save it to `$HOME/.hms-mirror/cfg/default.yaml`
- Use the [template yaml](#) for reference and create a `default.yaml` in the running users `$HOME/.hms-mirror/cfg` directory.

You'll need jdbc driver jar files that are *specific* to the clusters you'll integrate with. If the **LEFT** cluster isn't the same version as the **RIGHT** cluster, don't use the same jdbc jar file especially when integrating Hive 1 and Hive 3 services. The Hive 3 driver is NOT backwardly compatible with Hive 1.

See the [running](#) section for examples on running `hms-mirror` for various environment types and connections.

Tips for Running `hms-miror`

Run in `screen` or `tmux`

This process can be a long running process. It depends on how much you've asked it to do. Having the application terminated because the `ssh` session to the edgenode timed out and you're computer went to sleep will be very disruptive.

Using either of these session state tools (or another of your choice) while running it on an edgenode will allow you to sign-off without disrupting the process AND reattach to see the interactive progress at a later point.

Use `dryrun` FIRST

Before you go and run a process the will make changes, try running `hms-mirror` with the `dryrun` option first. The report generated at the end of the job will provide insight into what issues (if any) you'll run across.

For example: When running the either the METADATA or STORAGE phase and setting `overwriteTable:` `"true"` in the configuration, the process will `ERROR` if the table exists in the RIGHT cluster AND it manages the data there. `hms-mirror` will not `DROP` tables that manage data, you'll need to do that manually. But the processes leading upto this check can be time consuming and a failure late in the process is annoying.

Start Small

Use `-db` (database) AND `-tf` (table filter) options to limit the scope of what your processing. Start with a test database that contains various table types you'd like to migrate.

Review the output report for details/problems you encountered in processing.

Since the process expects the user to have adequate permissions in both clusters, you may find you have some prework to do before continuing.

RETRY (WIP-NOT FULLY IMPLEMENTED YET)

When you run `hms-mirror` we write a `retry` file out to the `$HOME/.hms-mirror/retry` directory with the same filename prefix as the config used to run the job. If you didn't specify a config then its using the `$HOME/.hms-mirror/cfg/default.yaml` file as a default. This results in a `$HOME/.hms-mirror/retry/default.retry` file.

If the job fails part way through you can rerun the process with the `--retry` option and it will load the *retry* file and retry any process that wasn't previously successful.

Running HMS Mirror

After running the `setup.sh` script, `hms-mirror` will be available in the `$PATH` in a default configuration.

Assumptions

1. This process will only 'migrate' EXTERNAL and MANAGED (non-ACID/Transactional) tables.
2. MANAGED tables replicated to the **RIGHT** cluster will be converted to "EXTERNAL" tables for the 'metadata' stage. They will be tagged as 'legacy managed' in the **RIGHT** cluster and will NOT be assigned the `external.table.purge=true` flag, yet. Once the table's data has been migrated, the table will be adjusted to be purgeable via `external.table.purge=true` to match the classic `MANAGED` table behavior.
3. The **RIGHT** cluster has 'line of sight' to the **LEFT** cluster.
4. The **RIGHT** cluster has been configured to access the **LEFT** cluster storage. See link clusters. This is the same configuration that would be required to support `distcp` from the **RIGHT** cluster to the **LEFT** cluster.
5. The movement of metadata/data is from the **LEFT** cluster to the **RIGHT** cluster.
6. With Kerberos, each cluster must share the same trust mechanism.
   - The **RIGHT** cluster must be Kerberized IF the **LEFT** cluster is.
   - The **LEFT** cluster does NOT need to be kerberized if the **RIGHT** cluster is kerberized.
7. The *LEFT* cluster does NOT have access to the **RIGHT** cluster.
8. The credentials use by 'hive' (doas=false) in the **RIGHT** cluster must have access to the required storage (hdfs) locations on the lower cluster.
   - If the **RIGHT** cluster is running impersonation (doas=true), that user must have access to the required storage (hdfs) locations on the lower cluster.

HELP

```
usage: hms-mirror
                version: ....
 -accept,--accept                       Accept ALL confirmations and
                                        silence prompts
 -cfg,--config <filename>               Config with details for the
                                        HMS-Mirror.  Default:
                                        $HOME/.hms-mirror/cfg/default
                                        .yaml
 -d,--data <strategy>                   Specify how the data will
                                        follow the schema.
                                        [SCHEMA_ONLY, LINKED, SQL,
                                        EXPORT_IMPORT, HYBRID,
                                        INTERMEDIATE, COMMON]
 -db,--database <databases>             Comma separated list of
```

```
                                            Databases (upto 100).
 -dbp,--db-prefix <prefix>                  Optional: A prefix to add to
                                            the RIGHT cluster DB Name.
                                            Usually used for testing.
 -e,--execute                               Execute actions request,
                                            without this flag the process
                                            is a dry-run.
 -h,--help                                  Help
 -is,--intermediate-storage <storage-path>  Intermediate Storage used
                                            with Data Strategy
                                            INTERMEDIATE.
 -o,--output-dir <outputdir>                Output Directory (default:
                                            $HOME/.hms-mirror/reports/<yy
                                            yy-MM-dd_HH-mm-ss>
 -r,--retry                                 Retry last incomplete run for
                                            'cfg'.  If none specified,
                                            will check for 'default'
 -ro,--read-only                            For SCHEMA_ONLY, COMMON, and
                                            LINKED data strategies set
                                            RIGHT table to NOT purge on
                                            DROP
 -sql,--sql-output                          Output the SQL to the report
 -tf,--table-filter <regex>                 Filter tables with name
                                            matching RegEx
```

Connections

`hms-mirror` connects to 3 endpoints. The hive jdbc endpoints for each cluster (2) and the `hdfs` environment configured on the running host. Which means you'll need:

- jdbc drivers to match the jdbc endpoints
- For **non** CDP 7.x environments and Kerberos connections, an edge node with the current hadoop libraries.

See the [config](#) section to setup the config file for `hms-mirror`.

Configuring the Libraries

Non-Kerberos Connections

The easiest connections are 'non-kerberos' jdbc connections, either to HS2 with AUTH models that aren't **Kerberos** or through a **Knox** proxy. Under these conditions, only the **standalone** jdbc drivers are required. Each of the cluster configurations contains an element `jarFile` to identify those standalone libraries.

```
    hiveServer2:
      uri: "<jdbc-url>"
      connectionProperties:
        user: "*****"
        password: "*****"
      jarFile: "<environment-specific-jdbc-standalone-driver>"
```

When dealing with clusters supporting different version of Hive (Hive 1 vs. Hive 3), the jdbc drivers aren't forward OR backward compatible between these versions. Hence, each jdbc jar file is loaded in a sandbox that allows us to use the same driver class, but isolate it between the two jdbc jars.

Place the two jdbc jar files in any directory **EXCEPT** `$HOME/.hms-mirror/aux_libs` and reference the full path in the `jarFile` property for that `hiveServer2` configuration.

SAMPLE Commandline

```
hms-mirror -db tpcds_bin_partitioned_orc_10
```

Kerberized Connections

`hms-mirror` relies on the Hadoop libraries to connect via 'kerberos'. If the clusters are running different versions of Hadoop/Hive, we can only support connecting to one of the clusters via kerberos. `hms-mirror` is built with the dependencies for Hadoop 3.1 (CDP 7.1.x). Kerberos connections are NOT supported in the 'sandbox' configuration we discussed above.

There are three scenarios for kerberized connections.

Scenario	LEFT Kerberized/Version	RIGHT Kerberized/Version	Notes	Sample Cor
1	No HDP2	Yes HDP 3 or CDP 7	1. The hadoop libs are built into `hms-mirror` for this scenario. 2. 'hms-mirror' needs to be run from a node on the HDP3/CDP cluster. 3. place the RIGHT cluster jdbc jar file in `$HOME/.hms-mirror/aux_libs` (yes this contradicts some earlier directions) 4. comment out the `jarFile` property for the RIGHT cluster hiveServer2 setting.	`hms-mirror -` `tpcds_bin_pa`
2	YES HDP 3 or CDP 7	YES HDP 3 or CDP 7	1. The hadoop libs are built into `hms-mirror` for this scenario. 2. 'hms-mirror' needs to be run from a node on the HDP3/CDP cluster. 3. place the RIGHT cluster jdbc jar file in $HOME/.hms-mirror/aux_libs	`hms-mirror -` `tpcds_bin_pa`

			(yes this contradicts some earlier directions) 4. comment out the `jarFile` property for the LEFT AND RIGHT cluster hiveServer2 settings.	
3	YES HDP2 or Hive 1	NO HDP 3 or CDP 7	Not Supported when `hms-mirror` run from the RIGHT cluster.	hms-mirror tpcds_bin_pa
4	YES HDP2 or Hive 1	YES HDP2 or Hive 1	1. The Kerberos credentials must be TRUSTED to both clusters 2. Add `--hadoop-classpath` as a commandline option to `hms-mirror`. This replaces the prebuilt Hadoop 3 libraries with the current environments Hadoop Libs. 3. Add the jdbc standalone jar file to `$HOME/.hms-mirror/aux_libs` 4. Comment out/remove the `jarFile` references for BOTH clusters in the configuration file.	hms-mirror tpcds_bin_pa --hadoop-cla

For kerberos jdbc connections, ensure you are using an appropriate Kerberized Hive URL.

```
jdbc:hive2://s03.streever.local:10000/;principal=hive/_HOST@STREEVER.LOCAL
```

Troubleshooting

If each JDBC endpoint is Kerberized and the connection to the LEFT or RIGHT is succesful, both NOT both and the program seems to just hang with no exception... it's most likely that the Kerberos ticket isn't TRUSTED across the two environments. You will only be able to support a kerberos connection to the cluster where the ticket is trusted. The other cluster connection will need to be anything BUT kerberos.

Add `--show-cp` to the `hms-mirror` commandline to see the classpath used to run.

The argument `--hadoop-classpath` allows us to replace the embedded Hadoop Libs (v3.1) with the libs of the current platform via a call to `hadoop classpath`. This is necessary to connect to kerberized Hadoop v2/Hive v1 environments.

Check the location and references to the JDBC jar files. General rules for Kerberos Connections:

- The jdbc jar file should be in the `$HOME/.hms-mirror/aux_libs`. For kerberos connections, we've seen issues attempting to load this jar in a sandbox, so this makes it available to the global classpath/loader.
- Get a kerberos ticket for the running user before launching `hms-mirror`.

Output

Application Report

When the process has completed a markdown report is generated with all the work that was done. The report location will be at the bottom of the output window. You will need a *markdown* renderer to read the report. Markdown is a textfile, so if you don't have a renderer you can still look at the report, it will just be harder to read.

The report location is displayed at the bottom on the application window when it's complete.

Action Report for LEFT and RIGHT Clusters

Under certain conditions, additional actions may be required to achieve the desired 'mirror' state. An output script for each cluster is created with these actions. These actions are NOT run by `hms-mirror`. They should be reviewed and understood by the owner of the dataset being `mirrored` and run when appropriate.

The locations are displayed at the bottom on the application window when it's complete.

SQL Execution Output

A SQL script of all the actions taken will be written to the local output directory.

Logs

There is a running log of the process in `$HOME/.hms-mirror/logs/hms-mirror.log`. Review this for details or issues of the running process.
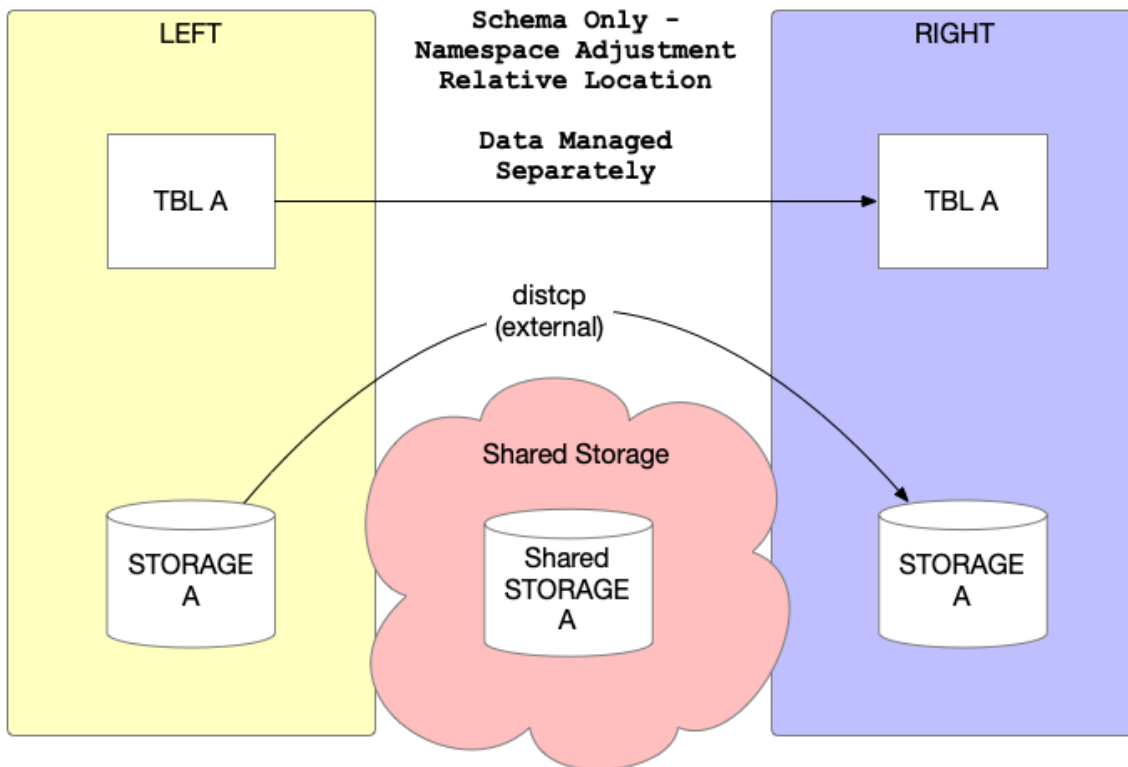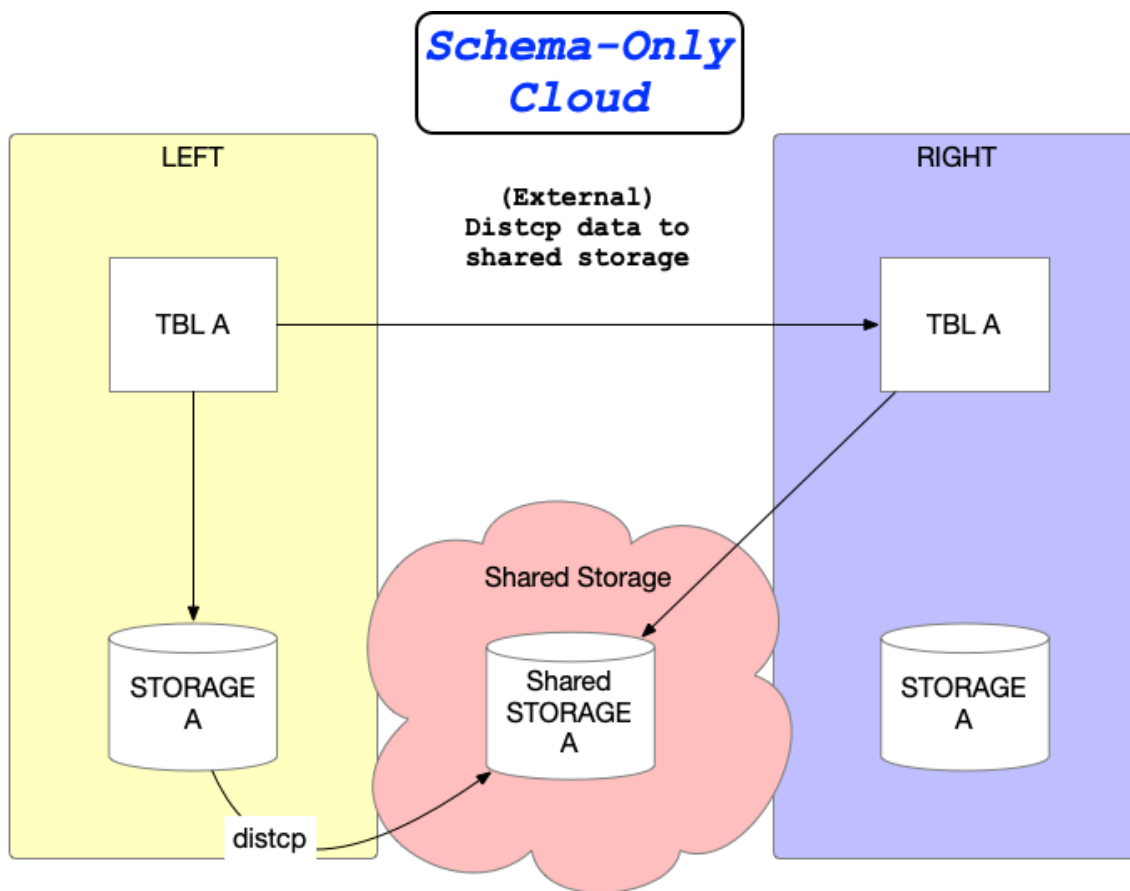
Strategies

Schema Only

Transfer the schema only, replace the location with the RIGHT clusters location namespace and maintain the relative path.

The data is transferred by an external process, like 'distcp'.

Schema-Only

LEFT

Schema Only -
Namespace Adjustment
Relative Location

Data Managed
Separately

TBL A → TBL A

distcp
(external)

STORAGE
A

Shared Storage

Shared
STORAGE
A

RIGHT

STORAGE
A

Figure shows diagram with "Schema-Only Cloud" at top. LEFT cluster (yellow) contains TBL A and STORAGE A. RIGHT cluster (purple) contains TBL A and STORAGE A. Shared Storage cloud (pink) contains Shared STORAGE A. "(External) Distcp data to shared storage" labels the transfer, with "distcp" arrow from LEFT STORAGE A to Shared STORAGE A.

Linked

Assumes the clusters are LINKED. We'll transfer the schema and leave the location as is on the new cluster.

This provides a means to test Hive on the RIGHT cluster using the LEFT clusters storage.
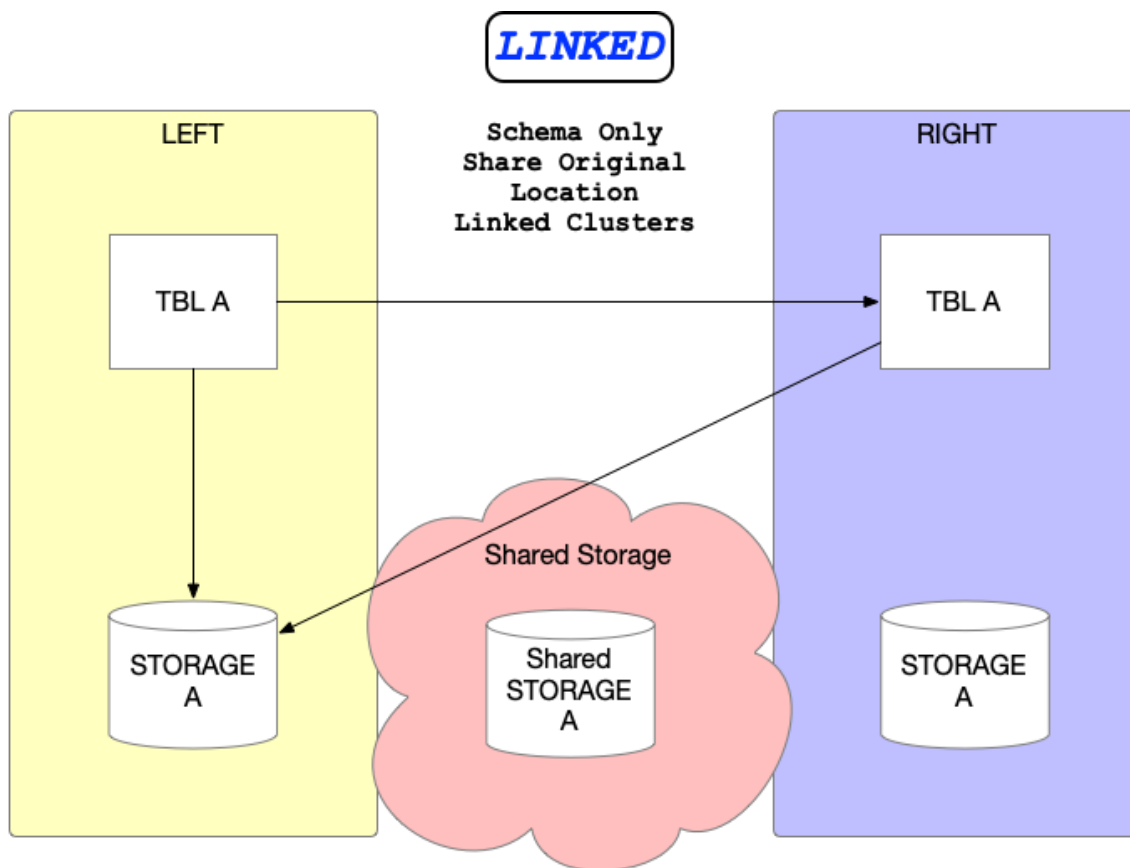
SQL

Assumes the clusters are LINKED. We'll use SQL to migrate the data from one cluster to another.

SQL / Export_Import Hybrid

LEFT

RIGHT

Schema and Data
Linked Clusters

TBL A

TBL A

SQL or EXPORT_IMPORT
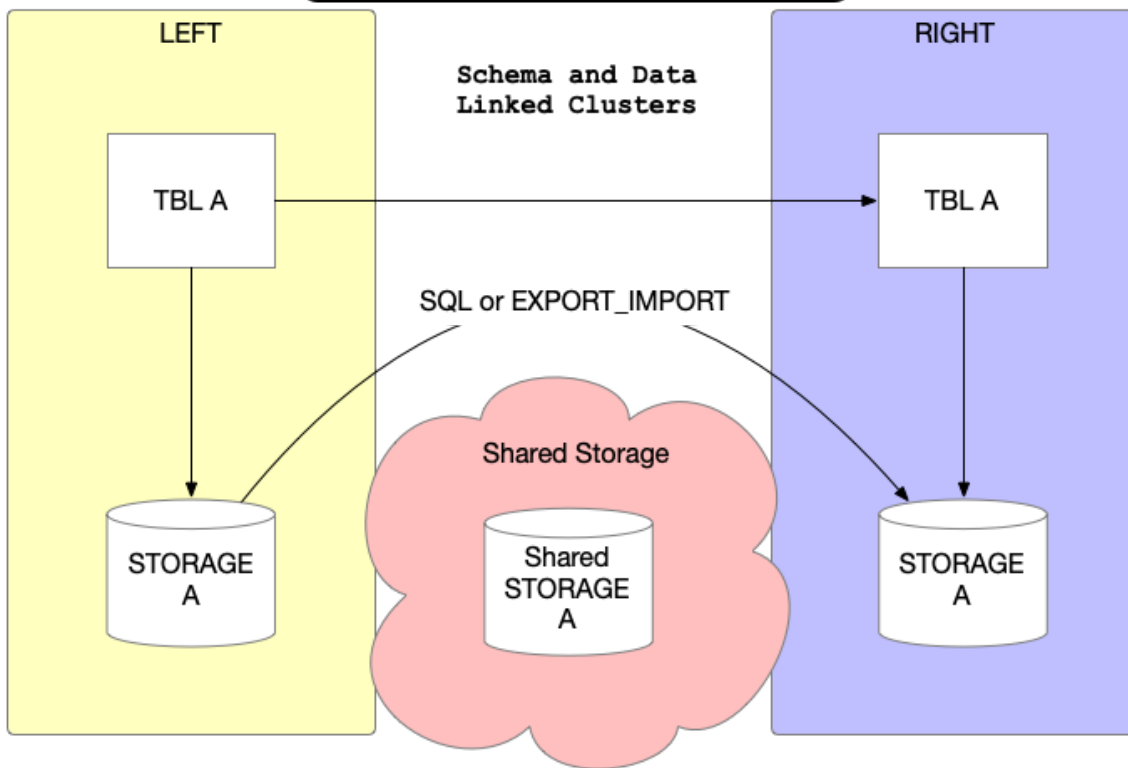
Shared Storage

STORAGE A

Shared STORAGE A

STORAGE A
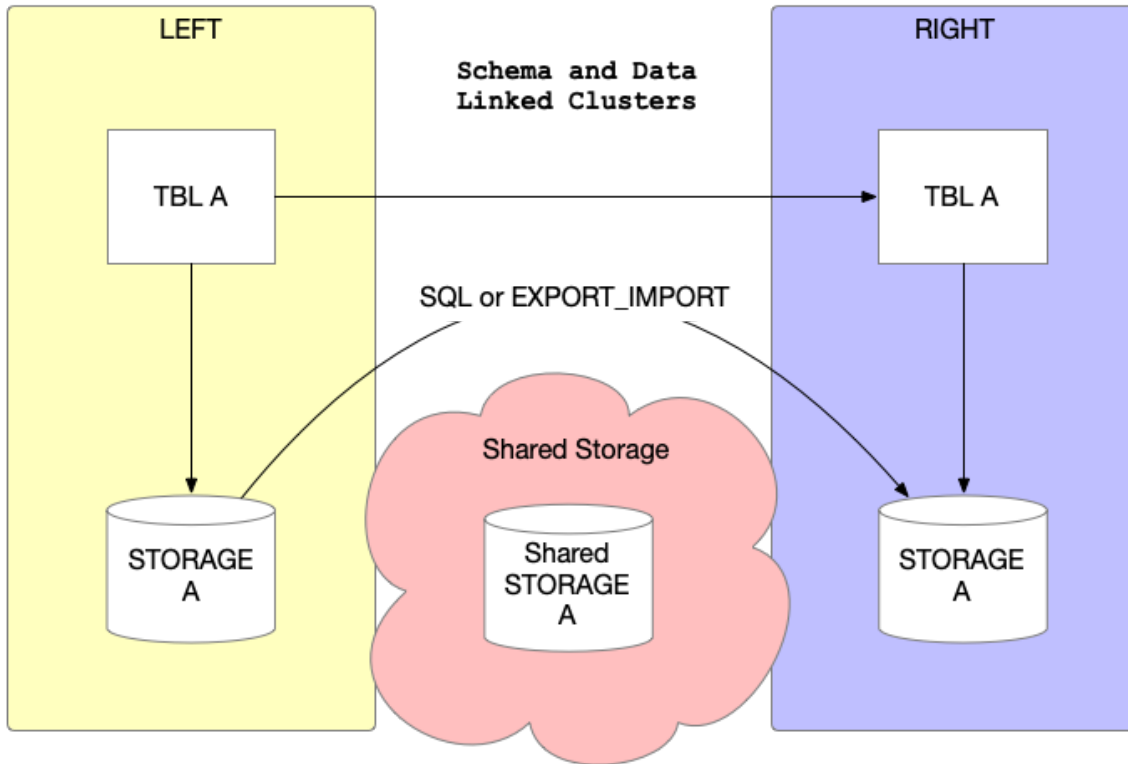
Export Import

Assumes the clusters are LINKED. We'll use EXPORT_IMPORT to get the data to the new cluster.

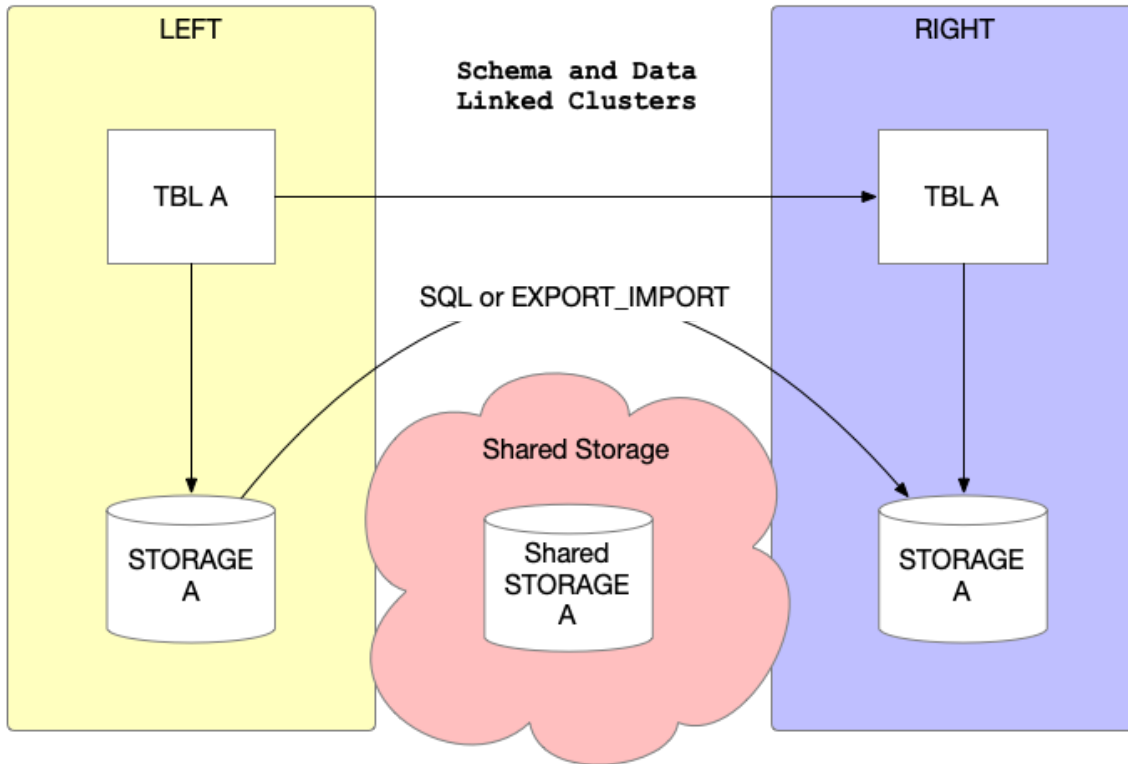EXPORT to a location on the LEFT cluster where the RIGHT cluster can pick it up with IMPORT.

Hybrid

Hybrid is a strategy that select either SQL or EXPORT_IMPORT for the tables data strategy depending on the criteria of the table.

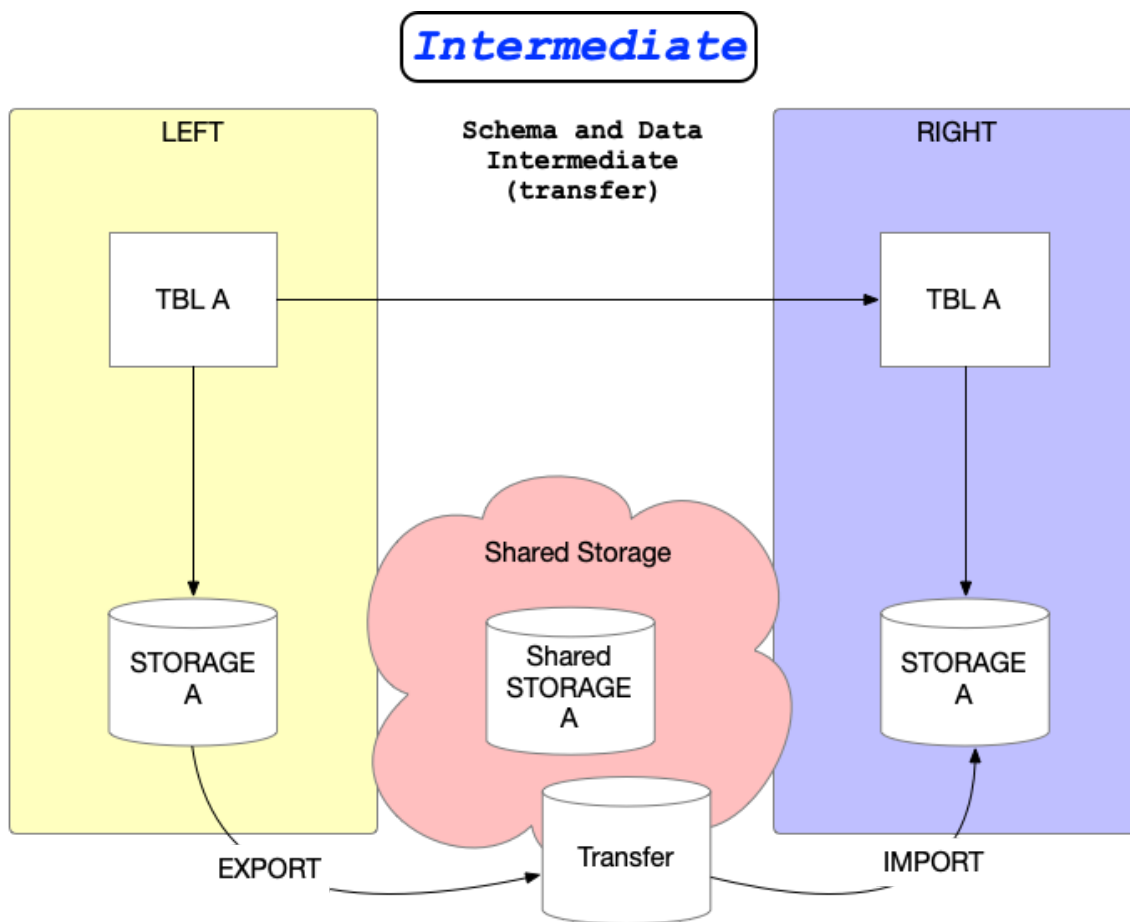**Intermediate**

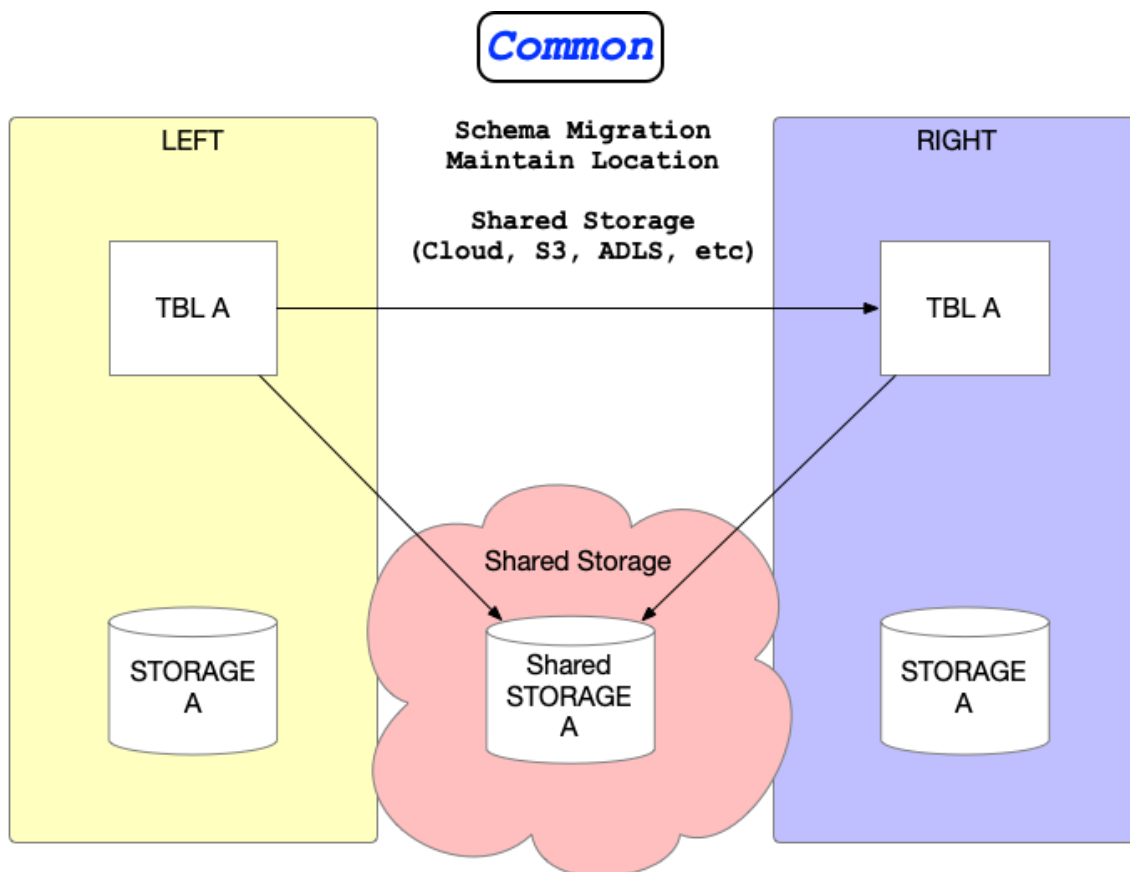The data migration with the schema will go through an intermediate storage location.

The process will EXPORT data on the LEFT cluster to the intermediate location with IMPORT from the RIGHT cluster.

Common

The data storage is shared between the two clusters and no data migration is required.

Schema's are transferred using the same location.

Issues

Table processing completed with `ERROR`

We make various checks as we perform the migrations and when those check don't pass, the result is an error.

Solution

In tips we suggest running with `--dryrun` first. This will catch the potential issues first, without taking a whole lot of time. Use this to remediate issues before executing.

If the scenario that cause the `ERROR` is known, a remediation summary will be in the output report under **Issues** for that table. Follow those instructions than rerun the process with `--retry`.

Connecting to HS2 via Kerberos

Connecting to an HDP cluster running 2.6.5 with Binary protocol and Kerberos triggers an incompatibility issue: `Unrecognized Hadoop major version number: 3.1.1.7.1....`

Solution

The application is built with CDP libraries (excluding the Hive Standalone Driver). When `kerberos` is the `auth` protocol to connect to **Hive 1**, it will get the application libs which will NOT be compatible with the older cluster.

Kerberos connections are only supported to the CDP cluster.

When connecting via `kerberos` , you will need to include the `--hadoop-classpath` when launching `hms-mirror` .
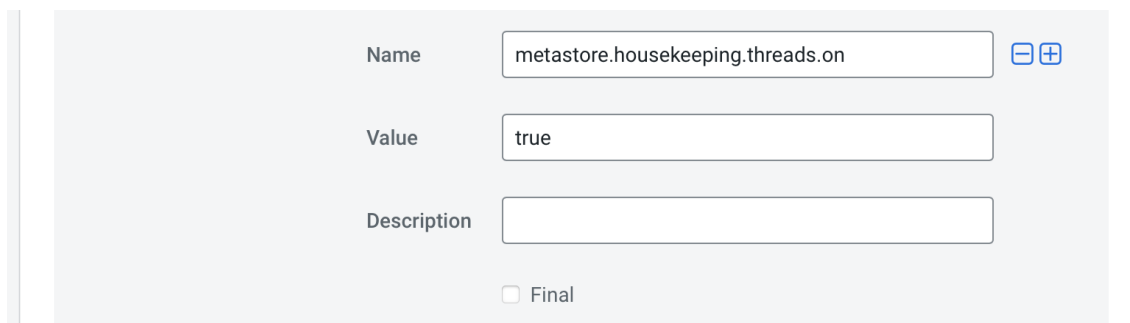
Auto Partition Discovery not working

I've set the `partitionDiscovery:auto` to `true` but the partitions aren't getting discovered.

Solution

In CDP Base/PVC versions < 7.1.6 have not set the house keeping thread that runs to activate this feature.

In the Hive metastore configuration in Cloudera Manager set `metastore.housekeeping.threads.on=true` in the *Hive Service Advanced Configuration Snippet (Safety Valve) for hive-site.xml*

HDFS Permissions Issues

Caused by: java.lang.RuntimeException: org.apache.hadoop.hive.ql.security.authorization.plugin.HiveAccessControlException:Permission denied: user [dstreev] does not have [ALL] privilege on [hdfs://HDP50/apps/hive/warehouse/tpcds_bin_partitioned_orc_10.db/web_site]

After checking permissions of 'dstreev': Found that the 'dstreev' user was NOT the owner of the files in these directories on the LEFT cluster. The user running the process needs to be in 'dfs.permissions.superusergroup' for the lower clusters 'hdfs' service. Ambari 2.6 has issues setting this property: https://jira.cloudera.com/browse/EAR-7805

Follow workaround above or add user to the 'hdfs' group. Or use Ranger to allow all access. On my cluster, with no Ranger, I had to use '/var/lib/ambari-server/resources/scripts/configs.py' to set it manually for Ambari.

```
sudo ./configs.py --host=k01.streever.local --port=8080 -u admin -p admin -n hdp50 -c hdfs-site -a set -k dfs.permissions.superusergroup -v hdfs_admin
```

YARN Submission stuck in ACCEPTED phase

The process uses a connection pool to Hive. If concurrency value for the cluster is too high, you may have reached the maximum ratio of AM (Application Masters) for the YARN queue.

Review the ACCEPTED jobs and review the jobs *Diagnostics* status for details on *why* the jobs is stuck.

Solution

Either of:

1. Reduce the concurrency in the configuration file for `hms-mirror`
2. Increase the AM ratio or Queue size to allow the jobs to be submitted. This can be done while the process is running.

Spark DFS Access

If you have problems accessing HDFS from `spark-shell` or `spark-submit` try adding the following configuration to spark:

```
--conf
spark.yarn.access.hadoopFileSystems=hdfs://<NEW_NAMESPACE>,hdfs://<OLD_NAMESPACE>
```