

..

Migrating Operational Database to CDP Public Cloud

Date published: 2021-10-25

Date modified: 2023-03-10

CLOUDERA

<https://docs.cloudera.com/>

Legal Notice

© Cloudera Inc. 2024. All rights reserved.

The documentation is and contains Cloudera proprietary information protected by copyright and other intellectual property rights. No license under copyright or any other intellectual property right is granted herein.

Unless otherwise noted, scripts and sample code are licensed under the Apache License, Version 2.0.

Copyright information for Cloudera software may be found within the documentation accompanying each component in a particular release.

Cloudera software includes software from various open source or other third party projects, and may be released under the Apache Software License 2.0 (“ASLv2”), the Affero General Public License version 3 (AGPLv3), or other license terms. Other software included may be released under the terms of alternative open source licenses. Please review the license and notice files accompanying the software for additional licensing information.

Please visit the Cloudera software product page for more information on Cloudera software. For more information on Cloudera support services, please visit either the Support or Sales page. Feel free to contact us directly to discuss your specific needs.

Cloudera reserves the right to change any products at any time, and without notice. Cloudera assumes no responsibility nor liability arising from the use of products, except as expressly agreed to in writing by Cloudera.

Cloudera, Cloudera Altus, HUE, Impala, Cloudera Impala, and other Cloudera marks are registered or unregistered trademarks in the United States and other countries. All other trademarks are the property of their respective owners.

Disclaimer: EXCEPT AS EXPRESSLY PROVIDED IN A WRITTEN AGREEMENT WITH CLOUDERA, CLOUDERA DOES NOT MAKE NOR GIVE ANY REPRESENTATION, WARRANTY, NOR COVENANT OF ANY KIND, WHETHER EXPRESS OR IMPLIED, IN CONNECTION WITH CLOUDERA TECHNOLOGY OR RELATED SUPPORT PROVIDED IN CONNECTION THEREWITH. CLOUDERA DOES NOT WARRANT THAT CLOUDERA PRODUCTS NOR SOFTWARE WILL OPERATE UNINTERRUPTED NOR THAT IT WILL BE FREE FROM DEFECTS NOR ERRORS, THAT IT WILL PROTECT YOUR DATA FROM LOSS, CORRUPTION NOR UNAVAILABILITY, NOR THAT IT WILL MEET ALL OF CUSTOMER’S BUSINESS REQUIREMENTS. WITHOUT LIMITING THE FOREGOING, AND TO THE MAXIMUM EXTENT PERMITTED BY APPLICABLE LAW, CLOUDERA EXPRESSLY DISCLAIMS ANY AND ALL IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO IMPLIED WARRANTIES OF MERCHANTABILITY, QUALITY, NON-INFRINGEMENT, TITLE, AND FITNESS FOR A PARTICULAR PURPOSE AND ANY REPRESENTATION, WARRANTY, OR COVENANT BASED ON COURSE OF DEALING OR USAGE IN TRADE.

Contents

- HBase Migration through Replication Manager.....4**
 - Cloudera Replication Plugin.....5
 - HBase migration prerequisites.....6
 - Migrating HBase data from CDH/HDP to COD CDP Public Cloud..... 12
 - Replication Manager..... 17

- Phoenix Replication to Cloudera Operational Database..... 18**
 - Replicating Phoenix Data Tables..... 18
 - Replicating Phoenix tables for versions lower than 4.14..... 18
 - Replicating Phoenix 4.14 and newer versions including 5.x..... 19
 - Replicating Phoenix Index Tables.....19

HBase Migration through Replication Manager

Apache HBase is a scalable, distributed, column-oriented datastore that provides real-time read/write random access to very large datasets hosted on HDFS. In CDP Operational Database or COD you use Apache HBase as a datastore with HDFS and/or S3/ABFS providing the storage infrastructure.

You can use one of the following methods to replicate HBase data based on your requirements:

Table 1: Replication methods

Methods	Description	When to use
Cloudera Replication Plugin for cluster versions that Replication Manager does not support.	You can prepare your data for migration, then set up the replication plugin and use a snapshot to migrate your data.	<p>The following list consolidates all the minimum supported versions of source and target cluster combinations for which you can use the replication plugin to replicate HBase data. The replication plugin is compatible with all the CDP Public Cloud releases.</p> <ul style="list-style-type: none"> From CDH 5.10 using CM 6.3.0 to CDP Public Cloud on AWS. See Replicate data from CDH 5 source clusters. From CDH 5.10 using CM 6.3.4 to CDP Public Cloud on Azure. See Replicate data from CDH 5 source clusters. From CDH 6.1 using CM 6.3.0 to CDP Public Cloud on AWS. See Replicate data from CDH 6 source clusters. From CDH 6.1 using CM 7.1.1/6.3.4 to CDP Public Cloud on Azure. See Replicate data from CDH 6 source clusters. CDP 7.1.1 using CM 7.1.1 to CDP Public Cloud on AWS and Azure. See Replicate data from CDP Private Cloud Base source clusters. HDP 2.6.5 and HDP 3.1.1 to CDP Public Cloud on AWS and Azure. See Replicate data from HDP 2 and HDP 3 source clusters. <p>For information about use cases that are not supported by Replication Manager, see support matrix.</p>

Methods	Description	When to use
Replication Manager	You can use Replication Manager to migrate HBase data that uses HBase replication through HBase replication policy.	<p>When the source cluster and target cluster meet the requirements of supported use cases. See caveats.</p> <p>The following list consolidates all the minimum supported versions of source and target cluster combinations for which you can use HBase replication policies to replicate HBase data.</p> <ul style="list-style-type: none"> From CDP 7.1.6 using CM 7.3.1 to CDP 7.2.14 Data Hub using CM 7.6.0. See Replicate data from CDP Private Cloud Base source clusters. From CDH 6.3.3 using CM 7.3.1 to CDP 7.2.14 Data Hub using CM 7.6.0. See Replicate data from CDH 6 source clusters. From CDH 5.16.2 using CM 7.4.4 (patch-5017) to COD 7.2.14. See Replicate data from CDH 5 source clusters. From COD 7.2.14 to COD 7.2.14. <p>See support matrix for more information.</p>

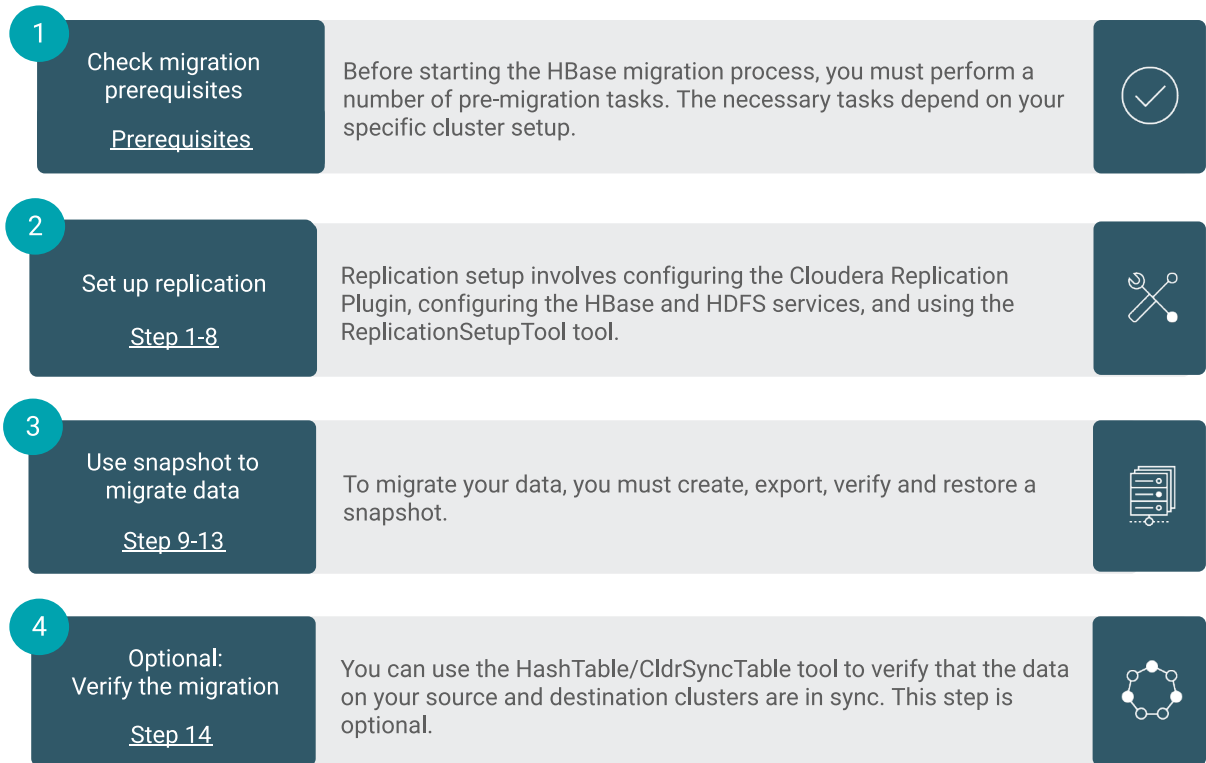
Related Information[HBase Replication Policy](#)[Creating HBase Replication Policy](#)

Cloudera Replication Plugin

You can migrate your HBase data from CDH or HDP to COD CDP Public Cloud. First you have to prepare your data for migration, then set up the replication plugin and use a snapshot to migrate your data.

Before starting the migration you have to understand and perform all the necessary data migration preparation tasks. Then, in the absence of Replication Manager support, you must perform certain tasks on both your source cluster (CDP or HDP) and your destination CDP cluster to ensure a successful migration.

The following is an overview of the HBase migration steps:



rect 45, 45, 219, 121 [HBase migration prerequisites](#)

rect 41, 185, 221, 254 [Step 1-8](#)

rect 39, 305, 217, 382 [Step 9-13](#)

rect 47, 433, 215, 509 [Step 14](#)

HBase migration prerequisites

Before migrating HBase data from CDP/HDP to COD CDP Public Cloud, you have to understand and perform all the necessary migration preparation tasks.

- If you are migrating from CDH, configure Ranger ACLs in CDP corresponding to the HBase ACLs in your existing CDH cluster.

For more information, see [Configure a resource-based service:HBase](#).

- Migrate all applications to use the new HBase-Spark connector because the Spark-HBase connector that you were using in CDH or HDP is no longer supported in CDP.

For more information, see [Using the HBase-Spark connector](#).

- Ensure that all data has been migrated to a supported encoding type before the migration. For more information, see [Removing PREFIX_TREE Data Block Encoding](#) on page 7 . This step is applicable for HBase 1.x when you are migrating data from HDP 2 or CDH 5. This step is not applicable for HBase 2.x.
- Ensure that your HFiles are compatible with the version Apache HBase in CDP. For more information, see [Validating HFiles](#) on page 9.
- Ensure that you upgrade any external or third-party co-processors manually because they are not automatically upgraded during migration. Ensure that your co-processor classes are compatible with CDP. For more information, see [Checking co-processor classes](#) on page 8.
- Review the deprecated APIs and incompatibilities when upgrading from HDP 2.x or CDH 5.x to CDP. For more information, see HBase in [Deprecation notices in Cloudera Runtime](#).

- Install Cloudera Replication Plugin on the source cluster.



Important: Contact your Cloudera account team to get access to the Cloudera Replication Plugin version compatible with your source cluster CDH or HDP version.

- On the target COD cluster, make sure ZooKeeper (2181) and RegionServer (16020) ports are opened for the IP addresses of all RegionServers in the source cluster.

For AWS, this involves editing the Security Groups which are in use by the COD cluster.

- Ensure that all the RegionServers' hosts in the COD cluster are resolvable by both the IP and name addresses by all the RegionServers' hosts in the source cluster.

Removing PREFIX_TREE Data Block Encoding

Before migrating to COD CDP Public Cloud, ensure that you have transitioned all the data to a supported encoding type.

About this task



Important:

Ensure that you complete all the pre-migration steps if you have Apache HBase installed in your existing CDH cluster.

The PREFIX_TREE data block encoding code is removed in COD CDP Public Cloud, meaning that HBase clusters with PREFIX_TREE enabled will fail. Therefore, before migrating to COD CDP Public Cloud you must ensure that all data has been transitioned to a supported encoding type.

The following pre-upgrade command is used for validation: `hbase pre-upgrade validate-dbe`

If your cluster is Kerberized, ensure that you run the kinit command as a hbase user before running the pre-upgrade commands. Ensure you have valid Kerberos credentials. You can list the Kerberos credentials using the klist command, and you can obtain credentials using the kinit command.

Before you begin

1. Download and distribute parcels for the target version of CDP.



Important: Do not activate the parcel yet.

If the downloaded parcel version is higher than the current Cloudera Manager version, the following error message displayed:

Error for parcel CDH-7.x.parcel : Parcel version 7.X is not supported by this version of Cloudera Manager. Upgrade Cloudera Manager to at least 7.X before using this version of the parcel.

You can safely ignore this error message.

2. Find the installed parcel at `/opt/cloudera/parcels`.

For example: `/opt/cloudera/parcels/CDH-7.1.1-1.cdh7.1.1.p0.3224867/bin/hbase`

Use the CDP parcel to run the pre-upgrade commands. Cloudera recommends that you run them on an HMaster host.

Procedure

1. Run the hbase pre-upgrade validate-dbe command using the new installation.

For example, if you have installed the CDH-7.1.1-1 parcel, you must run the following command:

```
/opt/cloudera/parcels/CDH-7.1.1-1.cdh7.1.1.p0.3224867/bin/hbase pre-upgrade validate-dbe
```

The commands check whether your table or snapshots use the PREFIX_TREE Data Block Encoding.

This command does not take much time to run because it validates only the table-level descriptors.

If PREFIX_TREE Data Block Encoding is not used, the following message is displayed:

```
The used Data Block Encodings are compatible with HBase 2.0.
```

If you see this message, your data block encodings are compatible, and you do not have to perform any more steps.

If PREFIX_TREE Data Block Encoding is used, a similar error message is displayed:

```
2018-07-13 09:58:32,028 WARN [main] tool.DataBlockEncodingValidator: Incompatible DataBlockEncoding for table: t, cf: f, encoding: PREFIX_TREE
```

If you got an error message, continue with Step 2 and fix all your PREFIX_TREE encoded tables.

2. Fix your PREFIX_TREE encoded tables using the old installation.

You can change the Data Block Encoding type to PREFIX, DIFF, or FAST_DIFF in your source cluster.

Our example validation output reported column family f of table t is invalid. Its Data Block Encoding type is changed to FAST_DIFF in this example:

```
hbase> alter 't', { NAME => 'f', DATA_BLOCK_ENCODING => 'FAST_DIFF' }
```

Checking co-processor classes

External co-processors are not automatically upgraded, you must upgrade them manually. Before migrating, ensure that your co-processors are compatible with the migration.

About this task



Important:

Ensure that you complete all the pre-migration steps if you have Apache HBase installed in your existing CDH cluster.

There are two ways to handle co-processor upgrade:

- Upgrade your co-processor jars manually before continuing the migration.
- Temporarily unset the co-processors and continue the migration.

Once they are manually upgraded, they can be reset.

Attempting to migrate without upgrading the co-processor jars can result in unpredictable behaviour such as HBase role start failure, HBase role crashing, or even data corruption.

If your cluster is Kerberized, ensure that you run the kinit command as a hbase user before running the pre-upgrade commands.

Procedure

1. Download and distribute parcels for target version of COD CDP Public Cloud.



Important: Do not activate the parcel yet.

If the downloaded parcel version is higher than the current Cloudera Manager version, the following error message displayed:

Error for parcel CDH-7.X.parcel : Parcel version 7.X is not supported by this version of Cloudera Manager. Upgrade Cloudera Manager to at least 7.X before using this version of the parcel.

You can safely ignore this error message.

2. Run the hbase pre-upgrade validate-cp commands to check if your co-processors are compatible with the migration.

Use the CDP parcel to run the pre-upgrade commands. Cloudera recommends that you run them on an HMaster host.

For example, you can check for co-processor compatibility on master:

```
$ /opt/cloudera/parcels/CDH-7.1.1-1.cdh7.1.1.p0.3224867/bin/hbase pre-upgrade validate-cp -jar /opt/cloudera/parcels/CDH-7.1.1-1.cdh7.1.1.p0.3224867/jars/ -config
```

Or, you can validate every table level co-processors where the table name matches to the .* regular expression:

```
$ /opt/cloudera/parcels/CDH-7.1.1-1.cdh7.1.1.p0.3224867/bin/hbase pre-upgrade validate-cp -table '.*'
```

Optionally, you can run the following command for a more detailed output:

```
HBASE_ROOT_LOGGER=DEBUG,console hbase pre-upgrade validate-cp -table '.*'
```

This way you can verify that all of the required tables were checked. The detailed output should contain lines like the following where test_table is a table on the server:

```
21/05/10 11:07:58 DEBUG coprocessor.CoprocessorValidator: Validating table test_table
```

3. Check the output to determine if your co-processors are compatible with the upgrade.

The output looks similar to the following:

```
$ hbase pre-upgrade validate-cp -config
... some output ...
$ echo $?
0
```

If echo \$? prints 0, the check was successful and your co-processors are compatible. A non-zero value means unsuccessful, your co-processors are not compatible.

Validating HFiles

Before migrating to COD CDP Public Cloud ensure that your HFiles are compatible with the version Apache HBase in CDP.

About this task



Important:

Ensure that you complete all the pre-migration steps if you have Apache HBase installed in your existing CDH cluster.

After converting all the PREFIX_TREE encoded HFiles to a supported format, there may be HFiles that are not compatible with HBase in CDP.

Use the following pre-upgrade commands to validate HFiles:

- `hbase pre-upgrade validate-hfile`
- `hbase hbck`

The validate-hfile tool lists all the corrupt HFiles with details. The hbck command identifies HFiles that are in a bad state.

If your cluster is Kerberized, ensure that you run the kinit command as a hbase user before running the pre-upgrade commands.

Before you begin

1. Download and distribute parcels for the target version of CDP.



Important: Do not activate the parcel yet.

If the downloaded parcel version is higher than the current Cloudera Manager version, the following error message displayed:

Error for parcel CDH-7.x.parcel : Parcel version 7.X is not supported by this version of Cloudera Manager. Upgrade Cloudera Manager to at least 7.X before using this version of the parcel.

You can safely ignore this error message.

2. Find the installed parcel at /opt/cloudera/parcels.

For example: /opt/cloudera/parcels/CDH-7.1.1-1.cdh7.1.1.p0.3224867/bin/hbase

Use the CDP parcel to run the pre-upgrade commands. Cloudera recommends that you run them on an HMaster host.

Procedure

1. Run the hbase pre-upgrade validate-hfile command using the new CDP installation.

This command checks every HFile (including snapshots) to ensure that the HFiles are not corrupted and have the compatible block encoding. It opens every HFile, and this operation can take a long time based on the size and number of HFiles in your cluster.

For example, if you have installed the CDH-7.1.1-1 parcel, you must run the following command:

```
/opt/cloudera/parcels/CDH-7.1.1-1.cdh7.1.1.p0.3224867/bin/hbase pre-upgrade validate-hfile
```

If there are no incompatible HFiles, the following message is displayed:

```
Checked 3 HFiles, none of them are corrupted.
```

There are no incompatible HFiles.

If you have an incompatible HFile, a similar error message to this is displayed:

```
2018-06-05 16:20:47,322 INFO [main] tool.HFileContentValidator: Corrupted
file: hdfs://example.com:8020/hbase/data/default/t/72ea7f7d625ee30f959897
d1a3e2c350/prefix/7e6b3d73263c4851bf2b8590a9b3791e
2018-06-05 16:20:47,383 INFO [main] tool.HFileCo
ntentValidator: Corrupted file: hdfs://example.com:8020/hbase/archive/da
ta/default/t/56be41796340b757eb7fff1eb5e2a905/f/29c641ae91c34fc3bee881f4
5436b6d1
```

If you get an error message, continue with Step 2 otherwise skip to Step 5.

2. Identify what kind of HFiles were reported in the error message.

The report can contain two different kinds of HFiles, they differ in their path:

- If an HFile is in /hbase/data then it belongs to a table.
- If an HFile is located under /hbase/archive/data then it belongs to a snapshot.

3. Fix the HFiles that belong to a table.

- a) Get the table name from the output.

Our example output showed the /hbase/data/default/t/... path, which means that the HFile belongs to the t table which is in the default namespace.

- b) Rewrite the HFiles by issuing a major compaction. Use the old installation.

```
shell> major_compact 't'
```

When compaction is finished, the invalid HFile disappears.

4. Fix the HFiles that belong to a snapshot that is needed. Use the old installation.

- a) Find the snapshot which refers the invalid HFile. You can do this using a shell script and the following example code. In our example output, the HFile is 29c641ae91c34fc3bee881f45436b6d1:

```
#!/bin/bash
invalid HFile
# Find the snapshot which refers to the
for snapshot in $(hbase snapshotinfo -
list-snapshots 2> /dev/null | cut -f 1 -d \|);
do
echo "checking snapshot named '${snapsho
t}'"
hbase snapshotinfo -snapshot "${snapsho
t}" -files 2> /dev/null | grep 29c641ae91c34fc3bee881f45436b6d1
done
```

The following output means that the invalid file belongs to the t_snap snapshot:

```
checking snapshot names 't__snap'
1.0 K t/56be41796340b757eb7fff1eb5
e2a905/f/29c641ae91c34fc3bee881f45436b6d1 (archive)
```

- b) Convert snapshot to another HFile format if data encoding is the issue using the following command:

```
# creating a new namespace for the cleanup process
hbase> create_namespace 'pre_upgrade_
cleanup'
# creating a new snapshot
hbase> clone_snapshot 't_snap', 'pre_
upgrade_cleanup:t'
```

```

hbase> alter 'pre_upgrade_cleanup:t',
{ NAME => 'f', DATA_BLOCK_ENCODING => 'FAST_DIFF' }
hbase> major_compact 'pre_upgrade_cleanup:t'

```



Important: Confirm if the major compaction is complete in the RegionServer Web user interface or the RegionServer logs before you run the following commands to remove invalid snapshots. If the major compaction process is not completed when you delete the snapshot, your files may get corrupted.

```

# removing the invalid snapshot
hbase> delete_snapshot 't_snap'
# creating a new snapshot
hbase> snapshot 'pre_upgrade_cleanup:t',
't_snap'

# removing temporary table
hbase> disable 'pre_upgrade_cleanup:t'
hbase> drop 'pre_upgrade_cleanup:t'
hbase> drop_namespace 'pre_upgrade_cleanup'

```

5. Run the `hbck` command on to identify HFiles in a bad state and remedy those HFiles.

Migrating HBase data from CDH/HDP to COD CDP Public Cloud

When you are migrating from CDH or HDP to COD CDP Public Cloud, you have to set up the Cloudera Replication Plugin and then use a snapshot to migrate your data to COD CDP Public Cloud. Once your data is migrated, optionally you can use the HashTable/CldrSyncTable tool to verify the migration.

About this task



Important: In this topic, we are discussing the manual steps to migrate the HBase data into the COD CDP Public Cloud. Alternatively, you can use the Replication Manager to migrate your HBase data. For more information, see *Using HBase replication policies*.

Before you begin

Ensure you have understood and performed all the necessary data migration preparation tasks listed in *HBase migration prerequisites*.

Procedure

1. Provide the Cloudera Replication Plugin by configuring HBase Client Environment Advanced Configuration Snippet (Safety Valve) for hbase-env.sh for Gateway and HBase Service Environment Advanced Configuration Snippet (Safety Valve) for RegionServers and Masters on the source cluster.

For CDH

- a. In Cloudera Manager, navigate to HBase Configuration .
- b. Find the HBase Client Environment Advanced Configuration Snippet (Safety Valve) for hbase-env.sh property and add the following configuration to provide the Cloudera Replication Plugin path for the Gateway:

```
HBASE_CLASSPATH=$HBASE_CLASSPATH:/opt/cloudera/parcels/cloudera-opdb-
-replication-[***REPLICATION_PLUGIN_VERSION***]-cdh[***CDH_VERSION**
*]-/lib/*
```

For example:

```
HBASE_CLASSPATH=$HBASE_CLASSPATH:/opt/cloudera/parcels/cloudera-opdb-
-replication-1.0-cdh5.14.4-/lib/*
```

- c. Find the HBase Service Environment Advanced Configuration Snippet (Safety Valve) property and add the same configuration to provide the Cloudera Replication Plugin path for RegionServers and Masters:

```
HBASE_CLASSPATH=$HBASE_CLASSPATH:/opt/cloudera/parcels/cloudera-opdb-
-replication-[***REPLICATION_PLUGIN_VERSION***]-cdh[***CDH_VERSION**
*]-/lib/*
```

For example:

```
HBASE_CLASSPATH=$HBASE_CLASSPATH:/opt/cloudera/parcels/cloudera-opdb-
-replication-1.0-cdh5.14.4-/lib/*
```

For more information on CDH parcels, see *Migrating HBase tables*.

For HDP

- a. In Ambari, navigate to CONFIGS ADVANCED Advanced hbase-env hbase-env template.
- b. Find the following line:

```
export HBASE_CLASSPATH=${HBASE_CLASSPATH}
```

- c. Modify the line to include the following configuration:

```
export HBASE_CLASSPATH=${HBASE_CLASSPATH}:/usr/hdp/cloudera-opdb-rep
lication-[***REPLICATION_PLUGIN_VERSION***]-hdp[***version***]-SNAPS
HOT/lib/*
```

For example:

```
export HBASE_CLASSPATH=${HBASE_CLASSPATH}:/usr/hdp/cloudera-opdb-rep
lication-1.0-hdp-2.6.5-SNAPSHOT/lib/*
```

2. Generate and distribute credentials.

- a) Create a machine user with the name hbase-replication (or any one of your choice) in User Management Service (UMS) on the CDP Management Console.
- b) Set your workload password.
- c) Add this machine user as an Environment User using Manage Access in the destination CDP environment.
- d) Perform FreeIPA synchronization for the destination environments.
- e) Verify credentials using the kinit `srv_$YOUR_REP_USER` (for example, if the specified user was *hbase-replication*, this would be `srv_hbase-replication`) command on any node.
- f) In the destination cluster, run the following command to generate the keystore:

```
hbase
com.cloudera.hbase.security.token.CldrReplicationSecurityTool -shared
key cloudera -password [***PASSWORD***] -keystore localjceks://file/tmp/
credentials.jceks
```

- g) Obtain hdfs user kerberos credentials on the destination cluster.

Ensure that you have root access to execute the following command.

```
kinit -kt /var/run/cloudera-scm-agent/process/`ls -l /var/run/cloudera-s
cm-agent/process/ | grep -i "namenode\|datanode" | sort -n | tail -1`/hd
fs.keytab hdfs/${hostname -f}
```

- h) Add the generated jceks file to related hdfs folder in the destination cluster:

```
hdfs dfs -mkdir /hbase-replication
hdfs dfs -put /tmp/credentials.jceks /hbase-replication
hdfs dfs -chown -R hbase:hbase /hbase-replication
```

- i) Copy the generated jceks file to any host of the source cluster.



Warning: You must copy the same jceks file to the source cluster instead of regenerating it.

- j) Add the copied jceks file to related hdfs folder in the source cluster (if source cluster is secured, you need hdfs kerberos credentials for destination cluster):

```
hdfs dfs -mkdir /hbase-replication
hdfs dfs -put /tmp/credentials.jceks /hbase-replication
hdfs dfs -chown -R hbase:hbase /hbase-replication
```

3. Set the `hbase.security.replication.credential.provider.path` property on the destination cluster using the HBase Service Advanced Configuration Snippet (Safety Valve) for `hbase-site.xml` property.

For example, if the path to the jceks file is `hdfs://ns1/hbase-replication/credentials.jceks`, set the following configuration:

```
<property>
<name>hbase.security.replication.credential.provider.path</name>
<value>cdprepjceks://hdfs@[***NAMESERVICE***]/hbase-replication/credenti
als.jceks</value>
</property>
```



Warning: You have to customize the value of the `hbase.security.replication.credential.provider.path` property. You might need to replace this sample "ns1" value with the actual name service value in case of High Availability, or with the active NameNode address otherwise.

4. In the source cluster, enable replication for each table that you want to replicate:

This can be done by setting the REPLICATION_SCOPE on the desired column families for that HBase table in the hbase shell.

```
$ hbase shell
alter 'my-table', {NAME=>'my-family', REPLICATION_SCOPE => '1'}
```

If you do not know what column families exist in your HBase table, use the describe command:

```
$ hbase shell
describe 'my-table'
```



Note: The enable_table_replication command is not supported in this configuration.

5. Set up the cloud services provider credentials for the source cluster using the core-site.xml HDFS configuration file.

a) Find the core-site.xml HDFS configuration file:

- Cloudera Manager: Navigate to HDFS Configuration and find the Cluster-wide Advanced Configuration Snippet (Safety Valve) for core-site.xml property.
- Ambari: Navigate to HDFS CONFIGS ADVANCED and find the custom core-site template.

b) Add the following configuration:

For AWS

Configure the following properties on the source cluster with the AWS keys and write permission to the bucket:

- fs.s3a.access.key
- fs.s3a.secret.key
- fs.s3a.endpoint

For example, fs.s3a.endpoint="s3.eu-central-1.amazonaws.com"

For ABFS

Configure your storage account access key by setting the fs.azure.account.key.[***ACCOUNT NAME***].blob.core.windows.net property to the access key:

```
<property> <name>fs.azure.account.key.[***ACCOUNT NAME***].blob
.core.windows.net</name>
<value>[***ACCOUNT NAME***]-ACCESS-KEY</value>
</property>
```

Access key can be obtained from the Access keys in your storage account settings.

6. Restart the RegionServers and add the client configurations on the source cluster:

- Cloudera Manager: Click Restart stale services.
- Ambari: Click Restart all required.

7. Use the ReplicationSetupTool tool on the source cluster to define the replication peer.

Run this command as the hbase user. Additionally, use the same keystore path that you provided in Step 3.

```
sudo -u hbase hbase org.apache.hadoop.hbase.client.replication.ReplicationSetupTool -clusterKey "zk-host-1,zk-host-2,zk-host-3:2181:/hbase" -endpointImpl "org.apache.hadoop.hbase.replication.regionserver.CldrHBaseInt
erClusterReplicationEndpoint" -peerId 1 -credentialPath "cdprepjceks://h
```

```
dfs@[***NAMESERVICE***/hbase-replication/credentials.jceks" -replicatio
nUser srv_hbase-replication
```

This example uses sudo. However, you can use kinit if it is appropriate for your source cluster setup.

The clusterKey parameter reflects the destination cluster's zookeeper quorum address. Here, *zk-host-1*, *zk-host-2*, *zk-host-3* represent the destination cluster Zookeeper hostnames.

NAMESERVICE is the source cluster nameservice for HA enabled cluster. It can also be the hostname of an active NameNode for a non-HA cluster.

8. Disable the replication peer in the hbase shell.

Disabling the peer before taking the snapshot ensures that incremental data written to your table is not lost while copying the HBase snapshot to COD.

```
$.bin/hbase shell
hbase> disable_peer '1'
```

9. Take a snapshot on the source cluster.

For Cloudera Manager

- a. In Cloudera Manager on the source cluster, select the HBase service.
- b. Click Table Browser.
- c. Select a table.
- d. Click Take Snapshot.
- e. Specify the name of the snapshot and click Take Snapshot.

For hbase shell

- a. You can take the snapshot through the hbase shell:

```
$ hbase shell
hbase> snapshot 'myTable', 'myTableSnapshot-122112'
```

10. Export the snapshot to the destination cluster using the ExportSnapshot tool.

You must run the ExportSnapshot command as the hbase user or as the user that owns the files. The ExportSnapshot tool executes a MapReduce job similar to distcp to copy files to the other cluster. The tool works at the file-system level, so the HBase cluster can be offline, but ensure that the HDFS cluster is online.

- a) Navigate to Cloudera Management Console CDP Operational Database your Database or Environment .
- b) Copy the Cloud Storage Location.
 - In the case of AWS, this is a s3a path: s3a://bucket/path/hbase
 - In the case of Azure blob storage (ABFS), this is an abfs path: abfs://<storagename>/datalake/<datahub_name>/hbase
- c) Run the following command in the hbase shell on the source cluster to export a snapshot from the source cluster to the destination cluster:

```
hbase org.apache.hadoop.hbase.snapshot.ExportSnapshot -snaps
hot [***SNAPSHOT NAME***] -copy-to [***DESTINATION***] -mappers 16
```

Replace [***DESTINATION***] with the s3a or abfs path obtained in *step b*.



Important: Snapshots must be enabled on the source and destination clusters. When you export a snapshot, the table's HFiles, logs, and the snapshot metadata are copied from the source cluster to the destination cluster.

It may be necessary to increase the value of the hbase.master.hfilecleaner.ttl property in the source cluster to work around known issues in HBase file cleaning due to known issues in these legacy products.

11. Verify that the snapshot exist in the COD database using the following command:

```
$ hbase shell
hbase> list_snapshots
```

12. Restore the snapshot into an HBase table using the following command:

```
$. /bin/hbase shell
hbase> restore_snapshot [***SNAPSHOT_NAME***]
```

13. Enable the peer on the source cluster to begin replicating data:

```
$. /bin/hbase shell
hbase> enable_peer '1'
```

14. Use the HashTable/CldrSyncTable tool to ensure that data is synchronized between your source and destination clusters.

- a) Run HashTable on the source cluster:

```
hbase org.apache.hadoop.hbase.mapreduce.HashTable [***TABLE
NAME***] [***HASH OUTPUT PATH***]
hdfs dfs -ls -R [***HASH OUTPUT PATH***]
```

Hashed indexes are generated on the source cluster.

- b) Run CldrSyncTable with the -cldr.cross.domain option on the source cluster:

```
hbase org.apache.hadoop.hbase.mapreduce.CldrSyncTable --cldr.cross.domain
--targetzkcluster=[***TARGET ZOOKEEPER QUORUM***]:[***TARGET ZOOKEEPER
PORT***]:[***TARGET ZOOKEEPER ROOT FOR HBASE***]
[***HASH OUTPUT PATH***] [***SOURCE TABLE NAME***] [***TARGET TABLE NAME
ON THE TARGET CLUSTER***]
```

Hash indexes are generated on the destination cluster and compared to the hash indexes generated on the source cluster. If the hash indexes are not identical, synchronization is run for that section of rows.

Related Information

[Using HBase replication policies](#)

[Migrating HBase tables](#)

Replication Manager

Cloudera Replication Manager is a key CDP service to migrate data between CDP environments, on-prem clusters, and clusters on cloud infrastructure services. Replication Manager provides a simple, easy-to-use, and feature-rich data movement capability to deliver data and metadata where it is needed and provides secure data backup with disaster recovery functionality.

Replication Manager provides HBase replication policies to replicate HBase data from source classic clusters to COD.

The following table lists the minimum versions that HBase replication policies support to replicate HBase data.

Table 2: Minimum supported versions for HBase replication policies

Lowest Supported Source Cloudera Manager Version	Lowest Supported Source CDH/CDP Version	Source Cluster Type	Lowest Supported Target Cloudera Manager Version	Lowest Supported Target CDP Version	Target Cluster Type
7.3.1	7.1.6	CDP Private Cloud Base	7.6.0	7.2.14	COD

Lowest Supported Source Cloudera Manager Version	Lowest Supported Source CDH/CDP Version	Source Cluster Type	Lowest Supported Target Cloudera Manager Version	Lowest Supported Target CDP Version	Target Cluster Type
7.3.1	6.3.3	CDH*	7.6.0	7.2.14	COD
7.4.4 (patch-5017)	5.16.2	CDH	-	7.2.14	COD
-	7.2.14	COD	-	7.2.14	COD
*Replication Manager supports HBase replication from non-kerberized source clusters only for CDH 6.3.x versions.					

Related Information

[HBase Replication Policy](#)

[Creating HBase Replication Policy](#)

Phoenix Replication to Cloudera Operational Database

Cloudera Operational Database (COD) provides both HBase and Phoenix. You might request a backported SchemaTool utility for HDP 2.6, HDP3, and Phoenix parcel for CDH6 from Cloudera support to reconstruct the original DDL.

Cloudera also provides the Cloudera Operational Database Replication plugin, which enables HBase replication from a number of products which also include HBase and Phoenix to COD, such as CDH 5, CDH 6, HDP 2.6, and HDP 3.1. You can replicate Phoenix tables to COD using the Replication plugin.

Currently, COD includes Apache Phoenix 5.1.1 while other products include a range of versions of Phoenix from 4.7.0 to 5.0.0.

Replicating Phoenix Data Tables

When you want to configure replication of Phoenix tables from all versions, you need to know the schema of the data to be replicated. However, all on-prem products which include Phoenix today do not have the ability to print the equivalent `CREATE TABLE` command for a table in Phoenix.

Cloudera has built some functionality to recreate the `CREATE TABLE` command for previously released versions of Phoenix as a standalone tool. Please contact your Cloudera representative to evaluate the ability to use this utility against your version of Phoenix.

Replicating Phoenix tables for versions lower than 4.14

For Phoenix versions lower than 4.14, you must treat both the Phoenix data tables and Phoenix index tables similar to any other HBase table.

Procedure

1. Copy the contents of your Phoenix data table in the source cluster to COD in the same manner that you would do for an HBase table (either using CDP Replication Manager or through the HBase ExportSnapshot utility). To copy the contents of the Phoenix data table, see [HBase migration steps 1-11](#).



Note: Ensure the name of the HBase table that you provide in snapshot operations is the same as your Phoenix table name. When creating the Phoenix table, ensure it appears within the quotation marks. Otherwise, it appears in capital letters.

```
hbase> disable 'myTable'
hbase> restore_snapshot 'myTableSnapshot-122112'
```

2. Validate that the data is readable by scanning the restored table in HBase through hbase shell before proceeding further.

3. Verify if the corresponding HBase table for your Phoenix data table exists in COD.
4. Obtain the DDL statement (CREATE TABLE) that you have used to create the table in your legacy system.
5. Run the CREATE TABLE command in phoenix-sqlline in COD with the option COLUMN_ENCODED_BYTES=0 appended to it.

```
For example, if the original create table statement is:
jdbc:phoenix> CREATE TABLE MY_DATA(rk VARCHAR NOT NULL PRIMARY KEY, col1
INTEGER) SALT_BUCKETS = 4;
The corresponding statement to run in COD will be:
jdbc:phoenix> CREATE TABLE MY_DATA(rk VARCHAR NOT NULL PRIMARY KEY, col1
INTEGER) SALT_BUCKETS = 4,
COLUMN_ENCODED_BYTES=0;
```

This option disables the [column encoding feature](#), which is enabled by default since Phoenix versions 4.14 till 5.0. This feature uses binary representations in the HBase column qualifier rather than the column name provided to Phoenix. If the data in the Phoenix table does not match this configuration, Phoenix does not display any data on query but the data appears if queried through the HBase APIs. Running a create table command when an HBase table already exists creates the corresponding internal Phoenix metadata while leaving all other data in place.

6. Validate that query using phoenix-sqlline. This must return the expected data from your Phoenix table.

Replicating Phoenix 4.14 and newer versions including 5.x

Phoenix 4.14 and 5.x have options which may or may not use column encoding, whereas the older versions did not have this feature. You must determine if your existing Phoenix table uses column encoding. Following are the various ways to check this.

About this task

You can also create an HBase replication policy using Replication Manager to replicate Phoenix 4.14 and 5.x versions into COD. For more information, refer to the second row of the *Replication methods* table in *HBase Migration through Replication Manager*.

Procedure

1. Follow the same steps as described for Phoenix older than 4.14, [steps 1 - 4](#), and modify [step 5](#) according to the following instructions.
2. Scan the raw data in HBase, and inspect if there are human-readable column names in the column qualifier. If you see human-readable names, you must set COLUMN_ENCODED_BYTES = 0. If you encounter a binary data ('\x00'), assume that COLUMN_ENCODED_BYTES must be set.
3. Inspect the create-table command from the legacy product. If you explicitly set COLUMN_ENCODED_BYTES in an earlier version, you must use the same value when re-creating the Phoenix table in COD.



Note: Consult the SchemaTool output if you are on a sufficiently new version of Phoenix.

4. Validate the query using phoenix-sqlline. This must return the expected data from your Phoenix table.

Related Information

[HBase Replication Policy](#)

[Creating HBase Replication Policy](#)

[HBase Migration through Replication Manager](#)

Replicating Phoenix Index Tables

The secondary indexing implementation in Phoenix changed drastically between versions lower than Phoenix 4.14 and higher. The manner in which HBase replication sends Phoenix data is fundamentally incompatible with the new style of Phoenix secondary index maintenance. Architecturally, it is most desirable that we replicate only

Phoenix data tables, and let the “local” HBase cluster maintain any secondary indexes for you. However, currently no “Phoenix-aware” replication strategy exists which is capable of automatically maintaining the Phoenix secondary indexes for replicated Phoenix data tables.

About this task

Follow the recommended steps in *Replicating Phoenix Data Tables* to replicate all Phoenix data tables before proceeding. For each index table that you want to replicate, perform the following steps:

1. Create the corresponding index in COD. Be sure to specify the same exact create index command that was used in the legacy product (both indexes columns and “include” columns). If you have a significant amount of data in the corresponding data table, include the `async` keyword in the create index command. The `async` keyword will create the table but not populate any data into this table.

```
jdbc:phoenix> CREATE INDEX SMALL_INDEX ON SMALL_TABLE(col1) INCLUDE (col2);
jdbc:phoenix> CREATE INDEX BIG_INDEX ON BIG_TABLE(col1) INCLUDE (col2) ASYNC;
```

2. Enable replication on the Phoenix index table in the source cluster by using the “alter” command in the HBase shell like enabling replication for HBase and Phoenix Data tables.

```
hbase> alter "BIG_INDEX", {NAME=>"0", REPLICATION_SCOPE=>1}
```

3. If you used the `ASYNC` keyword in the create index in COD, now you have to use `IndexTool` to build the index. This will launch a MapReduce job to build in the index in parallel, rather than synchronously from a single client as is normally done. An example `IndexTool` invocation would be done on the command line for the COD cluster (using the HBase client tarball) like the following:

```
$ hbase org.apache.phoenix.mapreduce.index.IndexTool --data-table BIG_TABLE --index-table BIG_INDEX --output-path /tmp/BIG_INDEX-files
```

4. Ensure the MapReduce job launched by this command completes before proceeding.
5. Validate the contents of the index table, as the index table is used for queries. For example, you can read a few rows from the index table directly:

```
jdbc:phoenix> SELECT * FROM BIG_INDEX LIMIT 10;
```

Alternatively, you could look at the explain plan for a query that uses the index (noting that `BIG_INDEX` is used instead of `BIG_TABLE` in this example)

```
jdbc:phoenix> EXPLAIN SELECT col1, col2 FROM BIG_TABLE;
```

Related Information

[Replicating Phoenix Data Tables](#)