Cloudera Runtime 7.2.9

# Using HBase Backup and Disaster Recovery

**Date published: 2020-02-29**
**Date modified: 2021-04-28**

# 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 backup and disaster recovery strategies.....4**

**Configuring HBase snapshots.....4**
About HBase snapshots.....4
Configure snapshots.....6
Manage HBase snapshots using Cloudera Manager.....6
Browse HBase tables.....7
Take HBase snapshots.....7
Store HBase snapshots on Amazon S3.....7
Configure HBase in Cloudera Manager to store snapshots in Amazon S3.....7
Configure the dynamic resource pool used for exporting and importing snapshots in Amazon S3.....8
HBase snapshots on Amazon S3 with Kerberos enabled.....9
Manage HBase snapshots on Amazon S3 in Cloudera Manager.....9
Delete HBase snapshots from Amazon S3.....9
Restore an HBase snapshot from Amazon S3.....10
Restore an HBase snapshot from Amazon S3 with a new name.....10
Manage Policies for HBase snapshots in Amazon S3.....10
Manage HBase snapshots using the HBase shell.....10
Shell commands.....11
Take a snapshot using a shell script.....11
Export a snapshot to another cluster.....12
Snapshot failures.....13
Information and debugging.....13

**Using HBase replication.....15**
Common replication topologies.....15
Notes about replication.....15
Replication requirements.....16
Deploy HBase replication.....16
Replication across three or more clusters.....17
Enable replication on a specific table.....17
Configure secure replication.....18
Configure bulk load replication.....20
Enable bulk load replication using Cloudera Manager.....20
Create empty table on the destination cluster.....22
Disable replication at the peer level.....22
Stop replication in an emergency.....23
Initiate replication when data already exist.....23
Replicate pre-exist data in an active-active deployment.....23
Effects of WAL rolling on replication.....24
Configure secure HBase replication.....25
Restore data from a replica.....26
Verify that replication works.....27
Replication caveats.....28

# HBase backup and disaster recovery strategies

Backup-and-restore is a standard set of operations for many databases. You must have an effective backup-and-restore strategy to ensure that you can recover data in case of data loss or failures.

HBase snapshot enables you to take a snapshot of a table without much impact on RegionServers, because snapshot, clone, and restore operations do not involve data copying. In addition, exporting a snapshot to another cluster has no impact on RegionServers.

If your data is already in an HBase cluster, replication is useful for getting the data into additional HBase clusters.

# Configuring HBase snapshots

HBase snapshot support enables you to take a snapshot of a table without much impact on RegionServers, because snapshot, clone, and restore operations do not involve data copying. In addition, exporting a snapshot to another cluster has no impact on RegionServers.

## About HBase snapshots

HBase snapshots allow you to clone a table without making data copies, and with minimal impact on RegionServers. Exporting the table to another cluster does not have any impact on the RegionServers.

In previous HBase releases, the only way to a back up or to clone a table was to use CopyTable or ExportTable, or to copy all the hfiles in HDFS after disabling the table. These methods have disadvantages:

- CopyTable and ExportTable can degrade RegionServer performance.
- Disabling the table means no reads or writes; this is usually unacceptable.

HBase snapshots allow you to clone a table without making data copies, and with minimal impact on RegionServers. Exporting the table to another cluster does not have any impact on the RegionServers.

### Use Cases

- Recovery from user or application errors
  - Useful because it may be some time before the database administrator notices the error.

    **Note:**

    The database administrator needs to schedule the intervals at which to take and delete snapshots. Use a script or management tool; HBase does not have this functionality.
  - The database administrator may want to save a snapshot before a major application upgrade or change.

    **Note:**

    Snapshots are not primarily used for system upgrade protection because they do not roll back binaries, and would not necessarily prevent bugs or errors in the system or the upgrade.
  - Recovery cases:
    - Roll back to previous snapshot and merge in reverted data.
    - View previous snapshots and selectively merge them into production.

- Backup
  - Capture a copy of the database and store it outside HBase for disaster recovery.
  - Capture previous versions of data for compliance, regulation, and archiving.
  - Export from a snapshot on a live system provides a more consistent view of HBase than CopyTable and ExportTable.
- Audit or report view of data at a specific time
  - Capture monthly data for compliance.
  - Use for end-of-day/month/quarter reports.
- Application testing
  - Test schema or application changes on similar production data from a snapshot and then discard.

    For example:

    1. Take a snapshot.
    2. Create a new table from the snapshot content (schema and data)
    3. Manipulate the new table by changing the schema, adding and removing rows, and so on. The original table, the snapshot, and the new table remain independent of each other.
- Offload work
  - Capture, copy, and restore data to another site
  - Export data to another cluster

### Where Snapshots Are Stored

Snapshot metadata is stored in the .hbase_snapshot directory under the hbase root directory (/hbase/.hbase-snapshot). Each snapshot has its own directory that includes all the references to the hfiles, logs, and metadata needed to restore the table.

hfiles required by the snapshot are in the /hbase/data/<namespace>/<tableName>/<regionName>/<familyName>/ location if the table is still using them; otherwise, they are in /hbase/.archive/<namespace>/<tableName>/<regionName>/<familyName>/.

### Zero-Copy Restore and Clone Table

From a snapshot, you can create a new table (clone operation) or restore the original table. These two operations do not involve data copies; instead, a link is created to point to the original hfiles.

Changes to a cloned or restored table do not affect the snapshot or (in case of a clone) the original table.

To clone a table to another cluster, you export the snapshot to the other cluster and then run the clone operation; see *Exporting a snapshot to another cluster*.

### Reverting to a Previous HBase Version

Snapshots do not affect HBase backward compatibility if they are not used.

If you use snapshots, backward compatibility is affected as follows:

- If you only take snapshots, you can still revert to a previous HBase version.
- If you use restore or clone, you cannot revert to a previous version unless the cloned or restored tables have no links. Links cannot be detected automatically; you would need to inspect the file system manually.

### Storage Considerations

Because hfiles are immutable, a snapshot consists of a reference to the files in the table at the moment the snapshot is taken. No copies of the data are made during the snapshot operation, but copies may be made when a compaction or deletion is triggered. In this case, if a snapshot has a reference to the files to be removed, the files are moved to an archive folder, instead of being deleted. This allows the snapshot to be restored in full.

Because no copies are performed, multiple snapshots share the same hfiles, butfor tables with lots of updates, and compactions, each snapshot could have a different set of hfiles.

## Configure snapshots

Snapshots are enabled by default, you must disable snapshots if you do not intend to use it. Snapshots do not affect HBase performance if they are not used.

Snapshots are on by default; to disable them, set the hbase.snapshot.enabled property in hbase-site.xml to false:

```
<property>
   <name>hbase.snapshot.enabled</name>
   <value>
      false
   </value>
</property>
```

To enable snapshots after you have disabled them, set hbase.snapshot.enabled to true.

**Note:**

If you have taken snapshots and then decide to disable snapshots, you must delete the snapshots before restarting the HBase master; the HBase master will not start if snapshots are disabled and snapshots exist.

## Manage HBase snapshots using Cloudera Manager

For HBase services, you can use the Table Browser tab to view the HBase tables associated with a service on your cluster. You can view the currently saved snapshots for your tables, and delete or restore them.

From the HBase Table Browser tab, you can:

- View the HBase tables for which you can take snapshots.
- Initiate immediate (unscheduled) snapshots of a table.
- View the list of saved snapshots currently maintained. These can include one-off immediate snapshots, as well as scheduled policy-based snapshots.
- Delete a saved snapshot.
- Restore from a saved snapshot.
- Restore a table from a saved snapshot to a new table (Restore As).

### Browsing HBase Tables

To browse the HBase tables to view snapshot activity:

1. From the Clusters tab, select your HBase service.
2. Go to the Table Browser tab.

### Managing HBase Snapshots

To take a snapshot:

1. Click a table.
2. Click Take Snapshot.
3. Specify the name of the snapshot, and click Take Snapshot.

To delete a snapshot, click and select Delete.

To restore a snapshot, click and select Restore.

**Warning:** If you use coprocessors, the coprocessor must be available on the destination cluster before restoring the snapshot.

To restore a snapshot to a new table, select Restore As from the menu associated with the snapshot, and provide a name for the new table.

**Warning:** If you "Restore As" to an existing table (that is, specify a table name that already exists), the existing table will be overwritten.

## Browse HBase tables

You can view the snapshot activity using Cloudera Manager.

### About this task

To browse the HBase tables to view snapshot activity:

### Procedure

1. From the Clusters tab, select your HBase service.
2. Go to the Table Browser tab.

## Take HBase snapshots

You can take a snapshot at any time using Cloudera Manager.

### About this task

To take a snapshot:

### Procedure

1. Click a table.
2. Click Take Snapshot.
3. Specify the name of the snapshot, and click Take Snapshot.

### Related Information

Notes about replication

# Store HBase snapshots on Amazon S3

HBase snapshots can be stored on the cloud storage service Amazon S3.

**Important:** When HBase snapshots are stored on, or restored from, Amazon S3, a MapReduce (MRv2) job is created to copy the HBase table data and metadata. The YARN service must be running on your Cloudera Manager cluster to use this feature.

To configure HBase to store snapshots on Amazon S3, you must have the following information:

- The access key ID for your Amazon S3 account.
- The secret access key for your Amazon S3 account.
- The path to the directory in Amazon S3 where you want your HBase snapshots to be stored.

You can improve the transfer of large snapshots to Amazon S3 by increasing the number of nodes due to throughput limitations of EC2 on a per node basis.

## Configure HBase in Cloudera Manager to store snapshots in Amazon S3

You must configure HBase before you can store snapshots on Amazon S3.

### About this task

Perform the following steps in Cloudera Manager

### Procedure

1. Open the HBase service page.
2. Scope HBASE (Service-Wide)
3. Select Category Backup .
4. Type S3 in the Search box.
5. Enter your Amazon S3 access key ID in the field AWS S3 access key ID for remote snapshots.
6. Enter your Amazon S3 secret access key in the field AWS S3 secret access key for remote snapshots.

   **Important:** If Amazon S3 access keys are rotated, the Cloudera Manager server must be restarted.

7. Enter the path to the location in Amazon S3 where your HBase snapshots will be stored in the field Amazon S3 Path for Remote Snapshots.

   **Warning:** Do not use the Amazon S3 location defined by the path entered in Amazon S3 Path for Remote Snapshots for any other purpose, or directly add or delete content there. Doing so risks corrupting the metadata associated with the HBase snapshots stored there. Use this path and Amazon S3 location only through Cloudera Manager, and only for managing HBase snapshots.

8. In a terminal window, log in to your Cloudera Manager cluster at the command line and create a /user/hbase directory in HDFS. Change the owner of the directory to hbase.

   For example:

   ```
   hdfs dfs -mkdir /user/hbase
   hdfs dfs -chown hbase /user/hbase
   ```

   **Note:** Amazon S3 has default rate limitation per prefix per bucket. The throughput can be limited to 3500 requests per second. Consider to use different prefixes on S3 per table namespace, or table if any of the following applies:

   - large number of tables
   - tables with a large number of store files or regions
   - frequent snapshot policy

## Configure the dynamic resource pool used for exporting and importing snapshots in Amazon S3

Dynamic resource pools are used to control the resources available for MapReduce jobs created for HBase snapshots on Amazon S3. By default, MapReduce jobs run against the default dynamic resource pool.

### About this task

To choose a different dynamic resource pool for HBase snapshots stored on Amazon S3, follow these steps:

### Procedure

1. Open the HBase service page.
2. Select Scope HBASE (Service-Wide) .
3. Select Category Backup .
4. Type Scheduler in the Search box.
5. Enter name of a dynamic resource pool in the Scheduler pool for remote snapshots in AWS S3 property.
6. Click Save Changes.

## HBase snapshots on Amazon S3 with Kerberos enabled

YARN should by default allow the hbase user to run MapReduce jobs even when Kerberos is enabled.

### About this task

If Kerberos is enabled on your cluster, and YARN does not allow the hbase user to run MapReduce jobs, perform the following steps

### Procedure

1. Open the YARN service page in Cloudera Manager.
2. Select  Scope NodeManager .
3. Select  Category Security .
4. In the Allowed System Users property, click the + sign and add hbase to the list of allowed system users.
5. Click Save Changes.
6. Restart the YARN service.

## Manage HBase snapshots on Amazon S3 in Cloudera Manager

You can manage your snaphots stored on Amazon S3 using Cloudera Manager.

### About this task

To take HBase snapshots and store them on Amazon S3, perform the following steps

### Procedure

1. On the HBase service page in Cloudera Manager, click the Table Browser tab.
2. Select a table in the Table Browser. If any recent local or remote snapshots already exist, they display on the right side.
3. In the dropdown for the selected table, click Take Snapshot.
4. Enter a name in the Snapshot Name field of the Take Snapshot dialog box.
5. If Amazon S3 storage is configured, the Take Snapshot dialog box Destination section shows a choice of Local or Remote S3. Select Remote S3.
6. Click Take Snapshot.

   While the Take Snapshot command is running, a local copy of the snapshot with a name beginning cm-tmp followed by an auto-generated filename is displayed in the Table Browser. This local copy is deleted as soon as the remote snapshot has been stored in Amazon S3. If the command fails without being completed, the temporary local snapshot might not be deleted. This copy can be manually deleted or kept as a valid local snapshot. To store a current snapshot in Amazon S3, run the Take Snapshot command again, selecting Remote S3 as the Destination, or use the HBase command-line tools to manually export the existing temporary local snapshot to Amazon S3.

## Delete HBase snapshots from Amazon S3

You can delete a snapshot stored on Amazon S3 using Cloudera Manager.

### About this task

To delete a snapshot stored on Amazon S3

### Procedure

1. Select the snapshot in the Table Browser.
2. Click the dropdown arrow for the snapshot.
3. Click Delete.

## Restore an HBase snapshot from Amazon S3

You can restore a snapshot that you have stored on Amazon S3.

### About this task

To restore a snapshot stored on Amazon S3

### Procedure

1. Select the table in the Table Browser.
2. Click Restore Table.
3. Choose Remote S3 and select the table to restore.
4. Click Restore.

   Cloudera Manager creates a local copy of the remote snapshot with a name beginning with cm-tmp followed by an auto-generated filename, and uses that local copy to restore the table in HBase. Cloudera Manager then automatically deletes the local copy. If the Restore command fails without completing, the temporary copy might not be deleted and can be seen in the Table Browser. In that case, delete the local temporary copy manually and re-run the Restore command to restore the table from Amazon S3.

## Restore an HBase snapshot from Amazon S3 with a new name

By restoring an HBase snapshot stored in Amazon S3 with a new name, you clone the table without affecting the existing table in HBase.

### About this task

To clone the table without affecting the existing table in HBase, perform the following steps

### Procedure

1. Select the table in the Table Browser.
2. Click Restore Table From Snapshot As.
3. In the Restore As dialog box, enter a new name for the table in the Restore As field.
4. Select Remote S3 and choose the snapshot in the list of available Amazon S3 snapshots.

## Manage Policies for HBase snapshots in Amazon S3

You can configure policies to automatically create snapshots of HBase tables on an hourly, daily, weekly, monthly or yearly basis. Snapshot policies for HBase snapshots stored in Amazon S3 are configured using the same procedures as for local HBase snapshots.

See the Cloudera Manager snapshot polices for more information. For snapshots stored in Amazon S3, you must also choose Remote S3 in the Destination section of the policy management dialog boxes.

**Note:** You can only configure a policy as Local or Remote S3 at the time the policy is created and cannot change the setting later. If the setting is wrong, create a new policy.

When you create a snapshot based on a snapshot policy, a local copy of the snapshot is created with a name beginning with cm-auto followed by an auto-generated filename. The temporary copy of the snapshot is displayed in the Table Browser and is deleted as soon as the remote snapshot has been stored in Amazon S3. If the snapshot procedure fails without being completed, the temporary local snapshot might not be deleted. This copy can be manually deleted or kept as a valid local snapshot. To export the HBase snapshot to Amazon S3, use the HBase command-line tools to manually export the existing temporary local snapshot to Amazon S3.

# Manage HBase snapshots using the HBase shell

You can manage snapshots by using the HBase shell, or you can use a shell script.

## Shell commands

The following table shows actions you can take from the shell.

| Action | Shell command | Comments |
|---|---|---|
| Take a snapshot of tableX called snap shotX | `snapshot 'tableX', 'snapshotX'` | Snapshots can be taken while a table is disabled, or while a table is online and serving traffic.<br>• If a table is disabled (using disable <table>), an offline snapshot is taken. This snapshot is managed by the master and fully consistent with the state when the table was disabled. This is the simplest and safest method, but it involves a service interruption because the table must be disabled to take the snapshot.<br>• In an online snapshot, the table remains available while the snapshot is taken, and incurs minimal performance degradation of normal read/write loads. This snapshot is managed by the master and run on the RegionServers. The current implementation—simple-flush snapshots—provides no causal consistency guarantees. Despite this shortcoming, it offers the same degree of consistency as CopyTable and is a significant improvement. |
| Restore snapshot snapshotX (replaces the source table content) | `restore_snapshot 'snapshotX'` | For emergency use only.<br>Restoring a snapshot replaces the current version of a table with different version. To run this command, you must disable the target table. The restore command takes a snapshot of the table (appending a timestamp code), and then clones data into the original data and removes data not in the snapshot. If the operation succeeds, the target table is enabled. |
| List all available snapshots | `list_snapshots` | |
| List all available snapshots starting with 'mysnapshot_' (regular expression) | `list_snapshots 'my_snapshot_.*'` | |
| Remove a snapshot called snapshotX | `delete_snapshot 'snapshotX'` | |
| Create a new table tableY from a snapshot snapshotX | `clone_snapshot 'snapshotX', 'tableY'` | Cloning a snapshot creates a new read/write table that serves the data kept at the time of the snapshot. The original table and the cloned table can be modified independently; new data written to one table does not show up on the other. |

## Take a snapshot using a shell script

You can take a snapshot using an operating system shell script, such as a Bash script, in HBase Shell noninteractive mode.

This example Bash script shows how to take a snapshot in this way. This script is provided as an illustration only; do not use in production.

```
#!/bin/bash
# Take a snapshot of the table passed as an argument
# Usage: snapshot_script.sh table_name
# Names the snapshot in the format snapshot-YYYYMMDD

# Parse the arguments
if [ -z $1 ]||[$1 == '-h' ]; then
```

```
 echo "Usage: $0 &lt;table&gt;"
 echo "       $0 -h"
 exit 1
fi

# Modify to suit your environment
export HBASE_PATH=/home/user/hbase
export DATE=`date +"%Y%m%d"`
echo "snapshot '$1', 'snapshot-$DATE'" | $HBASE_PATH/bin/hbase shell -n
status=$?
if [$status -ne 0]; then
 echo "Snapshot may have failed: $status"
fi
exit $status
```

HBase Shell returns an exit code of 0 on successA non-zero exit code indicates the possibility of failure, not a definite failure. Your script should check to see if the snapshot was created before taking the snapshot again, in the event of a reported failure.

## Export a snapshot to another cluster

You can export any snapshot from one cluster to another. Exporting the snapshot copies the table's hfiles, logs, and the snapshot metadata, from the source cluster to the destination cluster.

Specify the -copy-from option to copy from a remote cluster to the local cluster or another remote cluster. If you do not specify the -copy-from option, the hbase.rootdir in the HBase configuration is used, which means that you are exporting from the current cluster. You must specify the -copy-to option, to specify the destination cluster.

**Note:** Snapshots must be enabled on the destination cluster.

The ExportSnapshot tool executes a MapReduce Job similar to distcp to copy files to the other cluster. It works at file-system level, so the HBase cluster can be offline.

Run ExportSnapshot as the hbase user or the user that owns the files. If the user, group, or permissions need to be different on the destination cluster than the source cluster, use the -chuser, -chgroup, or -chmod options as in the second example below, or be sure the destination directory has the correct permissions. In the following examples, replace the HDFS server path and port with the appropriate ones for your cluster.

To copy a snapshot called MySnapshot to an HBase cluster srv2 (hdfs://srv2:8020/hbase) using 16 mappers:

```
hbase org.apache.hadoop.hbase.snapshot.ExportSnapshot -snapshot MySnapshot -
copy-to hdfs://srv2:<hdfs_port>/hbase -mappers 16
```

To export the snapshot and change the ownership of the files during the copy:

```
hbase org.apache.hadoop.hbase.snapshot.ExportSnapshot -snapshot MySnapshot -
copy-to hdfs://srv2:<hdfs_port>/hbase -chuser MyUser -chgroup MyGroup -chmod
 700 -mappers 16
```

You can also use the Java -D option in many tools to specify MapReduce or other configuration properties. For example, the following command copies MY_SNAPSHOT to hdfs://cluster2/hbase using groups of 10 hfiles per mapper:

```
hbase org.apache.hadoop.hbase.snapshot.ExportSnapshot -Dsnapshot.export.defa
ult.map.group=10 -snapshot MY_SNAPSHOT -copy-to hdfs://cluster2/hbase
```

(The number of mappers is calculated as TotalNumberOfHFiles/10.)

To export from one remote cluster to another remote cluster, specify both -copy-from and -copy-to parameters.

You can then reverse the direction to restore the snapshot back to the first remote cluster.

```
hbase org.apache.hadoop.hbase.snapshot.ExportSnapshot -snapshot snapshot-tes
t -copy-from hdfs://machine1/hbase -copy-to hdfs://machine2/my-backup
```

To specify a different name for the snapshot on the target cluster, use the -target option.

```
hbase org.apache.hadoop.hbase.snapshot.ExportSnapshot -snapshot snapshot-tes
t -copy-from hdfs://machine1/hbase -copy-to hdfs://machine2/my-backup -targe
t new-snapshot
```

## Snapshot failures

Region moves, splits, and other metadata actions that happen while a snapshot is in progress can cause the snapshot to fail. The software detects and rejects corrupted snapshot attempts.

### Information and debugging

You can use the SnapshotInfo tool to get information about a snapshot, including status, files, disk usage, and debugging information.

Examples:

Use the -h option to print usage instructions for the SnapshotInfo utility.

```
$ hbase org.apache.hadoop.hbase.snapshot.SnapshotInfo -h
Usage: bin/hbase org.apache.hadoop.hbase.snapshot.SnapshotInfo [options]
 where [options] are:
  -h|-help                Show this help and exit.
  -remote-dir             Root directory that contains the snapshots.
  -list-snapshots         List all the available snapshots and exit.
  -snapshot NAME          Snapshot to examine.
  -files                  Files and logs list.
  -stats                  Files and logs stats.
  -schema                 Describe the snapshotted table.
```

Use the -list-snapshots option to list all snapshots and exit.

```
$ hbase org.apache.hadoop.hbase.snapshot.SnapshotInfo -list-snapshots
SNAPSHOT             | CREATION TIME        | TABLE NAME
snapshot-test        |  2014-06-24T19:02:54 | test
```

Use the -remote-dir option with the -list-snapshots option to list snapshots located on a remote system.

```
$ hbase org.apache.hadoop.hbase.snapshot.SnapshotInfo -remote-dir s3a://mybu
cket/mysnapshot-dir -list-snapshots
SNAPSHOT |   CREATION TIME   | TABLE NAME
snapshot-test        2014-05-01 10:30    myTable
```

Use the -snapshot option to print information about a specific snapshot.

```
$ hbase org.apache.hadoop.hbase.snapshot.SnapshotInfo -snapshot test-snapsho
t
Snapshot Info
----------------------------------------
   Name: test-snapshot
   Type: DISABLED
  Table: test-table
Version: 0
Created: 2012-12-30T11:21:21
```

```
**************************************************************
```

Use the -snapshot with the -stats options to display additional statistics about a snapshot.

```
$ hbase org.apache.hadoop.hbase.snapshot.SnapshotInfo -stats -snapshot snaps
hot-test
Snapshot Info
----------------------------------------
   Name: snapshot-test
   Type: FLUSH
  Table: test
 Format: 0
Created: 2014-06-24T19:02:54

1 HFiles (0 in archive), total size 1.0k (100.00% 1.0k shared with the sou
rce table)
```

Use the -schema option with the -snapshot option to display the schema of a snapshot.

```
$ hbase org.apache.hadoop.hbase.snapshot.SnapshotInfo  -schema -snapshot sna
pshot-test
Snapshot Info
----------------------------------------
   Name: snapshot-test
   Type: FLUSH
  Table: test
 Format: 0
Created: 2014-06-24T19:02:54

Table Descriptor
----------------------------------------
'test', {NAME => 'cf', DATA_BLOCK_ENCODING => 'FAST_DIFF', BLOOMFILTER => '
ROW', REPLICATION_SCOPE => '0',
COMPRESSION => 'GZ', VERSIONS => '1', TTL => 'FOREVER', MIN_VERSIONS => '0',
 KEEP_DELETED_CELLS => 'false',
BLOCKSIZE => '65536', IN_MEMORY => 'false', BLOCKCACHE => 'true'}
```

Use the -files option with the -snapshot option to list information about files contained in a snapshot.

```
$ hbase org.apache.hadoop.hbase.snapshot.SnapshotInfo -snapshot test-snapsho
t -files
Snapshot Info
----------------------------------------
   Name: test-snapshot
   Type: DISABLED
  Table: test-table
Version: 0
Created: 2012-12-30T11:21:21

Snapshot Files
----------------------------------------
   52.4k test-table/02ba3a0f8964669520cf96bb4e314c60/cf/bdf29c39da2a4f2b81
889eb4f7b18107 (archive)
   52.4k test-table/02ba3a0f8964669520cf96bb4e314c60/cf/1e06029d0a2a4a70905
1b417aec88291 (archive)
   86.8k test-table/02ba3a0f8964669520cf96bb4e314c60/cf/506f601e14dc4c74a058
be5843b99577 (archive)
   52.4k test-table/02ba3a0f8964669520cf96bb4e314c60/cf/5c7f6916ab724eacb
cea218a713941c4 (archive)
  293.4k test-table/02ba3a0f8964669520cf96bb4e314c60/cf/aec5e33a6564441d9b
d423e31fc93abb (archive)
```

```
   52.4k test-table/02ba3a0f8964669520cf96bb4e314c60/cf/97782b2fbf0743edaac
d8fef06ba51e4 (archive)
6 HFiles (6 in archive), total size 589.7k (0.00% 0.0 shared with the source
 table)
0 Logs, total size 0.0
```

# Using HBase replication

If your data is already in an HBase cluster, replication is useful for getting the data into additional HBase clusters.

## Common replication topologies

You must learn about the common replication topologies before you configure and deploy HBase replication.

Some common replication topologies are:

- A central source cluster might propagate changes to multiple destination clusters, for failover or due to geographic distribution.
- A source cluster might push changes to a destination cluster, which might also push its own changes back to the original cluster.
- Many different low-latency clusters might push changes to one centralized cluster for backup or resource-intensive data-analytics jobs. The processed data might then be replicated back to the low-latency clusters.
- Multiple levels of replication can be chained together to suit your needs.

### Related Information

Apache documentation: Cluster replication

Replication caveats

## Notes about replication

You must consider a few things before using HBase replication.

- The timestamps of the replicated HLog entries are kept intact. In case of a collision (two entries identical as to row key, column family, column qualifier, and timestamp) only the entry arriving later write will be read.
- Increment Column Values (ICVs) are treated as simple puts when they are replicated. In the case where each side of replication is active (new data originates from both sources, which then replicate each other), this may be undesirable, creating identical counters that overwrite one another. Therefore Cloudera recommends that different cluster do not write increments to the same coordinates. For more information about this issue, see https://issues.apache.org/jira/browse/HBase-2804.
- Make sure the source and destination clusters are time-synchronized with each other. Cloudera recommends you use Network Time Protocol (NTP).
- Some changes are not replicated and must be propagated through other means, such as Snapshots or CopyTable.
  - Data that existed in the active cluster before replication was enabled.
  - Operations that bypass the WAL, such as when using BulkLoad or API calls such as writeToWal(false).
  - Table schema modifications.

### Related Information

Take HBase snapshots

Use CopyTable

## Replication requirements

Before configuring replication, make sure your environment meets all requirements.

- Replication is supported from CDH 5 to CDP Private Cloud Base , and from CDP Private Cloud Base to CDH 5, if the version of CDP Private Cloud Base is 7.1 or higher.
- Each host in both clusters must be able to reach every other host, including those in the ZooKeeper cluster.
- Every table that contains families that are scoped for replication must exist on each cluster and have exactly the same name. If your tables do not yet exist on the destination cluster, see *Create empty table on the destination cluster*.

### Related Information

Create empty table on the destination cluster

Deploy HBase replication

## Deploy HBase replication

Once you have source and destination clusters, you can enable and configure HBase replication to use it as your data import method.

### About this task

**Important:** You need to run this as the HBase superuser, usually called "hbase". To run replication-related HBase commands, you must have HBase user permissions. If ZooKeeper uses Kerberos, configure HBase Shell to authenticate to ZooKeeper using Kerberos before attempting to run replication-related commands. No replication-related ACLs are available at this time.

### Procedure

1. Configure and start the source and destination clusters.
2. Create tables with the same names and column families on both the source and destination clusters, so that the destination cluster knows where to store data it receives. All hosts in the source and destination clusters have to be reachable to each other. see *Create empty table on the destination cluster*.
3. On the source cluster, enable replication in Cloudera Manager, or by setting hbase.replication to true in hbase-site.xml.
4. Obtain Kerberos credentials as the HBase principal. Substitute your fully.qualified.domain.name and realm in the following command:

   ```
   kinit -k -t /etc/hbase/conf/hbase.keytab
   hbase/fully.qualified.domain.name@YOUR-REALM.COM
   ```

5. On the source cluster, in HBase Shell, add the destination cluster as a peer, using the add_peer command.

   ```
   add_peer 'ID', 'CLUSTER_KEY'
   ```

   To compose the CLUSTER_KEY, use the following template:

   ```
   hbase.zookeeper.quorum:hbase.zookeeper.property.clientPort:zookeeper.zno
   de.parent
   ```

   For example:

   ```
   host1.com,host2.com,host3.com:2181:/hbase
   ```

6. On the source cluster, configure each column family to be replicated by setting its REPLICATION_SCOPE to 1, using commands such as the following in HBase Shell:

```
hbase> disable 'example_table'
hbase> alter 'example_table', {NAME => 'example_family', REPLICATION_SCOPE
 => '1'}
hbase> enable 'example_table'
```

7. Verify that replication is occurring by examining the logs on the source cluster for messages such as the following.

```
Considering 1 rs, with ratio 0.1
Getting 1 rs from peer cluster # 0
Choosing peer 192.0.2.49:62020
```

8. To verify the validity of replicated data, use the included VerifyReplication MapReduce job on the source cluster, providing it with the ID of the replication peer and table name to verify. Other options are available, such as a time range or specific families to verify.

   The command has the following form:

```
hbase org.apache.hadoop.hbase.mapreduce.replication.VerifyReplication [-
-starttime=timestamp1] [--stoptime=timestamp] [--families=comma separated
 list of families] <peerId> <tablename>
```

   The VerifyReplication command prints GOODROWS and BADROWS counters to indicate rows that did and did not replicate correctly.

**Related Information**

Replication requirements

Create empty table on the destination cluster

## Replication across three or more clusters

When configuring replication among three or more clusters, Cloudera recommends to enable KEEP_DELETED_CELLS on column families in the destination cluster, where REPLICATION_SCOPE=1 in the source cluster.

The following commands show how to enable this configuration using HBase Shell.

- On the source cluster:

```
create 't1',{NAME=>'f1', REPLICATION_SCOPE=>1}
```

- On the destination cluster:

```
create 't1',{NAME=>'f1', KEEP_DELETED_CELLS=>'true'}
```

## Enable replication on a specific table

You can enable HBase replication for a specific table on the source cluster.

To enable replication for a specific table on the source cluster, run the enable_table_replication <*table*> command from the HBase shell on a cluster where a peer has been configured.

Running enable_table_replication <*table*> does the following:

1. Verifies that the table exists on the source cluster.
2. If the table does not exist on the remote cluster, uses the peer configuration to duplicate the table schema, inclusing splits, on the remote cluster.
3. Enables replication on that table.

# Configure secure replication

Secure replication configuration is the same whether your clusters are all in the same realm or not, with the exception of the last step.

## About this task

The last step involves setting up custom secure replication configurations per peer. This can be convenient when you need to replicate to a cluster that uses different cross-realm authentication rules than the source cluster. For example, a cluster in Realm A may be allowed to replicate to Realm B and Realm C, but Realm B may not be allowed to replicate to Realm C. If you do not need this feature, skip the last step.

To use this feature, service-level principals and keytabs (specific to HBase) must be specified when you create the cluster peers using HBase Shell.

**Important:** Secure HBase replication is not supported if the source or the destination cluster uses custom ZooKeeper server principal. The default ZooKeeper principal is zookeeper.

## Procedure

1. Set up Kerberos on your cluster.

   **Note:** HBase peer-to-peer replication from a non-Kerberized cluster to a Kerberized cluster is not supported.

2. If necessary, configure Kerberos cross-realm authentication:
   a. At the comman line, use the list_principals command to list the kdc, admin_server, and default_domain for each realm.
   b. Add this information to each cluster using Cloudera Manager:
      1. In Cloudera Manager, select the HDFS service.
      2. Click the Configuration tab.
      3. Find the Truster Kerberos Realms property.
      4. Add the target and source.
      5. Restart HDFS.
3. Configure ZooKeeper.
4. Configure the following HDFS parameters on both cluster:

```
<!-- In hdfs-site.xml or advanced configuration snippet -->
<property>
  <name>dfs.encrypt.data.transfer</name>
  <value>true</value>
</property>
<property>
  <name>dfs.data.transfer.protection</name>
  <value>privacy</value>
</property>

<!-- In core-site.xml or advanced configuration snippet -->
<property>
  <name>hadoop.security.authorization</name>
  <value>true</value>
</property>
<property>
  <name>hadoop.rpc.protection</name>
  <value>privacy</value>
</property>
<property>
```

```
  <name>hadoop.security.crypto.cipher.suite</name>
  <value>AES/CTR/NoPadding</value>
</property>
<property>
  <name>hadoop.ssl.enabled</name>
  <value>true</value>
</property>
```

If you use Cloudera Manager to manage your cluster, do not set these properties directly in configuration files, because Cloudera Manager will overwrite or ignore these settings. You must set these properties in Cloudera Manager.

For brevity, the Cloudera Manager setting names are not listed here, but you can search by property name. For instance, in the HDFS service configuration screen, search for dfs.encrypt.data.transfer. The Enable Data Transfer Encryption setting is shown. Selecting the box is equivalent to setting the value to true.

5. Configure the following HBase parameters on both clusters, using Cloudera Manager or in hbase-site.xml if you do not use Cloudera Managert.

```
<!-- In hbase-site.xml -->
<property>
  <name>hbase.rpc.protection</name>
  <value>privacy</value>
</property>
<property>
  <name>hbase.thrift.security.qop</name>
  <value>auth-conf</value>
</property>
<property>
  <name>hbase.thrift.ssl.enabled</name>
  <value>true</value>
</property>
<property>
  <name>hbase.rest.ssl.enabled</name>
  <value>true</value>
</property>
<property>
  <name>hbase.ssl.enabled</name>
  <value>true</value>
</property>
<property>
  <name>hbase.security.authentication</name>
  <value>kerberos</value>
</property>
<property>
  <name>hbase.security.authorization</name>
  <value>true</value>
</property>
<property>
  <name>hbase.secure.rpc.engine</name>
  <value>true</value>
</property>
```

6. Add the cluster peers using the simplified add_peer syntax:

```
add_peer 'ID', 'CLUSTER_KEY'
```

7. If you need to add any peers which require custom security configuration, modify the add_peer syntax, using the following examples as a model.

```
add_peer 'vegas', CLUSTER_KEY => 'zk1.vegas.example.com:2181:/hbase',
         CONFIG => {'hbase.master.kerberos.principal'           => 'hbase/
_HOST@TO_VEGAS',
```

```
                      'hbase.regionserver.kerberos.principal'     => 'hbase/
_HOST@TO_VEGAS',
                      'hbase.regionserver.keytab.file'            => '/key
tabs/vegas_hbase.keytab',
                      'hbase.master.keytab.file'                  => '/keyta
bs/vegas_hbase.keytab'},
          TABLE_CFS => {"tbl" => [cf1']}

add_peer 'atlanta', CLUSTER_KEY => 'zk1.vegas.example.com:2181:/hbase',
         CONFIG => {'hbase.master.kerberos.principal'           => 'hbase/
_HOST@TO_ATLANTA',
                      'hbase.regionserver.kerberos.principal'     => 'hbase/
_HOST@TO_ATLANTA',
                      'hbase.regionserver.keytab.file'            => '/ke
ytabs/atlanta_hbase.keytab',
                      'hbase.master.keytab.file'                  => '/ke
ytabs/atlanta_hbase.keytab'},
          TABLE_CFS => {"tbl" => [cf2']}
```

# Configure bulk load replication

Bulk loading is the process of preparing and loading HFiles directly into HBase RegionServers bypassing the write path. If you bulk load data into HBase frequently and want to replicate this data, you must configure and use bulk load replication.

Because bulk loading data bypasses the write path, and this process does not generate WALs, your data will not be replicated to the backup cluster.

## Enable bulk load replication using Cloudera Manager

You can enable bulk load replication using Cloudera Manager.

### Before you begin

If you have enabled Kerberos cross-realm authentication:

1. At the command line, use the list_principals command to list the kdc, admin_server, and default_domain for each realm.
2. Add this information to each cluster using Cloudera Manager. For each cluster, go to HDFS Configuration Trusted Kerberos Realms . Add the target and source realms. This requires a restart of HDFS.
3. Enter a Reason for change, and then click Save Changes to commit the changes. Restart the role and service when Cloudera Manager prompts you to restart.

### Procedure

1. Go to the source cluster from which you want to replicate the buck loaded data.
2. Go to HBase Configuration .
3. Select Scope > (Service-Wide).
4. Locate the HBase Service Advanced Configuration Snippet (Safety Valve) for hbase-site.xml property or search for it by typing its name in the Search box.

5. Add the following property values:
   - Name: hbase.replication.bulkload.enabled

     Value: true

     Description: Enable bulk load replication
   - Name: hbase.replication.cluster.id

     Value: source1

     Description: Provide a source cluster-ID. For example, source1.
6. Manually copy and paste the source cluster's HBase client configuration files in the target cluster where you want the data to be replicated. Copy core-site.xml, hdfs-site.xml, and hbase-site.xml to the target cluster. Do this for all RegionServers.
7. Go to the target cluster where you want the data to be replicated.
8. Go to HBase Configuration .
9. Select Scope > (Service-Wide).
10. Locate the HBase Service Advanced Configuration Snippet (Safety Valve) for hbase-site.xml property or search for it by typing its name in the Search box.
11. Add the following property value:
    - Name: hbase.replication.conf.dir

      Value: /opt/cloudera/fs_conf

      Description: Path to the configuration directory where the source cluster's configuration files have been copied. The path for copying the configuration file is [hbase.replication.conf.dir]/[hbase.replication.cluster.id], i.e.:/opt/cloudera/fs_conf/source/
12. Restart the HBase service on both clusters to deploy the new configurations.

    **Important:**
    - In the target cluster, ensure that you copy the configuration files to the path set in [hbase.replication.conf.dir]. There, you must create a directory with the [hbase.replication.cluster.id] name.
    - Make sure to set the correct file permissions to hbase user using the command:

      ```
      chown -R hbase:hbase /opt/cloudera/fs_conf/source
      ```
13. Add the peer to the source cluster as you would with normal replication.
    - In the HBase shell, add the target cluster as a peer using the following command:

      ```
      add_peer '1', CLUSTER_KEY => '<cluster_name>:<hbase_port>:/hbase'
      ```
    - Enable the replication for the table to which you will be bulk loading data using the command:

      ```
      enable_table_replication 'IntegrationTestBulkLoad'
      ```
    - Alternatively, you can allow replication on a column family using the command:

      ```
      disable 'IntegrationTestBulkLoad'
      alter 'IntegrationTestBulkLoad', {NAME => 'D', REPLICATION_SCOPE => '1'}
      enable 'IntegrationTestBulkLoad'
      ```

    You can verify if BulkLoad Replication is working in your set up by following the example in this blog post:

    https://blog.cloudera.com/blog/2013/09/how-to-use-hbase-bulk-loading-and-why/

## Create empty table on the destination cluster

You can create table on the destination cluster by extracting the schema using HBase Shell.

### About this task

If the table to be replicated does not yet exist on the destination cluster, you must create it.

### Procedure

1. On the source cluster, describe the tahble using HBase Shell.
   The ouput is reformatted for readability:

```
hbase> describe acme_users

Table acme_users is ENABLED
acme_users
COLUMN FAMILIES DESCRIPTION
{NAME => 'user', DATA_BLOCK_ENCODING => 'NONE', BLOOMFILTER => 'NONE',
REPLICATION_SCOPE => '0', VERSIONS => '3', COMPRESSION => 'NONE',
MIN_VERSIONS => '0', TTL => 'FOREVER', KEEP_DELETED_CELLS => 'FALSE',
BLOCKSIZE => '65536', IN_MEMORY => 'false', BLOCKCACHE => 'false'}
```

2. Copy the output and make the following changes:
   - For the TTL, change FOREVER to org.apache.hadoop.hbase.HConstants::FOREVER.
   - Add the word CREATE before the table name.
   - Remove the line COLUMN FAMILIES DESCRIPTION and everything above the table name.

   A command like the following:

```
"CREATE 'cme_users' ,

{NAME => 'user', DATA_BLOCK_ENCODING => 'NONE', BLOOMFILTER => 'NONE',
REPLICATION_SCOPE => '0', VERSIONS => '3', COMPRESSION => 'NONE',
MIN_VERSIONS => '0', TTL => org.apache.hadoop.hbase.HConstants::FOREVER,
 KEEP_DELETED_CELLS => 'FALSE',
BLOCKSIZE => '65536', IN_MEMORY => 'false', BLOCKCACHE => 'false'}
```

3. Paste the command into HBase Shell on the destination cluster.
   The table is created.

### Related Information

Replication requirements
Deploy HBase replication

## Disable replication at the peer level

You can disable replication for a specific peer.

Use the command disable_peer ("<peerID>") to disable replication for a specific peer. This will stop replication to the peer, but the logs are kept for future reference.

**Note:** This log accumulation is a powerful side effect of the disable_peer command and can be used to your advantage.

To re-enable the peer, use the command enable_peer(<"peerID">). Replication resumes.

Examples:

- To disable peer 1:

```
disable_peer("1")
```

- To re-enable peer 1:

```
enable_peer("1")
```

## Stop replication in an emergency

If replication is causing serious problems, you can stop it while the clusters are running.

Open the shell on the source cluster and use the disable_peer command for each peer, then the disable_table_replication command. For example:

```
hbase> disable_peer("1")
   hbase> disable_table_replication 'table_name'
```

Already queued edits will be replicated after you use the disable_table_replication command, but new entries will not.

To start replication again, use the enable_peer command.

## Initiate replication when data already exist

You can initiate replication when data already exist by taking advantage of the accumulation that happens when a replication peer is disabled.

### About this task

You may need to start replication from some point in the past. For example, suppose you have a primary HBase cluster in one location and are setting up a disaster-recovery (DR) cluster in another. To initialize the DR cluster, you need to copy over the existing data from the primary to the DR cluster, so that when you need to switch to the DR cluster you have a full copy of the data generated by the primary cluster. Once that is done, replication of new data can proceed as normal.

### Procedure

1. Start replication.
2. Add the destination cluster as a peer.
3. Immediately disable it using disable_peer.
4. Take a snapshot of the table on the source cluster and export it.
   The snapshot command flushes the table from memory.
5. Import and restore the snapshot on the destination cluster.
6. Run enable_peer to re-enable the destination cluster.

## Replicate pre-exist data in an active-active deployment

You can replicate pre-exist data in an active-active deployment if you run the copyTable job before starting the replication.

### About this task

You have to run the copyTable job before starting the replication. If you start the job after enabling replication, the second cluster will re-send the data to the first cluster, because copyTable does not edit the clusterId in the mutation objects.

### Procedure

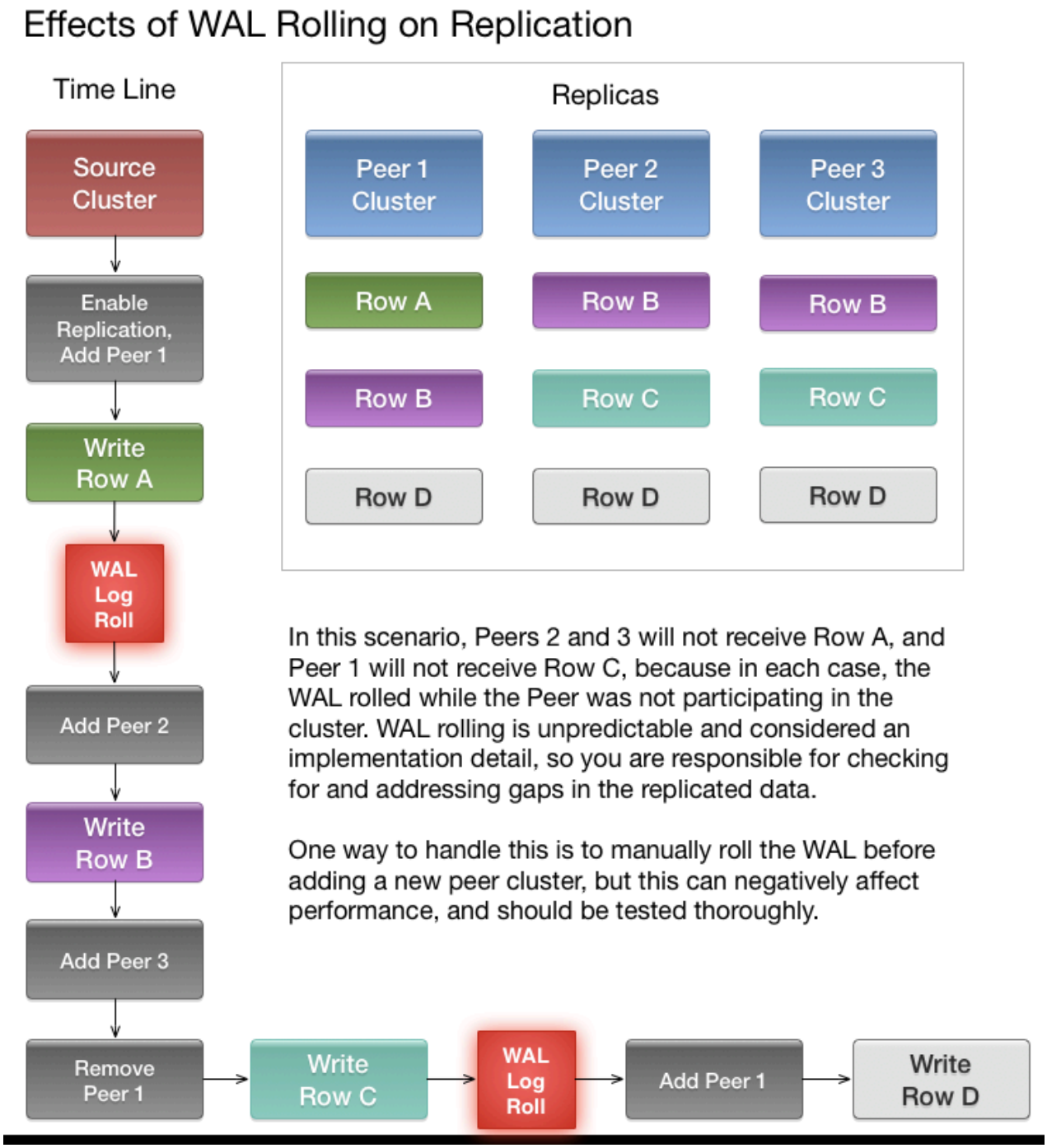
1. Run the copyTable job and note the start timestamp of the job.
2. Start replication.
3. Run the copyTable job again with a start time equal to the start time you noted in step 1.
   Some data being pushed back and forth between the two clusters; but it minimizes the amount of data.

## Effects of WAL rolling on replication

Adding and removing peer clusters with unpredictable WAL rolling occurring have effects on the replication.

When you add a new peer cluster, it only receives new writes from the source cluster since the last time the WAL was rolled.

The following diagram shows the consequences of adding and removing peer clusters with unpredictable WAL rolling occurring. Follow the time line and notice which peer clusters receive which writes. Writes that occurred before the WAL is rolled are not retroactively replicated to new peers that were not participating in the cluster before the WAL was rolled.




## Configure secure HBase replication

You must configure cross realm support for Kerberos, ZooKeeper, and Hadoop to configure secure HBase replication.

### About this task

There must be at least one common encryption mode between the two realms.

**Note:** HBase peer-to-peer replication from a non-Kerberized cluster to a Kerberized cluster is not supported.

### Procedure

1. Create krbtgt principals for the two realms.

   For example, if you have two realms called EXAMPLE.com and COMPANY.TEST, you need to add the following principelas: krbtgt/EXAMPLE.COM@COMPANY.TEST and krbtgt/COMPANY.TEST@EXAMPLE.COM

2. Add the two principals at both realms.

```
kadmin: addprinc -e "<enc_type_list>" krbtgt/EXAMPLE.COM@COMPANY.TEST
kadmin: addprinc -e "<enc_type_list>" krbtgt/COMPANY.TEST@EXAMPLE.COM
```

Add rules creating short names in ZooKeeper:

3. Add a system level property in java.env, defined in the conf directory.

   The following example rule illustrates how to add support for the realm called EXAMPLE.COM and have two members in the principal (such as service/instance@EXAMPLE.COM):

```
-Dzookeeper.security.auth_to_local=RULE:[2:\$1@\$0](.*@\\QEXAMPLE.COM\\E
$)s/@\\QEXAMPLE.COM\\E$//DEFAULT
```

   This example adds support for the EXAMPLE.COM realm in a different realm. So, in the case of replication, you must add a rule for the primary cluster realm in the replica cluster realm. DEFAULT is for defining the default rule

Add rules for creating short names in the Hadoop processes:

4. Add the hadoop.security.auth_to_local property in the core-site.xml file in the replica cluster.

   For example to add support for the EXAMPLE.COM realm:

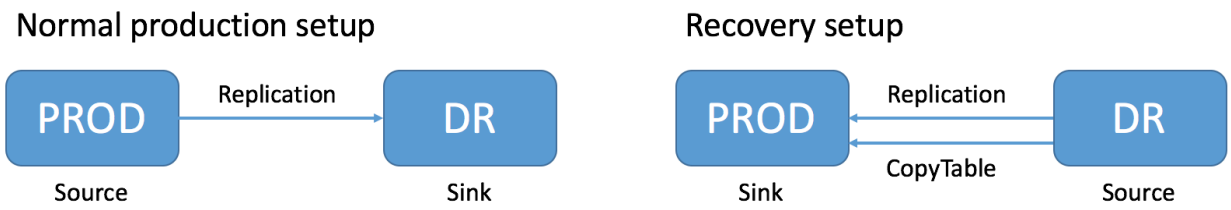
```
<property>
  <name>hadoop.security.auth_to_local</name>
  <value>
    RULE:[2:$1@$0](.*@\QEXAMPLE.COM\E$)s/@\QEXAMPLE.COM\E$//
  DEFAULT
  </value>
</property>
```

## Restore data from a replica

Recover HBase data from a replicated cluster in a disaster recovery scenario.

### About this task

One of the main reasons for replications is to be able to restore data, whether during disaster recovery or for other reasons. During restoration, the source and sink roles are reversed. The source is the replica cluster, and the sink is the cluster that needs restoration. This can be confusing, especially if you are in the middle of a disaster recovery scenario. The following image illustrates the role reversal between normal production and disaster recovery.




### Procedure

1. Change the value of the column family property REPLICATION_SCOPE on the sink to 0 for each column to be restored, so that its data will not be replicated during the restore operation.
2. Change the value of the column family property REPLICATION_SCOPE on the source to 1 for each column to be restored, so that its data will be replicated.
3. Use the copyTable or distcp commands to import the data from the backup to the sink cluster.
4. Add the sink as a replication peer to the source, using the add_peer command.
5. If you used distcp in step 3 on page 27,restart or rolling restart both clusters.
   The RegionServers pick up the new files.

When restoration is complete, do the following:

6. Change the REPLICATION_SCOPE values back to their values before initiating the restoration.

## Verify that replication works

Confirm data has been replicated from a source cluster to a remote destination cluster.

### Procedure

1. Install and configure YARN on the source cluster.

   If YARN cannot be used in the source cluster, configure YARN on the destination cluster to verify replication.

   If neither the source nor the destination clusters can have YARN installed, you can configure the tool to use local mode; however, performance and consistency could be negatively impacted.
2. Ensure that you have the required permissions:
   - You have sudo permissions to run commands as the hbase user, or a user with admin permissions on both clusters.
   - You are an hbase user configured for submitting jobs with YARN.

   **Note:** To use the hbase user in a secure cluster, use Cloudera Manager to add the hbase user as a YARN whitelisted user. For a new installation, the hbase user is already added to the whitelisted users. In addition, /user/hbase should exist on HDFS and owned as the hbase user, because YARN will create a job staging directory there.
3. Run the VerifyReplication command:

```
src-node$ sudo -u hbase hbase org.apache.hadoop.hbase.mapreduce.replicat
ion.VerifyReplication peer1 table1
...
        org.apache.hadoop.hbase.mapreduce.replication.VerifyReplication$V
erifier$Counters
                BADROWS=2
                CONTENT_DIFFERENT_ROWS=1
                GOODROWS=1
                ONLY_IN_PEER_TABLE_ROWS=1
        File Input Format Counters
```

```
                Bytes Read=0
        File Output Format Counters
                Bytes Written=0
```

The following table describes the VerifyReplication counters:

### Table 1: VerifyReplication Counters

| Counter | Description |
|---|---|
| GOODROWS | Number of rows. On both clusters, and all values are the same. |
| CONTENT_DIFFERENT_ROWS | The key is the same on both source and destination clusters for a row, but the value differs. |
| ONLY_IN_SOURCE_TABLE_ROWS | Rows that are only present in the source cluster but not in the destination cluster. |
| ONLY_IN_PEER_TABLE_ROWS | Rows that are only present in the destination cluster but not in the source cluster. |
| BADROWS | Total number of rows that differ from the source and destination clusters; the sum of CONTENT_DIFFERENT_ROWS + ONLY_IN_SOURCE_TABLE_ROWS + ONLY_IN_PEER_TABLE_ROWS |

By default, VerifyReplication compares the entire content of table1 on the source cluster against table1 on the destination cluster that is configured to use the replication peer peer1.

Use the following options to define the period of time, versions, or column families

### Table 2: VerifyReplication Counters

| Option | Description |
|---|---|
| --starttime=<timestamp> | Beginning of the time range, in milliseconds. Time range is forever if no end time is defined. |
| --endtime=<timestamp> | End of the time range, in milliseconds. |
| --versions=<versions> | Number of cell versions to verify. |
| --families=<cf1,cf2,..> | Families to copy; separated by commas. |

The following example, verifies replication only for rows with a timestamp range of one day:

```
src-node$ sudo -u hbase hbase org.apache.hadoop.hbase.mapreduce.replicat
ion.VerifyReplication --starttime=1472499077000 --endtime=1472585477000 --
families=c1  peer1 table1
```

## Replication caveats

Note these caveats when you use HBase replication.

- Two variables govern replication: hbase.replication and a replication znode. Stopping replication (using disable_table_replication as above) sets the znode to false. Two problems can result:
  - If you add a new RegionServer to the active cluster while replication is stopped, its current log will not be added to the replication queue, because the replication znode is still set to false. If you restart replication at this point (using enable_peer), entries in the log will not be replicated.
  - Similarly, if a log rolls on an existing RegionServer on the active cluster while replication is stopped, the new log will not be replicated, because the replication znode was set to false when the new log was created.

- In the case of a long-running, write-intensive workload, the destination cluster may become unresponsive if its meta-handlers are blocked while performing the replication. Cloudera Runtime has three properties to deal with this problem:
  - hbase.regionserver.replication.handler.count - the number of replication handlers in the destination cluster (default is 3). Replication is now handled by separate handlers in the destination cluster to avoid the above-mentioned sluggishness. Increase it to a high value if the ratio of active to passive RegionServers is high.
  - replication.sink.client.retries.number - the number of times the HBase replication client at the sink cluster should retry writing the WAL entries (default is 1).
  - replication.sink.client.ops.timeout - the timeout for the HBase replication client at the sink cluster (default is 20 seconds).
- For namespaces, tables, column families, or cells with associated ACLs, the ACLs themselves are not replicated. The ACLs need to be re-created manually on the target table. This behavior opens up the possibility for the ACLs could be different in the source and destination cluster.

### Related Information

Common replication topologies