Hortonworks Data Platform

User Guides

(Aug 13, 2013)

docs.hortonworks.com

Hortonworks Data Platform : User Guides

The Hortonworks Data Platform, powered by Apache Hadoop, is a massively scalable and 100% open source platform for storing, processing and analyzing large volumes of data. It is designed to deal with data from many sources and formats in a very quick, easy and cost-effective manner. The Hortonworks Data Platform consists of the essential set of Apache Hadoop projects including MapReduce, Hadoop Distributed File System (HDFS), HCatalog, Pig, Hive, HBase, Zookeeper and Ambari. Hortonworks is the major contributor of code and patches to many of these projects. These projects have been integrated and tested as part of the Hortonworks Data Platform release process and installation and configuration tools have also been included.

Unlike other providers of platforms built using Apache Hadoop, Hortonworks contributes 100% of our code back to the Apache Software Foundation. The Hortonworks Data Platform is Apache-licensed and completely open source. We sell only expert technical support, training and partner-enablement services. All of our technology is, and will remain free and open source.

Please visit the Hortonworks Data Platform page for more information on Hortonworks technology. For more information on Hortonworks services, please visit either the Support or Training page. Feel free to Contact Us directly to discuss your specific needs.

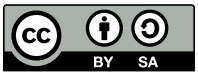

Table of Contents

List of Tables

1. HBase Snapshots

Prior to version HBase version 0.94.6, in order to perform backup or clone a table involved only two mechanisms - either perform CopyTable/ExportTable operations or copy all the hfiles in HDFS after disabling the table. These methods to backup or clone tables either degraded RegionServer performance (Copy/Export Table) or users were required to disable the table (which means no further read or write operations could occur on the disabled table).

HBase Snapshots lets you take a snapshot of a table without impacting RegionServers adversely. Snapshot, Clone, and restore operations do not involve data copying. Also, Exporting the snapshot to another cluster does not impact the Region Servers.

Use this document to configure and use HBase snapshots for performing backups or cloning HBase tables:

1.1. Configuration

To turn on the snapshot support just set the `hbase.snapshot.enabled` property to true. (Snapshots are enabled by default in HBase v 0.95 or later and are disabled by default in HBase v 0.94.6 or later)

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

1.2. Take a Snapshot

You can take a snapshot of a table regardless of whether it is enabled or disabled. The snapshot operation does not involve copying any data.

```
  $ C:\...\hbase..\bin\hbase.cmd shell
  hbase> snapshot 'myTable', 'myTableSnapshot-122112'
```

1.3. Listing Snapshots

List all snapshots taken (by printing the names and relative information).

```
$ C:\...\hbase..\bin\hbase.cmd shell
hbase> list_snapshots
```

1.4. Deleting Snapshots

You can remove a snapshot and the files retained for that snapshot will be removed if these files are no longer required.

```
$ C:\...\hbase..\bin\hbase.cmd shell
hbase> delete_snapshot 'myTableSnapshot-122112'
```

1.5. Clone a table from snapshot

From a snapshot you can create a new table (clone operation) with the same data, that you had, when the snapshot was taken. The clone operation, does not involve data copies. A change to the cloned table does not impact either the snapshot or the original table.

```
$ C:\...\hbase..\bin\hbase.cmd shell
hbase> clone_snapshot 'myTableSnapshot-122112', 'myNewTestTable'
```

1.6. Restore a snapshot

The restore operation requires the table to be disabled. The table will be restored to the state at the time when the snapshot was taken (this operation might change both data and schema).

```
$ C:\...\hbase..\bin\hbase.cmd shell
hbase> disable 'myTable'
hbase> restore_snapshot 'myTableSnapshot-122112'
```

Note

Replication works at log level and snapshots at file-system level. Therefore, after a restore operation, the replicas will have a state that is different from the master. If you want to perform the restore operation, you must stop replication and perform bootstrap operation again.

In case if you experience partial data-loss (due to client failure), instead of performing a full restore (full restore requires disabling the table), you can clone the table from the snapshot and use a Map-Reduce job to copy the data that you need from the cloned table to the main table.

1.7. Export to another cluster

The ExportSnapshot tool copies all the data related to a snapshot (hfiles, logs, snapshot metadata) to another cluster. The tool executes a Map-Reduce job, similar to distcp, to copy files between the two clusters, and because it works at file-system level the hbase cluster does not have to be online.

The HBase Snapshot Export tool must be run as HBase service user. The HBase Snapshot Export tool must have temporary directory, specified by "hbase.tmp.dir" (for example, /grid/0/var/log/hbase), created on HDFS with HBase user as the owner.

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

```
$ bin/hbase class org.apache.hadoop.hbase.snapshot.tool.ExportSnapshot -
snapshot MySnapshot -copy-to hdfs://srv2:8020/hbase -mappers 16
```

2. User Guide - HDFS Snapshots

HDFS Snapshots are read-only point-in-time copies of the file system. Snapshots can be taken on a subtree of the file system or the entire file system. Some common use cases of snapshots are data backup, protection against user errors and disaster recovery.

The implementation of HDFS Snapshots is efficient:

1. Snapshot creation is instantaneous: the cost is **O(1)** excluding the inode lookup time.
2. Additional memory is used only when modifications are made relative to a snapshot: memory usage is **O(M)** where **M** is the number of modified files/directories.
3. Blocks in datanodes are not copied: the snapshot files record the block list and the file size. There is no data copying.
4. Snapshots do not adversely affect regular HDFS operations: modifications are recorded in reverse chronological order so that the current data can be accessed directly. The snapshot data is computed by subtracting the modifications from the current data.

In this document:

2.1. Snapshottable Directories

Snapshots can be taken on any directory once the directory has been set as **snapshottable**. A snapshottable directory is able to accommodate 65,536 simultaneous snapshots. There is no limit on the number of snapshottable directories. Administrators may set any directory to be snapshottable. If there are snapshots in a snapshottable directory, the directory can be neither deleted nor renamed before all the snapshots are deleted.

2.2. Snapshot Paths

For a snapshottable directory, the path component **".snapshot"** is used for accessing its snapshots. Suppose `/foo` is a snapshottable directory, `/foo/bar` is a file/directory in `/foo`, and `/foo` has a snapshot `s0`. Then, the path `/foo/.snapshot/s0/bar` refers to the snapshot copy of `/foo/bar`. The usual API and CLI can work with the ".snapshot" paths. The following are some examples:

- Listing all the snapshots under a snapshottable directory: `hadoop dfs -ls /foo/.snapshot`
- Listing the files in snapshot `s0`: `hadoop dfs -ls /foo/.snapshot/s0`
- Copying a file from snapshot `s0`: `hadoop dfs -cp /foo/.snapshot/s0/bar /tmp`

The name ".snapshot" is now a reserved file name in HDFS so that users cannot create a file/directory with ".snapshot" as the name. If ".snapshot" is used in a previous version of HDFS, it must be renamed before upgrade; otherwise, upgrade will fail.

2.3. Snapshot Operations

Snapshot operations are grouped into the following two categories:

- Administrator Operations
- User Operations

2.3.1. Administrator Operations

The operations described in this section require superuser privileges.

- Allow Snapshots: Allowing snapshots of a directory to be created. If the operation completes successfully, the directory becomes snapshottable.
  - Command:

```
hadoop dfsadmin -allowSnapshot $path
```

  - Arguments:

Table 2.1. Administrator Operations - Allow Snapshots

Parameter name	Description
path	The path of the snapshottable directory.

See also the corresponding Java API `void allowSnapshot(Path path)` in `HdfsAdmin`.

- Disallow Snapshots: Disallowing snapshots of a directory to be created. All snapshots of the directory must be deleted before disallowing snapshots.
  - Command:

```
hadoop dfsadmin -disallowSnapshot $path
```

  - Arguments:

Table 2.2. Administrator Operations - Disallow Snapshots

Parameter name	Description
path	The path of the snapshottable directory.

See also the corresponding Java API `void disallowSnapshot(Path path)` in `HdfsAdmin`.

2.3.2. User Operations

The section describes user operations. Note that HDFS superuser can perform all the operations without satisfying the permission requirement in the individual operations.

- Create Snapshots: Create a snapshot of a snapshottable directory. This operation requires owner privilege to the snapshottable directory.
  - Command:

```
hadoop dfs -createSnapshot $path $snapshotName
```

  - Arguments:

Table 2.3. User Operations - Create Snapshots

Parameter name	Description
path	The path of the snapshottable directory.
snapshotName	The snapshot name, which is an optional argument. When it is omitted, a default name is generated using a timestamp with the format "'s'yyyyMMdd-HHmmss.SSS", e.g. "s20130412-151029.033".

See also the corresponding Java API `Path createSnapshot(Path path)` and `Path createSnapshot(Path path, String snapshotName)` in `FileSystem`. The snapshot path is returned in these methods.

- Delete Snapshots: Delete a snapshot of from a snapshottable directory. This operation requires owner privilege of the snapshottable directory.
  - Command:

```
hadoop dfs -deleteSnapshot $path $snapshotName
```

  - Arguments:

Table 2.4. User Operations - Delete Snapshots

Parameter name	Description
path	The path of the snapshottable directory.
snapshotName	The snapshot name.

See also the corresponding Java API `void deleteSnapshot(Path path, String snapshotName)` in `FileSystem`.

- Rename Snapshots: Rename a snapshot. This operation requires owner privilege of the snapshottable directory..
  - Command:

```
hadoop dfs -renameSnapshot $path $oldName $newName
```

  - Arguments:

Table 2.5. User Operations - Rename Snapshots

Parameter name	Description
path	The path of the snapshottable directory.
oldName	The old snapshot name.
newName	The new snapshot name.

See also the corresponding Java API `void renameSnapshot(Path path, String oldName, String newName)` in `FileSystem`.

- Get Snapshottable Directory Listing: Get all the snapshottable directories where the current user has permission to take snapshots.
  - Command:

```
hadoop lsSnapshottableDir $path $snapshotName
```

  - Arguments:

Table 2.6. User Operations - Get Snapshottable Directory Listing

Parameter name	Description
path	The path of the snapshottable directory.
snapshotName	The snapshot name.

See also the corresponding Java API `SnapshottableDirectoryStatus[] getSnapshottableDirectoryListing()` in `DistributedFileSystem`.

- Get Snapshots Difference Report: Get the differences between two snapshots. This operation requires read access privilege for all files/directories in both snapshots.
  - Command:

```
hadoop snapshotDiff $path $fromSnapshot $toSnapshot
```

  - Arguments:

Table 2.7. User Operations - Get Snapshots Difference Report

Parameter name	Description
path	The path of the snapshottable directory.
fromSnapshot	The name of the starting snapshot.
toSnapshot	The name of the ending snapshot.

3. HDFS Permissions

The Hadoop Distributed File System (HDFS) enforces permissions the same way on Windows and Linux deployments. The HDFS permissions model for files and directories shares much of the POSIX model; each file and directory is associated with an owner and a group.

3.1. HDFS and the HadoopUsers Group

On each node that HDP is installed, HDP sets up a `HadoopUsers` group and creates a `hadoop` user in that group. The `hadoop` user is the superuser in HDP. This user:

1. Is the owner of the HDP services installed on each Windows Server node.
2. Is the HDFS superuser. This superuser can modify the permissions of any HDFS directory or file, regardless of owner.
3. Is the Oozie proxy user.
4. Is the WebHCat proxy user.

Note

HDP depends on user accounts on each cluster node for enforcing access rules to the data in HDFS.

3.1.1. Active Directory Groups and Users

HDP resolves membership in the machine's local groups, and skip groups coming from Active Directory. Although Active Directory groups are unsupported, Active Directory users are supported. You can create local groups on all nodes in the cluster, and manage group membership individually on each node. These local groups can contain Active Directory users.

For a Windows domain user, such as `CORP\$win_username`, the Hadoop code ignores the domain portion and treats the user identity as just the username, `$win_username`. File ownership in HDFS and job submissions display as `$win_username`. Consequently, if the cluster is joined to multiple domain controllers, and the same username exists in multiple domains, Hadoop assumes they are the same user: `DOMAIN1\$win_username = DOMAIN2\$win_username = DOMAIN3\$win_username.`

3.1.2. Protecting Against Impersonation

If a user account can create new users on machines that have direct access to the HDP cluster, then those users can create a `hadoop` user and get administrative access to HDP services.

You can manage user access in Windows similar to a Linux non-secured cluster by:

- Putting all of the cluster nodes behind a firewall.

- Only allowing HDFS client access and MapReduce job submission from specific machines (or a specific subnet).
- Giving users accounts on machines with non-admininistrator permissions.