

Cloudera Runtime 7.2.0

Accessing Apache HBase

Date published: 2020-02-29

Date modified: 2023-04-05

CLOUdera

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

Legal Notice

© Cloudera Inc. 2025. 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 Shell overview.....	4
Virtual machine options for HBase Shell.....	4
Script with HBase Shell.....	4
Use HBase command-line utilities.....	5
Use the Java API.....	11
Use the Apache Thrift Proxy API.....	12
Use the Hue HBase app.....	17

HBase Shell overview

You can use the HBase Shell from the command line interface to communicate with HBase.

In CDP, you need to SSH into an HBase node before you can use the HBase Shell. For example, to SSH into an HBase node with the IP address 10.10.10.10, you must use the command:

```
ssh <USERNAME>@10.10.10.10
```



Note: You must use your IPA password for authentication.

After you have started HBase, you can access the database in an interactive way by using the HBase Shell, which is a command interpreter for HBase which is written in Ruby. Always run HBase administrative commands such as the HBase Shell, hbck, or bulk-load commands as the HBase user (typically hbase).

```
hbase shell
```

You can use the following commands to get started with the HBase shell:

- To get help and to see all available commands, use the help command.
- To get help on a specific command, use help "command". For example:

```
hbase> help "create"
```

- To remove an attribute from a table or column family or reset it to its default value, set its value to nil. For example, use the following command to remove the KEEP_DELETED_CELLS attribute from the f1 column of the users table:

```
hbase> alter 'users', { NAME => 'f1', KEEP_DELETED_CELLS => nil }
```

- To exit the HBase Shell, type quit.

Virtual machine options for HBase Shell

You can set variables for the virtual machine running HBase Shell, by using the HBASE_SHELL_OPTS environment variable. This example sets several options in the virtual machine.

This example sets several options in the virtual machine.

```
$ HBASE_SHELL_OPTS="-verbose:gc -XX:+PrintGCApplicationStoppedTime -XX:+PrintGCDateStamps  
-XX:+PrintGCDetails -Xloggc:$HBASE_HOME/logs/gc-hbase.log" ./bin/hbase  
shell
```

Script with HBase Shell

You can use HBase shell in your scripts. You can also write Ruby scripts for use with HBase Shell. Example Ruby scripts are included in the hbase-examples/src/main/ruby/ directory.

The non-interactive mode allows you to use HBase Shell in scripts, and allow the script to access the exit status of the HBase Shell commands. To invoke non-interactive mode, use the `-n` or `--non-interactive` switch. This small example script shows how to use HBase Shell in a Bash script.

```
#!/bin/bash
echo 'list' | hbase shell -n
status=$?
if [ $status -ne 0 ]; then
    echo "The command may have failed."
fi
```

Successful HBase Shell commands return an exit status of 0. However, an exit status other than 0 does not necessarily indicate a failure, but should be interpreted as unknown. For example, a command may succeed, but while waiting for the response, the client may lose connectivity. In that case, the client has no way to know the outcome of the command. In the case of a non-zero exit status, your script should check to be sure the command actually failed before taking further action.

You can use the `get_splits` command, which returns the split points for a given table:

```
hbase> get_splits 't2'
Total number of splits = 5

=> ["", "10", "20", "30", "40"]
```

Use HBase command-line utilities

Besides the HBase Shell, HBase includes several other command-line utilities, which are available in the `hbase/bin/` directory of each HBase host. This topic provides basic usage instructions for the most commonly used utilities.

PerformanceEvaluation

The PerformanceEvaluation utility allows you to run several preconfigured tests on your cluster and reports its performance. To run the PerformanceEvaluation tool, use the `bin/hbase pecommand`.

```
$ hbase pe

Usage: java org.apache.hadoop.hbase.PerformanceEvaluation \
  <OPTIONS> [-D<property=value>]* <command> <nclients>

Options:
  nomapreduce      Run multiple clients using threads (rather than use mapreduce)
  rows             Rows each client runs. Default: One million
  size             Total size in GiB. Mutually exclusive with --rows. Default: 1.0.
  sampleRate       Execute test on a sample of total rows. Only supported by randomRead.
  traceRate        Default: 1.0
  t: 0             Enable HTrace spans. Initiate tracing every N rows. Default: 0
  table            Alternate table name. Default: 'TestTable'
  multiGet         If >0, when doing RandomRead, perform multiple gets instead of single gets.
                  Default: 0
  compress         Compression type to use (GZ, LZ0, ...). Default: 'NONE'
  flushCommits     Used to determine if the test should flush the table. Default: false
  writeToWAL       Set writeToWAL on puts. Default: True
```

autoFlush	Set autoFlush on htable. Default: False
oneCon	all the threads share the same connection. Default: False
presplit	Create presplit table. Recommended for accurate perf analy
sis (see	guide). Default: disabled
inmemory	Tries to keep the HFiles of the CF inmemory as far as possi
ble. Not	guaranteed that reads are always served from memory. Defa
ult: false	
usetags	Writes tags along with KVs. Use with HFile V3. Default:
false	
numoftags	Specify the no of tags that would be needed. This works o
nly if usetags	is true.
filterAll	Helps to filter out all the rows on the server side there
by not returning	anything back to the client. Helps to check the server si
de performance.	Uses FilterAllFilter internally.
latency	Set to report operation latencies. Default: False
bloomFilter	Bloom filter type, one of [NONE, ROW, ROWCOL]
valueSize	Pass value size to use: Default: 1024
valueRandom	Set if we should vary value size between 0 and 'valueSiz
e'; set on read	for stats on size: Default: Not set.
valueZipf	Set if we should vary value size between 0 and 'valueSize'
in zipf form:	Default: Not set.
period	Report every 'period' rows: Default: opts.perClientRunRo
ws / 10	
multiGet	Batch gets together into groups of N. Only supported by ran
domRead.	Default: disabled
addColumnns	Adds columns to scans/gets explicitly. Default: true
replicas	Enable region replica testing. Defaults: 1.
splitPolicy	Specify a custom RegionSplitPolicy for the table.
randomSleep	Do a random sleep before each get between 0 and entered v
alve. Defaults: 0	
columns	Columns to write per row. Default: 1
caching	Scan caching to use. Default: 30
Note: -D properties will be applied to the conf used.	
For example:	
-Dmapreduce.output.fileoutputformat.compress=true	
-Dmapreduce.task.timeout=60000	
Command:	
append	Append on each row; clients overlap on keyspace so some c
oncurrent	operations
checkAndDelete	CheckAndDelete on each row; clients overlap on keyspace so
some concurrent	operations
checkAndMutate	CheckAndMutate on each row; clients overlap on keyspace so
some concurrent	operations
checkAndPut	CheckAndPut on each row; clients overlap on keyspace so s
ome concurrent	operations
filterScan	Run scan test using a filter to find a specific row based
on it's value	(make sure to use --rows=20)
increment	Increment on each row; clients overlap on keyspace so some
concurrent	operations

```

randomRead      Run random read test
randomSeekScan  Run random seek and scan 100 test
randomWrite     Run random write test
scan            Run scan test (read every row)
scanRange10     Run random seek scan with both start and stop row (max 10
rows)
scanRange100    Run random seek scan with both start and stop row (max 100
rows)
scanRange1000   Run random seek scan with both start and stop row (max 1000
rows)
scanRange10000  Run random seek scan with both start and stop row (max 1
0000 rows)
sequentialRead  Run sequential read test
sequentialWrite Run sequential write test
Args:
nclients        Integer. Required. Total number of clients (and HRegionS
ervers)
running: 1 <= value <= 500
Examples:
To run a single client doing the default 1M sequentialWrites:
$ bin/hbase org.apache.hadoop.hbase.PerformanceEvaluation sequentialWrite 1
To run 10 clients doing increments over ten rows:
$ bin/hbase org.apache.hadoop.hbase.PerformanceEvaluation --rows=10 --noma
pred increment 10

```

LoadTestTool

The LoadTestTool utility load-tests your cluster by performing writes, updates, or reads on it. To run the LoadTest Tool, use the `bin/hbase ltt` command. To print general usage information, use the `-h` option.

```

$ bin/hbase ltt -h

Options:
-batchupdate      Whether to use batch as opposed to separate
updates for every column in a row
-bloom <arg>      Bloom filter type, one of [NONE, ROW, ROWC
OL]
-compression <arg> Compression type, one of [LZO, GZ, NONE, SN
APPY, LZ4]
-data_block_encoding <arg> Encoding algorithm (e.g. prefix compress
ion) to use for data blocks in the test column family, one of
[NONE, PREFIX, DIFF, FAST_DIFF, PREFIX_T
REE].
-deferredlogflush Enable deferred log flush.
-encryption <arg> Enables transparent encryption on the test
table, one of [AES]
-families <arg>    The name of the column families to use se
parated by comma
-generator <arg>   The class which generates load for the too
l. Any args for this class can be passed as colon separated after c
lass name
-h, --help        Show usage
-in_memory        Tries to keep the HFiles of the CF inmemory
as far as possible. Not guaranteed that reads are always served fro
m inmemory
-init_only        Initialize the test table only, don't do
any loading

```

-key_window <arg> and writes for concurrent	The 'key window' to maintain between reads
-max_read_errors <arg> erate before terminating all	write/read workload. The default is 0. The maximum number of read errors to tol
-mob_threshold <arg> will use the MOB write path	reader threads. The default is 10. Desired cell size to exceed in bytes that
-multiget_batchsize <arg> arate gets for every	Whether to use multi-gets as opposed to sep
-multiput eparate puts for every	column in a row Whether to use multi-puts as opposed to s
-num_keys <arg> -num_regions_per_server <arg> er. Defaults to 5.	column in a row The number of keys to read/write Desired number of regions per region serv
-num_tables <arg> is specified, load test tool	A positive integer number. When a number n
value becomes table name prefix.	will load n table parallely. -tn parameter
_n	Each table name is in format <tn>_1...<tn>
-read <arg> -reader <arg> -region_replica_id <arg> -region_replication <arg> -regions_per_server <arg> is specified, load test tool	<verify_percent>[:<#threads=20>] The class for executing the read requests Region replica id to do the reads from Desired number of replicas per region A positive integer number. When a number n
er server	will create the test table with n regions p
-skip_init already exists	Skip the initialization; assume test table
-start_key <arg> ex). The default value is 0.	The first key to read/write (a 0-based ind
-tn <arg> -update <arg> to ignore nonce collisions=0>]	The name of the table to read or write <update_percent>[:<#threads=20>][:<#whether
-updater <arg> -write <arg> eads=20>]	The class for executing the update requests <avg_cols_per_key>:<avg_data_size>[:<#thr
-writer <arg> -zk <arg> ithout port numbers	The class for executing the write requests ZK quorum as comma-separated host names w
-zk_root <arg>	name of parent znode in zookeeper

wal

The wal utility prints information about the contents of a specified WAL file. To get a list of all WAL files, use the HDFS command `hadoop fs -ls -R /hbase/WALs`. To run the wal utility, use the `bin/hbase wal` command. Run it without options to get usage information.

```
hbase wal
usage: WAL <filename...> [-h] [-j] [-p] [-r <arg>] [-s <arg>] [-w <arg>]
-h,--help          Output help message
-j,--json          Output JSON
-p,--printvals     Print values
-r,--region <arg> Region to filter by. Pass encoded region name; e.g.
                  '9192caead6a5a20acb4454ffbc79fa14'
-s,--sequence <arg> Sequence to filter by. Pass sequence number.
-w,--row <arg>    Row to filter by. Pass row name.
```


hfile

The hfile utility prints diagnostic information about a specified hfile, such as block headers or statistics. To get a list of all hfiles, use the HDFS command `hadoop fs -ls -R /hbase/data`. To run the hfile utility, use the `bin/hbase hfilecommand`. Run it without options to get usage information.

```
$ hbase hfile

usage: HFile [-a] [-b] [-e] [-f <arg> | -r <arg>] [-h] [-i] [-k] [-m] [-p]
          [-s] [-v] [-w <arg>]
-a,--checkfamily           Enable family check
-b,--printblocks           Print block index meta data
-e,--printkey              Print keys
-f,--file <arg>           File to scan. Pass full-path; e.g.
                           hdfs://a:9000/hbase/hbase:meta/12/34
-h,--printblockheaders     Print block headers for each block.
-i,--checkMobIntegrity     Print all cells whose mob files are missing
-k,--checkrow              Enable row order check; looks for out-of-order
                           keys
-m,--printmeta             Print meta data of file
-p,--printkv               Print key/value pairs
-r,--region <arg>         Region to scan. Pass region name; e.g.
                           'hbase:meta,,1'
-s,--stats                 Print statistics
-v,--verbose               Verbose output; emits file and meta data
                           delimiters
-w,--seekToRow <arg>      Seek to this row and print all the kvs for this
                           row only
```

hbck

The hbck utility checks and optionally repairs errors in HFiles.



Warning: Running hbck with any of the `-fix` or `-repair` commands is dangerous and can lead to data loss. Contact Cloudera support before running it.

To run hbck, use the `bin/hbase hbck` command. Run it with the `-h` option to get more usage information.

```
-----
NOTE: As of HBase version 2.0, the hbck tool is significantly changed.
In general, all Read-Only options are supported and can be used
safely. Most -fix/ -repair options are NOT supported. Please see usage
below for details on which options are not supported.
-----
```

```
Usage: fsck [opts] {only tables}
where [opts] are:
  -help Display help options (this)
  -details Display full report of all regions.
  -timelag <timeInSeconds> Process only regions that have not experienced
any metadata updates in the last <timeInSeconds> seconds.
  -sleepBeforeRerun <timeInSeconds> Sleep this many seconds before checking
if the fix worked if run with -fix
  -summary Print only summary of the tables and status.
  -metaonly Only check the state of the hbase:meta table.
  -sidelineDir <hdfs://> HDFS path to backup existing meta.
  -boundaries Verify that regions boundaries are the same between META and
store files.
  -exclusive Abort if another hbck is exclusive or fixing.
```

Datafile Repair options: (expert features, use with caution!)

```
-checkCorruptHFiles    Check all Hfiles by opening them to make sure the
                        y are valid
-sidelineCorruptHFiles  Quarantine corrupted HFiles.  implies -checkCorru
                        ptHFiles
```

Replication options

```
-fixReplication    Deletes replication queues for removed peers
```

Metadata Repair options supported as of version 2.0: (expert features, use with caution!)

```
-fixVersionFile    Try to fix missing hbase.version file in hdfs.
-fixReferenceFiles  Try to offline lingering reference store files
-fixHFileLinks     Try to offline lingering HFileLinks
-noHdfsChecking    Don't load/check region info from HDFS. Assumes hbas
e:meta region info is good. Won't check/fix any HDFS issue, e.g. hole, orpha
n, or overlap
-ignorePreCheckPermission  ignore filesystem permission pre-check
```

NOTE: Following options are NOT supported as of HBase version 2.0+.

```
UNSUPPORTED Metadata Repair options: (expert features, use with caution!)
-fix                                Try to fix region assignments.  This is for backwards
compatibility
-fixAssignments    Try to fix region assignments.  Replaces the old -fix
-fixMeta           Try to fix meta problems.  This assumes HDFS region inf
o is good.
-fixHdfsHoles      Try to fix region holes in hdfs.
-fixHdfsOrphans    Try to fix region dirs with no .regioninfo file in hdfs
-fixTableOrphans   Try to fix table dirs with no .tableinfo file in hdfs
(online mode only)
-fixHdfsOverlaps   Try to fix region overlaps in hdfs.
-maxMerge <n>      When fixing region overlaps, allow at most <n> regions
to merge. (n=5 by default)
-sidelineBigOverlaps  When fixing region overlaps, allow to sideline big
overlaps
-maxOverlapsToSideline <n>  When fixing region overlaps, allow at most <
n> regions to sideline per group. (n=2 by default)
-fixSplitParents   Try to force offline split parents to be online.
-removeParents     Try to offline and sideline lingering parents and keep
daughter regions.
-fixEmptyMetaCells Try to fix hbase:meta entries not referencing any
region (empty REGIONINFO_QUALIFIER rows)
```

UNSUPPORTED Metadata Repair shortcuts

```
-repair                Shortcut for -fixAssignments -fixMeta -fixHdfsHoles -
fixHdfsOrphans -fixHdfsOverlaps -fixVersionFile -sidelineBigOverlaps -fixRef
erenceFiles -fixHFileLinks
-repairHoles           Shortcut for -fixAssignments -fixMeta -fixHdfsHoles
```

clean

After you have finished using a test or proof-of-concept cluster, the `hbase clean` utility can remove all HBase-related data from ZooKeeper and HDFS.



Warning: The `hbase clean` command destroys data. Do not run it on production clusters, or unless you are absolutely sure you want to destroy the data.

To run the `hbase clean` utility, use the `bin/hbase clean` command. Run it with no options for usage information.

```
$ bin/hbase clean
```

```
Usage: hbase clean (--cleanZk|--cleanHdfs|--cleanAll)
Options:
```

```
--cleanZk    cleans hbase related data from zookeeper.
--cleanHdfs  cleans hbase related data from hdfs.
--cleanAll   cleans hbase related data from both zookeeper and hdfs.
```

Use the Java API

You can use the Apache HBase Java API to communicate with Apache HBase. The Java API is one of the most common ways to communicate with HBase.

The following sample uses Apache HBase APIs to create a table and put a row into that table. The table name, column family name, qualifier (or column) name, and a unique ID for the row are defined. Together, these define a specific cell. Next, the table is created and the text “Hello, World!” is inserted into this cell.

```
import java.io.IOException;
import org.apache.hadoop.conf.Configuration;
import org.apache.hadoop.hbase.HBaseConfiguration;
import org.apache.hadoop.hbase.TableName;
import org.apache.hadoop.hbase.client.*;
import org.apache.hadoop.hbase.util.Bytes;

public class CreateAndPut {
    private static final TableName TABLE_NAME = TableName.valueOf("test_table_example");
    private static final byte[] CF_NAME = Bytes.toBytes("test_cf");
    private static final byte[] QUALIFIER = Bytes.toBytes("test_column");
    private static final byte[] ROW_ID = Bytes.toBytes("row01");
    public static void createTable(final Admin admin) throws IOException {
        if(!admin.tableExists(TABLE_NAME)) {
            TableDescriptor desc = TableDescriptorBuilder.newBuilder(TABLE_NAME)
                .setColumnFamily(ColumnFamilyDescriptorBuilder.of(CF_NAME))
                .build();
            admin.createTable(desc);
        }
    }

    public static void putRow(final Table table) throws IOException {
        table.put(new Put(ROW_ID).addColumn(CF_NAME, QUALIFIER, Bytes.toBytes("Hello, World!")));
    }

    public static void main(String[] args) throws IOException {
        Configuration config = HBaseConfiguration.create();

        try (Connection connection = ConnectionFactory.createConnection(config); Admin admin = connection.getAdmin()) {
            createTable(admin);

            try (Table table = connection.getTable(TABLE_NAME)) {
                putRow(table);
            }
        }
    }
}
```

Use the Apache Thrift Proxy API

The Apache Thrift library provides cross-language client-server remote procedure calls (RPCs), using *Thrift bindings*.

Prepare Thrift server and client before using Thrift Proxy API

A *Thrift binding* is client code generated by the Apache Thrift Compiler for a target language (such as Python) that allows communication between the Thrift server and clients using that client code. HBase includes an Apache Thrift Proxy API, which allows you to write HBase applications in Python, C, C++, or another language that Thrift supports. The Thrift Proxy API is slower than the Java API and may have fewer features. To use the Thrift Proxy API, you need to configure and run the HBase Thrift server on your cluster. You also need to install the [Apache Thrift compiler](#) on your development system.

After the Thrift server is configured and running, generate *Thrift bindings* for the language of your choice, using an IDL file. An HBase IDL file named `HBase.thrift` is included as part of HBase. After generating the bindings, copy the Thrift libraries for your language into the same directory as the generated bindings. In the following Python example, these libraries provide the `thrift.transport` and `thrift.protocol` libraries. These commands show how you might generate the *Thrift bindings* for Python and copy the libraries on a Linux system.

After installation of the thrift compiler, verify that the thrift compiler version is newer than 0.9.0 by running the `thrift -version` command. You need to find the `Hbase.thrift` file from the HBase node or copy it to co-locate with the Thrift compiler. Perform the following steps:

```
mkdir HBaseThrift
cd HBaseThrift/
thrift -gen py /path/to/Hbase.thrift
mv gen-py/* .
rm -rf gen-py/
mkdir thrift
cp -rp ~/Downloads/thrift/lib/py/src/* ./thrift/
```

As a result, the HBase thrift Python bindings appears as follows:

```
HbaseThrift/
|-- hbased
|   |-- constants.py
|   |-- Hbase.py
|   |-- Hbase-remote
|   |-- __init__.py
|   |-- ttypes.py
|-- __init__.py
|-- thrift
|   |-- compat.py
|   |-- ext
|   |   |-- binary.cpp
|   |   |-- binary.h
|   |   |-- compact.cpp
|   |   |-- compact.h
|   |   |-- endian.h
|   |   |-- module.cpp
|   |   |-- protocol.h
|   |   |-- protocol.tcc
|   |   |-- types.cpp
|   |   |-- types.h
|   |-- __init__.py
|   |-- protocol
|   |   |-- __init__.py
|   |   |-- TBase.py
|   |   |-- TBinaryProtocol.py
```

```

|-- TCompactProtocol.py
|-- THeaderProtocol.py
|-- TJSONProtocol.py
|-- TMultiplexedProtocol.py
|-- TProtocolDecorator.py
|-- TProtocol.py
-- server
|-- __init__.py
|-- THttpServer.py
|-- TNonblockingServer.py
|-- TProcessPoolServer.py
|-- TServer.py
-- Thrift.py
-- TMultiplexedProcessor.py
-- transport
|-- __init__.py
|-- sslcompat.py
|-- THeaderTransport.py
|-- THttpClient.py
|-- TSocket.py
|-- TSSLSocket.py
|-- TTransport.py
|-- TTwisted.py
|-- TZlibTransport.py
-- TRecursive.py
-- TCons.py
-- TSerialization.py
-- TTornado.py

```

Introduction to example codes

Choose the right class and functions along with the right configurations for HBase.

Classes and functions

- Transport level: `TBufferedTransport`, `TFrameTransport`, `TSaslTransport`, and `THttpClient`.
- Protocol level: `TBinaryProtocol` and `TCompactProtocol`.

Configurations for HBase thrift

HBase thrift configurations

Property	Default value (secured)	Default value (unsecured)	Description
<code>hbase.thrift.support.proxyuser</code>	true	true	Use this to allow proxy users on the thrift gateway, which is mainly needed for doAs functionality.
<code>hbase.regionserver.thrift.framed</code>	true	true	Use framed transport. When using the <code>THsHaServer</code> or <code>TNonblockingServer</code> , framed transport is always used irrespective of this configuration value.
<code>hbase.regionserver.thrift.compact</code>	true	true	Use the <code>TCompactProtocol</code> instead of the default <code>TBinaryProtocol</code> . <code>TCompactProtocol</code> is a binary protocol that is more compact than the default and typically more efficient.
<code>hbase.regionserver.thrift.http</code>	true	true	Use this to enable HTTP server usage on thrift, which is mainly needed for doAs functionality.

Property	Default value (secured)	Default value (unsecured)	Description
hbase.thrift.security.qop	auth_conf	none	If this is set, HBase Thrift Server authenticates its clients. HBase Proxy User Hosts and Groups must be configured to allow specific users to access HBase through Thrift Server.
hbase.thrift.ssl.enabled	true	false	Encrypt communication between clients and HBase Thrift Server over HTTP using Transport Layer Security (TLS) (formerly known as Secure Socket Layer (SSL)).

Example-1 THttpClient in Secure Cluster

Let us consider that the cluster is secured with the configuration properties mentioned in the *HBase thrift configurations* table under the *Default value (secured)* column.

Before proceeding, ensure that the following applications are installed on your system.

- python 3.6.8 and python 3-devel
- pip 21.3.1
- virtualenv 20.17.1

Perform the following steps:

1. Install virtualenv using pip3.

```
pip3 install virtualenv
```

2. Create a new virtual environment named *PY3ENV*.

```
virtualenv py3env
```

3. Activate the virtual environment.

```
source py3env/bin/activate
```

4. Install the required Python packages and their specific versions. Consider you are inside the python3 virtual environment.

```
pip3 install kerberos==1.3.1 pure-sasl==0.6.2 setuptools==59.6.0 six==1.16.0 wheel==0.37.1
```

This ensures that you have all the necessary dependencies and packages installed to proceed with your project.

```
from thrift.transport import THttpClient
from thrift.protocol import TBinaryProtocol
from hbase.Hbase import Client
from subprocess import call
import ssl
import kerberos
import os

# Get the env parameters
def get_env_params():
    # Replace with your own parameters
    hostname='your_hbase_thrift_hostname'
    cert_file="your_cert_file"
    key_file="your_key_file"
    ca_file="your_ca_file"
    key_pw='your_key_pw'
```

```

keytab_file='your_keytab'
principal = 'your_principal'
return hostname, cert_file, key_file, ca_file, keytab_file, principal, key_pw

# Check if a valid Kerberos ticket is already present in the cache
def check_kerberos_ticket():
    ccache_file = os.getenv('KRB5CCNAME')
    if ccache_file:
        ccache = CCache.load_ccache(ccache_file)
        if ccache.get_principal() and not ccache.get_principal().is_anonymous():
            return True
    return False

# Obtain a Kerberos ticket by running kinit from keytab
def kinit(keytab_file, principal):
    call(['kinit', '-kt', keytab_file, principal])
# Authenticate with Kerberos
def kerberos_auth():
    __, krb_context = kerberos.authGSSClientInit("HTTP")
    kerberos.authGSSClientStep(krb_context, "")
    negotiate_details = kerberos.authGSSClientResponse(krb_context)
    headers = {'Authorization': 'Negotiate ' + negotiate_details, 'Content-Type': 'application/binary'}
    return headers

# Initialize an SSL context with certificate verification enabled
def get_ssl_context():
    ssl_context = ssl.create_default_context()
    ssl_context.load_cert_chain(certfile=cert_file, keyfile=key_file, password=key_pw)
    ssl_context.load_verify_locations(cafile=ca_file)
    return ssl_context
if __name__ == '__main__':
    hostname, cert_file, key_file, ca_file, keytab_file, principal, key_pw = get_env_params()
    # Check if a valid Kerberos ticket is not in the cache, then kinit.
    if not check_kerberos_ticket():
        kinit(keytab_file, principal)

# Create a THttpClient instance with the SSL context and custom headers
httpClient = THttpClient.THttpClient('https://' + hostname + ':9090/', ssl_context=get_ssl_context())
httpClient.setCustomHeaders(headers=kerberos_auth())

# Initialize TBinaryProtocol with THttpClient
protocol = TBinaryProtocol.TBinaryProtocol(httpClient)

# Create HBase client
client = Client(protocol)
# Retrieve list of HBase tables
tables = client.getTableNames()
print(tables)
# Close connection
httpClient.close()

```

Example-2 THttpClient in Unsecured Cluster

Let us consider that the cluster is unsecured with the configuration properties mentioned in the *HBase thrift configurations* table under the *Default value (unsecured)* column.

```
from thrift.transport import THttpClient
```

```

from thrift.protocol import TBinaryProtocol
from hbase.Hbase import Client
# Replace with your own parameters
hostname = 'your_hbase_thrift_server_hostname'

# Initialize THttpClient
httpClient = THttpClient.THttpClient('http://' + hostname + ':9090/')

# Initialize TBinaryProtocol with THttpClient
protocol = TBinaryProtocol.TBinaryProtocol(httpClient)

# Create HBase client
client = Client(protocol)

# Retrieve list of HBase tables
tables = client.getTableNames()
print(tables)

# Close connection
httpClient.close()

```

Example-3 TSaslClientTransport in Secure Cluster without HTTP

If you do not use THttpClient and want to use TSaslClientTransport for legacy compatibility reasons, ensure that you set `hbase.regionserver.thrift.http` property to false. The other settings could be same as the configuration properties mentioned in the *HBase thrift configurations* table under the *Default value (secured)* column.

```

from thrift.transport import TSocket
from thrift.transport import TTransport
from thrift.protocol import TBinaryProtocol
from thrift.protocol import TCompactProtocol
from hbase import Hbase

'''
Assume you already kinit the hbase principal, or you can use the function
in example-1 to kinit.
'''

# Replace with your own parameters
thrift_host = 'your_hbase_thrift_server_hostname'
thrift_port = 9090

# Initialize TSocket and TTransport
socket = TSocket.TSocket(thrift_host, thrift_port)
transport=TTransport.TSaslClientTransport(socket,host=thrift_host,service='hbase',mechanism='GSSAPI')

# Initialize TCompactProtocol with TTransport
protocol = TCompactProtocol.TCompactProtocol(transport)

# Create HBase client
client = Hbase.Client(protocol)

# Open connection and retrieve list of HBase tables
transport.open()
tables = client.getTableNames()
print(tables)

# Close connection
transport.close()

```

Cloudera recommends you to use the HTTP options (Example-1 and Example-2). You can consider the Example-3 for legacy compatibility issues where some old applications might not rewrite the codes. This is because Hue is using

HTTP mode to interact with HBase thrift, and if you disable the HTTP mode, Hue might not work properly with HBase.

Known bugs while using TSaslClientTransport with Kerberos enabled CDP versions

Upstream JIRA [HBASE-21652](#), where a bug is introduced related to Kerberos principal handling. The affected versions are CDP 7.1.6 and earlier. The versions containing the fix are 7.1.7, 7.2.11, and later.

Related Information

[Using the HBase Thrift Interface, Part 1](#)

[Using the HBase Thrift Interface, Part 2](#)

[Python interaction with HBase Thrift proxy in Secured Cluster](#)

[Apache Thrift document](#)

Use the Hue HBase app

Hue is a web-based interactive query editor that enables you to interact with data warehouses. You can use the HBase Browser application in Hue to create and browse HBase tables.

The HBase Hue app enables you to insert a new row or bulk upload CSV files, TSV files, and type data into your table. You can also insert columns into your row. If you need more control or data about your cell, you can use the full editor to edit a cell.

The screenshot displays the Hue HBase app interface. On the left, a sidebar shows a list of tables: customers, hbase_table_1, hbase_table_2, key (int), value (string), hbase_table_3, sample_07, sample_08, transactions1g, and web_logs. The main area shows a table with columns 'cf1: val' and 'cf1: purchase'. The table contains three rows of data: Krishna with value 12, Eva with value 1000, and Anna with value null. The interface includes a search bar at the top, a 'Query' dropdown, and a 'Switch Cluster' button. At the bottom, there are buttons for 'Drop Rows', 'Bulk Upload', and 'New Row'.

cf1: val	cf1: purchase
Krishna	12
Eva	1000
Anna	

Fetches 10 entries starting from null in 13.397 seconds.

If you are using the HBase Thrift interface, Hue fits in between the Thrift Server and the HBase client, and the Thrift Server assumes that all HBase operations come from the hue user and not the client. To ensure that users in Hue are only allowed to perform HBase operations assigned to their own credentials, and not those of the hue user, you must enable doAs Impersonation for the HBase Browser Application.