HBase Commands

Learning outcomes:

This document explains the basic commands of Hbase. The various basic command line data definition and data manipulation commands are introduced in the document. At the end of the activity, you should be able run basic commands in Hbase and understand tables, column families & columns in Hbase.

Overview

The HBase Shell is the command-line interface to your HBase cluster(s).The shell provides both client and administrative operations, mirroring.

1. Starting the hbase shell

```
hbase shell
```

```
[yugandhar@v ~]$ hbase shell
HBase Shell; enter 'help<RETURN>' for list of supported commands.
Type "exit<RETURN>" to leave the HBase Shell
Version 1.1.2.2.6.1.0-129, r718c773662346de98a8ce6fd3b5f64e279cb87d4, Wed May 31 03:2
7:31 UTC 2017
```

The shell is based on JRuby, the Java Virtual Machine-based implementation of Ruby. More specifically, it uses the Interactive Ruby Shell (IRB), which is used to enter Ruby commands and get an immediate response. HBase ships with Ruby scripts that extend the IRB with specific commands, related to the Java-based APIs.

2. Exiting the hbase shell

```
hbase(main):001:0> exit
[yugandhar@v ~]$
```

General Commands:

Status

Returns various levels of information contained in the ClusterStatus class.

```
hbase(main):008:0> status
1 active master, 0 backup masters, 20 servers, 1 dead, 0.3000 average load
```

version

Returns the current version, repository revision, and compilation date of the HBase cluster.

```
hbase(main):009:0> version
1.1.2.2.6.1.0-129, r718c773662346de98a8ce6fd3b5f64e279cb87d4, Wed May 31 03:27:31 UTC
 2017
```

whoami

The command shows the current OS user and group membership known to hbase about the shell user.

```
hbase(main):010:0> whoami
yugandhar (auth:SIMPLE)
    groups: yugandhar
```

Data Definition Commands:

create

Creates a new table

Ex: Create a new table ‘employee’ with column families ‘personal’ and ‘professional’

```
create ‘employee’,’personal’,’professional’
```

OR

```
hbase(main):001:0> create 'employee', {NAME => 'personal'}, {NAME => 'professional'}
0 row(s) in 5.2020 seconds

=> Hbase::Table - employee
```

“Personal” and “Professional” in the above statement are column families

describe

Describes the table by providing the information of the table and the column families.

```
hbase(main):003:0> describe 'employee'
Table employee is ENABLED
employee
COLUMN FAMILIES DESCRIPTION
{NAME => 'personal', BLOOMFILTER => 'ROW', VERSIONS => '5', IN_MEMORY => 'false', KEE
P_DELETED_CELLS => 'FALSE', DATA_BLOCK_ENCODING => 'NONE', TTL => 'FOREVER', COMPRESS
ION => 'NONE', MIN_VERSIONS => '0', BLOCKCACHE => 'true', BLOCKSIZE => '65536', REPLI
CATION_SCOPE => '0'}
{NAME => 'professional', BLOOMFILTER => 'ROW', VERSIONS => '1', IN_MEMORY => 'false',
 KEEP_DELETED_CELLS => 'FALSE', DATA_BLOCK_ENCODING => 'NONE', TTL => 'FOREVER', COMP
RESSION => 'NONE', MIN_VERSIONS => '0', BLOCKCACHE => 'true', BLOCKSIZE => '65536', R
EPLICATION_SCOPE => '0'}
2 row(s) in 0.1610 seconds
```

`disable`

disable an existing HBase table. Disabled tables will not be deleted from HBase but they are not available for regular access. This table is excluded from the list command and we cannot run any other command except either enable or drop commands on disabled tables. Disabling is like deleting the tables temporarily.

```
hbase(main):004:0> disable 'employee'
0 row(s) in 2.3670 seconds
```

`enable`

Used to enable a table which might be currently disabled.

```
hbase(main):005:0> enable 'employee'
0 row(s) in 2.4660 seconds
```

`disable_all`

Disable all the tables matching the given regular expression.

```
hbase(main):006:0> disable_all 'emp.*'
employee

Disable the above 1 tables (y/n)?
y
1 tables successfully disabled
```

`enable_all`

Enable all the tables matching the given regex

```
hbase(main):007:0> enable_all 'emp.*'
employee

Enable the above 1 tables (y/n)?
y
1 tables successfully enabled
```

`exists`

To check the existence of an Hbase Table.

```
hbase(main):018:0> exists 'employee'
Table employee does exist

0 row(s) in 0.0190 seconds
```

`is_disabled`

To know whether an HBase table is disabled or not.

```
hbase(main):017:0> is_disabled 'employee'
false
0 row(s) in 0.0210 seconds
```

`is_enabled`

To know whether an HBase table is enabled or not.

```
hbase(main):020:0> is_enabled 'employee'
true
0 row(s) in 0.0210 seconds
```

`list`

List all the tables in hbase. Optional regular expression parameter could be used to filter the output.

```
hbase(main):021:0> list
TABLE
ATLAS_ENTITY_AUDIT_EVENTS
atlas_titan
employee
3 row(s) in 0.0180 seconds

=> ["ATLAS_ENTITY_AUDIT_EVENTS", "atlas_titan", "employee"]
```

`alter`

add/modify/delete column families, as well as change table configuration.

Add/Change column family:
to change or add the ‘account’ column family in table ‘employee’ from current value to keep a maximum of 5 cell VERSIONS, do:

```
hbase(main):027:0* alter 'employee', NAME=>'account',VERSIONS => 5
Updating all regions with the new schema...
1/1 regions updated.
Done.
0 row(s) in 2.8270 seconds
```

Delete the column family from the table employee.

To delete the ‘account’ column family in table ‘employee’, use one of:

```
hbase(main):028:0> alter 'employee', 'delete' => 'account'
Updating all regions with the new schema...
0/1 regions updated.
1/1 regions updated.
Done.
0 row(s) in 4.0640 seconds
```

Data Manipulation Commands:

append

Appends a cell ‘value’ at specified table/row/column coordinates.

```
hbase(main):029:0> append 'employee','101','personal:address','hyderabad',ATTRIBUTES=
>{'area'=>'Gachibowli'}
0 row(s) in 0.1610 seconds
```

count

Count the number of rows in a table. Return value is the number of rows. This operation may take a long time.

```
hbase(main):030:0> count 'employee'
1 row(s) in 0.0660 seconds

=> 1
```

put

Put a cell ‘value’ at specified table/row/column and optionally timestamp coordinates.

```
hbase(main):031:0> put 'employee','101','personal:id','104'
0 row(s) in 0.0920 seconds

hbase(main):032:0> put 'employee','104','personal:id','106'
0 row(s) in 0.0220 seconds
```

scan

Scans the table and shows the column families, columns associated with the row key.

```
hbase(main):033:0> scan 'employee'
ROW                   COLUMN+CELL
 101                  column=personal:address, timestamp=1502433662795, value=hydera
                      bad
 101                  column=personal:id, timestamp=1502433742900, value=104
 104                  column=personal:id, timestamp=1502433775192, value=106
2 row(s) in 0.0340 seconds
```

get

Get row or cell contents. Pass table name, row, and optionally a dictionary of column(s), timestamp, timerange and versions.

```
hbase(main):034:0> get 'employee','101'
COLUMN                  CELL
 personal:address       timestamp=1502433662795, value=hyderabad
 personal:id            timestamp=1502433742900, value=104
2 row(s) in 0.0510 seconds

hbase(main):035:0> get 'employee','1'
COLUMN                  CELL
0 row(s) in 0.0140 seconds

hbase(main):036:0> get 'employee','101',{COLUMNS =>'personal:id'}
COLUMN                  CELL
 personal:id            timestamp=1502433742900, value=104
1 row(s) in 0.0220 seconds
```

```
hbase(main):037:0> put 'employee','101','personal:name','Kamlesh'
0 row(s) in 0.0460 seconds

hbase(main):038:0> put 'employee','101','personal:profession','Data Scientist'
0 row(s) in 0.0200 seconds

hbase(main):039:0> get 'employee','101',{COLUMN=>['personal:id','personal:name','pers
onal:profession']}
COLUMN                  CELL
 personal:id            timestamp=1502433742900, value=104
 personal:name          timestamp=1502433992093, value=Kamlesh
 personal:profession    timestamp=1502434022063, value=Data Scientist
3 row(s) in 0.0300 seconds
```

Changing the values for ‘personal’ column family and the column name ‘profession’

```
hbase(main):040:0> put 'employee','101','personal:profession','Sr.Data Scientist'
0 row(s) in 0.0250 seconds

hbase(main):041:0> scan 'employee'
ROW                     COLUMN+CELL
 101                    column=personal:address, timestamp=1502433662795, value=hydera
                        bad
 101                    column=personal:id, timestamp=1502433742900, value=104
 101                    column=personal:name, timestamp=1502433992093, value=Kamlesh
 101                    column=personal:profession, timestamp=1502434256708, value=Sr.
                        Data Scientist
 104                    column=personal:id, timestamp=1502433775192, value=106
2 row(s) in 0.0370 seconds
```

Alter the ‘employee’ table with 5 versions on the ‘personal’ column family.

```
hbase(main):042:0> alter 'employee',NAME => 'personal' , VERSIONS => 5
Updating all regions with the new schema...
0/1 regions updated.
1/1 regions updated.
Done.
0 row(s) in 3.9300 seconds

hbase(main):043:0> put 'employee','101','personal:profession','Senior Data Scientist'
0 row(s) in 0.0270 seconds

hbase(main):044:0> put 'employee','101','personal:profession','Senior DS'
0 row(s) in 0.0210 seconds

hbase(main):045:0> put 'employee','101','personal:profession','Sr Data Scientist'
0 row(s) in 0.0230 seconds

hbase(main):046:0> scan 'employee',{COLUMN=>'personal:profession',VERSIONS=>3}
ROW                   COLUMN+CELL
 101                  column=personal:profession, timestamp=1502434473321, value=Sr
                      Data Scientist
```

truncate

Disables, drops and recreates the specified table.

```
hbase(main):047:0> truncate 'employee'
Truncating 'employee' table (it may take a while):
 - Disabling table...
 - Truncating table...
0 row(s) in 5.5750 seconds

hbase(main):048:0> scan 'employee'
ROW                   COLUMN+CELL
0 row(s) in 0.3780 seconds
```

Bulk Load in HBase.

To bulk upload create the following table ‘Blog’, column families ‘info’ and ‘content’.

```
[yugandhar@v ~]$ hbase shell
HBase Shell; enter 'help<RETURN>' for list of supported commands.
Type "exit<RETURN>" to leave the HBase Shell
Version 1.1.2.2.6.1.0-129, r718c773662346de98a8ce6fd3b5f64e279cb87d4, Wed May 31 03:2
7:31 UTC 2017

hbase(main):001:0> create 'Blog',{NAME => 'info'},{NAME => 'content'}
0 row(s) in 2.6430 seconds

=> Hbase::Table - Blog
```

Import the data(present in HDFS) into the table using the following command.

```
hbase org.apache.hadoop.hbase.mapreduce.ImportTsv -
Dimporttsv.columns=HBASE_ROW_KEY,info:title,info:author,info:
date,content:post Blog /user/yugandhar/bulkload.txt
```

```
[yugandhar@v ~]$ hbase org.apache.hadoop.hbase.mapreduce.ImportTsv -Dimporttsv.columns=HBASE_ROW_KEY,info:title,info:author,info:date,content:post Blog /user/yugandhar/bulkload.txt
2017-08-11 12:33:35,482 INFO  [main] zookeeper.RecoverableZooKeeper: Process identifier=hconnection-0x776b83cc connecting to ZooKeeper ensemble=b.insofe.edu.in:2181,c.insofe.edu.in:2181,a.insofe.edu.in:2181,e.insofe.edu.in:2181,f.insofe.edu.in:2181
2017-08-11 12:33:35,489 INFO  [main] zookeeper.ZooKeeper: Client environment:zookeeper.version=3.4.6-129--1, built on 05/31/2017 03:01 GMT
2017-08-11 12:33:35,489 INFO  [main] zookeeper.ZooKeeper: Client environment:host.name=v.insofe.edu.in
2017-08-11 12:33:35,489 INFO  [main] zookeeper.ZooKeeper: Client environment:java.version=1.8.0_112
2017-08-11 12:33:35,489 INFO  [main] zookeeper.ZooKeeper: Client environment:java.vendor=Oracle Corporation
2017-08-11 12:33:35,489 INFO  [main] zookeeper.ZooKeeper: Client environment:java.home=/usr/jdk64/jdk1.8.0_112/jre
2017-08-11 12:33:35,489 INFO  [main] zookeeper.ZooKeeper: Client environment:java.class.path=/usr/hdp/2.6.1.0-129/hbase/conf:/usr/jdk64/jdk1.8.0_112/lib/tools.jar:/usr/hdp/2.6.1.0-129/hbase:/usr/hdp/2.6.1.0-129/hbase/lib/activation-1.1.jar:/usr/hdp/2.6.1.0-129/hbase/lib/aopalliance-1.0.jar:/usr/hdp/2.6.1.0-129/hbase/lib/apacheds-i18n-2.0.0-M15.jar:/usr/hdp/2.6.1.0-129/hbase/lib/apacheds-kerberos-codec-2.0.0-M15.jar:/usr/hdp/2.6.1.0-129/hbase/lib/api-asn1-api-1.0.0-M20.jar:/usr/hdp/2.6.1.0-129/hbase/lib/ap
```

After completion of the bulk upload check the tables.

```
hbase(main):002:0> list
TABLE
ATLAS_ENTITY_AUDIT_EVENTS
Blog
atlas_titan
employee
4 row(s) in 0.0570 seconds

=> ["ATLAS_ENTITY_AUDIT_EVENTS", "Blog", "atlas_titan", "employee"]
```