After successful installation of HBase, we get an interactive shell to execute various commands. The Apache HBase Shell is (J)Ruby's IRB with some HBase particular commands added. Anything you can do in IRB, you should be able to do in the HBase Shell. To run the HBase shell, do as follows:

```
$ ./bin/hbase shell
```

Type `help` and then `<RETURN>` to see a listing of shell commands and options.

HBase shell commands

General HBase shell commands

Status

This command will display details about the system status like a number of servers present in the cluster, active server count, and average load value. You can also pass parameters depending on how detailed status you want. The parameters can be 'summary', 'simple', or 'detailed'. The default is ‘summary’.

Its syntax is as follows:

```
hbase(main):003:0> status 'simple'
hbase(main):004:0> status 'detailed'
```

- ## version

  This command will display the currently used HBase version

```
hbase(main):005:0> version
```


- ## Table help

  This command guides you what and how to use table-referenced commands. It will give table manipulations commands like put, get and all other commands information.

```
hbase(main):006:0> table_help
```

- ## whoami

  Shows the current hbase user.

```
hbase(main):007:0> whoami
```

HBase Table Management commands

- ## create

  To create a table use the create command, specifying the table name and the Column Family name. The syntax to create a table in HBase shell is shown below.

```
create '<table name>','<column family>'
```

  Example : Lets create a table named student. It has two column families: “name” and “id”.

```
hbase(main):001:0> create 'student', 'name', 'id'
```

```
0 row(s) in 1.9820 seconds
```

- ## list

  "list" command will display all the tables that are present or created in HBase

  ```
  hbase(main):001:0> list
  ```

  Note: We can filter output values from tables by passing optional regular expression parameters

- ## describe

  As the name suggests, it will give information about table name with column families, associated filters, versions and some more details.

  ```
  describe '<table name>'
  ```

- ## disable

  If table needs to be deleted or dropped, it has to disable first, and `disable` command will disable the named table.

  ```
  disable '<table name>'
  ```

- ## disable_all

  This command will disable all the tables matching the given regex.

  ```
  disable_all "matching regex"
  ```

  Example: disable all tables with names starting with 's'

  ```
  hbase(main):003:0> disable_all 's.*'
  ```

- ## enable

  This command will enabled the named table

  ```
  enable '<table name>'
  ```

- ## show_filters

  This command shows all the filters present in HBase.

- ## drop

  This command is used to drop the named table. Table must first be disabled.

  ```
  drop '<table name>'
  ```

  Example:

  ```
  hbase(main):007:0> drop 'student'
  ```


- ## drop_all

  This command is used to drop all of the tables matching the given regex

  ```
  drop_all "matching regex"
  ```

  Example: drop all tables with names starting with 's'

  ```
  hbase(main):009:0> drop_all 's.*'
  ```

- ## is_enabled

  is_enabled command will check either the table is enabled or not.

  ```
  is_enabled '<table name>'
  ```

- ## exists

  This command checks whether the named table exists or not

  ```
  exists '<table name>'
  ```

- ## alter

  This command alters the column family schema. You can pass table name and a dictionary specifying new column family schema.

  Examples:

  To change the column family from col1 to col2 in a table called table1, use this

  ```
  hbase(main):010:0> alter 'table1', NAME=>'col2', VERSIONS=>3
  ```

```
hbase(main):011:0> alter 'table1', {NAME=>'col2', VERSIONS=>3},{NAME=>'c
ol3', VERSIONS=>5}
```

And if you want to delete the column name 'col3', then:

```
hbase(main):011:0> alter 'table1', 'delete' =>' col3'
```

- ## alter_status

Use this command, to get the status of the alter command, which shows the number of regions of the table that have received the updated schema pass table name

```
alter_status '<table name>'
```


Data manipulation commands

- ## count

count command will retrieve the count of a number of rows in a table. Current count is shown every 1000 rows by default. Count interval may be optionally specified. Count command will work fast when it is configured with right Cache. Example

```
hbase(main):01:0>count 'corejavaguru', INTERVAL => 100000
hbase(main):02:0> count 'corejavaguru', CACHE=> 1000
hbase(main):03:0> count 'corejavaguru', INTERVAL =>10, CACHE=> 1000
```

It's quite fast when configured with the right CACHE.

```
hbase> count '<tablename>', CACHE => 1000
```

The above count fetches 1000 rows at a time. Set CACHE lower if your rows are big. Default is to fetch one row at a time.

- ## Put

Using this command you can put a cell 'value' at specified table/row/column and optionally timestamp coordinates. To put a cell value into table 'table1' at row 'row1' under column 'col1' marked with the time 'tsp1', do:

- ## get

  Use get command to get row or cell contents. You can pass table name, row, and optionally a dictionary of column(s), timestamp, timerange and versions to the command. Examples:

```
hbase(main):05:0> get 't1', 'r1', 'c1'
hbase(main):06:0> get 't1', 'r1', 'c1', 'c2'
hbase(main):07:0> get 't1', 'r1', ['c1', 'c2']
hbase(main):08:0> get 't1', 'r1', {TIMERANGE => [ts1, ts2]}
hbase(main):09:0> get 't1', 'r1', {COLUMN => 'c1'}
hbase(main):10:0> get 't1', 'r1', {COLUMN => ['c1', 'c2', 'c3']}
hbase(main):11:0> get 't1', 'r1', {COLUMN => 'c1', TIMESTAMP => ts1}
hbase(main):12:0> get 't1', 'r1', {COLUMN => 'c1', TIMERANGE => [ts1, ts
2], VERSIONS => 4}
hbase(main):13:0> get 't1', 'r1', {COLUMN => 'c1', TIMESTAMP => ts1, VER
SIONS => 4}
hbase(main):14:0> get 't1', 'r1', {FILTER => "ValueFilter(=, 'binary:abc
')"}
hbase(main):15:0> get 't1', 'r1'
```

- ## delete

  This command can be used to delete cell value at specified table/row/column and optionally timestamp coordinates. To delete a cell from 'table1' at row 'row1' under column 'col1' marked with the time 'ts1', do

```
hbase(main):16:0> delete 't1', 'r1', 'c1', ts1
```

- ## deleteall

  delete command deletes all cells in a given row. You can pass a table name, row, and optionally a column and timestamp to the command. Examples:

```
hbase(main):17:0> deleteall 't1', 'r1'
hbase(main):18:0> deleteall 't1', 'r1', 'c1'
hbase(main):19:0> deleteall 't1', 'r1', 'c1', ts1
```

- ## truncate

  truncate command disables, drops and recreates the specified table.

```
hbase(main):20:0> truncate 'table1'
```

This command increments a cell 'value' at specified table/row/column coordinates. To increment a cell value in table 't1' at row 'r1' under column 'c1' by 1 (can be omitted) or 10 do:

```
hbase(main):21:0> incr 't1', 'r1', 'c1'
hbase(main):22:0> incr 't1', 'r1', 'c1', 1
hbase(main):23:0> incr 't1', 'r1', 'c1', 10
```

- ## scan

This command scans entire table and displays the table contents. You can pass table name and optionally a dictionary of scanner specifications. Scanner specifications may include one or more of: TIMERANGE, FILTER, LIMIT, STARTROW, STOPROW, TIMESTAMP, MAXLENGTH etc.

```
hbase(main):24:0> scan  'table1'
```


Cluster Replication Commands

- ## add_peer

Add peers to cluster to replicate using this command. Example:

```
hbase(main):01:0> add_peer '1', "server1.corejavaguru.com:2181:/hbase"
```

- ## remove_peer

remove_peer command stops the specified replication stream and deletes all the meta information kept about it. Example:

```
hbase(main):02:0> remove_peer '1'
```

- ## list_peers

This command lists all the replication peer clusters

- ## enable_peer

  enable_peer command restarts the replication to the specified peer cluster

```
hbase(main):04:0> enable_peer '1'
```

- ## disable_peer

  disable_peer command Stops the replication stream to the specified cluster, but still keeps track of new edits to replicate.

```
hbase(main):05:0> disable_peer '1'
```

- ## start_replication

  Restarts all the replication features.

```
hbase(main):06:0> start_replication
```

- ## stop_replication

  tops all the replication feature.

```
hbase(main):07:0> stop_replication
```

  Note: start/stop replication is only meant to be used in critical load situations.

Security Commands

The HBase shell has been extended to provide simple commands for editing and updating user permissions. The following commands have been added for access control list management:

- ## grant

  This command grants specific rights such as read, write, execute, create and admin on a table for a certain user. The syntax of grant command is as follows:

<permissions> is zero or more letters from the set "RWCA": READ('R'), WRITE('W'), CREATE('C'), ADMIN('A'). Given below is an example that grants all privileges to a user named corejavaguru.

```
hbase(main):01:0> grant 'corejavaguru', 'RWXCA'
```

- ## revoke

This command is used to revoke a user's access rights. Syntax is as below:

```
revoke <user> <table> [ <column family> [ <column qualifier> ] ]
```

Example:

```
hbase(main):02:0> revoke 'corejavaguru'
```

- ## user_permission

The `user_permission` command shows all access permissions for the current user for a given table:

```
user_permission <table>
```

JAVA

BIG DATA

← Previous
HBase Configuration

Next →
HBase-Java API Class

about | Contact Us