— Learn HBase

Welcome to Apache HBase blog.

Home About

Search

March 2, 2013

Uncategorized

39 Comments

HBase shell commands

As told in HBase introduction, HBase provides Extensible jruby-based (JIRB) shell as a feature to execute some commands(each command represents one functionality).

HBase shell commands are mainly categorized into 6 parts

1) General HBase shell commands

status	Show cluster status. Can be 'summary', 'simple', or 'detailed'. The default is 'summary'. **hbase> status** **hbase> status 'simple'** **hbase> status 'summary'** **hbase> status 'detailed'**
version	Output this HBase version**Usage:** **hbase> version**
whoami	Show the current hbase user.**Usage:** **hbase> whoami**

2) Tables Management commands

alter	Alter column family schema; pass table name and a dictionary specifying new column family schema. Dictionaries are described on the main help command output. Dictionary must include name of column family to alter.For example, to change or add the 'f1' column family in 't1' from current value to keep a maximum of 5 cell VERSIONS, do: **hbase> alter 't1', NAME => 'f1', VERSIONS => 5** You can operate on several column families: **hbase> alter 't1', 'f1', {NAME => 'f2', IN_MEMORY => true}, {NAME => 'f3', VERSIONS => 5}** To delete the 'f1' column family in table 't1', use one of:hbase> alter 't1', NAME = 'f1', METHOD => 'delete' **hbase> alter 't1', 'delete' => 'f1'** You can also change table-scope attributes like MAX_FILESIZE, READONLY, MEMSTORE_FLUSHSIZE, DEFERRED_LOG_FLUSH, etc. These can be put at th end; for example, to change the max size of a region to 128MB, do: **hbase> alter 't1', MAX_FILESIZE => '134217728'** You can add a table coprocessor by setting a table coprocessor attribute: **hbase> alter 't1', 'coprocessor'=>'hdfs:///foo.jar|com.foo.FooRegionObserver|1001|arg1=1,ar**

Search

Recent Posts

HBase shell commands
HBase introduction

Recent Comments

rajeshhcu32 on HBase shell commands

Leviya on HBase shell commands

Nagesh Kumar on HBase shell commands

siva on HBase shell commands

DineshKumar on HBase shell commands

Archives

March 2013

Categories

Uncategorized

Meta

Register
Log in
Entries RSS
Comments RSS
WordPress.com

	Since you can have multiple coprocessors configured for a table, a sequence number will be automatically appended to the attribute name to uniquely identify it. The coprocessor attribute must match the pattern below in order for the framework to understand how to load the coprocessor classes: [coprocessor jar file location] | class name | [priority] | [arguments] You can also set configuration settings specific to this table or column family: **hbase> alter 't1', CONFIGURATION => {'hbase.hregion.scan.loadColumnFamiliesOnDemand' => 'true'}** **hbase> alter 't1', {NAME => 'f2', CONFIGURATION => {'hbase.hstore.blockingStoreFiles' => '10'}}** You can also remove a table-scope attribute: **hbase> alter 't1', METHOD => 'table_att_unset', NAME => 'MAX_FILESIZE'** **hbase> alter 't1', METHOD => 'table_att_unset', NAME => 'coprocessor$1'** There could be more than one alteration in one command: **hbase> alter 't1', { NAME => 'f1', VERSIONS => 3 }, { MAX_FILESIZE => '134217728' }, { METHOD => 'delete', NAME => 'f2' }, OWNER => 'johndoe', METADATA => { 'mykey' => 'myvalue' }**
create	Create table; pass table name, a dictionary of specifications per column family, and optionally a dictionary of table configuration. **hbase> create 't1', {NAME => 'f1', VERSIONS => 5}** **hbase> create 't1', {NAME => 'f1'}, {NAME => 'f2'}, {NAME => 'f3'}** **hbase> # The above in shorthand would be the following:** **hbase> create 't1', 'f1', 'f2', 'f3'** **hbase> create 't1', {NAME => 'f1', VERSIONS => 1, TTL => 2592000, BLOCKCACHE => true}** **hbase> create 't1', {NAME => 'f1', CONFIGURATION => {'hbase.hstore.blockingStoreFiles' => '10'}}** Table configuration options can be put at the end.
describe	Describe the named table. **hbase> describe 't1'**
disable	Start disable of named table **hbase> disable 't1'**
disable_all	Disable all of tables matching the given regex **hbase> disable_all 't.*'**
is_disabled	verifies Is named table disabled **hbase> is_disabled 't1'**
drop	Drop the named table. Table must first be disabled **hbase> drop 't1'**
drop_all	Drop all of the tables matching the given regex **hbase> drop_all 't.*'**
enable	Start enable of named table **hbase> enable 't1'**

enable_all	Enable all of the tables matching the given regex **hbase> enable_all 't.*'**
is_enabled	verifies Is named table enabled **hbase> is_enabled 't1'**
exists	Does the named table exist **hbase> exists 't1'**
list	List all tables in hbase. Optional regular expression parameter could be used to filter the output **hbase> list** **hbase> list 'abc.*'**
show_filters	Show all the filters in hbase. **hbase> show_filters**
alter_status	Get the status of the alter command. Indicates the number of regions of the tabl have received the updated schema Pass table name. **hbase> alter_status 't1'**
alter_async	Alter column family schema, does not wait for all regions to receive the schema changes. Pass table name and a dictionary specifying new column family schema. Dictionaries are described on the main help command output. Dictionary must include name of column family to alter. To change or add the 'f1' column family in table 't1' from defaults to instead keep a maximum of 5 cell VERSIONS, do:hbase> alter_async 't1', NA => 'f1', VERSIONS => 5To delete the 'f1' column family in table 't1', do: **hbase> alter_async 't1', NAME => 'f1', METHOD => 'delete'or a shorter version:hbase> alter_async 't1', 'delete' => 'f1'** You can also change table-scope attributes like MAX_FILESIZE MEMSTORE_FLUSHSIZE, READONLY, and DEFERRED_LOG_FLUSH. For example, to change the max size of a family to 128MB, do: **hbase> alter 't1', METHOD => 'table_att', MAX_FILESIZE => '134217728'** There could be more than one alteration in one command: **hbase> alter 't1', {NAME => 'f1'}, {NAME => 'f2', METHOD => 'delete'}** To check if all the regions have been updated, use alter_status <table_name>

3) Data Manipulation commands

count	Count the number of rows in a table. Return value is the number of rows. This operation may take a LONG time (Run '$HADOOP_HOME/bin/hadoop jar hbase.jar rowcount' to run a counting mapreduce job). Current count is shown every 1000 rows by default. Count interval may be optionally specified. Scan caching is enabled on count scans by default. Default cache size is 10 rows. If your rows are small in size, you may want to increase this parameter. Examples:hbase> count 't1' **hbase> count 't1', INTERVAL => 100000** **hbase> count 't1', CACHE => 1000** **hbase> count 't1', INTERVAL => 10, CACHE => 1000** The same commands also can be run on a table reference. Suppose you had a reference t to table 't1', the corresponding commands would be:hbase> t.count

	hbase> t.count INTERVAL => 100000 **hbase> t.count CACHE => 1000** **hbase> t.count INTERVAL => 10, CACHE => 1000**
delete	Put a delete cell value at specified table/row/column and optionally timestamp coordinates. Deletes must match the deleted cell's coordinates exactly. When scanning, a delete cell suppresses older versions. To delete a cell from 't1' at row 'r1' under column 'c1' marked with the time 'ts1', do: **hbase> delete 't1', 'r1', 'c1', ts1** The same command can also be run on a table reference. Suppose you had a reference t to table 't1', the corresponding command would be:hbase> t.delete 'r1', 'c1', ts1
deleteall	Delete all cells in a given row; pass a table name, row, and optionally a column and timestamp. Examples:hbase> deleteall 't1', 'r1' **hbase> deleteall 't1', 'r1', 'c1'** **hbase> deleteall 't1', 'r1', 'c1', ts1** The same commands also can be run on a table reference. Suppose you had a reference t to table 't1', the corresponding command would be:hbase> t.deleteall 'r1' **hbase> t.deleteall 'r1', 'c1'** **hbase> t.deleteall 'r1', 'c1', ts1**
get	Get row or cell contents; pass table name, row, and optionally a dictionary of column(s), timestamp, timerange and versions. Examples: **hbase> get 't1', 'r1'** **hbase> get 't1', 'r1', {TIMERANGE => [ts1, ts2]}** **hbase> get 't1', 'r1', {COLUMN => 'c1'}** **hbase> get 't1', 'r1', {COLUMN => ['c1', 'c2', 'c3']}** **hbase> get 't1', 'r1', {COLUMN => 'c1', TIMESTAMP => ts1}** **hbase> get 't1', 'r1', {COLUMN => 'c1', TIMERANGE => [ts1, ts2], VERSIONS => 4}** **hbase> get 't1', 'r1', {COLUMN => 'c1', TIMESTAMP => ts1, VERSIONS => 4}** **hbase> get 't1', 'r1', {FILTER => "ValueFilter(=, 'binary:abc')"}** **hbase> get 't1', 'r1', 'c1'** **hbase> get 't1', 'r1', 'c1', 'c2'** **hbase> get 't1', 'r1', ['c1', 'c2']** Besides the default 'toStringBinary' format, 'get' also supports custom formatting by column. A user can define a FORMATTER by adding it to the column name in the get specification. The FORMATTER can be stipulated:1. either as a org.apache.hadoop.hbase.util.Bytes method name (e.g, toInt, toString) 2. or as a custom class followed by method name: e.g. 'c(MyFormatterClass).format'.Example formatting cf:qualifier1 and cf:qualifier2 both as Integers: hbase> get 't1', 'r1' {COLUMN => ['cf:qualifier1:toInt', 'cf:qualifier2:c(org.apache.hadoop.hbase.util.Bytes).toInt'] } Note that you can specify a FORMATTER by column only (cf:qualifer). You cannot specify a FORMATTER for all columns of a column family.The same commands also can be run on a reference to a table (obtained via get_table or create_table). Suppose you had a reference t to table 't1', the corresponding commands would be: **hbase> t.get 'r1'** **hbase> t.get 'r1', {TIMERANGE => [ts1, ts2]}** **hbase> t.get 'r1', {COLUMN => 'c1'}** **hbase> t.get 'r1', {COLUMN => ['c1', 'c2', 'c3']}** **hbase> t.get 'r1', {COLUMN => 'c1', TIMESTAMP => ts1}** **hbase> t.get 'r1', {COLUMN => 'c1', TIMERANGE => [ts1, ts2],**

	VERSIONS => 4} **hbase> t.get 'r1', {COLUMN => 'c1', TIMESTAMP => ts1, VERSIONS => 4}** **hbase> t.get 'r1', {FILTER => "ValueFilter(=, 'binary:abc')"}** **hbase> t.get 'r1', 'c1'** **hbase> t.get 'r1', 'c1', 'c2'** **hbase> t.get 'r1', ['c1', 'c2']**
get_counter	Return a counter cell value at specified table/row/column coordinates. A cell cell should be managed with atomic increment function oh HBase and the data should be binary encoded. Example: **hbase> get_counter 't1', 'r1', 'c1'** The same commands also can be run on a table reference. Suppose you had a reference t to table 't1', the corresponding command would be: **hbase> t.get_counter 'r1', 'c1'**
incr	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> incr 't1', 'r1', 'c1'** **hbase> incr 't1', 'r1', 'c1', 1** **hbase> incr 't1', 'r1', 'c1', 10** The same commands also can be run on a table reference. Suppose you had a reference t to table 't1', the corresponding command would be:hbase> t.incr 'r1', 'c1' **hbase> t.incr 'r1', 'c1', 1** **hbase> t.incr 'r1', 'c1', 10**
put	Put a cell 'value' at specified table/row/column and optionally timestamp coordinates. To put a cell value into table 't1' at row 'r1' under column 'c1' marked with the time 'ts1', do: **hbase> put 't1', 'r1', 'c1', 'value', ts1** The same commands also can be run on a table reference. Suppose you had a reference t to table 't1', the corresponding command would be: **hbase> t.put 'r1', 'c1', 'value', ts1**
scan	Scan a table; 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, or COLUMNS, CACHEIf no columns are specified, all columns will be scanned. To scan all members of a column family, leave the qualifier empty as in 'col_family:'.The filter can be specified in two ways: 1. Using a filterString – more information on this is available in the Filter Language document attached to the HBASE-4176 JIRA 2. Using the entire package name of the filter.Some examples:hbase> scan '.META.' **hbase> scan '.META.', {COLUMNS => 'info:regioninfo'}** **hbase> scan 't1', {COLUMNS => ['c1', 'c2'], LIMIT => 10, STARTROW => 'xyz'}** **hbase> scan 't1', {COLUMNS => 'c1', TIMERANGE => [1303668804, 1303668904]}** **hbase> scan 't1', {FILTER => "(PrefixFilter ('row2') AND (QualifierFilter (>=, 'binary:xyz'))) AND (TimestampsFilter (123, 456))"}** **hbase> scan 't1', {FILTER => org.apache.hadoop.hbase.filter.ColumnPaginationFilter.new(1, 0)}** For experts, there is an additional option — CACHE_BLOCKS — which switches block caching for the scanner on (true) or off (false). By default it is enabled. Examples:hbase> scan 't1', {COLUMNS => ['c1', 'c2'], CACHE_BLOCKS => false} Also for experts, there is an advanced option — RAW — which instructs the scanner to return all cells (including delete markers and uncollected deleted

	cells). This option cannot be combined with requesting specific COLUMNS. Disabled by default. Example: **hbase> scan 't1', {RAW => true, VERSIONS => 10}** Besides the default 'toStringBinary' format, 'scan' supports custom formatting by column. A user can define a FORMATTER by adding it to the column name in the scan specification. The FORMATTER can be stipulated: 1. either as a org.apache.hadoop.hbase.util.Bytes method name (e.g, toInt, toString) 2. or as a custom class followed by method name: e.g. 'c(MyFormatterClass).format'. Example formatting cf:qualifier1 and cf:qualifier2 both as Integers: hbase> scan 't1', {COLUMNS => ['cf:qualifier1:toInt', 'cf:qualifier2:c(org.apache.hadoop.hbase.util.Bytes).toInt'] } Note that you can specify a FORMATTER by column only (cf:qualifer). You cannot specify a FORMATTER for all columns of a column family. Scan can also be used directly from a table, by first getting a reference to a table, like such: **hbase> t = get_table 't'** **hbase> t.scan** Note in the above situation, you can still provide all the filtering, columns, options, etc as described above.
truncate	Disables, drops and recreates the specified table. Examples: **hbase>truncate 't1'**

4) HBase surgery tools

assign	Assign a region. Use with caution. If region already assigned, this command will do a force reassign. For experts only. Examples: **hbase> assign 'REGION_NAME'**
balancer	Trigger the cluster balancer. Returns true if balancer ran and was able to tell the region servers to unassign all the regions to balance (the re-assignme async). Otherwise false (Will not run if regions in transition). Examples: **hbase> balancer**
balance_switch	Enable/Disable balancer. Returns previous balancer state. Examples: **hbase> balance_switch true** **hbase> balance_switch false**
close_region	Close a single region. Ask the master to close a region out on the cluster or if 'SERVER_NAME' is supplied, ask the designated hosting regionserver to close the region directly. Closing a region, the master expects 'REGIONNAM to be a fully qualified region name. When asking the hosting regionserver to directly close a region, you pass the regions' encoded name only. A region name looks like this:TestTable,0094429456,1289497600452.527db22f95c8a9e0116f0cc13c6 trailing period is part of the regionserver name. A region's encoded name is the hash at the end of a region name; e.g. 527db22f95c8a9e0116f0cc13c6 (without the period). A 'SERVER_NAME' is its host, port plus startcode. For example: host187.example.com,60020,1289493121758 (find servername in r or when you do detailed status in shell). This command will end up running close on the region hosting regionserver. The close is done without the master's involvement (It will not know of the close). Once closed, region will stay closed. Use assign to reopen/reassign. Use unassign or move to assign the region elsewhere on cluster. Use with caution. For experts only. Examples:hbase> close_region 'REGIONNAME' **hbase> close_region 'REGIONNAME', 'SERVER_NAME'**

compact	Compact all regions in passed table or pass a region row to compact an individual region. You can also compact a single column family within a region. Examples: Compact all regions in a table: **hbase> compact 't1'** Compact an entire region: **hbase> compact 'r1'** Compact only a column family within a region: **hbase> compact 'r1', 'c1'** Compact a column family within a table: **hbase> compact 't1', 'c1'**
flush	Flush all regions in passed table or pass a region row to flush an individual region. For example:hbase> flush 'TABLENAME' **hbase> flush 'REGIONNAME'**
major_compact	Run major compaction on passed table or pass a region row to major compact an individual region. To compact a single column family within a region specify the region name followed by the column family name. Examples: Compact all regions in a table: **hbase> major_compact 't1'** Compact an entire region: **hbase> major_compact 'r1'** Compact a single column family within a region: **hbase> major_compact 'r1', 'c1'** Compact a single column family within a table: **hbase> major_compact 't1', 'c1'**
move	Move a region. Optionally specify target regionserver else we choose one at random. NOTE: You pass the encoded region name, not the region name s this command is a little different to the others. The encoded region name is the hash suffix on region names: e.g. if the region name were TestTable,0094429456,1289497600452.527db22f95c8a9e0116f0cc13c6803 the encoded region name portion is 527db22f95c8a9e0116f0cc13c680396 A server name is its host, port plus startcode. For example: host187.example.com,60020,1289493121758 Examples:hbase> move 'ENCODED_REGIONNAME' **hbase> move 'ENCODED_REGIONNAME', 'SERVER_NAME'**
split	Split entire table or pass a region to split individual region. With the second parameter, you can specify an explicit split key for the region. Examples: **split 'tableName'** **split 'regionName' # format: 'tableName,startKey,id'** **split 'tableName', 'splitKey'** **split 'regionName', 'splitKey'**
unassign	Unassign a region. Unassign will close region in current location and then reopen it again. Pass 'true' to force the unassignment ('force' will clear all in-memory state in master before the reassign. If results in double assignment use hbck -fix to resolve. To be used by experts). Use with caution. For expert use only. Examples:hbase> unassign 'REGIONN **hbase> unassign 'REGIONNAME', true**
hlog_roll	Roll the log writer. That is, start writing log messages to a new file. The name of the regionserver should be given as the parameter. A 'server_name' is the host, port plus startcode of a regionserver. For example: host187.example.com,60020,1289493121758 (find servername in master ui or when you do detailed status in shell) **hbase>hlog_roll**
zk_dump	Dump status of HBase cluster as seen by ZooKeeper. Example: **hbase>zk_dump**

5) Cluster replication tools

add_peer	Add a peer cluster to replicate to, the id must be a short and the cluster key is composed like this: hbase.zookeeper.quorum:hbase.zookeeper.property.clientPort:zookeeper.zn This gives a full path for HBase to connect to another cluster. Examples:hbase> add_peer '1', "server1.cie.com:2181:/hbase" **hbase> add_peer '2', "zk1,zk2,zk3:2182:/hbase-prod"**
remove_peer	Stops the specified replication stream and deletes all the meta information kept about it. Examples: **hbase> remove_peer '1'**

list_peers	List all replication peer clusters. **hbase> list_peers**
enable_peer	Restarts the replication to the specified peer cluster, continuing from where it was disabled.Examples: **hbase> enable_peer '1'**
disable_peer	Stops the replication stream to the specified cluster, but still keeps track of new edits to replicate.Examples: **hbase> disable_peer '1'**
start_replication	Restarts all the replication features. The state in which each stream starts in is undetermined. WARNING: start/stop replication is only meant to be used in critical load situations. Examples: **hbase> start_replication**
stop_replication	Stops all the replication features. The state in which each stream stops in is undetermined. WARNING: start/stop replication is only meant to be used in critical load situations. Examples: **hbase> stop_replication**

6) Security tools

grant	Grant users specific rights. Syntax : grantpermissions is either zero or more letters from the set "RWXCA". READ('R'), WRITE('W'), EXEC('X'), CREATE('C'), ADMIN('A')For example:hbase> grant 'bobsmith', 'RWXCA' **hbase> grant 'bobsmith', 'RW', 't1', 'f1', 'col1'**
revoke	Revoke a user's access rights. Syntax : revoke For example: **hbase> revoke 'bobsmith', 't1', 'f1', 'col1'**
user_permission	Show all permissions for the particular user. Syntax : user_permission For example:hbase> user_permission **hbase> user_permission 'table1'**

← **Previous post**

39 comments

Vimal said:
August 16, 2013
9:06 am

Excellent content .. Thanks

Reply ↓

Sarav said:
December 1, 2013
7:12 pm

excellent COmpilation

Reply ↓

Pingback: [转]HBase Shell commands(HBase Shell 命令） -超级用户

faraz said:
January 13, 2014

6:21 am

```
USER="root"
PASSWORD="abc1234"
db=faraz
table=tree
mysqldump -u$USER -p$PASSWORD $db $table > /tmp/tablename.sql
mysql -u$USER -p$PASSWORD $db -e "truncate table $table"
```

this script is truncating the single table of a database named as faraz, but i wana truncate multiple tables, what syntax i need to use ?????

Reply ↓

Dinesh Sakote said:
June 9, 2016
9:14 pm

in the line
hbase> get 't1', 'r1' {COLUMN => ['cf:qualifier1:toInt', 'cf:qualifier2:c(org.apache.hadoop.hbase.util.Bytes).toInt'] }

A ',' is missing after 'r1'

Reply ↓

Pritesh said:
March 30, 2014
11:33 am

As we had configured HBase 0.94.1 pseudo Distributed mode Hadoop 1.0.3 & It's working fine but when tried put operation once data is stored into one row with column family.then for next row when we tried to store for next Data it overwrites the previous data. & we have to create new object of put every time for each file which does't seem Feasible. so we would like to know how to insert record automatically in next row in hbase?

Reply ↓

Abhishek said:
February 9, 2015
5:02 pm

you might not be using unique rowkey. It the rowkey is same the latest data will overwrite.

Reply ↓

rajeshhcu32 said:
March 30, 2014
3:23 pm

Hi Pritesh, If you want to do puts in bulk, you can prepare list of puts and do put all together. If you feel its difficult just ask the same question at HBase user mailing list(user@hbase.apache.org).
Thanks.

Reply ↓

OpenKB said:
May 20, 2014
6:06 pm

Nice reference. I will try to build my tests on all hbase shell commands also.

Reply ↓

Bin said:
July 16, 2014
7:40 pm

Nice post.
I'm wondering why you specify both the timestamp and versions during get
t.get 'r1′, {COLUMN => 'c1′, TIMESTAMP => ts1, VERSIONS => 4}
Since TIMESTAMP has been set to ts1, it is obvious that only (up to) one version can match and so, VERSIONS => 4 will be useless.
Instead, use following to get multiple versions of cells back
t.get 'r1′, {COLUMN => 'c1′, VERSIONS => 4}

Reply ↓

Jahid said:
July 22, 2014
7:11 pm

Nice post.Great

Reply ↓

Pingback: HBase tables from Hive | Kevin's Blog

Johnk120 said:

September 3, 2014
9:00 pm

Hey very cool blog!! Guy.. Beautiful.. Wonderful.. I will bookmark your website and take the feeds additionallyKI am satisfied to search out numerous useful information right here in the publish, we want develop more techniques on this regard, thank you for sharing dcdkdcekgfda

Reply ↓

Naveen said:
September 30, 2014
11:47 am

any command for renaming a table

Reply ↓

Pingback: A binary rowkey in HBase shell | I love green and blue

Sandip Adkar said:
January 16, 2015
11:42 am

Nice one!! Helped lot

Reply ↓

Ramkumar said:
May 22, 2015
5:09 pm

great job. thanks. do we have similar compilation for other hadoop projects or other nosql databases?
thanks

Reply ↓

prabu said:
July 26, 2015
11:36 am

How to rename the column ?

Reply ↓

AshwinGupta said:
September 11, 2015
6:16 am

Very good understanding of hbase. I believe, i am having now after reading your blog.

Thanks for supporting open source tools.

Reply ↓

Pingback: Emerging technologies relating to big data | IT Technologies

Navdeep Singh said:
January 20, 2016
5:41 am

what is habse shell command to add column to existing hbase table safely.

Reply ↓

SRIHARI said:
January 28, 2016
6:25 pm

THANK U SIR….REALLY USEFUL SIR

Reply ↓

rajeev sinha said:
February 2, 2016
1:52 am

I am very new findthe topicinteresting and useful

Reply ↓

kiran said:
April 7, 2016
12:49 am

Really nice

Reply ↓

srikanth kulkarni said:
April 11, 2016
5:08 pm

Thank you!

Reply ↓

KV said:
July 5, 2016
6:17 pm

Great article.

Reply ↓

paresh said:
August 23, 2016
12:49 pm

awesome material for learning hbase

Reply ↓

ankitbaldua said:
October 12, 2016
10:36 am

Great Compilation

Reply ↓

ankitbaldua said:
October 12, 2016
10:44 am

Reblogged this on Big Data – Baldua and commented:
Great Compilation of HBASE Shell Commands

Reply ↓

Umesh patil said:
October 19, 2016
10:32 am

I m try to insert a large data in hbase1.2.2 in a single node cluster but it get stuck takes to time after that it shows an exception that could not found location which location it does mean. I checked hbase gui the write request stop at 294 but it does not write the whole data.. Plz help.. why its happening

Reply ↓

Piyush said:
November 11, 2016
1:21 pm

awesome contents together at on place.. perfectly done !!

Reply ↓

Pingback: HBase shell commands – Bhavesh Gadoya

Learn hadoop big data training courses said:
April 20, 2017
5:23 am

Hi,
I must appreciate you for providing such a valuable content for us. This is one amazing piece of article. Helped a lot in increasing my knowledge on Hadoop.
Thanks,
Jeswika,
Learn hadoop big data training courses

Reply ↓

sarika said:
April 24, 2017
5:30 am

Thanks so much for writing this article. This is probably the best one by far. Easy to understand and educate myself on blog commenting and how the best way to go about it. Thanks a lot really appreciate you sharing this with us.

Reply ↓

DineshKumar said:
April 28, 2017
5:03 am

In Hadoop we have option to format name node. Similarly do we have option to format HBase?

Reply ↓

rajeshhcu32 said:
July 26, 2017
10:24 am

Yes. You can use $HBASE_HOME/bin/hbase clean –cleanAll
It will clean both hdfs and zookeeper data related to HBase.

Reply ↓

siva said:
May 7, 2017
2:27 am

really gr8 work bro thanks alot….
i tried many materials to understand small thing they give pages and pages of story.

Reply ↓

Nagesh Kumar said:
June 12, 2017
12:28 pm

Thank you for sharing the information here. Its much informative and really i got some valid information. You have posted the amazing article on Hbase

Reply ↓

Leviya said:
July 25, 2017
4:33 am

Thank you somuch for the information. The information you provided is very helpful for Hbase Learners.

Reply ↓

Leave a Reply

Enter your comment here...