24: Docker Tutorial: HBase (i.e. NoSQL DB) Shell on Cloudera quickstart

Posted on June 16, 2019

This extends Docker Tutorial: BigData on Cloudera quickstart via Docker.

Step 1: Run the container on a command line.

```
~/projects/docker-hadoop]$ docker run --hostname=qu
--privileged=true -t -i -v /Users/arulkumarankumar
--publish-all=true -p 8888:8888 -p 80:80 -p 7180:7
cloudera/quickstart /usr/bin/docker-quickstart
```

300+ Java Interview FAQs

300+ Java FAQs 🔥

16+ Java Key Areas Q&As

150+ Java Architect FAQs

80+ Java Code Quality Q&As

150+ Java Coding Q&As

300+ Big Data Interview FAQs

300+ Big Data FAQs 🔥

Tutorials - Big Data

TUT - ⏯ Starting Big Data

TUT - Starting Spark & Scala

hbase cli

We are going to focus on "**shell**" command to create a table, column-families, and columns, etc.

```
[root@quickstart /]# hbase
Usage: hbase [] <command></command> []
Options:
  --config DIR    Configuration direction to use.
  --hosts HOSTS   Override the list in 'regionserv
  --auth-as-server Authenticate to ZooKeeper using

Commands:
Some commands take arguments. Pass no args or -h f
  shell           Run the HBase shell
  hbck            Run the hbase 'fsck' tool
  snapshot        Create a new snapshot of a table
  snapshotinfo    Tool for dumping snapshot inform
  wal             Write-ahead-log analyzer
  hfile           Store file analyzer
  zkcli           Run the ZooKeeper shell
  upgrade         Upgrade hbase
  master          Run an HBase HMaster node
  regionserver    Run an HBase HRegionServer node
  zookeeper       Run a Zookeeper server
  rest            Run an HBase REST server
  thrift          Run the HBase Thrift server
  thrift2         Run the HBase Thrift2 server
  clean           Run the HBase clean up script
  classpath       Dump hbase CLASSPATH
  mapredcp        Dump CLASSPATH entries required
  pe              Run PerformanceEvaluation
  ltt             Run LoadTestTool
  version         Print the version
  CLASSNAME       Run the class named CLASSNAME
[root@quickstart /]#

```

Step 2: Enter the hbase shell.

```
[root@quickstart /]# hbase shell
hbase(main):001:0>

```

Create a table

TUT - Starting with Python
TUT - Kafka
TUT - Pig
TUT - Apache Storm
TUT - Spark Scala on Zeppelin
TUT - Cloudera
TUT - Cloudera on Docker
TUT - File Formats
TUT - Spark on Docker
TUT - Flume
TUT - Hadoop (HDFS)
TUT - HBase (NoSQL)
TUT - Hive (SQL)
TUT - Hadoop & Spark
TUT - MapReduce
TUT - Spark and Scala
TUT - Spark & Java
TUT - PySpark on Databricks
TUT - Zookeeper

Step 3: You create a table with the “create” command, which requires a “table” name and a “column family” name.

```
hbase(main):004:0> create 'employees', 'personal',
```

Table “employees” is created with 2 column families “personal” and “professional”.

list & describe tables

```
hbase(main):005:0> list
TABLE
employees
1 row(s) in 0.0090 seconds

=> ["employees"]
hbase(main):006:0>
```

```
hbase(main):014:0> describe 'employees'
Table employees is ENABLED
employees
COLUMN FAMILIES DESCRIPTION
{NAME => 'personal', DATA_BLOCK_ENCODING => 'NONE
=> 'NONE', MIN_VERSIONS => '0', TTL => 'FOREVER',
KCACHE => 'true'}
{NAME => 'professional', DATA_BLOCK_ENCODING => 'N
ION => 'NONE', MIN_VERSIONS => '0', TTL => 'FOREVE
BLOCKCACHE => 'true'}
2 row(s) in 0.1040 seconds

hbase(main):015:0>
```

put data into a table

Step 4: put data into a table. You must specify, table name, row key, and then column name prefixed by the **column family** name and a value.

```
hbase(main):015:0> put 'employees', '101', 'perso
0 row(s) in 0.0980 seconds

hbase(main):016:0> put 'employees', '101', 'perso
0 row(s) in 0.0150 seconds

hbase(main):017:0> put 'employees', '101', 'profe
0 row(s) in 0.0090 seconds

hbase(main):018:0>
```

scan data from a table

Step 4: scan for data by specifying the table name.

```
hbase(main):015:0> hbase(main):018:0> scan 'employ
ROW                                COLUMN+CELL
 101                               column=personal:
 101                               column=personal:
 101                               column=professio
1 row(s) in 0.0320 seconds

hbase(main):019:0>
```

put more data

```
hbase(main):023:0> put 'employees', '102', 'perso
0 row(s) in 0.0120 seconds

hbase(main):024:0> put 'employees', '102', 'perso
0 row(s) in 0.0120 seconds

hbase(main):025:0> put 'employees', '102', 'perso
0 row(s) in 0.0090 seconds

hbase(main):026:0> put 'employees', '102', 'profe
0 row(s) in 0.0100 seconds

hbase(main):027:0>
```

get data by row key

Step 5: Get a specific record by row key. get 'table_name','row_key'.

```
hbase(main):027:0> get 'employees', '102'
COLUMN                                CELL
 personal:firstname                   timestamp=15606
 personal:middlename                  timestamp=15606
 personal:surname                     timestamp=15606
 professional:department              timestamp=15606
4 row(s) in 0.0250 seconds

hbase(main):028:0>
```

Access via REST API

Let's exit out of hbase shell, and access the table we had created using "curl", which is a REST API client.

Table schema

```
[root@quickstart /]# curl -I -H "Accept: text/xml"
HTTP/1.1 200 OK
Content-Length: 0
Cache-Control: no-cache
Content-Type: text/xml
```

```
curl -vi -X GET \
         -H "Accept: text/xml" \
         "http://quickstart.cloudera:8070/employee
```

Outputs:

```
* About to connect() to quickstart.cloudera port
*   Trying 172.17.0.2... connected
* Connected to quickstart.cloudera (172.17.0.2) p
> GET /employees/schema HTTP/1.1
> User-Agent: curl/7.19.7 (x86_64-redhat-linux-gn
> Host: quickstart.cloudera:8070
> Accept: text/xml
>
< HTTP/1.1 200 OK
HTTP/1.1 200 OK
< Cache-Control: no-cache
```

```
Cache-Control: no-cache
< Content-Type: text/xml
Content-Type: text/xml
< Content-Length: 609
Content-Length: 609

<
* Connection #0 to host quickstart.cloudera left
* Closing connection #0
<!--?xml version="1.0" encoding="UTF-8" standalone
```

Table get

Get the value of a single row. Values are Base-64 encoded.

```
curl -vi -X GET \
         -H "Accept: text/xml" \
         "http://quickstart.cloudera:8070/employees
```

Output:

```
* About to connect() to quickstart.cloudera port 8
*   Trying 172.17.0.2... connected
* Connected to quickstart.cloudera (172.17.0.2) po
> GET /employees/101 HTTP/1.1
> User-Agent: curl/7.19.7 (x86_64-redhat-linux-gnu
> Host: quickstart.cloudera:8070
> Accept: text/xml
>
< HTTP/1.1 200 OK
HTTP/1.1 200 OK
< Content-Type: text/xml
Content-Type: text/xml
< Content-Length: 351
Content-Length: 351

<
* Connection #0 to host quickstart.cloudera left
* Closing connection #0
<!--?xml version="1.0" encoding="UTF-8" standalone
```

How do you know the REST port?

```
[root@quickstart /]# cat /etc/hbase/conf/hbase-site
```

```
...
<configuration>
  <property>
    <name>hbase.rest.port</name>
    <value>8070</value>
    <description>The port for the HBase REST serve
  </property>
...
</configuration>

```

‹ 23: Docker Tutorial: Apache Spark (spark-submit) in Python 3 with virtual env on Cloudera quickstart

25: Docker Tutorial: HBase (i.e. NoSQL) Java API on Cloudera quickstart ›

Disclaimer