

Java-Success.com

Prepare to fast-track, choose & go places with 800+ Java & Big Data Q&As with lots of code & diagrams.

[Home](#) [Why? ▾](#) [300+ Java FAQs ▾](#) [300+ Big Data FAQs ▾](#) [Courses ▾](#)[👤 Membership ▾](#) [Your Career ▾](#)[Home](#) > [bigdata-success.com](#) > [Tutorials - Big Data](#) > [TUT - Cloudera on Docker](#) > 24:

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

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.

```
1 ~/projects/docker-hadoop]$ docker run --hostname=q
2 --privileged=true -t -i -v /Users/arulkumarankumar
3 --publish-all=true -p 8888:8888 -p 80:80 -p 7180:7
4 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.

```

1 [root@quickstart /]# hbase
2 Usage: hbase <command> <command>
3 Options:
4   --config DIR      Configuration direction to use.
5   --hosts HOSTS     Override the list in 'regionserv
6   --auth-as-server  Authenticate to ZooKeeper using
7
8 Commands:
9 Some commands take arguments. Pass no args or -h
10  shell              Run the HBase shell
11  hbck               Run the hbase 'fsck' tool
12  snapshot           Create a new snapshot of a table
13  snapshotinfo       Tool for dumping snapshot inform
14  wal                Write-ahead-log analyzer
15  hfile              Store file analyzer
16  zkcli              Run the ZooKeeper shell
17  upgrade            Upgrade hbase
18  master             Run an HBase HMaster node
19  regionserver       Run an HBase HRegionServer node
20  zookeeper          Run a Zookeeper server
21  rest               Run an HBase REST server
22  thrift             Run the HBase Thrift server
23  thrift2            Run the HBase Thrift2 server
24  clean              Run the HBase clean up script
25  classpath          Dump hbase CLASSPATH
26  mapredcp           Dump CLASSPATH entries required
27  pe                 Run PerformanceEvaluation
28  ltt                Run LoadTestTool
29  version            Print the version
30  CLASSNAME          Run the class named CLASSNAME
31 [root@quickstart /]#
32

```

Step 2: Enter the hbase shell.

```

1 [root@quickstart /]# hbase shell
2 hbase(main):001:0>
3

```

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

800+ Java Interview Q&As

300+ Core Java Q&As



300+ Enterprise Java Q&As



150+ Java Frameworks Q&As



120+ Companion Tech Q&As



Tutorials - Enterprise Java



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

```
1 hbase(main):004:0> create 'employees', 'personal',
2
```

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

list & describe tables

```
1 hbase(main):005:0> list
2 TABLE
3 employees
4 1 row(s) in 0.0090 seconds
5
6 => ["employees"]
7 hbase(main):006:0>
8
```

```
1 hbase(main):014:0> describe 'employees'
2 Table employees is ENABLED
3 employees
4 COLUMN FAMILIES DESCRIPTION
5 {NAME => 'personal', DATA_BLOCK_ENCODING => 'NONE
6 => 'NONE', MIN_VERSIONS => '0', TTL => 'FOREVER',
7 KCACHE => 'true'}
8 {NAME => 'professional', DATA_BLOCK_ENCODING => 'I
9 ION => 'NONE', MIN_VERSIONS => '0', TTL => 'FOREVI
10 BLOCKCACHE => 'true'}
11 2 row(s) in 0.1040 seconds
12
13 hbase(main):015:0>
14
```

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.

```
1 hbase(main):015:0> put 'employees', '101', 'person
2 0 row(s) in 0.0980 seconds
3
4 hbase(main):016:0> put 'employees', '101', 'person
5 0 row(s) in 0.0150 seconds
6
7 hbase(main):017:0> put 'employees', '101', 'profes
8 0 row(s) in 0.0090 seconds
9
10 hbase(main):018:0>
11
```

scan data from a table

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

```
1 hbase(main):015:0> hbase(main):018:0> scan 'employe
2 ROW COLUMN+CELL
3 101 column=personal:
4 101 column=personal:
5 101 column=profession
6 1 row(s) in 0.0320 seconds
7
8 hbase(main):019:0>
9
```

put more data

```
1 hbase(main):023:0> put 'employees', '102', 'person
2 0 row(s) in 0.0120 seconds
3
4 hbase(main):024:0> put 'employees', '102', 'person
5 0 row(s) in 0.0120 seconds
6
7 hbase(main):025:0> put 'employees', '102', 'person
8 0 row(s) in 0.0090 seconds
9
10 hbase(main):026:0> put 'employees', '102', 'profes
11 0 row(s) in 0.0100 seconds
12
13 hbase(main):027:0>
14
```

get data by row key

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

```

1 hbase(main):027:0> get 'employees', '102'
2 COLUMN                                CELL
3 personal:firstname                    timestamp=1560600000
4 personal:middlename                  timestamp=1560600000
5 personal:surname                     timestamp=1560600000
6 professional:department              timestamp=1560600000
7 4 row(s) in 0.0250 seconds
8
9 hbase(main):028:0>
10

```

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

```

1 [root@quickstart /]# curl -I -H "Accept: text/xml"
2 HTTP/1.1 200 OK
3 Content-Length: 0
4 Cache-Control: no-cache
5 Content-Type: text/xml
6

```

```

1 curl -vi -X GET \
2     -H "Accept: text/xml" \
3     "http://quickstart.cloudera:8070/employees/
4

```

Outputs:

```

1 * About to connect() to quickstart.cloudera port 8070
2 * Trying 172.17.0.2... connected
3 * Connected to quickstart.cloudera (172.17.0.2) port 8070
4 > GET /employees/schema HTTP/1.1
5 > User-Agent: curl/7.19.7 (x86_64-redhat-linux-gnu)
6 > Host: quickstart.cloudera:8070
7 > Accept: text/xml
8 >
9 < HTTP/1.1 200 OK
10 HTTP/1.1 200 OK
11 < Cache-Control: no-cache

```

```

12 Cache-Control: no-cache
13 < Content-Type: text/xml
14 Content-Type: text/xml
15 < Content-Length: 609
16 Content-Length: 609
17
18 <
19 * Connection #0 to host quickstart.cloudera left
20 * Closing connection #0
21 <!--?xml version="1.0" encoding="UTF-8" standalone=
22

```

Table get

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

```

1 curl -vi -X GET \
2     -H "Accept: text/xml" \
3     "http://quickstart.cloudera:8070/employees/101"
4

```

Output:

```

1 * About to connect() to quickstart.cloudera port 8070
2 * Trying 172.17.0.2... connected
3 * Connected to quickstart.cloudera (172.17.0.2) port 8070
4 > GET /employees/101 HTTP/1.1
5 > User-Agent: curl/7.19.7 (x86_64-redhat-linux-gnu)
6 > Host: quickstart.cloudera:8070
7 > Accept: text/xml
8 >
9 < HTTP/1.1 200 OK
10 HTTP/1.1 200 OK
11 < Content-Type: text/xml
12 Content-Type: text/xml
13 < Content-Length: 351
14 Content-Length: 351
15
16 <
17 * Connection #0 to host quickstart.cloudera left
18 * Closing connection #0
19 <!--?xml version="1.0" encoding="UTF-8" standalone=
20

```

How do you know the REST port?

```

1 [root@quickstart /]# cat /etc/hbase/conf/hbase-site.xml

```

```
1  ...
2  <configuration>
3    <property>
4      <name>hbase.rest.port</name>
5      <value>8070</value>
6      <description>The port for the HBase REST server</description>
7    </property>
8  ...
9  </configuration>
10
```

◀ 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

The contents in this Java-Success are copyrighted and from EmpoweringTech pty ltd. The EmpoweringTech pty ltd has the right to correct or enhance the current content without any prior notice. These are general advice only, and one needs to take his/her own circumstances into consideration. The EmpoweringTech pty ltd will not be held liable for any damages caused or alleged to be caused either directly or indirectly by these materials and resources. Any trademarked names or labels used in this blog remain the property of their respective trademark owners. Links to external sites do not imply endorsement of the linked-to sites. [Privacy Policy](#).