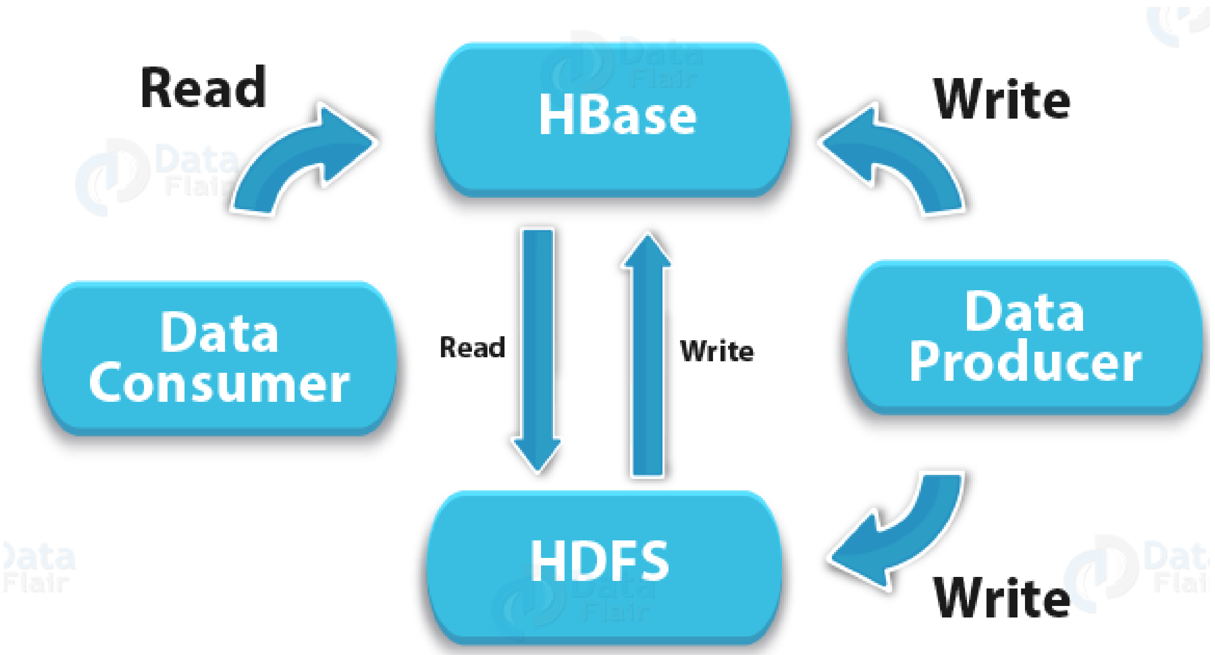
HBase

HBase is an open source, multidimensional, distributed, scalable and a ***NoSQL database*** written in Java. HBase runs on top of ***HDFS*** (Hadoop Distributed File System) and provides BigTable like capabilities to Hadoop. It is designed to provide a fault tolerant way of storing large collection of sparse data sets.

Since, HBase achieves high throughput and low latency by providing faster Read/Write Access on huge data sets. Therefore, HBase is the choice for the applications which require fast & random access to large amount of data.

It provides compression, in-memory operations and Bloom filters (data structure which tells whether a value is present in a set or not) to fulfil the requirement of fast and random read-writes.



History

- Initially, in Nov 2006, Google released the paper on BigTable.
- The first HBase prototype was created as a Hadoop contribution in the year Feb 2007.

- The first usable HBase was released in the same year Oct 2007 along with Hadoop 0.15.0.
- HBase became the subproject of Hadoop, in Jan 2008.
- In the year 2010, May HBase became Apache top-level project.

HBase Installation

1. **Standalone mode installation** (No dependency on Hadoop system)

    - This is default mode of HBase
    - It runs against local file system
    - It doesn't use Hadoop HDFS
    - Only HMaster daemon can run
    - Not recommended for production environment
    - Runs in single JVM

2. **Pseudo-Distributed mode installation** (Single node Hadoop system + HBase installation)

    - It runs on Hadoop HDFS
    - All Daemons run in single node
    - Recommend for production environment

3. **Fully Distributed mode installation** (Multi node Hadoop environment + HBase installation)

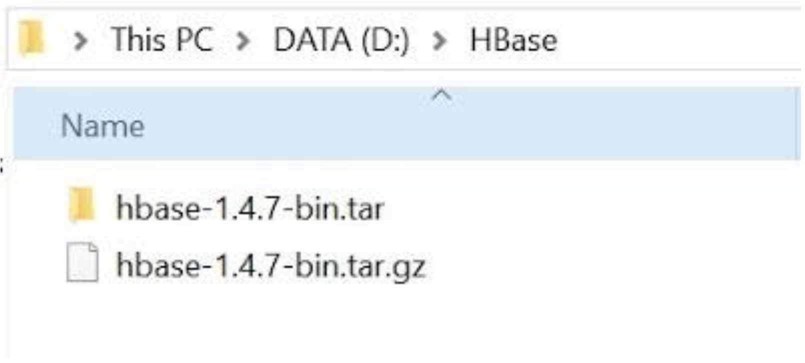
    - It runs on Hadoop HDFS
    - All daemons going to run across all nodes present in the cluster
    - Highly recommended for production environment

HBase - Standalone mode installation

STEP - 1: Extract the HBase file

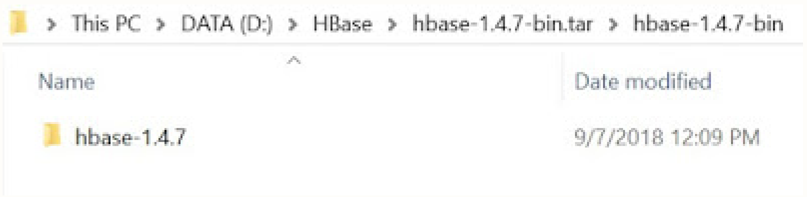
Extract file hbase-1.4.7-bin.tar.gz and place under "D:\HBase", you can use any preferred location –

[1] You will get again a tar file post extraction –



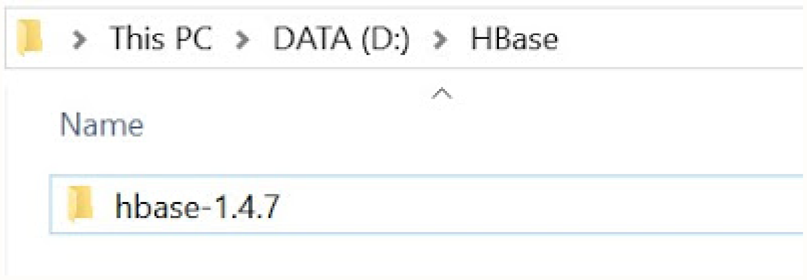
hbase-1.4.7-bin.tar
hbase-1.4.7-bin.tar.gz

[2] Go inside of hbase-1.4.7-bin.tar folder and extract again –

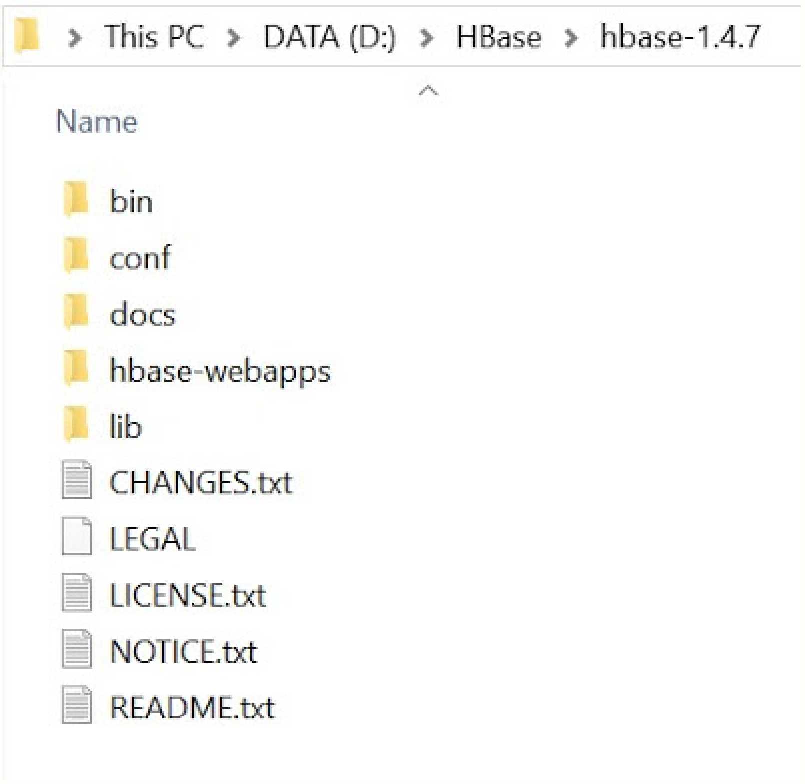

This PC > DATA (D:) > HBase > hbase-1.4.7-bin.tar > hbase-1.4.7-bin

Name	Date modified
hbase-1.4.7	9/7/2018 12:09 PM

[3] Copy the leaf folder "hbase-1.4.7" and move to the root folder "D:\HBase" and removed all other files and folders –



hbase-1.4.7

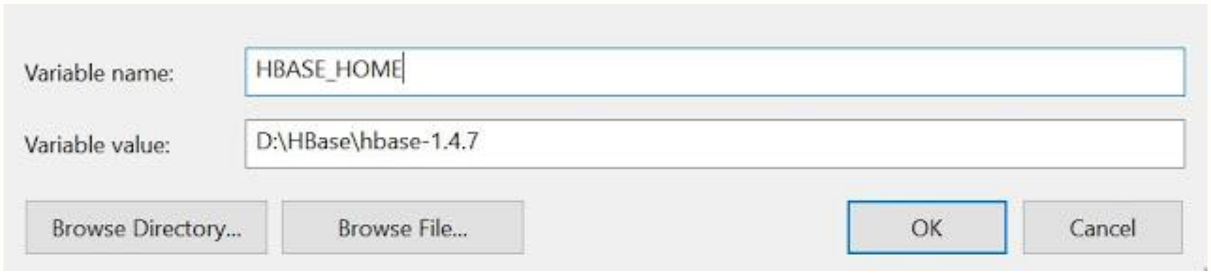


```
bin
conf
docs
hbase-webapps
lib
CHANGES.txt
LEGAL
LICENSE.txt
NOTICE.txt
README.txt
```

STEP - 2: Configure Environment variable

Set the path for the following Environment variable (User Variables

HBASE_HOME - D:\HBase\hbase-1.4.7



Variable name:	HBASE_HOME
Variable value:	D:\HBase\hbase-1.4.7

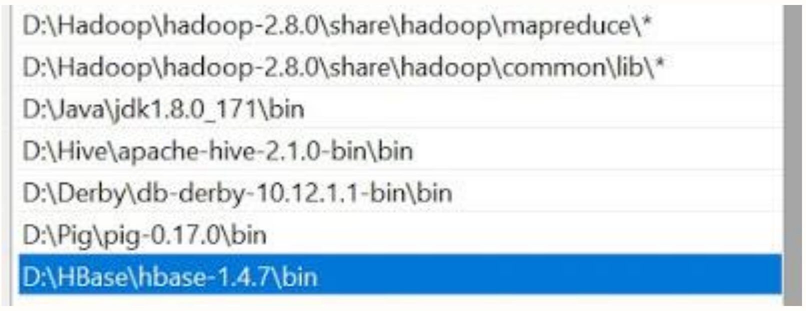
STEP - 3: Configure System variable

Next onward need to set System variable, including Hive bin directory path –

Variable: Path

Value:

- D:\HBase\hbase-1.4.7\bin



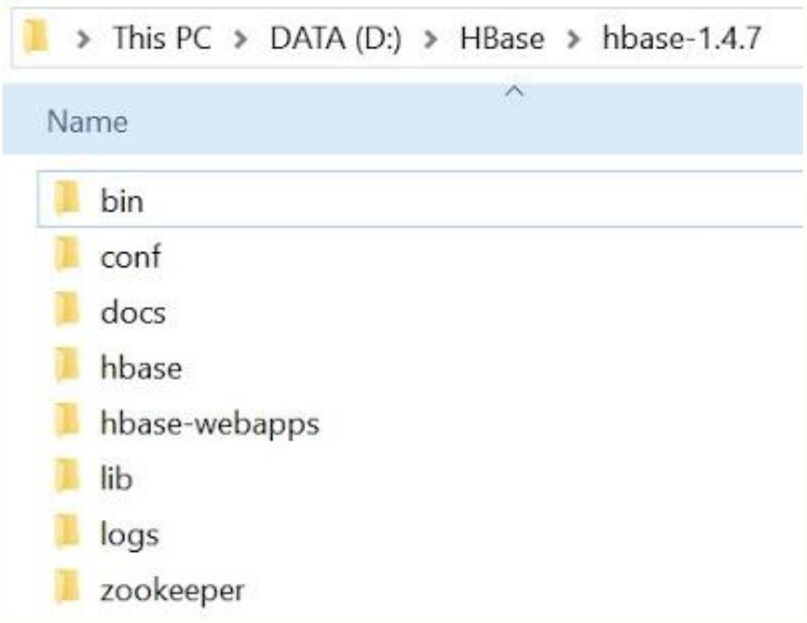
```
D:\Hadoop\hadoop-2.8.0\share\hadoop\mapreduce\*
D:\Hadoop\hadoop-2.8.0\share\hadoop\common\lib\*
D:\Java\jdk1.8.0_171\bin
D:\Hive\apache-hive-2.1.0-bin\bin
D:\Derby\db-derby-10.12.1.1-bin\bin
D:\Pig\pig-0.17.0\bin
D:\HBase\hbase-1.4.7\bin
```

STEP - 4: Create required folders

Create some dedicated folders -

1. Create folder "hbase" under “D:\HBase\hbase-1.4.7”.
2. Create folder "zookeeper" under “D:\HBase\hbase-1.4.7”.

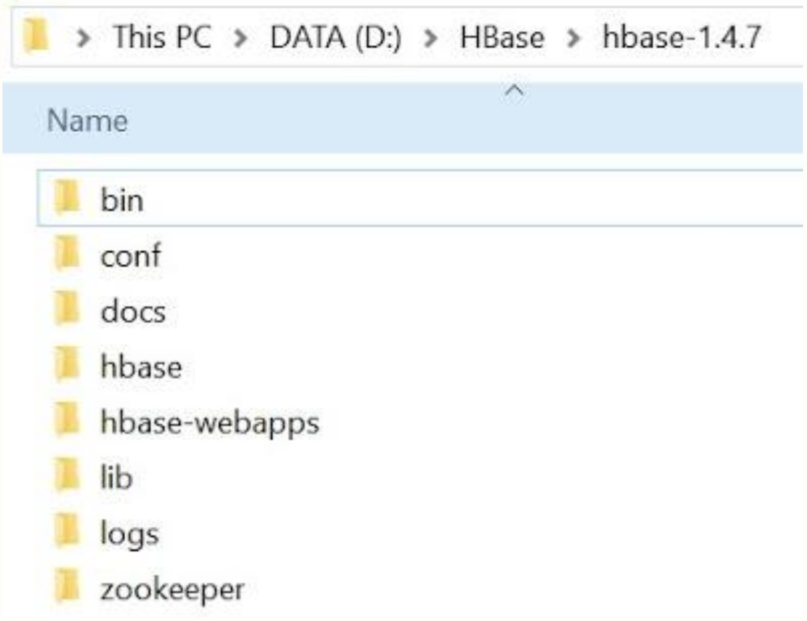
For example -



This PC > DATA (D:) > HBase > hbase-1.4.7

Name

bin
conf
docs
hbase
hbase-webapps
lib
logs
zookeeper



```
bin
conf
docs
hbase
hbase-webapps
lib
logs
zookeeper
```

STEP - 5: Configured required files

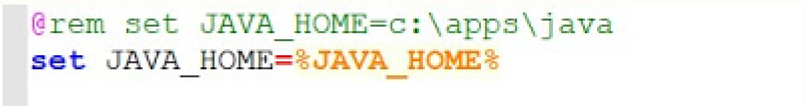
Next, essential to configure two key files with minimal required details –

- hbase-env.cmd
- hbase-site.xml

[1] Edit file D:/HBase/hbase-1.4.7/conf/hbase-env.cmd, mention JAVA_HOME path in the location and save this file.

@rem set JAVA_HOME=c:\apps\java

set JAVA_HOME=%JAVA_HOME



```
@rem set JAVA_HOME=c:\apps\java
set JAVA_HOME=%JAVA_HOME%
```

[2] Edit file D:/HBase/hbase-1.4.7/conf/hbase-site.xml, paste below xml paragraph and save this file.

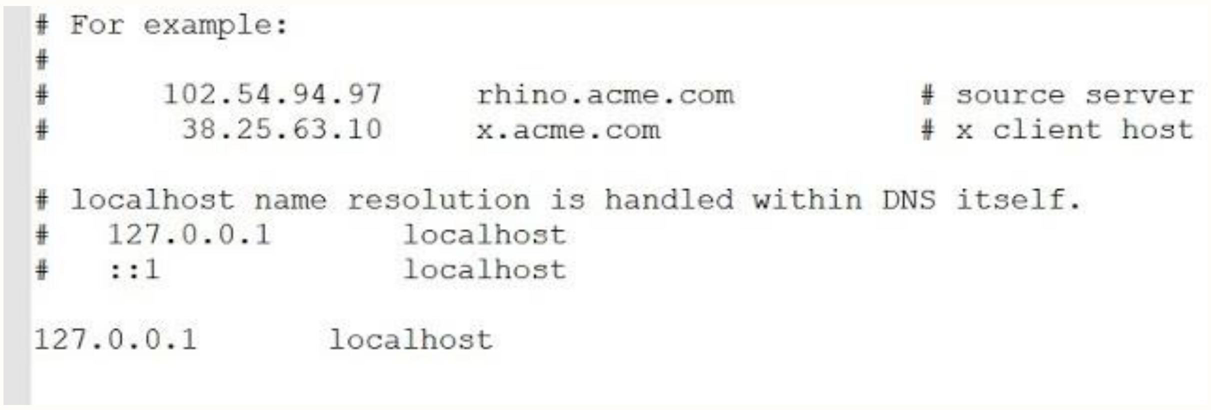
```
<configuration>

 <property>
  <name>hbase.rootdir</name>
  <value>file:///D:/HBase/hbase-1.4.7/hbase</value>
```

```
</property>

<property>

 <name>hbase.zookeeper.property.dataDir</name>
 <value>/D:/HBase/hbase-1.4.7/zookeeper</value>
</property>

<property>
 <name> hbase.zookeeper.quorum</name>
 <value>127.0.0.1</value>
</property>
</configuration>
```

All HMaster and ZooKeeper activities point out to this hbase-site.xml.

[3] Edit file hosts (C: /Windows/System32/drivers/etc/hosts), mention localhost IP and save this file.

```
127.0.0.1      localhost
```

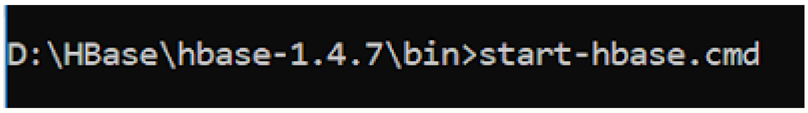


```
# For example:
#
#      102.54.94.97     rhino.acme.com          # source server
#       38.25.63.10     x.acme.com              # x client host

# localhost name resolution is handled within DNS itself.
#   127.0.0.1       localhost
#   ::1             localhost

127.0.0.1       localhost
```

STEP - 6: Start HBase

Here need to start HBase first -

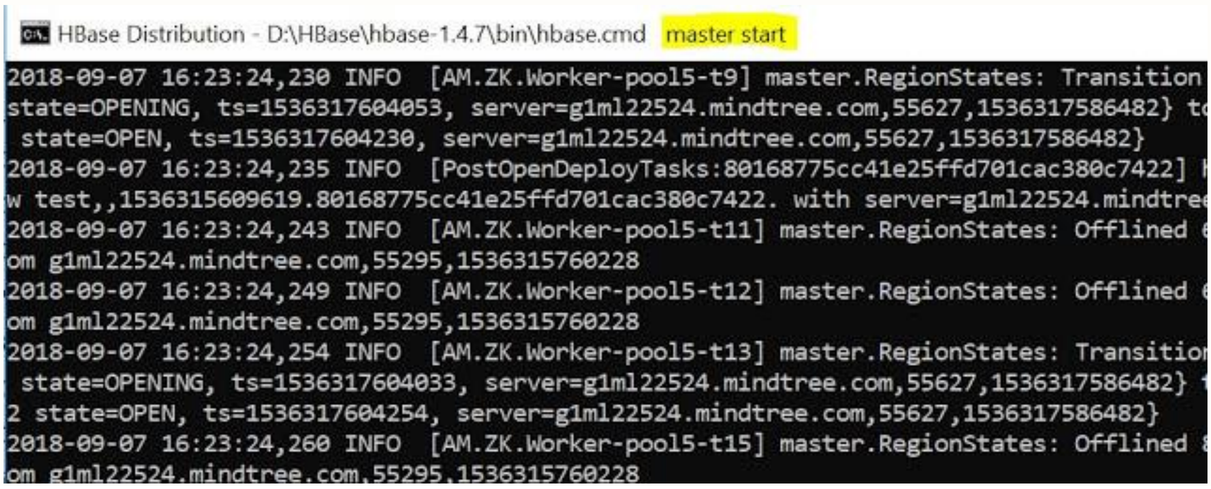
Open command prompt and change directory to “D:\HBase\hbase-1.4.7\bin" and type "**start-hbase.cmd**" to start HBase.



```
D:\HBase\hbase-1.4.7\bin>start-hbase.cmd
```

It will open a separate instances of cmd for following tasks –

- HBase Master

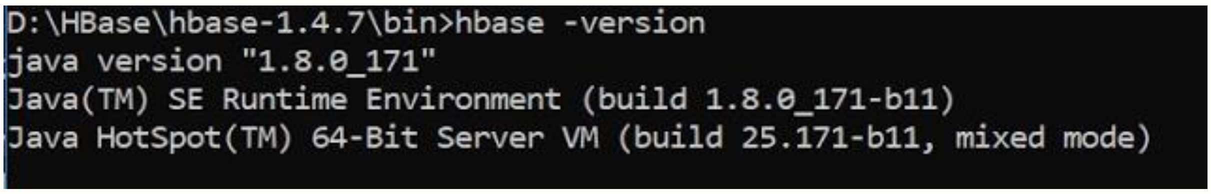


```
HBase Distribution - D:\HBase\hbase-1.4.7\bin\hbase.cmd  master start
2018-09-07 16:23:24,230 INFO  [AM.ZK.Worker-pool5-t9] master.RegionStates: Transition
state=OPENING, ts=1536317604053, server=g1ml22524.mindtree.com,55627,1536317586482} to
 state=OPEN, ts=1536317604230, server=g1ml22524.mindtree.com,55627,1536317586482}
2018-09-07 16:23:24,235 INFO  [PostOpenDeployTasks:80168775cc41e25ffd701cac380c7422] 
w test,,1536315609619.80168775cc41e25ffd701cac380c7422. with server=g1ml22524.mindtre
2018-09-07 16:23:24,243 INFO  [AM.ZK.Worker-pool5-t11] master.RegionStates: Offlined 
om g1ml22524.mindtree.com,55295,1536315760228
2018-09-07 16:23:24,249 INFO  [AM.ZK.Worker-pool5-t12] master.RegionStates: Offlined 
om g1ml22524.mindtree.com,55295,1536315760228
2018-09-07 16:23:24,254 INFO  [AM.ZK.Worker-pool5-t13] master.RegionStates: Transitio
 state=OPENING, ts=1536317604033, server=g1ml22524.mindtree.com,55627,1536317586482} 
2 state=OPEN, ts=1536317604254, server=g1ml22524.mindtree.com,55627,1536317586482}
2018-09-07 16:23:24,260 INFO  [AM.ZK.Worker-pool5-t15] master.RegionStates: Offlined 
om g1ml22524.mindtree.com,55295,1536315760228
```

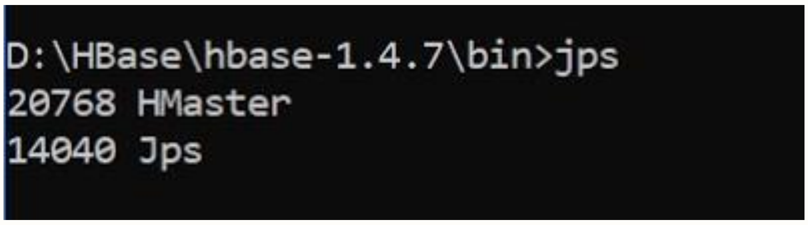
STEP - 7: Validate HBase

Post successful execution of HBase, verify the installation using following commands –

- hbase –version
- jps



```
D:\HBase\hbase-1.4.7\bin>hbase -version
java version "1.8.0_171"
Java(TM) SE Runtime Environment (build 1.8.0_171-b11)
Java HotSpot(TM) 64-Bit Server VM (build 25.171-b11, mixed mode)
```

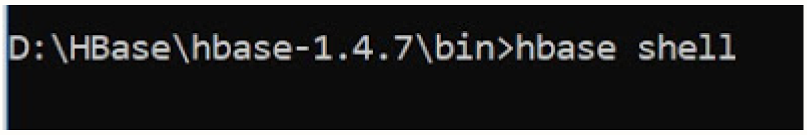


```
D:\HBase\hbase-1.4.7\bin>jps
20768 HMaster
14040 Jps
```

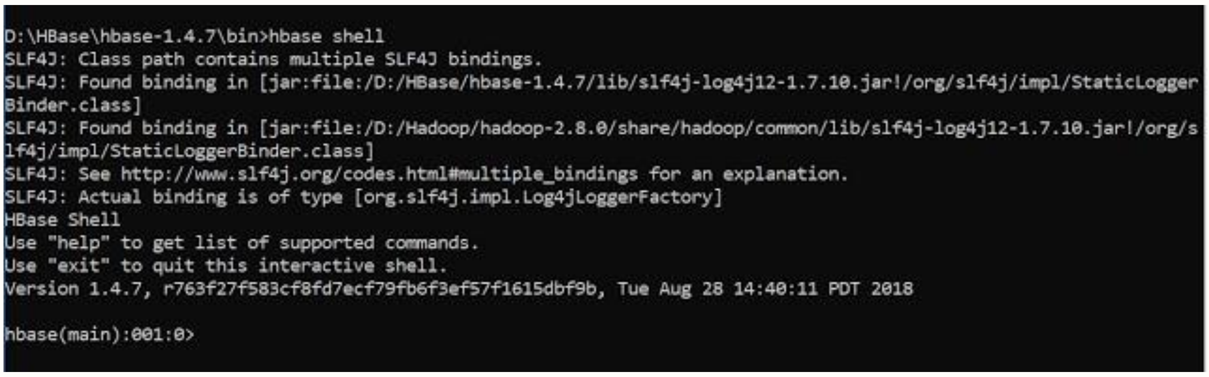
If we can see HMaster is in running mode, then our installation is okay.

STEP - 8: Execute HBase Shell

The standalone mode does not require Hadoop daemons to start. HBase can run independently. HBase shell can start by using "hbase shell" and it will enter into interactive shell mode –



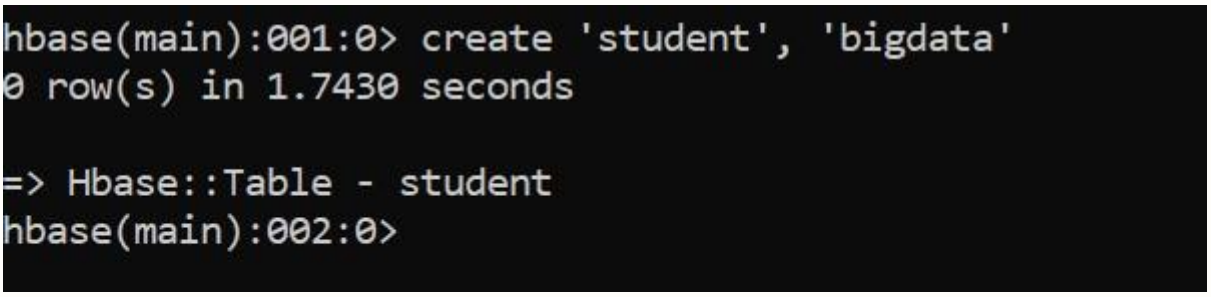
```
D:\HBase\hbase-1.4.7\bin>hbase shell
```



```
D:\HBase\hbase-1.4.7\bin>hbase shell
SLF4J: Class path contains multiple SLF4J bindings.
SLF4J: Found binding in [jar:file:/D:/HBase/hbase-1.4.7/lib/slf4j-log4j12-1.7.10.jar!/org/slf4j/impl/StaticLogger
Binder.class]
SLF4J: Found binding in [jar:file:/D:/Hadoop/hadoop-2.8.0/share/hadoop/common/lib/slf4j-log4j12-1.7.10.jar!/org/s
lf4j/impl/StaticLoggerBinder.class]
SLF4J: See http://www.slf4j.org/codes.html#multiple_bindings for an explanation.
SLF4J: Actual binding is of type [org.slf4j.impl.Log4jLoggerFactory]
HBase Shell
Use "help" to get list of supported commands.
Use "exit" to quit this interactive shell.
Version 1.4.7, r763f27f583cf8fd7ecf79fb6f3ef57f1615dbf9b, Tue Aug 28 14:40:11 PDT 2018

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

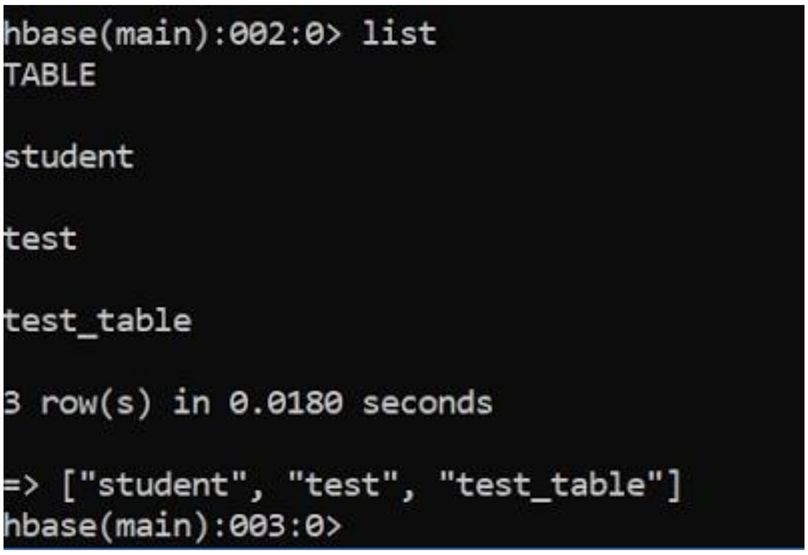
STEP-9: Some hands on activities

[1] Create a simple table
create 'student', 'bigdata'



```
hbase(main):001:0> create 'student', 'bigdata'
0 row(s) in 1.7430 seconds

=> Hbase::Table - student
hbase(main):002:0>
```

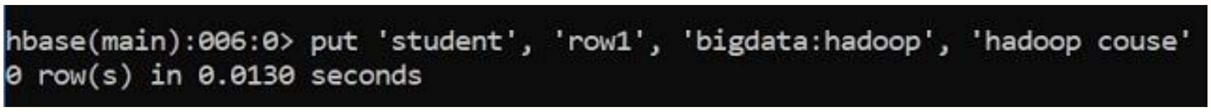
[2] List the table has been created
list



```
hbase(main):002:0> list
TABLE

student

test

test_table

3 row(s) in 0.0180 seconds

=> ["student", "test", "test_table"]
hbase(main):003:0>
```

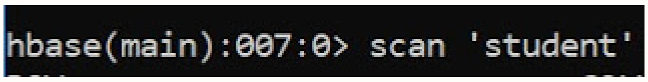
[3] Insert some data to above created table

put ‘tablename’, ‘rowname’, ‘columnvalue’, ‘value’
put 'student', 'row1', 'bigdata:hadoop', 'hadoop couse'

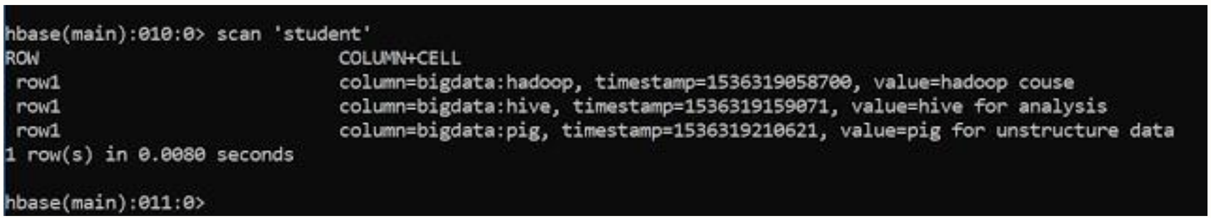


```
hbase(main):006:0> put 'student', 'row1', 'bigdata:hadoop', 'hadoop couse'
0 row(s) in 0.0130 seconds
```

[4] List all rows in the table
scan 'student'



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



```
hbase(main):010:0> scan 'student'
ROW                          COLUMN+CELL
 row1                        column=bigdata:hadoop, timestamp=1536319058700, value=hadoop couse
 row1                        column=bigdata:hive, timestamp=1536319159071, value=hive for analysis
 row1                        column=bigdata:pig, timestamp=1536319210621, value=pig for unstructure data
1 row(s) in 0.0080 seconds

hbase(main):011:0>
```

HBase Shell

HBase contains a shell using which you can communicate with HBase. HBase uses the Hadoop File System to store its data. It will have a master server and region servers. The data storage will be in the form of regions (tables). These regions will be split up and stored in region servers.

The master server manages these region servers and all these tasks take place on HDFS. Given below are some of the commands supported by HBase Shell.

General Commands

- **status** - Provides the status of HBase, for example, the number of servers.
- **version** - Provides the version of HBase being used.
- **table_help** - Provides help for table-reference commands.
- **whoami** - Provides information about the user.

Data Definition Language

These are the commands that operate on the tables in HBase.

- **create** - Creates a table.
- **list** - Lists all the tables in HBase.
- **disable** - Disables a table.
- **is_disabled** - Verifies whether a table is disabled.
- **enable** - Enables a table.
- **is_enabled** - Verifies whether a table is enabled.
- **describe** - Provides the description of a table.
- **alter** - Alters a table.
- **exists** - Verifies whether a table exists.
- **drop** - Drops a table from HBase.
- **drop_all** - Drops the tables matching the 'regex' given in the command.
- **Java Admin API** - Prior to all the above commands, Java provides an Admin API to achieve DDL functionalities through programming. Under **org.apache.hadoop.hbase.client** package, HBaseAdmin and HTableDescriptor are the two important classes in this package that provide DDL functionalities.

Data Manipulation Language

- **put** - Puts a cell value at a specified column in a specified row in a particular table.
- **get** - Fetches the contents of row or a cell.
- **delete** - Deletes a cell value in a table.
- **deleteall** - Deletes all the cells in a given row.
- **scan** - Scans and returns the table data.
- **count** - Counts and returns the number of rows in a table.
- **truncate** - Disables, drops, and recreates a specified table.
- **Java client API** - Prior to all the above commands, Java provides a client API to achieve DML functionalities, **CRUD** (Create Retrieve Update Delete) operations and

more through programming, under org.apache.hadoop.hbase.client package. **HTable Put** and **Get** are the important classes in this package.