Quick HBase Standalone Mode Demonstration

This guide describes the setup of a standalone HBase instance running against the local filesystem. This is not based on Hadoop Cluster (which is not an appropriate configuration for a production instance of HBase), but will allow you to experiment with HBase. For more detail, please check https://hbase.apache.org/book.html section 2 (Quick Start - Standalone HBase). This demo has been test on Linux and Mac OS (the screenshots as follows are from experiments on Mac). For Windows user, http://ics.upjs.sk/~novotnyr/blog/334/setting-up-hbase-on-windows describes how to setup HBase based on Cygwin, but authors has not tested it yet.

1. JDK install

HBase is supported by JDK. Java 1.7+ is required. You need to find JAVA_HOME environment variable.

For Linux user, you can find it via

```
hadoop@masi-10:~$ export JAVA_HOME=$(readlink -f /usr/bin/java | sed "s:bin/java::")
hadoop@masi-10:~$ source ~/.bashrc
hadoop@masi-10:~$ echo "$JAVA_HOME"
/usr/lib/jvm/java-7-openjdk-amd64
```

/usr/lib/jvm/java-7-openjdk-amd64 is your JAVA_HOME path.

For Mac user,

```
obo@Shunxings-MacBook-Pro:~$ export JAVA_HOME=$(/usr/libexec/java_home)
obo@Shunxings-MacBook-Pro:~$ source ~/.bash_profile
obo@Shunxings-MacBook-Pro:~$  echo $JAVA_HOME
/Library/Java/JavaVirtualMachines/jdk1.8.0_45.jdk/Contents/Home
```

/Library/Java/JavaVirtualMachines/jdk1.8.0_45.jdk/Contents/Home

2. Download HBase source

Official suggested mirror download site is: http://download.nextag.com/apache/hbase/. This tutorial is tested on stable/hbase-1.1.4-bin.tar.gz

Uncompress the package, and change current directory to HBase directory

```
obo@Shunxings-MacBook-Pro:~/Downloads$ tar -zxf hbase-1.1.4-bin.tar.gz
obo@Shunxings-MacBook-Pro:~/Downloads$ cd hbase-1.1.4
obo@Shunxings-MacBook-Pro:~/Downloads/hbase-1.1.4$ ls
CHANGES.txt     LICENSE.txt     README.txt      conf/           hbase-webapps/
LEGAL           NOTICE.txt      bin/            docs/           lib/
obo@Shunxings-MacBook-Pro:~/Downloads/hbase-1.1.4$
```

3. Configuration

In Standalone mode, you need to edit 2 files. Now you are login to your HBase home directory $HBASE_HOME

3.1 $HBASE_HOME/conf/hbase-env.sh

You need to set your JAVA_HOME environment variable found from section-1 in hbase-env.sh.

```
# The java implementation to use.  Java 1.7+ required.
export JAVA_HOME=/Library/Java/JavaVirtualMachines/jdk1.8.0_45.jdk/Contents/Home
```

3.2 $HBASE_HOME/conf/hbase-site.sh

The HBase stores data in /tmp by default. Many servers are configured to delete the contents of /tmp upon reboot, so you should store the data elsewhere. Edit hbase-site.xml with the XML format as follows.

```
<configuration>
        <property>
                <name>hbase.rootdir</name>
                <value>file:///Users/obo/Downloads/hbase-1.1.4/hbase_data/hbase</value>
        </property>
</configuration>
```

Once you set your hbase.rootdir, you do not need to create the directory. HBase will do this for you.

4. Start HBase

Before you start HBase, there is no Java Virtual Machine (JVM) Process (get by command 'jps'). You can start your HBase by running the script of $HBASE_HOME/ bin/start-hbase.sh

Then you can see there is JVM process HMaster, it means your HBase is started successfully.

```
obo@Shunxings-MacBook-Pro:~/Downloads/hbase-1.1.4$ jps
3872 Jps
obo@Shunxings-MacBook-Pro:~/Downloads/hbase-1.1.4$ bin/start-hbase.sh
starting master, logging to /Users/obo/Downloads/hbase-1.1.4/bin/../logs/hbase-obo-master-Shunxings-MacBook-Pro.local.out
Java HotSpot(TM) 64-Bit Server VM warning: ignoring option PermSize=128m; support was removed in 8.0
Java HotSpot(TM) 64-Bit Server VM warning: ignoring option MaxPermSize=128m; support was removed in 8.0
obo@Shunxings-MacBook-Pro:~/Downloads/hbase-1.1.4$ jps
3934 HMaster
3966 Jps
obo@Shunxings-MacBook-Pro:~/Downloads/hbase-1.1.4$
```

5. HBase Shell

HBase contains a shell using which you can communicate with HBase via command 'bin/hbase shell'. The experiment in section-6 are all executed within HBase Shell.

```
obo@Shunxings-MacBook-Pro:~/Downloads/hbase-1.1.4$ bin/hbase shell
2016-04-04 20:14:10,829 WARN  [main] util.NativeCodeLoader: Unable to load native-hadoop library for your platform... using buil
tin-java classes where applicable
HBase Shell; enter 'help<RETURN>' for list of supported commands.
Type "exit<RETURN>" to leave the HBase Shell
Version 1.1.4, r14c0e77956f9bb4c6edf0378474264843e4a82c3, Wed Mar 16 21:18:26 PDT 2016

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

6. HBase Shell Command experiment

Before starting to do experiment, several concept of HBase data model should be introduced [reference by 'Introduction to HBase Schema Design edit by Amandeep Khurana'].

Concept	Description
Table	HBase organizes data into tables. Table names are Strings and composed of characters that are safe for use in a file system path
Row	Within a table, data is stored according to its row. Rows are identified uniquely by their row key. Row keys do not have a data type and are always treated as a byte[ ] (byte array).
Column Family	Data within a row is grouped by column family. Column families also impact the physical arrangement of data stored in HBase. For this reason, they must be defined up front and are not easily modified. Every row in a table has the same column families, although a row need not store data in all its families. Column families are Strings and composed of characters that are safe for use in a file system path.
Column Qualifier	Data within a column family is addressed via its column qualifier, or simply, column. Column qualifiers need not be specified in advance. Column qualifiers need not be consistent between rows. Like row keys, column qualifiers do not have a data type and are always treated as a byte[ ].
Cell:	A combination of row key, column family, and column qualifier uniquely identifies a cell. The data stored in a cell is referred to as that cell's value. Values also do not have a data type and are always treated as a byte[ ].
Timestamp:	Values within a cell are versioned. Versions are identified by their version number, which by default is the timestamp of when the cell was written. If a timestamp is not specified during a write, the current timestamp is used. If the timestamp is not specified for a read, the latest one is returned. The number of cell value versions retained by HBase is configured for each column family. The default number of cell versions is three.

For instance, in table as follows, 'Personalinfo' and 'Publicinfo' are Column Family, 'name', 'major' and 'hobby' are column qualifier. The row is identified as 'row1' and 'row2'

Row	Personalinfo		Publicinfo
	name	major	hobby
row1	Amy	CS	Baseball
row2	Bob	EE	Basketball

6.1 General Commands
The related commands are: status, version, whoami

6.1.1 status
Provides the status of HBase, for example, the number of servers. In standalone mode, there is only one server.

```
hbase(main):002:0> status
1 servers, 0 dead, 2.0000 average load
```

6.1.2 version
Provides the version of HBase being used.

```
hbase(main):003:0> version
1.1.4, r14c0e77956f9bb4c6edf0378474264843e4a82c3, Wed Mar 16 21:18:26 PDT 2016
```

6.1.3 whoami
Provides information about the user.

```
hbase(main):004:0> whoami
obo (auth:SIMPLE)
    groups: staff, com.apple.sharepoint.group.1, everyone, localaccounts, _appserverusr, admin, _appserveradm, _lpadmin, _appsto
re, _lpoperator, _developer, com.apple.access_screensharing, com.apple.access_ssh-disabled
```

6.2 Data Definition Language (DDL)
The commands in this sections are used to operate on the tables in HBase. The commands we test are: list, create, disable, enable, describe, alter, drop

6.2.1 list
Lists all the tables in HBase. Initially, there is no table in HBase, so the result is empty.

```
hbase(main):006:0> list
TABLE
0 row(s) in 0.0100 seconds

=> []
```

6.2.2 create

Creates a table. The format is:
create 'TABLE_NAME', 'COLUMN_FAMILY_1','COLUMN_FAMILY_2',...,'COLUMN_FAMILY_N'
In demo, we create a table named 'user', with a column family 'personalinfo'

```
hbase(main):007:0> create 'user','personalinfo'
0 row(s) in 1.2590 seconds

=> Hbase::Table - user
```

6.2.3 disable

Disables a table. If you want to delete a table or change its settings, you need to disable the table first.

```
hbase(main):008:0> disable 'user'
0 row(s) in 2.3280 seconds
```

6.2.4 enable

Re-enable a disabled table.

```
hbase(main):009:0> enable 'user'
0 row(s) in 0.8570 seconds
```

6.2.5 describe

Provides the description of a table. i.e. If table is enabled, Column family description.

```
hbase(main):013:0> describe 'user'
Table user is ENABLED
user
COLUMN FAMILIES DESCRIPTION
{NAME => 'personalinfo', BLOOMFILTER => 'ROW', VERSIONS => '1', IN_MEMORY => 'false', KEEP_DELETED_CELLS => 'FALSE', DATA_BLOCK_
ENCODING => 'NONE', TTL => 'FOREVER', COMPRESSION => 'NONE', MIN_VERSIONS => '0', BLOCKCACHE => 'true', BLOCKSIZE => '65536', RE
PLICATION_SCOPE => '0'}
1 row(s) in 0.0190 seconds
```

6.2.6 alter

Alters a table. We can alter a table's structure by adding/deleting column family. Please remember, before alter a table, we need to disable it first.

(1) Add a new column family

We first disable the table 'user', add a new column family 'publicinfo', and re-enable table 'user'. Finally we use 'describe' command to check the updated table structure information.

```
hbase(main):018:0> disable 'user'
0 row(s) in 2.2690 seconds

hbase(main):019:0> alter 'user',{NAME=>'publicinfo',METHOD=>'delete'}
Updating all regions with the new schema...
1/1 regions updated.
Done.
0 row(s) in 1.9230 seconds

hbase(main):020:0> enable 'user'
0 row(s) in 0.8400 seconds

hbase(main):021:0> describe 'user'
Table user is ENABLED
user
COLUMN FAMILIES DESCRIPTION
{NAME => 'personalinfo', BLOOMFILTER => 'ROW', VERSIONS => '1', IN_MEMORY => 'false', KEEP_DELETED_CELLS => 'FALSE', DATA_BLOCK_
ENCODING => 'NONE', TTL => 'FOREVER', COMPRESSION => 'NONE', MIN_VERSIONS => '0', BLOCKCACHE => 'true', BLOCKSIZE => '65536', RE
PLICATION_SCOPE => '0'}
1 row(s) in 0.0070 seconds
```

(2) Delete a column family

In this test, we delete column family 'publicinfo'

```
hbase(main):014:0> disable 'user'
0 row(s) in 2.2570 seconds

hbase(main):015:0> alter 'user','publicinfo'
Updating all regions with the new schema...
1/1 regions updated.
Done.
0 row(s) in 1.9140 seconds

hbase(main):016:0> enable 'user'
0 row(s) in 0.8380 seconds

hbase(main):017:0> describe 'user'
Table user is ENABLED
user
COLUMN FAMILIES DESCRIPTION
{NAME => 'personalinfo', BLOOMFILTER => 'ROW', VERSIONS => '1', IN_MEMORY => 'false', KEEP_DELETED_CELLS => 'FALSE', DATA_BLOCK_
ENCODING => 'NONE', TTL => 'FOREVER', COMPRESSION => 'NONE', MIN_VERSIONS => '0', BLOCKCACHE => 'true', BLOCKSIZE => '65536', RE
PLICATION_SCOPE => '0'}
{NAME => 'publicinfo', BLOOMFILTER => 'ROW', VERSIONS => '1', IN_MEMORY => 'false', KEEP_DELETED_CELLS => 'FALSE', DATA_BLOCK_EN
CODING => 'NONE', TTL => 'FOREVER', COMPRESSION => 'NONE', MIN_VERSIONS => '0', BLOCKCACHE => 'true', BLOCKSIZE => '65536', REPL
ICATION_SCOPE => '0'}
2 row(s) in 0.0160 seconds
```

6.2.7 drop

Drops (delete)a table from HBase. Before delete a table, we have to disable it first, or an error occurs shown as follows. Before we drop table 'user', using command 'list' we can see there is 1 table in HBase, and after dropping it, there is no more table exists.

```
hbase(main):028:0> list
TABLE
user
1 row(s) in 0.0100 seconds

=> ["user"]
hbase(main):029:0> drop 'user'

ERROR: Table user is enabled. Disable it first.

Here is some help for this command:
Drop the named table. Table must first be disabled:
  hbase> drop 't1'
  hbase> drop 'ns1:t1'


hbase(main):030:0> disable 'user'
0 row(s) in 2.2520 seconds

hbase(main):031:0> drop 'user'
0 row(s) in 1.2620 seconds

hbase(main):032:0> list
TABLE
0 row(s) in 0.0070 seconds

=> []
```

6.3 Data Manipulation Language (DML)

The commands in this section can manipulate data store in a HBase table.
The test commands are: put, get, delete, scan, count

6.3.1 put

```
hbase(main):033:0> create 'user','personalinfo'
0 row(s) in 1.2300 seconds

=> Hbase::Table - user
hbase(main):034:0> put 'user','row1','personalinfo:name','Amy'
0 row(s) in 0.0620 seconds

hbase(main):035:0> put 'user','row2','personalinfo:name','Bob'
0 row(s) in 0.0050 seconds

hbase(main):036:0> put 'user','row1','personalinfo:major','CS'
0 row(s) in 0.0030 seconds

hbase(main):037:0> put 'user','row2','personalinfo:major','EE'
0 row(s) in 0.0020 seconds
```

Puts a cell value at a specified column in a specified row in a particular table. We first create a 'user' table as 6.2.2 does. Then we put data into two rows, 'row1' and 'row2'. For each row, two column qualifiers are specified: 'name' and 'major'.

So after putting 4 records into HBase, actually we create a table as

Row	Personalinfo	
	name	major
row1	Amy	CS
row2	Bob	EE

6.3.2 get

Fetches the contents of row or a cell.

(1) Get designated column family's column qualifier. e.g. 'personalinfo:name'

```
hbase(main):038:0> get 'user','row1','personalinfo:name'
COLUMN                          CELL
 personalinfo:name              timestamp=1459820672469, value=Amy
1 row(s) in 0.0190 seconds
```

(2) Get all values of a designated column family. e.g. 'personalinfo'

```
hbase(main):039:0> get 'user','row1','personalinfo'
COLUMN                          CELL
 personalinfo:major             timestamp=1459820715506, value=CS
 personalinfo:name              timestamp=1459820672469, value=Amy
2 row(s) in 0.0030 seconds
```

(3) Get values by timestamp

If same 'put' is applied on same row's column family qualifier's value, the operation is treated as an update. As follows, we update row1's name as Amelia.

```
hbase(main):045:0> put 'user','row1','personalinfo:name','Amelia'
0 row(s) in 0.0090 seconds

hbase(main):046:0> get 'user','row1','personalinfo:name'
COLUMN                          CELL
 personalinfo:name              timestamp=1459821032331, value=Amelia
1 row(s) in 0.0070 seconds
```

Moreover, we can get the record via timestamp.

```
hbase(main):047:0> get 'user','row1',{COLUMN=>'personalinfo:name',TIMESTAMP=>1459821032331}
COLUMN                          CELL
 personalinfo:name              timestamp=1459821032331, value=Amelia
1 row(s) in 0.0050 seconds
```

6.3.3 count

Counts and returns the number of rows in a table. The 'count' operation will go through all data of a table if we do not specify a column qualifier. If we store large data on each rows, the speed is slow. In this demo we cannot present it.

```
hbase(main):048:0> count 'user'
2 row(s) in 0.0150 seconds

=> 2
```

6.3.4 scan

Scans and returns the table data.

```
hbase(main):049:0> scan 'user'
ROW                           COLUMN+CELL
 row1                         column=personalinfo:major, timestamp=1459820715506, value=CS
 row1                         column=personalinfo:name, timestamp=1459821032331, value=Amelia
 row2                         column=personalinfo:major, timestamp=1459820728493, value=EE
 row2                         column=personalinfo:name, timestamp=1459820684856, value=Bob
2 row(s) in 0.0060 seconds
```

6.3.5 delete

Deletes a cell value in a table.

(1) Delete designated column family's column qualifier. e.g. 'personalinfo:name', 'personalinfo:major'

```
hbase(main):050:0> delete 'user','row1','personalinfo:name'
0 row(s) in 0.0130 seconds

hbase(main):051:0> scan 'user'
ROW                           COLUMN+CELL
 row1                         column=personalinfo:major, timestamp=1459820715506, value=CS
 row2                         column=personalinfo:major, timestamp=1459820728493, value=EE
 row2                         column=personalinfo:name, timestamp=1459820684856, value=Bob
2 row(s) in 0.0060 seconds

hbase(main):052:0> delete 'user','row1','personalinfo:major'
0 row(s) in 0.0030 seconds

hbase(main):053:0> scan 'user'
ROW                           COLUMN+CELL
 row2                         column=personalinfo:major, timestamp=1459820728493, value=EE
 row2                         column=personalinfo:name, timestamp=1459820684856, value=Bob
1 row(s) in 0.0110 seconds
```

(2) Delete designated row. e.g. 'row2'

```
hbase(main):056:0> deleteall 'user','row2'
0 row(s) in 0.0030 seconds

hbase(main):057:0> scan 'user'
ROW                           COLUMN+CELL
0 row(s) in 0.0050 seconds
```

6.4 table_help - Provides help for table-reference commands.

```
hbase(main):058:0> table_help
Help for table-reference commands.

You can either create a table via 'create' and then manipulate the table via commands like 'put', 'get', etc.
See the standard help information for how to use each of these commands.

However, as of 0.96, you can also get a reference to a table, on which you can invoke commands.
For instance, you can get create a table and keep around a reference to it via:

   hbase> t = create 't', 'cf'

Or, if you have already created the table, you can get a reference to it:

   hbase> t = get_table 't'

You can do things like call 'put' on the table:

  hbase> t.put 'r', 'cf:q', 'v'

which puts a row 'r' with column family 'cf', qualifier 'q' and value 'v' into table t.

To read the data out, you can scan the table:

  hbase> t.scan

which will read all the rows in table 't'.

Essentially, any command that takes a table name can also be done via table reference.
Other commands include things like: get, delete, deleteall,
get_all_columns, get_counter, count, incr. These functions, along with
the standard JRuby object methods are also available via tab completion.

For more information on how to use each of these commands, you can also just type:

   hbase> t.help 'scan'

which will output more information on how to use that command.

You can also do general admin actions directly on a table; things like enable, disable,
flush and drop just by typing:

   hbase> t.enable
   hbase> t.flush
   hbase> t.disable
   hbase> t.drop

Note that after dropping a table, your reference to it becomes useless and further usage
```

6.5 Exit HBase Shell and Stop HBase

In HBase shell, type command 'exit' to close shell interface. Running the script of $HBASE_HOME/bin/stop-hbase.sh to stop all HBase JVM process.

```
hbase(main):059:0> exit
obo@Shunxings-MacBook-Pro:~/Downloads/hbase-1.1.4$ jps
4877 Jps
3934 HMaster
obo@Shunxings-MacBook-Pro:~/Downloads/hbase-1.1.4$ bin/stop-hbase.sh
stopping hbase..................
obo@Shunxings-MacBook-Pro:~/Downloads/hbase-1.1.4$ jps
4978 Jps
```