Lab: Using HBase

About This Lab

Objective:	To become familiar with HBase shell operations
Successful outcome:	Explore the structure and perform some command line HBase operations
Related lesson:	HBase Essentials: Apache HBase Overview

Lab Steps

Perform the following steps:

1. Launch the HBase Shell
    a. Issue the following command:

    ```
    $ hbase shell
    ```

    b. A prompt will appear that looks like:

    ```
    $ hbase shell

    hbase>
    ```

2. Get the version of hbase

    a. Issue the command:

    ```
    hbase> version
    ```

3. Get the status of hbase

    a. Issue the command:

```
hbase> status

hbase> status
1 servers, 0 dead, 4.0000 average load
```

    This shows there is only one server in the hbase "cluster."

    b. Get detailed status by adding 'detailed' to the status command:

```
hbase> status 'detailed'
version 0.96.0.2.0.6.0-76-hadoop2
0 regionsInTransition
master coprocessors: [ ]
1 live servers
    sandbox.hortonworks.com:60020 1390046418833
        requestsPerSecond=0.0, numberOfOnlineRegions=4,
usedHeapMB=102, maxHeapMB=1004, numberOfStores=4,
numberOfStorefiles=4, storefileUncompressedSizeMB=14,
storefileSizeMB=14, compressionRatio=1.0000, memstoreSizeMB=0,
storefileIndexSizeMB=0, readRequestsCount=1231,
writeRequestsCount=46, rootIndexSizeKB=16, totalStaticIndexSizeKB=8,
totalStaticBloomSizeKB=64, totalCompactingKVs=37,
currentCompactedKVs=37, compactionProgressPct=1.0, coprocessors=[ ]


"ambarismoketest,,1382317477828.8f2af39f4875a4539e3876c7b122ad8d."
            numberOfStores=1, numberOfStorefiles=1,
storefileUncompressedSizeMB=0, storefileSizeMB=0, memstoreSizeMB=0,
storefileIndexSizeMB=0, readRequestsCount=0, writeRequestsCount=0,
rootIndexSizeKB=0, totalStaticIndexSizeKB=0,
totalStaticBloomSizeKB=0, totalCompactingKVs=0,
currentCompactedKVs=0, compactionProgressPct=NaN
        "hbase:meta,,1"
            numberOfStores=1, numberOfStorefiles=1,
storefileUncompressedSizeMB=0, storefileSizeMB=0, memstoreSizeMB=0,
storefileIndexSizeMB=0, readRequestsCount=1223,
writeRequestsCount=14, rootIndexSizeKB=0, totalStaticIndexSizeKB=0,
totalStaticBloomSizeKB=0, totalCompactingKVs=37,
currentCompactedKVs=37, compactionProgressPct=1.0

"hbase:namespace,,1382307437528.dd774cb1b0266abb03b555f05f0fc334."
            numberOfStores=1, numberOfStorefiles=1,
storefileUncompressedSizeMB=0, storefileSizeMB=0, memstoreSizeMB=0,
storefileIndexSizeMB=0, readRequestsCount=6, writeRequestsCount=0,
rootIndexSizeKB=0, totalStaticIndexSizeKB=0,
totalStaticBloomSizeKB=0, totalCompactingKVs=0,
currentCompactedKVs=0, compactionProgressPct=NaN
```

```
        "users,,1390143562217.f241cb20cfd0eef7a212ca1d8128d103."
            numberOfStores=1, numberOfStorefiles=1,
storefileUncompressedSizeMB=14, storefileSizeMB=14,
compressionRatio=1.0000, memstoreSizeMB=0, storefileIndexSizeMB=0,
readRequestsCount=2, writeRequestsCount=32, rootIndexSizeKB=16,
totalStaticIndexSizeKB=8, totalStaticBloomSizeKB=64,
totalCompactingKVs=0, currentCompactedKVs=0,
compactionProgressPct=NaN
0 dead servers
```

c. Hbase is not a relational database. The operations available for data stored in hbase are the following:

   `Get`: retrieves a row or a subset of a row

   `Put`: Add or update a row or a subset of a row

   `Scan`: retrieve a range of sequential rows

   `Delete`: remove a row or a subset of a row

4. Determine what user you are connected as
   a. Use the `whoami` command:

```
hbase> whoami
train (auth:SIMPLE)
```

5. Ask HBase for help

   a. Run the `help` command and view the output

```
hbase(main):027:0> help
HBase Shell, version 0.96.0.2.0.6.0-76-hadoop2,
re6d7a56f72914d01e55c0478d74e5cfd3778f231, Thu Oct 17 18:15:20 PDT
2013
```

   b. Type `'help "COMMAND"'`, (e.g. `'help "get"'` - the quotes are necessary) for help on a specific command

   Commands are grouped. Type 'help "COMMAND_GROUP"', (e.g. 'help "general"') for help on a command group.

c. `ddl` stands for Data Definition Language. So help on creating tables defining column family attributes will be displayed by typing:

```
hbase> help 'ddl'

hbase> help 'get'
hbase> help 'put'
hbase> help 'scan'
hbase> help 'delete'
```

6. Create a table

a. Use the `create` command

```
hbase> create 't1','cf1'
0 row(s) in 0.5070 seconds
```

b. List the table you just created using the `list` command

```
hbase> list
```

7. Put some data in the table using the `put` command

```
hbase> put 't1', '1','cf1:name','yourname'
0 row(s) in 0.1480 seconds
```

8. Scan the table using the `scan` command

```
hbase> scan 't1'
ROW              COLUMN+CELL
1                column=cf1:name, timestamp=1390166020067, value=yourname
```

9. Add another row

```
hbase> put 't1', '2','cf1:name','theRainInSpain'


hbase> scan 't1'
```

10. Change your name

```
hbase> put 't1', '1','cf1:name','your_new_name'
0 row(s) in 0.0070 seconds
```

11. Drop the table

```
hbase> disable 't1'
0 row(s) in 1.2840 seconds

hbase> drop 't1'
0 row(s) in 0.1630 seconds
```

12. Create a table that stores more than one version of a column

```
hbase> create 't1', {NAME => 'f1', VERSIONS => 2}
```

This creates table t1, with column family f1 and any data stored in column family f1 will be permitted to have up to 2 versions. Versions beyond 2 will be deleted oldest first.

13. Insert multiple versions of a column.

```
hbase> put 't1','1', 'f1:name','name1'

hbase> put 't1','1', 'f1:name','name2'
```

14. Scan the table requesting multiple versions (note different timestamps)

```
hbase> scan 't1',{VERSIONS => 2}
ROW              COLUMN+CELL
 1               column=f1:name, timestamp=1390167231632, value=name2
 1               column=f1:name, timestamp=1390167226238, value=name1
```

15. Add a third value for the column identifier `f1:name`

```
hbase> put 't1','1', 'f1:name','name3'
0 row(s) in 0.0080 seconds
```

16. Scan again

```
hbase> scan 't1',{VERSIONS => 2}
ROW             COLUMN+CELL
 1              column=f1:name, timestamp=1390167445021, value=name3
 1              column=f1:name, timestamp=1390167231632, value=name2
1 row(s) in 0.0110 seconds
```

Try to scan for three versions

```
hbase> scan 't1',{VERSIONS => 3}
ROW              COLUMN+CELL
 1               column=f1:name, timestamp=1390167445021, value=name3
 1               column=f1:name, timestamp=1390167231632, value=name2
1 row(s) in 0.0170 seconds
```

Gets vs. Scans: If the table is large, the scan operation uses a lot of resources. Hbase was designed for the optimal lookup to be a single row get.

17. Exit from the HBase shell

```
hbase> exit
```

RESULT:

You have now performed some command line HBase operations.

Lab: Importing Tables from MySQL into HBase

About This Lab

Objective:	To import data from MySQL into HBase
Successful outcome:	Explore the classroom Lab Environment and import a table from a relational DataBase MySQL into HBase using Sqoop
Related lesson:	HBase Essentials: Apache HBase Overview

Lab Steps

Perform the following steps:

1. Start a web browser and connect to the url: `http://localhost:60010`

    a. The page should show the following:

    HMaster the Master process is running on sandbox.hortonworks.com.
    A RegionServer a worker process for HBase is running on sandbox.hortonworks.com
    Additional summary information about version and storage directories.

2. Create an HBase table using sqoop.

    a. Our data for this lab is currently in a MySQL database table. Sqoop is a command line tool to simplify importing database tables into hadoop. Typically the data source is a relational database and the destination is hadoop. The hadoop destination defaults to flat files stored into hdfs. Sqoop has many options available to format the data when stored into hdfs. We will use sqoop to create an HBase table.

    In this case we want an HBase table.

    b. Verify that sqoop is working and start with a simple command to perform a basic flat file import of the data.

    Run this command and make it is all on one line:

```
$ sqoop import --connect jdbc:mysql://localhost/orders --table
users_sample --username train -P
```

    The password when prompted is "`hadoop`"

c. Verify the data is present

```
$ hadoop fs -ls
drwx------   - train hdfs        0 2014-01-16 05:56 .Trash
drwx------   - train hdfs        0 2014-01-16 05:58 .staging
drwxr-xr-x   - train hdfs        0 2014-01-16 05:58 users_sample
```

The directory users_sample is where the sqoop job put the data.

d. Take a look at the content of the Directory users..

```
$ hadoop fs -ls users_sample Found 5 items
-rw-r--r--   3 train hdfs          0 2014-01-16 05:58 users_sample/_SUCCESS
-rw-r--r--   3 train hdfs    1217083 2014-01-16 05:58 users/part-m-00000
-rw-r--r--   3 train hdfs    1227912 2014-01-16 05:58 users/part-m-00001
-rw-r--r--   3 train hdfs    1228280 2014-01-16 05:58 users/part-m-00002
-rw-r--r--   3 train hdfs    1228908 2014-01-16 05:58 users/part-m-00003
```

e. The directory contains 4 files.

Why 4 files?

Sqoop runs 4 separate tasks to import the dataset, a way to do parallel processing,each worker imports about 1/4 of the rows in the table.

It is common in Hadoop to manage content as a directory of files, rather than a single large file. Often the name of the file is not as meaningful as the location of the file. Also because we had 4 separate tasks doing parallel work to import this data it would be challenging to have 4 processes potentially on different machines writing to the same file.

f. Take a look at one of the files. The `tail` option to the `fs` command allows us to view a few lines at the end of a file.

```
$ hadoop fs -tail users_sample/part-m-00000
```

```
$ hadoop fs -tail users_sample/part-m-00000
1,Larry E Schwarz,3084 Cody Ridge
Road,Loco,OK,73442,LarryESchwarz@teleworm.us,580-537-8691
2,Rosa M Akers,1009 Lost Creek
Road,Morgantown,PA,12345,RosaMAkers@superrito.com,610-913-4691
3,Michele J Cameron,1774 Ridge Road,Dodge
City,KS,67801,MicheleJCameron@fleckens.hu,620-682-1902
4,Mary O Duprey,4093 Adonais
Way,Atlanta,GA,30308,MaryODuprey@jourrapide.com,678-398-6820
5,Kenneth A Aguilar,3725 Plainfield
Avenue,Utica,NY,13502,KennethAAguilar@teleworm.us,315-606-7662
```

Looks like ID, name, address, city, state, postal code, email and phone.

g. Sqoop defaults to running a distributed request against the database with 4 workers. Sqoop runs a MapReduce job that does not require a reduce phase, also called a map-only job. Sqoop defaults to writing comma delimited text files into hdfs.

3. Remove the data generated by sqoop

```
$ hadoop fs -rm -r users_sample
```

4. Use sqoop to import the database table into an HBase table

   a. Run the following command keeping it on a single line:

   ```
   $ sqoop import --connect jdbc:mysql://localhost/orders --table
   users_sample --hbase-table users --hbase-create-table
   --column-family a --username train -P
   ```

   b. This command tells sqoop to import the table from MySQL, create an HBase table named users, and put all the data in column family a.

5. Use the HBase command line shell to view the HBase table

   a. Launch the HBase shell with this command

   ```
   $ hbase shell
   ```

   b. Ask HBase for a list of tables

   ```
   hbase> list
   TABLE
   ambarismoketest
   users
   2 row(s) in 2.9240 seconds

   => ["ambarismoketest", "users"]
   ```

   c. Reading this in HBase terminology we see:

   For Rowkey 1 has a column identifier address, in column Family a (a:address) with a timestamp of 1390143578758 and a value of 3084 Cody Ridge Road

   The timestamp is the number of milliseconds since Jan 1 1970.

   The last 3 digits have been stripped off as the date command expects second granularity and our java timestamp is milliseconds.

d. Exit the HBase Shell

```
hbase> exit
```

RESULT:

You now have imported a table from a relational DataBase MySQL into HBase using Sqoop.

Lab: Using HBase Shell Commands

About This Lab

Objective:	To use shell commands to view and manipulate rows and columns
Successful outcome:	Retrieval of full and partial rows and columns
Related lesson:	HBase Essentials: HBase Command Line Interface

Lab Steps

Perform the following steps:

1. Get row 12

   a. Typing `get 'users',12` will retrieve all values for any data keys for row 12.

```
   hbase> get 'users','12'

 COLUMN       CELL

 a:address   timestamp=1390143578758, value=4732 Sweetwood Drive
 a:city      timestamp=1390143578758, value=Denver
 a:email     timestamp=1390143578758, value=DaleLDomingo@teleworm.us
 a:name      timestamp=1390143578758, value=Dale L Domingo
 a:phone     timestamp=1390143578758, value=303-338-8457
 a:state     timestamp=1390143578758, value=CO
 a:zipcode   timestamp=1390143578758, value=80014 7 row(s) in 0.0200
seconds
```

Notes

The line `return` or `enter` is the command delimiter, you do not need a semi colon or any other command delimiter. The whole command on a single line followed by `enter`.

This shows us that row key 12 has an address in column family a (a:address) that was created at timestamp 1390143578758 with the value of 4732 Sweetwood Drive

Rowkey 12 also has name column in column family a with a timestamp of 1390143578758, and a value of Dale L Domingo

Rowkey 12 also has a phone column in column family a with a timestamp=1390143578758, and a value of 303-338-8457

You have now retrieved all of the data for row key 12 in the 'users' table.

2. Getting parts of a row instead of the whole row

    a. Retrieve just the name for user 1

    ```
    hbase> get 'users','1', {COLUMN=>'a:name'}
    COLUMN          CELL
    a:name          timestamp=1390143578758, value=Larry E Schwarz
    ```

    Specifying a:name states that we want the name column in column family "a"

3. Retrieve all the columns in column family 'a' for row key 1.

```
hbase> get 'users','1', {COLUMN=>'a'}

COLUMN              CELL

 a:address          timestamp=1390143578758, value=3084 Cody Ridge Road
 a:city             timestamp=1390143578758, value=Loco
 a:email            timestamp=1390143578758, value=LarryESchwarz@teleworm.us
 a:name             timestamp=1390143578758, value=Larry E Schwarz
 a:phone            timestamp=1390143578758, value=580-537-8691
 a:state            timestamp=1390143578758, value=OK
 a:zipcode          timestamp=1390143578758, value=73442
```

Exit from the HBase Shell

RESULT:

You have now retrieved full and partial rows and columns in HBase