**Login to BigVM**

* ssh 10.27.163.14 => password

**Find IP of MoVE**

* hostname -I => 10.27.128.90

**Transfer file from MoVE to BigVM**

* scp file\_name BigVM\_IP\_address:destination\_folder => scp test.csv 10.27.163.14:bigvm\_folder

**Transfer file from BigVM to MoVE**

* scp file\_name MoVE\_address:destination\_folder => scp test.csv 10.27.128.90:move\_folder

**Copy external CSV file to BigVM**

* wget https://www.data.vic.gov.au/data/dataset/38f43f13-d988-419a-8e99-39ff6c41e5f2/resource/00378584-3157-42f7-9370-a8f0e71ecfc1/download/.fileswateruse.csv -O <different\_file\_name>.csv

**Copy external ZIP file to BigVM**

* wget [https://data.gov.au/dataset/67265383-0ecc-4523-8ffd-02790297a65a/resource/2f236165-69a4-4d6a-9da6-5b39bfa70737/download/taxstats-1994-95-to-2008-09.zip -O taxstats-1994-95-to-2008-09.zip](https://data.gov.au/dataset/67265383-0ecc-4523-8ffd-02790297a65a/resource/2f236165-69a4-4d6a-9da6-5b39bfa70737/download/taxstats-1994-95-to-2008-09.zip%20-O%20taxstats-1994-95-to-2008-09.zip) -O file\_name.zip
* unzip file\_name.zip -d folder\_name

**Start HDFS (in order)**

* start-dfs-dh => NameNode (password), DataNode (password), SecondaryNameNode (password)
* stop-dfs.dh stops HDFS

**Start YARN (in order)**

* start-yarn.sh => ResourceManager (password), NodeManager (password)
* stop-yarn.sh stops YARN

**Start HBASE (in order)**

* start-hbase.sh => ZooKeeper (password), regiosServer (password)
* stop-hbase.sh stops HBASE

**Create directory**

* hadoop fs -mkdir -p /tmp/student\_id

**List Files & Folders**

* hadoop fs -ls . ==> lists content of user folder on HDFS
* hadoop fs -ls ==> no dot, lists current directory
* hadoop fs -ls -R ==> displays all files in subdirectories of path

**Upload Files to HDFS**

* Hadoop fs -put <local\_src1> <local\_src2> etc <hdfs-dest>
* Example - hadoop fs -put \*file.txt newDir
* Alternative – hadoop fs -copyFromLocal <local-src> <hdfs\_dest>

**Download Files from HDFS**

* ls \*file.txt ==> check if text files exist in local directory
* rm \*file.txt ==> remove text files from local directory before download
* hadoop fs -get <hdfs-src>
* Example - hadoop fs -get newDir/\* ==> download from hdfs-src to local-src
* Alternative – hadoop fs -copyToLocal <hdfs-src> <local-dest>

**Copy Files in HDFS**

* hadoop fs -cp <hdfs-source> <hdfs-dest> ==> -f can force file to overwrite
* Example – hadoop fs – cp newDir/anotherFile.txt newDir/anotherDir

**Move Files in HDFS (deletes in source directory)**

* hadoop fs -mv <hdfs-source> < hdfs-dest>
* Example – hadoop fs -mv newDir/newFile.txt newDir/anotherDir

**Remove Files in HDFS**

* hadoop fs -rm <args> - remove files
* hadoop fs -rmdir <args> - remove directory
* hadoop fs -rm -R <args> - recursive removal of folder and all its contents

**1- Data Definition Commands**

These commands operate on the tables in HBase to create, alter or drop a table.

**Create a table:**

* create '<table name>','<column family>'

**Enable/disable a table:**

* enable '<table name>'
* disable '<table name>'

**Changes to an existing table.**

For example, one can change the maximum number of cells of a column family, set/delete table scope operators, and delete a column family from a table.

**Changing the maximum number of cell to N:**

* alter '<table name>', NAME => '<column family>', VERSIONS => N

**Delete a column family from a table:**

* alter '<table name>', 'delete' => '<column family>'

**Make the table read-only:**

* alter '<table name>', READONLY

**Drop a disabled table from HBase:**

* drop '<table name>'

**Drop the tables matching the ‘regex’ given in the command. The tables should be disabled before dropping them.**

* drop\_all 'regular expression'

**2- Data Manipulation Commands**

The following commands manipulate (e.g., insert, update, delete) the data in the tables.

**Inserts a new row to the specified table**

* put '<table name>','row','<colfamily:colname>','<value>'
* put ‘insurance’,’999998’,’house:line’,’Tree House’

**Set a cell value at a specified column in a specified row in a particular table.**

* put '<table name>','row','<colfamily:colname>','<new value>'

**Reads contents of a specific row:**

* get '<table name>','row'

**Reads contents of a specific cell:**

* get '<table name>', 'rowid', {COLUMN => 'column family:column name'}
* get '<table name>', 'rowid', {FILTER => "ValueFilter(=, 'binary:705600')"}

**Delete a cell value in a table.:**

* delete '<table name>', '<row>', '<column name >', '<time stamp>'
* delete ‘insurance’,’999999’,’house:line’ – must have column name

**Delete all the cells in a given row of a table:**

* deleteall '<table name>', '<row>'
* deleteall ‘insurance’,’999998’

**Disable, drop, and recreate a specified table, all at once:**

* truncate '<table name>'

**List all the tables previously created in HBase.**

* list

**Return the table data.**

* scan'<table name>' – returns all rows (beware!)
* scan 'insurance', {FILTER => "ValueFilter(=, 'binary:705600') OR ValueFilter(=, 'binary:315000')"}
* scan 'insurance', {FILTER => "ValueFilter(=, 'substring:Steel')"} - starting with Steel (Steel Frame)
* scan 'insurance', {FILTER => "RowFilter(>=, 'binary:999900')"} – all rows >= 999900
* scan 'insurance', {FILTER => "TimestampsFilter(1527406053326)"}
* scan 'insurance', {FILTER => "SingleColumnValueFilter('house','point\_granularity',>=,'binary:5') AND SingleColumnValueFilter('house','point\_granularity','<=','binary:7')"}

**Count the number of rows in a table.**

* count '<table name>' INTERVAL => 10000

**Verify whether a table exists.**

* exists '<table name>'

**Return the description of a table.**

* describe '<table name>'

**Verify whether a table is enabled/disabled.**

* is\_enabled '<table name>'
* is\_disabled '<table name>'