[Hbase – 2.4]

Last Updated: 16th May 2022.

Hbase with Phoenix:
Hadoop Version : 3.1.2 , Hbase Version : 2.4.11 and Phoenix : 5.1.2 (phoenix-hbase-2.4.0-5.1.2-bin.tar.gz)

Hbase cluster only:
Hbase version : 3.1.2 and Hadoop : 3.1.2

https://community.cloudera.com/t5/Community-Articles/Compression-in-HBase/ta-p/247244

1. pre-requisite

You can install Hbase either in VM or in Docker container.

1) VM

Start your VM. Use Centos or Redhat Linux version above 6.0

2) On Docker:

Only Hbase – Cluster

```
# docker run --name hbase0 --hostname hmaster.master.com --privileged -p 2181:2181 -p 16000:16000 -p 16010:16010 -p 16002:16002 -p 16012:16012 -p 16020:16020 -p 16030:16030 -p 8080:8080 -p 8085:8085 -p 9090:9090 -p 9095:9095  -i -t  centos:7 /usr/sbin/init
```

Hbase & YARN Cluster

```
#docker run --name hbase0 --hostname hmaster.master.com --privileged -p 2181:2181 -p 16000:16000 -p 16010:16010 -p 16002:16002 -p 16012:16012 -p 16020:16020 -p 16030:16030 -p 8080:8080 -p 8085:8085 -p 8088:8088 -p 9870:9870 -p 9864:9864 -i -t  centos:7 /usr/sbin/init
```

2. Hbase Installation – 60 Minutes

HBase requires that a JDK be installed. Use jdk - jdk-8u121-linux-x64.tar.gz

Extract it and specify the JAVA_HOME in ~/.bashrc as mention below.

export PATH=$PATH:/opt/jdk/bin:/opt/hbase/bin
export JAVA_HOME=/opt/jdk
Extract the downloaded file, and change to the newly-created directory.

```
$ tar xzvf hbase-3.0.0-SNAPSHOT-bin.tar.gz –C /opt
#mv hbase-3.0.0-SNAPSHOT-bin hbase
$ cd hbase/
```

You must set the `JAVA_HOME` environment variable before starting HBase. To make this easier, HBase lets you set it within the *conf/hbase-env.sh* file.

```
# Set environment variables here.
# The java implementation to use.
export JAVA_HOME=/opt/jdk
```

The *bin/start-hbase.sh* script is provided as a convenient way to start HBase.
Issue the command, and if all goes well, a message is logged to standard output showing that HBase started successfully.

```
[root@4f5607f478bd hbase]# bin/start-hbase.sh
running master, logging to /opt/hbase/bin/../logs/hbase--master-4f5607f478bd.out
[root@4f5607f478bd hbase]# tail -f logs/hbase--master-4f5607f478bd.out
WARNING: An illegal reflective access operation has occurred
WARNING: Illegal reflective access by org.apache.hadoop.hbase.util.UnsafeAvailChecker (file:/opt/hbase/lib/hbase-common-2.3.2.jar) to method
java.nio.Bits.unaligned()
WARNING: Please consider reporting this to the maintainers of org.apache.hadoop.hbase.util.UnsafeAvailChecker
WARNING: Use --illegal-access=warn to enable warnings of further illegal reflective access operations
WARNING: All illegal access operations will be denied in a future release
```

Check the log in case of any issue.

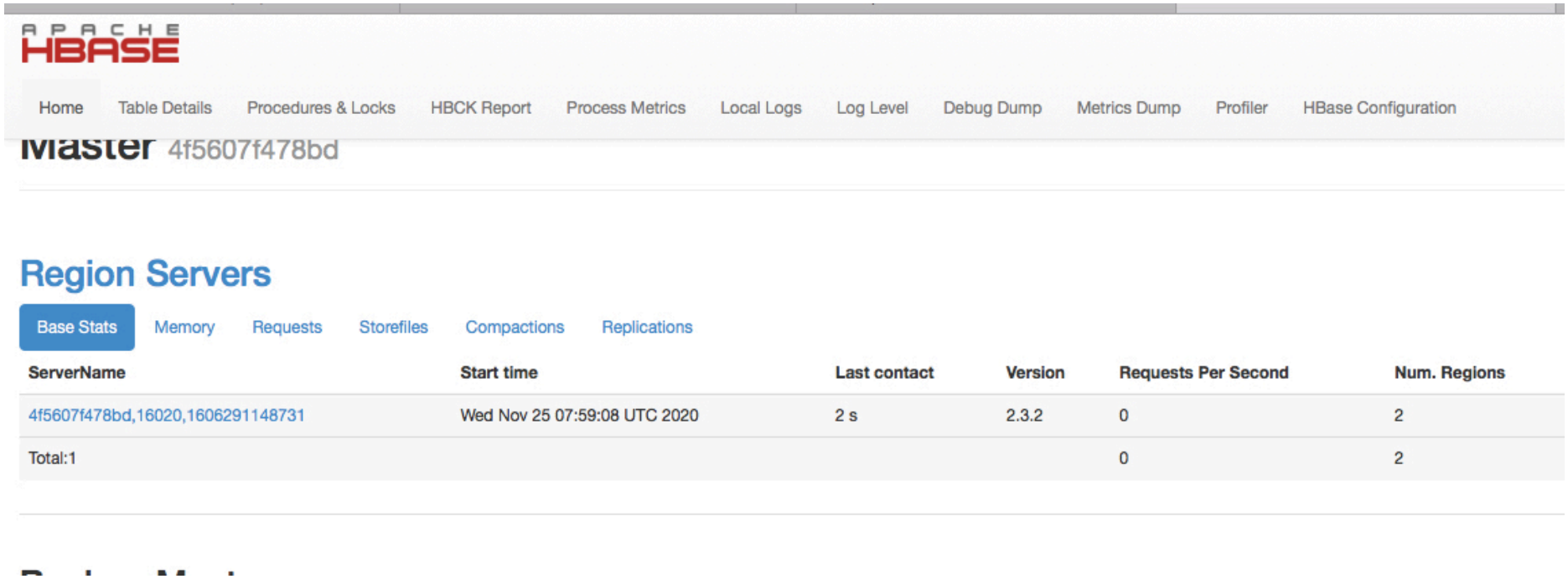
```
2020-11-25 08:00:16,204 INFO  [ReadOnlyZKClient-127.0.0.1:2181@0x5943051a] zookeeper.ZooKeeper: Session: 0x100000ae1d20002 closed
2020-11-25 08:00:16,205 INFO  [ReadOnlyZKClient-127.0.0.1:2181@0x5943051a-EventThread] zookeeper.ClientCnxn: EventThread shut down for sessio
n: 0x100000ae1d20002
[root@4f5607f478bd logs]# ls
SecurityAuth.audit  hbase--master-4f5607f478bd.log  hbase--master-4f5607f478bd.out  hbase--master-4f5607f478bd.out.1
[root@4f5607f478bd logs]# pwd
/opt/hbase/logs
[root@4f5607f478bd logs]#
```

You can use the jps command to verify that you have one running process called HMaster.

```
[root@4f5607f478bd logs]# jps
626 Jps
199 HMaster
[root@4f5607f478bd logs]#
```

In standalone mode HBase runs all daemons within this single JVM, i.e. the HMaster, a single HRegionServer, and the ZooKeeper daemon.

Go to *http://localhost:16010* to view the HBase Web UI



---------- Lab Ends Here ----------

3. Using the HBase Shell & Data Access – 60 minutes

Procedure: Use HBase For the First Time

1. Connect to HBase.

   Connect to your running instance of HBase using the `hbase shell` command, located in the *bin/* directory of your HBase install. In this example, some usage and version information that is printed when you start HBase Shell has been omitted. The HBase Shell prompt ends with a `>` character.

   ```
   $ ./bin/hbase shell
   ```

2. Display HBase Shell Help Text.

   Type `help` and press Enter, to display some basic usage information for HBase Shell, as well as several example commands. Notice that table names, rows, columns all must be enclosed in quote characters.

```
ses where applicable
HBase Shell
Use "help" to get list of supported commands.
Use "exit" to quit this interactive shell.
For Reference, please visit: http://hbase.apache.org/2.0/book.html#shell
Version 2.3.2, r8dc7de1c75ab615b8992b928150e8b476cb053e3, Sun Sep 20 13:28:26 UTC 2020
Took 0.0022 seconds
hbase(main):001:0> help
HBase Shell, version 2.3.2, r8dc7de1c75ab615b8992b928150e8b476cb053e3, Sun Sep 20 13:28:26 UTC 2020
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.

COMMAND GROUPS:
  Group name: general
  Commands: processlist, status, table_help, version, whoami

  Group name: ddl
  Commands: alter, alter_async, alter_status, clone_table_schema, create, describe, disable, disable_all, drop, drop_all, enable, enable_all,
 exists, get_table, is_disabled, is_enabled, list, list_regions, locate_region, show_filters
```

3. Create a table.

   Use the `create` command to create a new table. You must specify the table name and the ColumnFamily name.

   ```
   hbase(main):001:0> create 'test', 'cf'
   ```

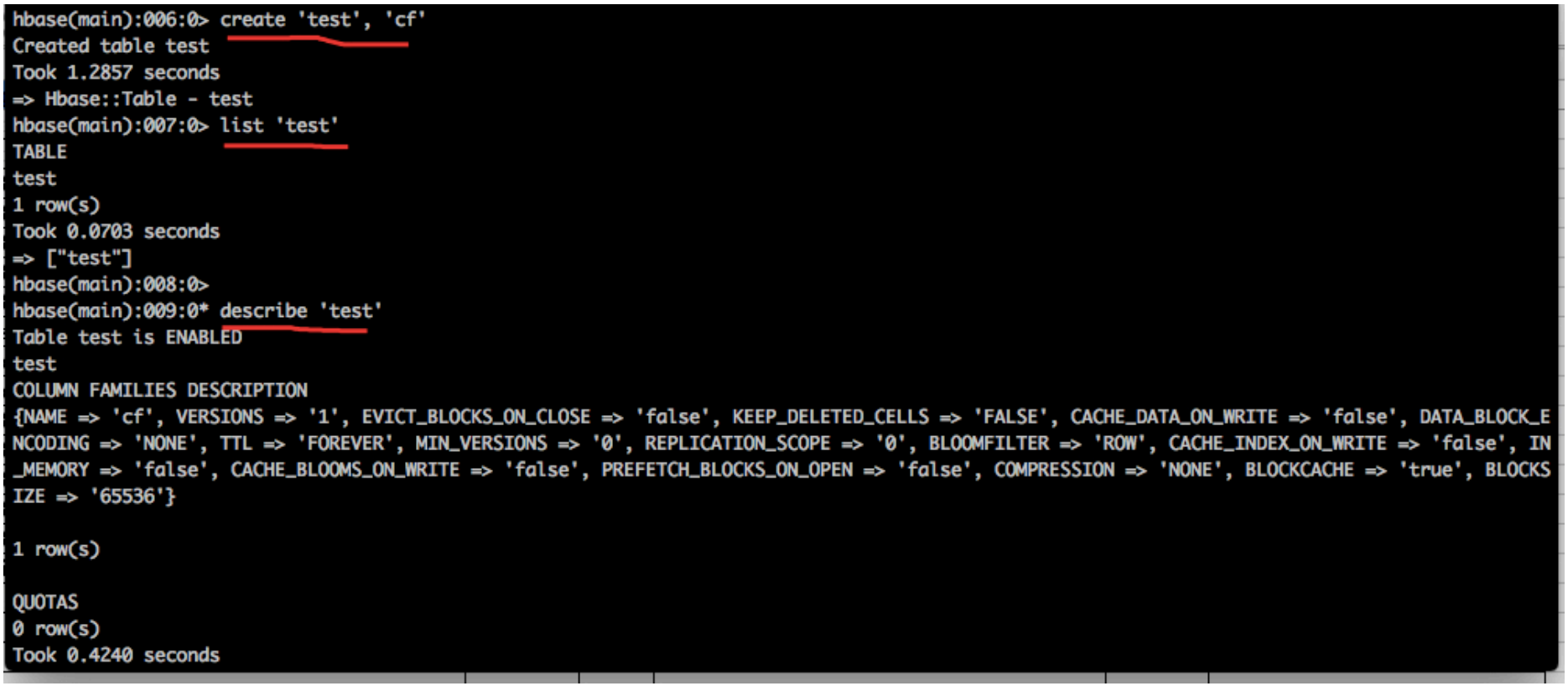
4. List Information About your Table

   Use the `list` command to confirm your table exists

   ```
   hbase(main):002:0> list 'test'
   ```

Now use the `describe` command to see details, including configuration defaults

```
hbase(main):003:0> describe 'test'
```



5. Put data into your table.

   To put data into your table, use the `put` command.

```
hbase(main):003:0> put 'test', 'row1', 'cf:a', 'value1'

hbase(main):004:0> put 'test', 'row2', 'cf:b', 'value2'

hbase(main):005:0> put 'test', 'row3', 'cf:c', 'value3'
```

   Here, we insert three values, one at a time. The first insert is at `row1`, column `cf:a`, with a value of `value1`. Columns in HBase are comprised of a column family prefix, `cf` in this example, followed by a colon and then a column qualifier suffix, `a`in this case.

6. Scan the table for all data at once.

   One of the ways to get data from HBase is to scan. Use the `scan` command to scan the table for data. You can limit your scan, but for now, all data is fetched.

```
hbase(main):006:0> scan 'test'
```

```
hbase(main):010:0>
hbase(main):011:0* put 'test', 'row1', 'cf:a', 'value1'
Took 0.0958 seconds
hbase(main):012:0> put 'test', 'row2', 'cf:b', 'value2'
Took 0.0077 seconds
hbase(main):013:0> put 'test', 'row3', 'cf:c', 'value3'
Took 0.0079 seconds
hbase(main):014:0> scan 'test'
ROW                                   COLUMN+CELL
 row1                                 column=cf:a, timestamp=2020-11-25T08:18:25.387, value=value1
 row2                                 column=cf:b, timestamp=2020-11-25T08:18:31.324, value=value2
 row3                                 column=cf:c, timestamp=2020-11-25T08:18:37.532, value=value3
3 row(s)
Took 0.0746 seconds
```

7. Get a single row of data.

   To get a single row of data at a time, use the `get` command.

   ```
   hbase(main):007:0> get 'test', 'row1'
   ```

8. Disable a table.

   If you want to delete a table or change its settings, as well as in some other situations, you need to disable the table first, using the `disable` command. You can re-enable it using the `enable` command.

   ```
   hbase(main):008:0> disable 'test'

   Try fetching information from the table. It should throw an exception.

   hbase(main):007:0> get 'test', 'row1'
   ```

```
hbase(main):015:0> get 'test', 'row1'
COLUMN                                CELL
 cf:a                                 timestamp=2020-11-25T08:18:25.387, value=value1
1 row(s)
Took 0.0681 seconds
hbase(main):016:0> disable 'test'
Took 0.4304 seconds
hbase(main):017:0> get 'test', 'row1'
COLUMN                                CELL

ERROR: Table test is disabled!

For usage try 'help "get"'

Took 0.4916 seconds
```

```
hbase(main):009:0> enable 'test'
```

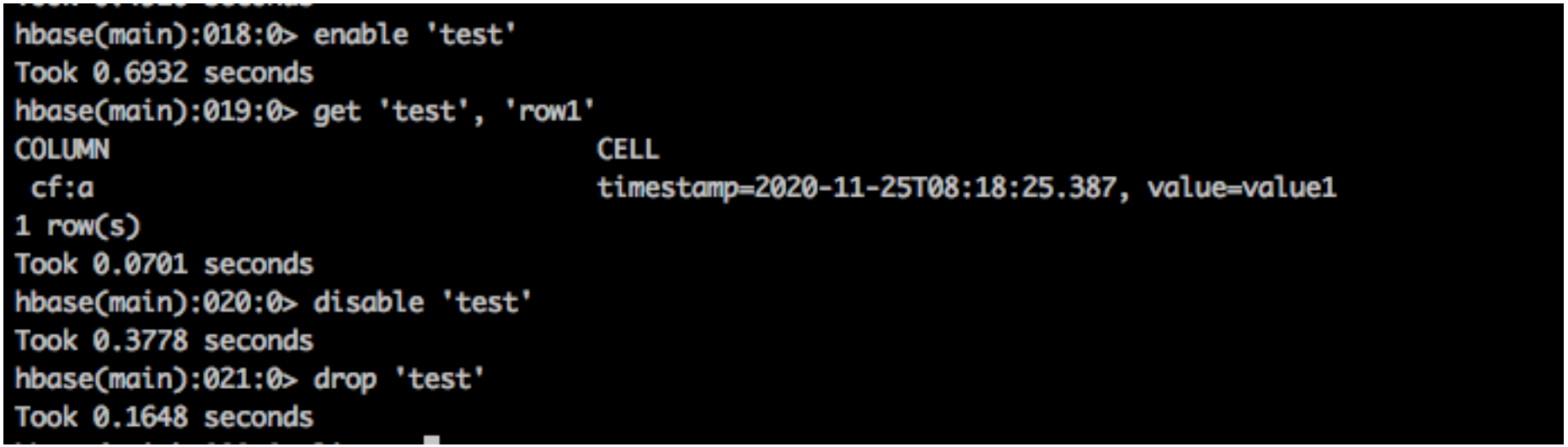
Disable the table again if you tested the `enable` command above:

```
hbase(main):010:0> disable 'test'
```

9. Drop the table.

   To drop (delete) a table, use the `drop` command.

```
hbase(main):011:0> drop 'test'
```



10. Exit the HBase Shell.

    To exit the HBase Shell and disconnect from your cluster, use the `quit` command. HBase is still running in the background.

Start the Shell :

#bin/hbase shell

create_namespace 'entertainment'

create 'entertainment:movie', {NAME => 'desc',VERSIONS => 2}, {NAME => 'media'}

list

```
hbase(main):001:0> create_namespace 'entertainment'
Took 0.7972 seconds
hbase(main):002:0> create 'entertainment:movie', {NAME => 'desc',VERSIONS => 2}, {NAME => 'media'}
Created table entertainment:movie
Took 0.6836 seconds
=> Hbase::Table - entertainment:movie
hbase(main):003:0> list
TABLE
entertainment:movie
1 row(s)
Took 0.0406 seconds
=> ["entertainment:movie"]
hbase(main):004:0>
```

alter 'entertainment:movie', NAME => 'desc', VERSIONS => 5

put 'entertainment:movie', 'row1', 'desc:title', 'Home Alone'

put 'entertainment:movie', 'row1', 'desc:title', 'Home Alone 2', 1274032629663

put 'entertainment:movie', 'row2', 'desc:title', 'Braveheart'

```
hbase(main):005:0> alter 'entertainment:movie', NAME => 'desc', VERSIONS => 5
Updating all regions with the new schema...
1/1 regions updated.
Done.
Took 1.8517 seconds
hbase(main):006:0> put 'entertainment:movie', 'row1', 'desc:title', 'Home Alone'
Took 0.1570 seconds
hbase(main):007:0> put 'entertainment:movie', 'row1', 'desc:title', 'Home Alone 2', 1274032629663
Took 0.0105 seconds
hbase(main):008:0> put 'entertainment:movie', 'row2', 'desc:title', 'Braveheart'
Took 0.0079 seconds
hbase(main):009:0>
```

Different ways of retrieving row information in the Hbase:

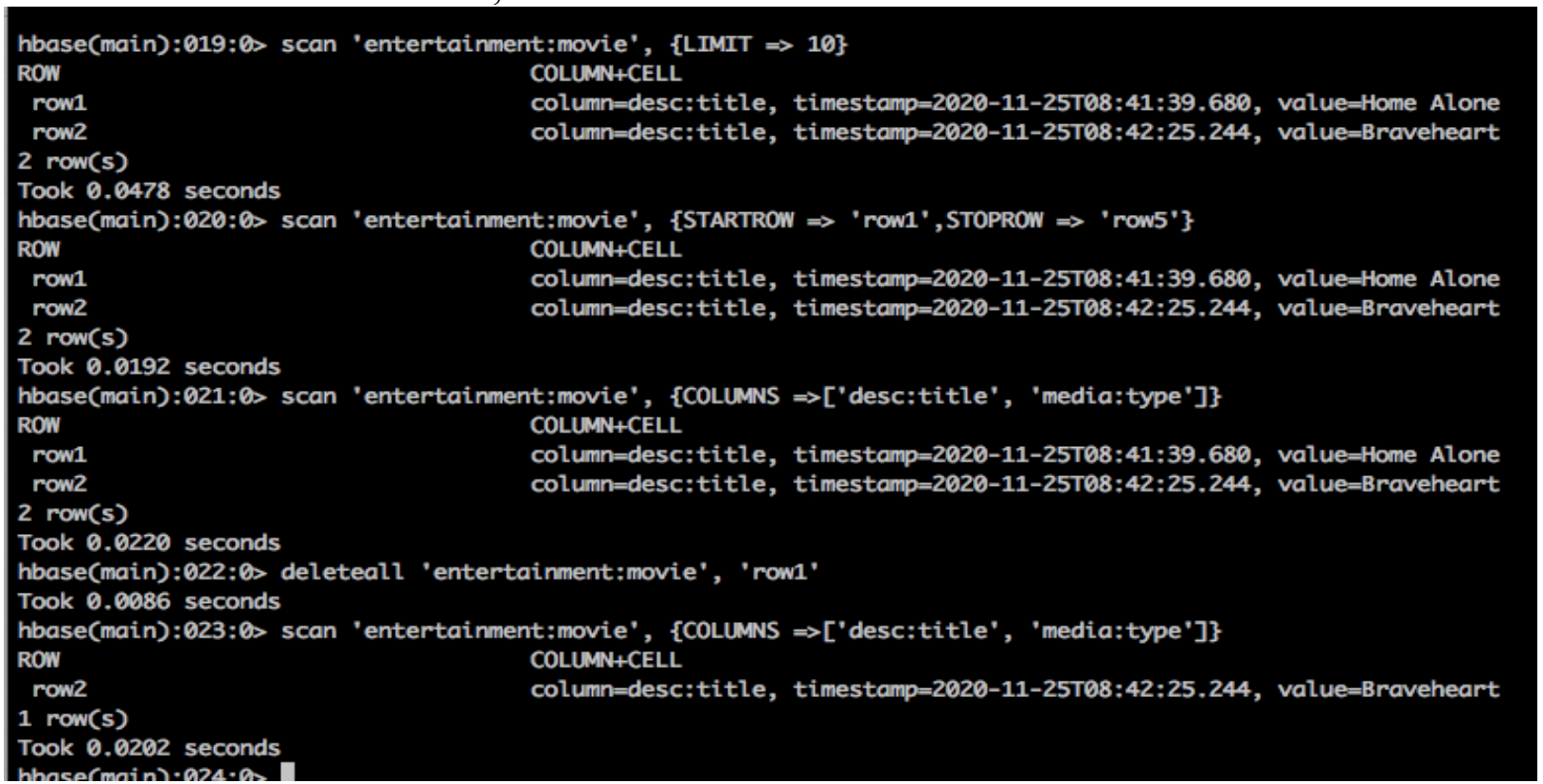
get 'entertainment:movie', 'row1'

```
get 'entertainment:movie', 'row1', {COLUMN => 'desc:title'}
get 'entertainment:movie', 'row1', {COLUMN => 'desc:title',VERSIONS => 2}
get 'entertainment:movie', 'row1', {COLUMN => ['desc']}
```

```
hbase(main):010:0> get 'entertainment:movie', 'row1'
COLUMN                                CELL
 desc:title                           timestamp=2020-11-25T08:41:39.680, value=Home Alone
1 row(s)
Took 0.0560 seconds
hbase(main):011:0> get 'entertainment:movie', 'row1', {COLUMN => 'desc:title'}
COLUMN                                CELL
 desc:title                           timestamp=2020-11-25T08:41:39.680, value=Home Alone
1 row(s)
Took 0.0174 seconds
hbase(main):012:0> get 'entertainment:movie', 'row1', {COLUMN => 'desc:title',VERSIONS => 2}
COLUMN                                CELL
 desc:title                           timestamp=2020-11-25T08:41:39.680, value=Home Alone
 desc:title                           timestamp=2010-05-16T17:57:09.663, value=Home Alone 2
1 row(s)
Took 0.0340 seconds
hbase(main):013:0> get 'entertainment:movie', 'row1', {COLUMN => ['desc']}
COLUMN                                CELL
 desc:title                           timestamp=2020-11-25T08:41:39.680, value=Home Alone
1 row(s)
Took 0.0280 seconds
hbase(main):014:0> get 'entertainment:movie', 'row1', {COLUMN => 'desc:title',VERSIONS => 3}
COLUMN                                CELL
 desc:title                           timestamp=2020-11-25T08:41:39.680, value=Home Alone
 desc:title                           timestamp=2010-05-16T17:57:09.663, value=Home Alone 2
```

Different ways of fetching rows :

scan 'entertainment:movie', {LIMIT => 10}
scan 'entertainment:movie', {STARTROW => 'row1',STOPROW => 'row5'}
scan 'entertainment:movie', {COLUMNS =>['desc:title', 'media:type']}
deleteall 'entertainment:movie', 'row1'



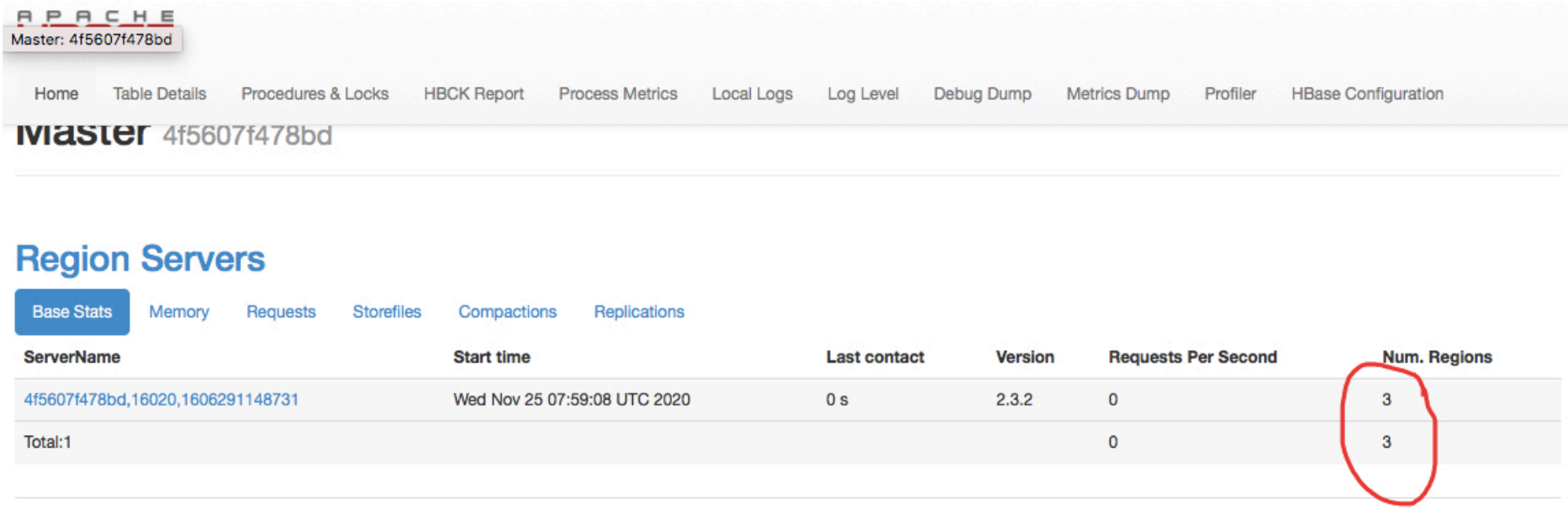
---------- Lab Ends Here ----------

4. Understanding Hbase Architecture – 30 Minutes

Determine the no of region in a region Server.

Using Web UI.:

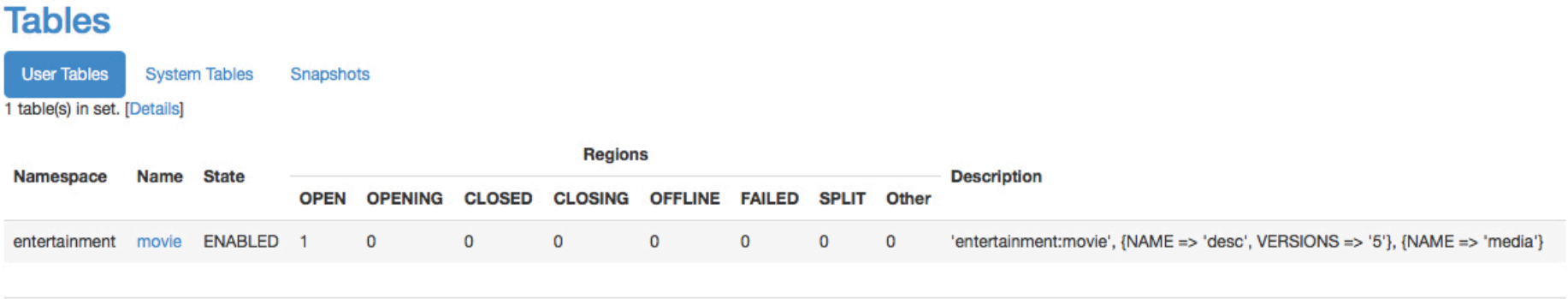
Determine the No of regions in our Existing Hbase Node.



In the above example, its having 3 regions in the node, 4f*.

How many tables are there? Scroll down till you find a section as shown below in the Admin web UI.

Click on the User Tables Tab : There will be a table, movie which we have created earlier.



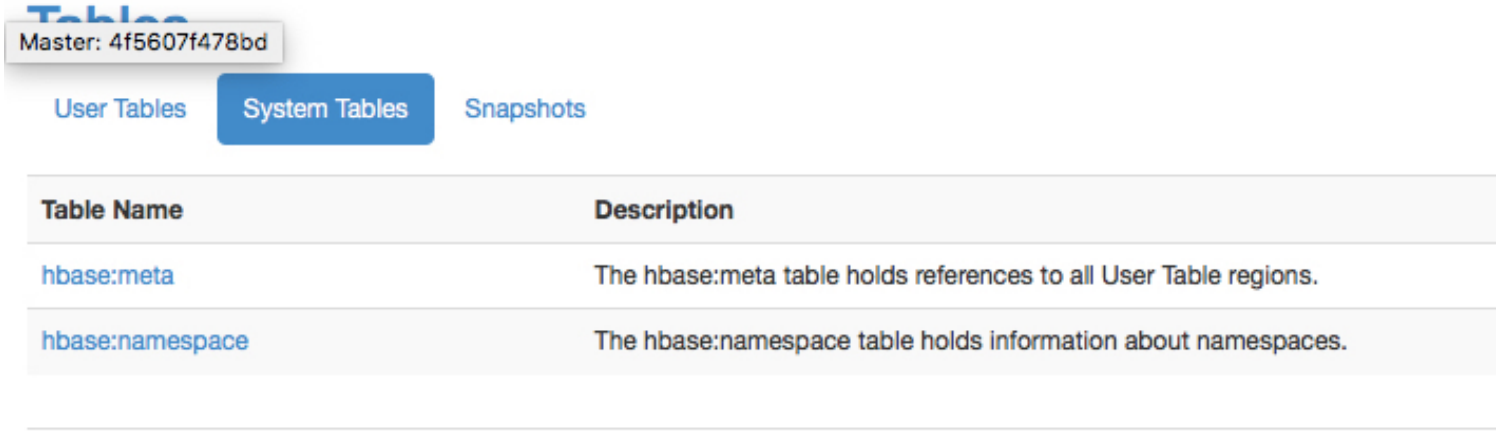
Tables

User Tables | System Tables | Snapshots

1 table(s) in set. [Details]

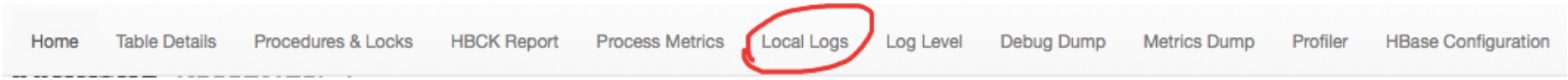
Namespace	Name	State	Regions							Description
			OPEN	OPENING	CLOSED	CLOSING	OFFLINE	FAILED	SPLIT	Other	
entertainment	movie	ENABLED	1	0	0	0	0	0	0	0	'entertainment:movie', {NAME => 'desc', VERSIONS => '5'}, {NAME => 'media'}

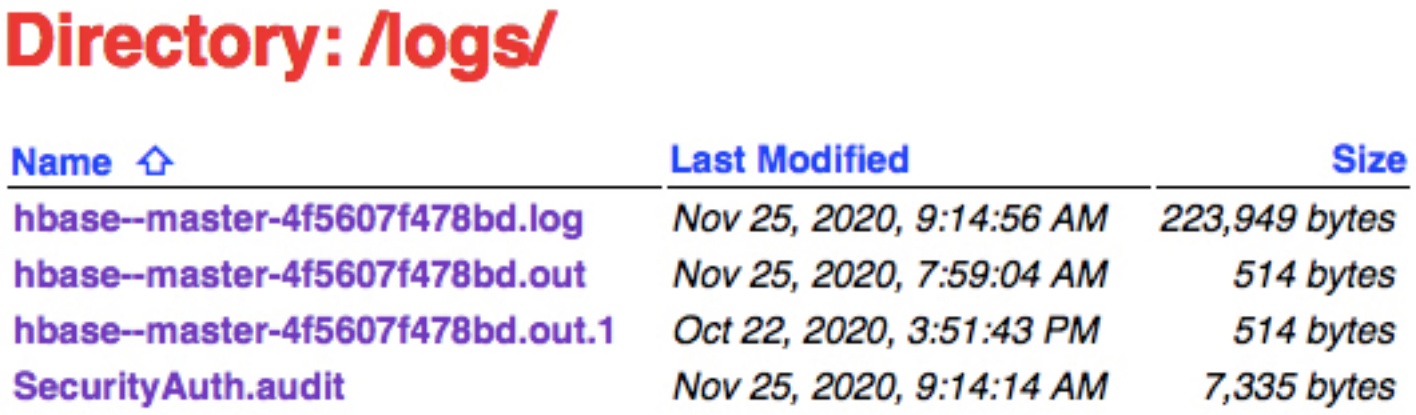
Click on the System Tables.



There are two system tables; meta and namepace.

Click on the Local Logs to determine the various Logs.





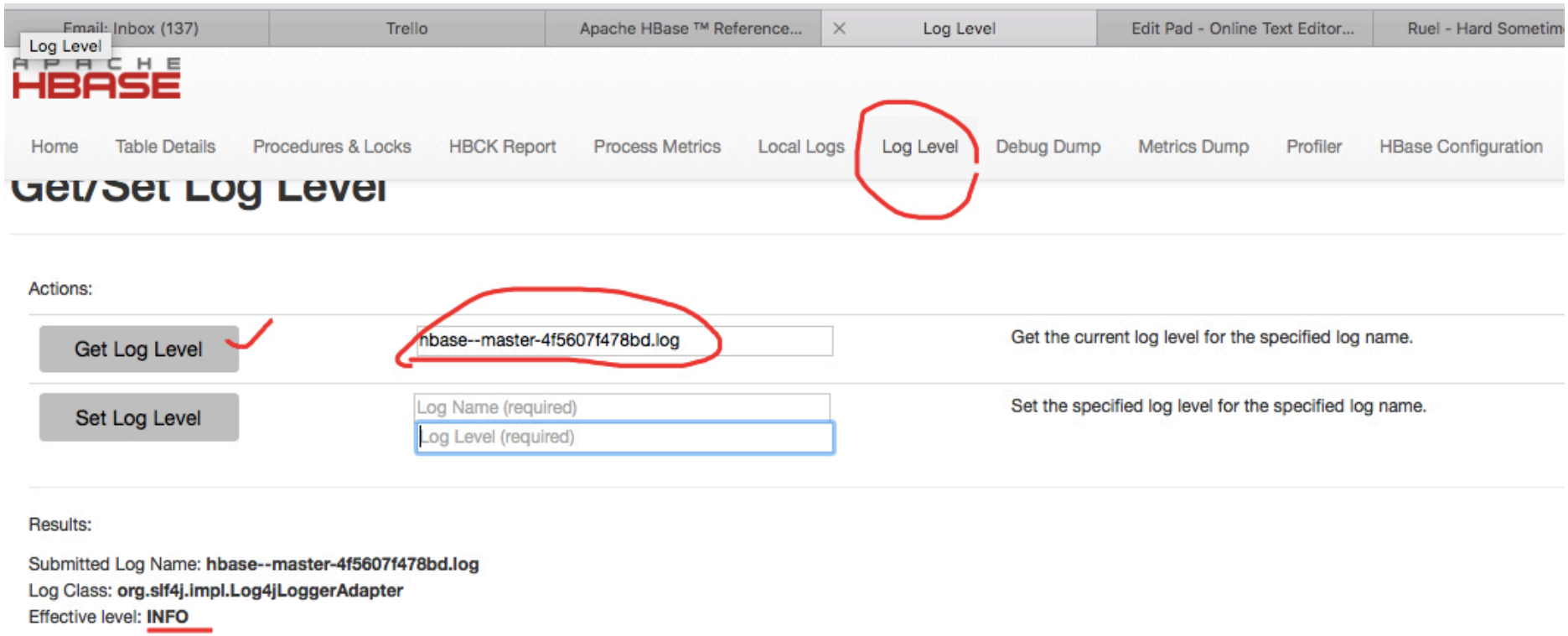
Click on the hbase-*.log to view its content.

127.0.0.1:16010/logs/hbase--master-4f5607f478bd.log

```
2020-11-25 09:09:05,011 INFO  [LruBlockCacheStatsExecutor] hfile.LruBlockCache: totalSize=156.93 KB, freeSize=199.05 MB, max=199.20 MB, blockCount=5, accesses=53, hits=42,
Accesses=47, cachingHits=38, cachingHitsRatio=80.85%, evictions=419, evicted=0, evictedPerRun=0.0
FO  [master/4f5607f478bd:16000.Chore.1] balancer.BaseLoadBalancer: Start Generate Balance plan for cluster.
FO  [master/4f5607f478bd:16000.Chore.1] master.HMaster: Normalizer ran successfully in PT0.011S. Submitted 0 plans, affecting 0 tables.
2020-11-25 09:12:27,429 INFO  [ReadOnlyZKClient-127.0.0.1:2181@0x066a7829] zookeeper.ZooKeeper: Initiating client connection, connectString=127.0.0.1:2181
sessionTimeout=10000 watcher=org.apache.hadoop.hbase.zookeeper.ReadOnlyZKClient$$Lambda$216/0x0000000800f4d440@7c8c7741
2020-11-25 09:12:27,431 INFO  [ReadOnlyZKClient-127.0.0.1:2181@0x066a7829] zookeeper.ClientCnxnSocket: jute.maxbuffer value is 4194304 Bytes
2020-11-25 09:12:27,431 INFO  [ReadOnlyZKClient-127.0.0.1:2181@0x066a7829] zookeeper.ClientCnxn: zookeeper.request.timeout value is 0. feature enabled=
2020-11-25 09:12:27,433 INFO  [ReadOnlyZKClient-127.0.0.1:2181@0x066a7829-SendThread(127.0.0.1:2181)] zookeeper.ClientCnxn: Opening socket connection to server
localhost/127.0.0.1:2181. Will not attempt to authenticate using SASL (unknown error)
2020-11-25 09:12:27,435 INFO  [ReadOnlyZKClient-127.0.0.1:2181@0x066a7829-SendThread(127.0.0.1:2181)] zookeeper.ClientCnxn: Socket connection established, initiating
session, client: /127.0.0.1:35672, server: localhost/127.0.0.1:2181
2020-11-25 09:12:27,438 INFO  [ReadOnlyZKClient-127.0.0.1:2181@0x066a7829-SendThread(127.0.0.1:2181)] zookeeper.ClientCnxn: Session establishment complete on server
localhost/127.0.0.1:2181, sessionid = 0x100000ae1d2000d, negotiated timeout = 10000
2020-11-25 09:13:27,512 INFO  [ReadOnlyZKClient-127.0.0.1:2181@0x066a7829] zookeeper.ZooKeeper: Session: 0x100000ae1d2000d closed
2020-11-25 09:13:27,513 INFO  [ReadOnlyZKClient-127.0.0.1:2181@0x066a7829-EventThread] zookeeper.ClientCnxn: EventThread shut down for session: 0x100000ae1d2000d
2020-11-25 09:14:04,742 INFO  [LruBlockCacheStatsExecutor] hfile.LruBlockCache: totalSize=156.93 KB, freeSize=199.05 MB, max=199.20 MB, blockCount=5, accesses=56, hits=45,
hitRatio=80.36%, , cachingAccesses=50, cachingHits=41, cachingHitsRatio=82.00%, evictions=449, evicted=0, evictedPerRun=0.0
2020-11-25 09:14:14,048 INFO  [master/4f5607f478bd:16000.Chore.1] master.HMaster: Normalizer ran successfully in PT0.012S. Submitted 0 plans, affecting 0 tables.
2020-11-25 09:14:14,052 INFO  [master/4f5607f478bd:16000.Chore.1] balancer.BaseLoadBalancer: Start Generate Balance plan for cluster.
2020-11-25 09:14:56,034 INFO  [master:store-WAL-Roller] wal.AbstractFSWAL: Rolled WAL
/opt/hbase/tmp/hbase/MasterData/WALs/4f5607f478bd,16000,1606291144491/4f5607f478bd%2C16000%2C1606291144491.1606294786757 with entries=0, filesize=83 B; new WAL
/opt/hbase/tmp/hbase/MasterData/WALs/4f5607f478bd,16000,1606291144491/4f5607f478bd%2C16000%2C1606291144491.1606295696011
2020-11-25 09:14:56,039 INFO  [master:store-WAL-Roller] wal.AbstractFSWAL: Archiving
file:/opt/hbase/tmp/hbase/MasterData/WALs/4f5607f478bd,16000,1606291144491/4f5607f478bd%2C16000%2C1606291144491.1606294786757 to
file:/opt/hbase/tmp/hbase/MasterData/oldWALs/4f5607f478bd%2C16000%2C1606291144491.1606294786757
2020-11-25 09:14:56,042 INFO  [master:store-WAL-Roller] region.MasterRegionUtils: Moved
file:/opt/hbase/tmp/hbase/MasterData/oldWALs/4f5607f478bd%2C16000%2C1606291144491.1606294786757 to
file:/opt/hbase/tmp/hbase/oldWALs/4f5607f478bd%2C16000%2C1606291144491.1606294786757$masterlocalwal$
```

You can look for errors in this file, if you are having issue.
As shown above, the current log level is INFO. You can determine the current log level using the Log Level tab in the Admin Web UI section.

Log Name :**hbase--master-4f5607f478bd.log** Click on Get Log Level.

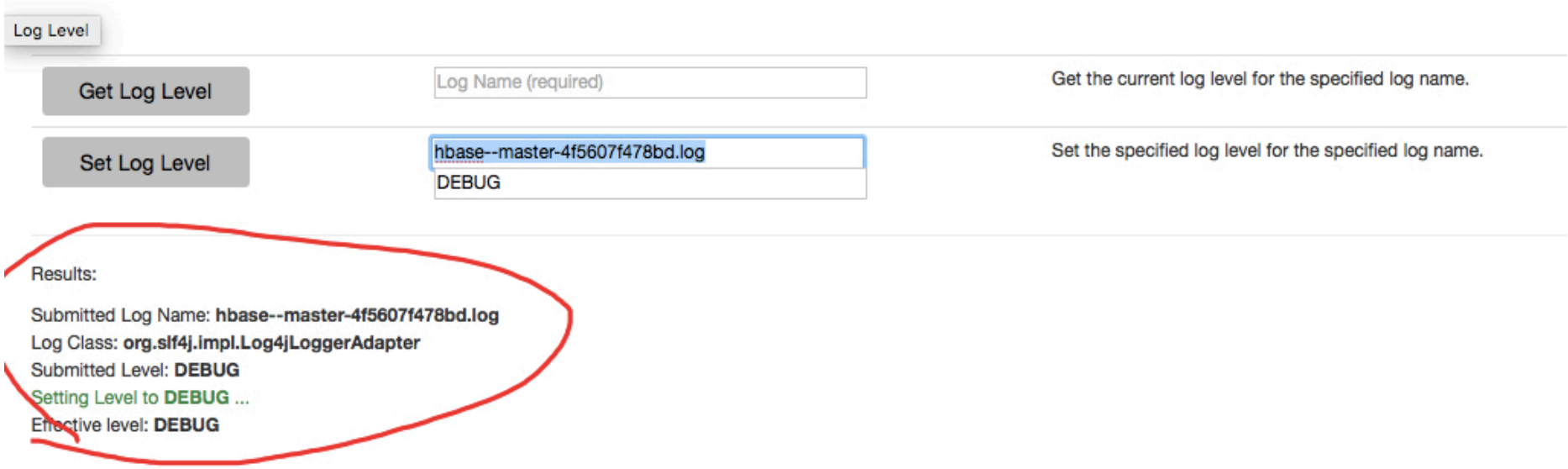


Its showing the current log level to INFO.
Let us change it to WARNING.

hbase--master-4f5607f478bd.log
DEBUG

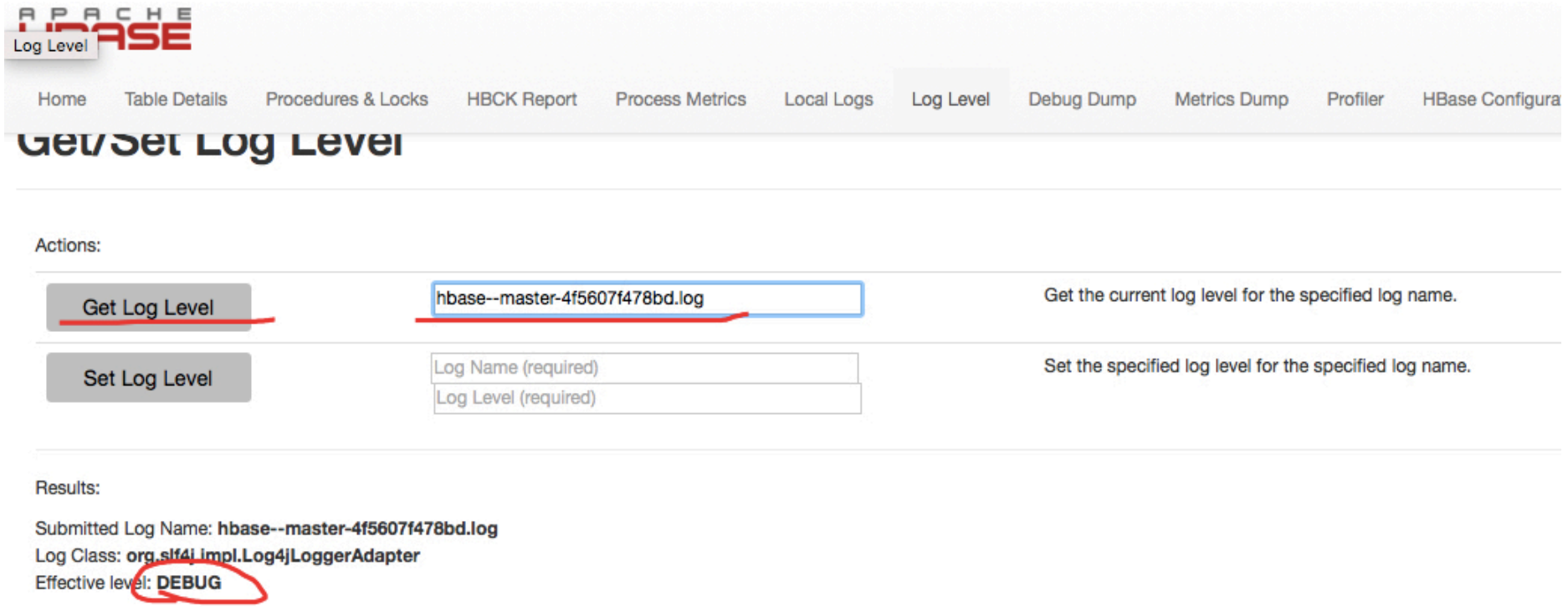
Click on the Set LOG LEVEL.

OSTech



It may take a few minutes.

Later verify the log level.



It should be DEBUG as shown above.

You can find a region for a particular table using the following option:

Click on table Details → Select the User tables e.x entertainment:movie

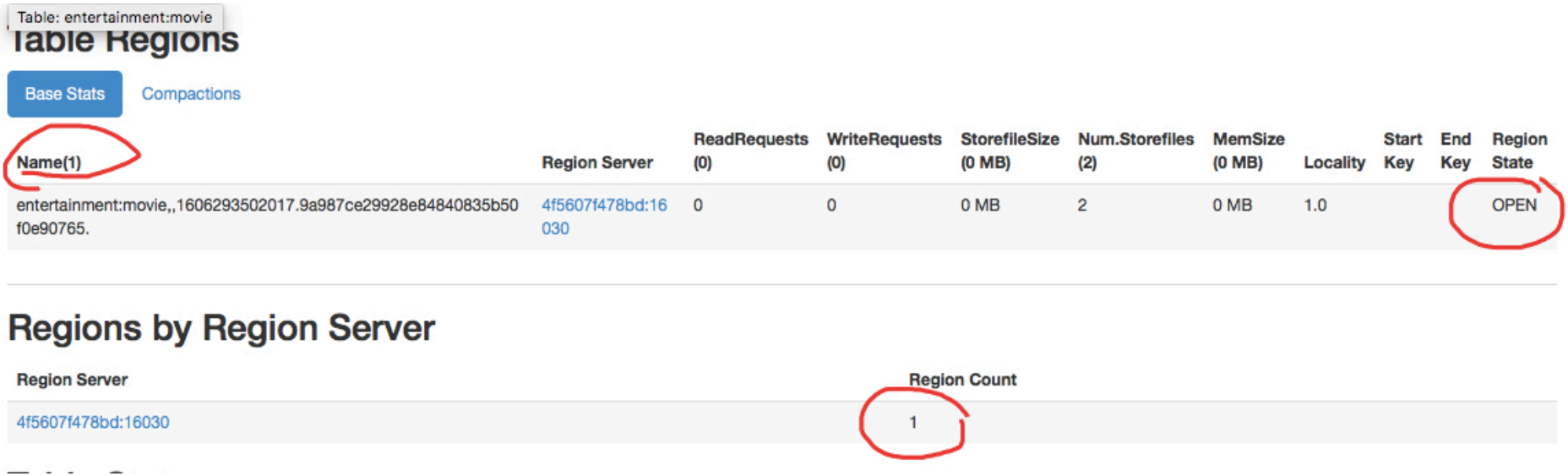


Table: entertainment:movie

Table Regions

Base Stats | Compactions

Name(1)	Region Server	ReadRequests (0)	WriteRequests (0)	StorefileSize (0 MB)	Num.Storefiles (2)	MemSize (0 MB)	Locality	Start Key	End Key	Region State
entertainment:movie,,1606293502017.9a987ce29928e84840835b50f0e90765.	4f5607f478bd:16030	0	0	0 MB	2	0 MB	1.0			OPEN

Regions by Region Server

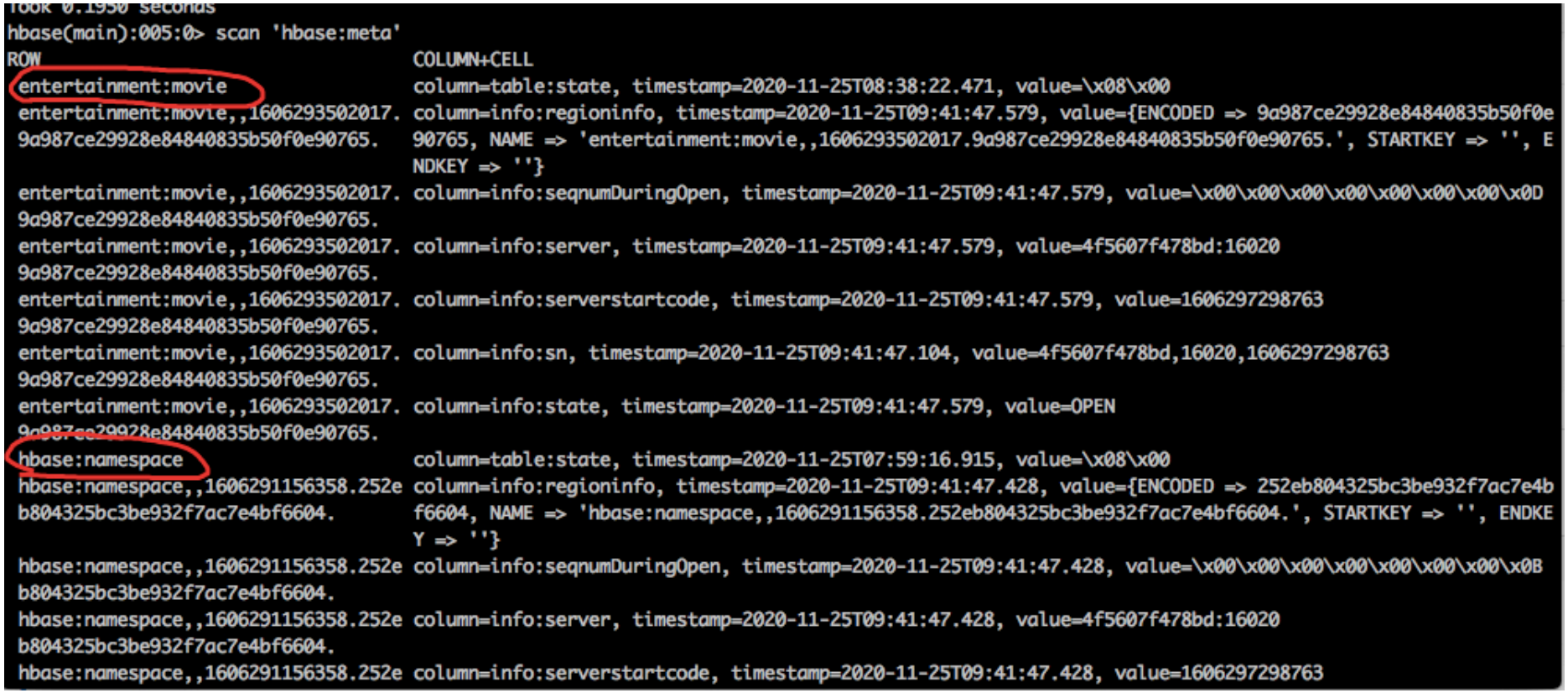
Region Server	Region Count
4f5607f478bd:16030	1

As shown above, there is a single region for this table which is open for request.

Understand the system tables:

Execute the following commands in the hbase shell.

describe 'hbase:meta'
scan 'hbase:meta'



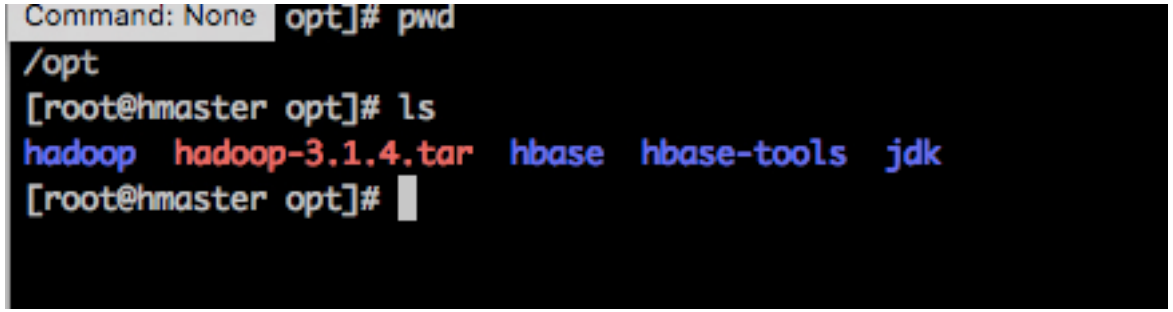
records of movie and namespace tables are store in the meta table as shown above.

---------- Lab Ends Here ----------

5. Hadoop Single Node Cluster – 60 Minutes

Hadoop : 3.1.2
JDK : jdk-8u221
https://www.apache.org/dyn/closer.cgi/hadoop/common/hadoop-3.1.4/hadoop-3.1.4.tar.gz

Extract in a folder and rename the folder as Hadoop as show below.



Prepare to Start the Hadoop Cluster

Unpack the downloaded Hadoop distribution. In the distribution, edit the file /opt/hadoop/`etc/hadoop/hadoop-env.sh` to define some parameters as follows or in the ~/.bashrc file:

```
export HDFS_NAMENODE_USER="root"
export HDFS_DATANODE_USER="root"
export HDFS_SECONDARYNAMENODE_USER="root"
export YARN_RESOURCEMANAGER_USER="root"
```

```
export YARN_NODEMANAGER_USER="root"
```

```
# set to the root of your Java installation

export JAVA_HOME=/opt/jdk/
```

```
# Technically, the only required environment variable is JAVA_HOME.
# All others are optional.  However, the defaults are probably not
# preferred.  Many sites configure these options outside of Hadoop,
# such as in /etc/profile.d

export HDFS_NAMENODE_USER="root"
export HDFS_DATANODE_USER="root"
export HDFS_SECONDARYNAMENODE_USER="root"
export YARN_RESOURCEMANAGER_USER="root"
export YARN_NODEMANAGER_USER="root

# The java implementation to use. By default, this environment
# variable is REQUIRED on ALL platforms except OS X!
export JAVA_HOME=/opt/jdk/

# Location of Hadoop.  By default, Hadoop will attempt to determine
# this location based upon its execution path.
# export HADOOP_HOME=
```

Try the following command:

cd /opt/hadoop

```
$ bin/hadoop
```

Pseudo-Distributed Operation

Hadoop can also be run on a single-node in a pseudo-distributed mode where each Hadoop daemon runs in a separate Java process.

Configuration

Use the following:

etc/hadoop/core-site.xml:

```
<configuration>
    <property>
        <name>fs.defaultFS</name>
        <value>hdfs://hmaster.master.com:9000</value>
    </property>
</configuration>
```

:etc/hadoop/hdfs-site.xml:

```
<?xml version="1.0" encoding="UTF-8"?>
<?xml-stylesheet type="text/xsl" href="configuration.xsl"?>
<!--
  Licensed under the Apache License, Version 2.0 (the "License");
  you may not use this file except in compliance with the License.
  You may obtain a copy of the License at

    http://www.apache.org/licenses/LICENSE-2.0

  Unless required by applicable law or agreed to in writing, software
  distributed under the License is distributed on an "AS IS" BASIS,
  WITHOUT WARRANTIES OR CONDITIONS OF ANY KIND, either express or implied.
  See the License for the specific language governing permissions and
  limitations under the License. See accompanying LICENSE file.
-->

<!-- Put site-specific property overrides in this file. -->

<configuration>
<property>
<name>dfs.namenode.name.dir</name>
<value>file:///opt/hdfs_hb0/namenode</value>
<description>NameNode directory for namespace and transaction logs storage.</description>
</property>
<property>
<name>dfs.datanode.data.dir</name>
<value>file:///opt/hdfs_hbo/datanode</value>
<description>DataNode directory</description>
</property>
<property>
<name>dfs.replication</name>
<value>2</value>
</property>
```

```
<property>
<name>dfs.permissions</name>
<value>false</value>
</property>
<property>
<name>dfs.datanode.use.datanode.hostname</name>
<value>false</value>
</property>
<property>
<name>dfs.namenode.datanode.registration.ip-hostname-check</name>
<value>false</value>
</property>
<property>
      <name>dfs.namenode.decommission.interval</name>
      <value>30</value>
<name>dfs.replication</name>
<value>2</value>
</property>
<property>
<name>dfs.permissions</name>
<value>false</value>
</property>
<property>
<name>dfs.datanode.use.datanode.hostname</name>
<value>false</value>
</property>
<property>
<name>dfs.namenode.datanode.registration.ip-hostname-check</name>
<value>false</value>
</property>
<property>
      <name>dfs.namenode.decommission.interval</name>
      <value>30</value>
  </property>
  <property>
      <name>dfs.client.datanode-restart.timeout</name>
      <value>30</value>
  </property>
```

```
</configuration>
```

Setup passphraseless ssh
yum install openssh*

change the password of root:
#passwd

Enter : root123

Start the ssh server.
systemctl start sshd

Now check that you can ssh to the localhost without a passphrase:

```
$ ssh hmaster.master.com
```

If you cannot ssh to host without a passphrase, execute the following commands and try ssh again:

```
$ ssh-keygen -t rsa -P '' -f ~/.ssh/id_rsa

$ cat ~/.ssh/id_rsa.pub >> ~/.ssh/authorized_keys

$ chmod 0600 ~/.ssh/authorized_keys
```

Create the necessary directory structure:

#mkdir -p /opt/hdfs_hb0/namenode

#mkdir -p /opt/hdfs_hb0/datanode

1. Format the filesystem:

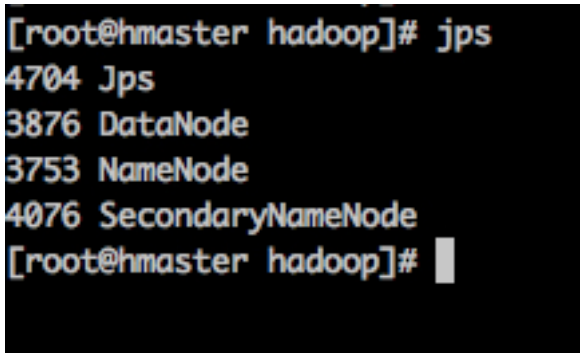
```
2.    $ bin/hdfs namenode -format
```

3. Start NameNode daemon and DataNode daemon:

```
4.    $ sbin/start-dfs.sh
```

The hadoop daemon log output is written to the `$HADOOP_LOG_DIR` directory (defaults to `$HADOOP_HOME/logs`).

Verify the process : jps



5. Browse the web interface for the NameNode; by default it is available at:
   - NameNode - `http://localhost:9870/`
6. Make the HDFS directories required to execute MapReduce jobs:

```
7.   $ bin/hdfs dfs -mkdir /user
```

```
   $ bin/hdfs dfs -mkdir /user/root
```

```
[root@hmaster hadoop]# bin/hdfs dfs -mkdir /user
[root@hmaster hadoop]# bin/hdfs dfs -mkdir /user/root
[root@hmaster hadoop]# bin/hdfs dfs -ls /
Found 1 items
drwxr-xr-x   - root supergroup          0 2020-11-27 14:53 /user
[root@hmaster hadoop]# bin/hdfs dfs -ls -R /
drwxr-xr-x   - root supergroup          0 2020-11-27 14:53 /user
drwxr-xr-x   - root supergroup          0 2020-11-27 14:53 /user/root
[root@hmaster hadoop]#
```

Following configuration are Optional – Not Required.

1. Configure parameters as follows:

etc/hadoop/mapred-site.xml:

```
<configuration>
```

```
    <property>
        <name>mapreduce.framework.name</name>
        <value>yarn</value>
    </property>
    <property>
        <name>mapreduce.application.classpath</name>

<value>$HADOOP_MAPRED_HOME/share/hadoop/mapreduce/*:$HADOOP_MAPRED_HOME/share/hadoop/mapreduce/lib/*</value>
    </property>
```

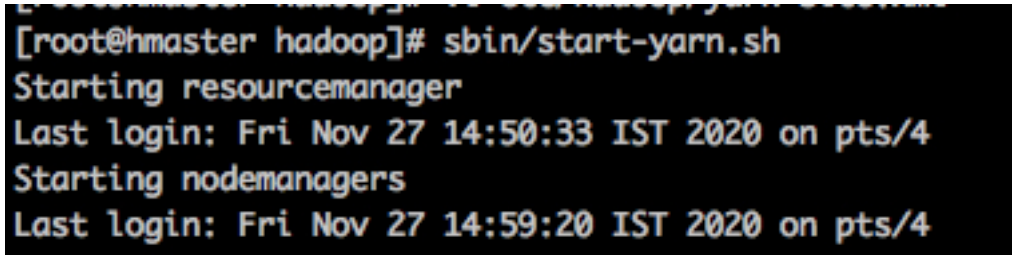
```
</configuration>
```

etc/hadoop/yarn-site.xml:

```
<configuration>
    <property>
        <name>yarn.nodemanager.aux-services</name>
        <value>mapreduce_shuffle</value>
    </property>
    <property>
        <name>yarn.nodemanager.env-whitelist</name>

<value>JAVA_HOME,HADOOP_COMMON_HOME,HADOOP_HDFS_HOME,HADOOP_CONF_DIR,CLASSPATH_PREPEND_DISTCACHE,HADOOP_YARN
_HOME,HADOOP_MAPRED_HOME</value>
    </property>
</configuration>
```

2. Start ResourceManager daemon and NodeManager daemon:

```
  $ sbin/start-yarn.sh
```



3. Browse the web interface for the ResourceManager; by default it is available at:
    - ResourceManager - `http://localhost:8088/`
4. Run a MapReduce job.
5. When you're done, stop the daemons with:

```
$ sbin/stop-yarn.sh
```

---------- Lab Ends Here ----------

6. Using HDFS – 30 Minutes

Most of your interaction with the system will be through commands supported by the bin/hdfs script. If you start a terminal and run this program with no arguments, it prints a help message. To try this, run the following command:

export PATH=$PATH:/opt/hadoop/bin:/opt/hadoop/sbin

$ hdfs

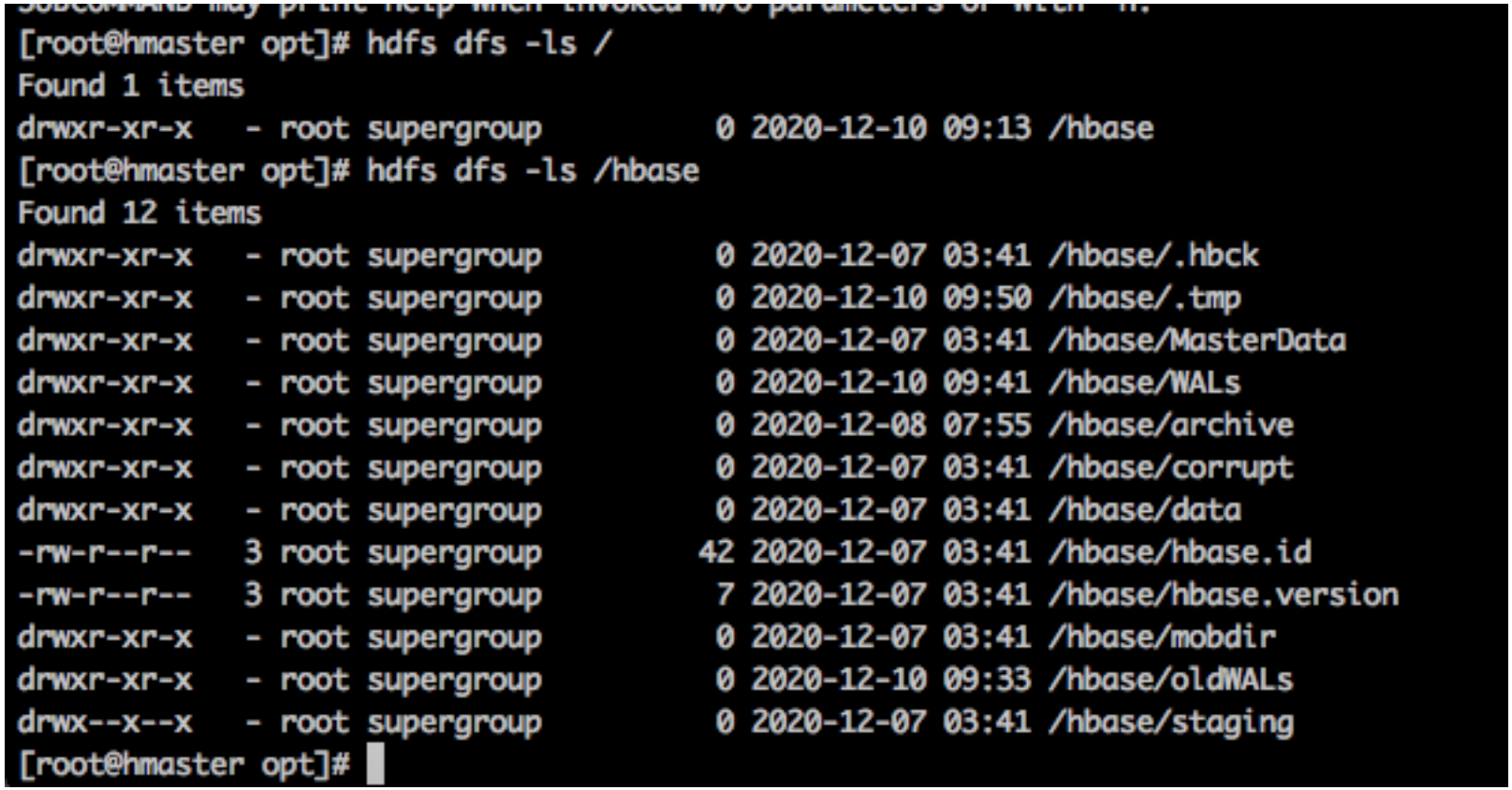
Enter:

$ hdfs dfs -ls /

This shows you the contents of the root directory in HDFS. There will be multiple entries, one of which is /user. Individual users have a “home” directory under. Folders may varies.

View the contents of the /hbase directory by entering the following:

$ hdfs dfs -ls /hbase



Enter the following:

$ hdfs dfs -ls /data/training

There are no files, it would display an error message.

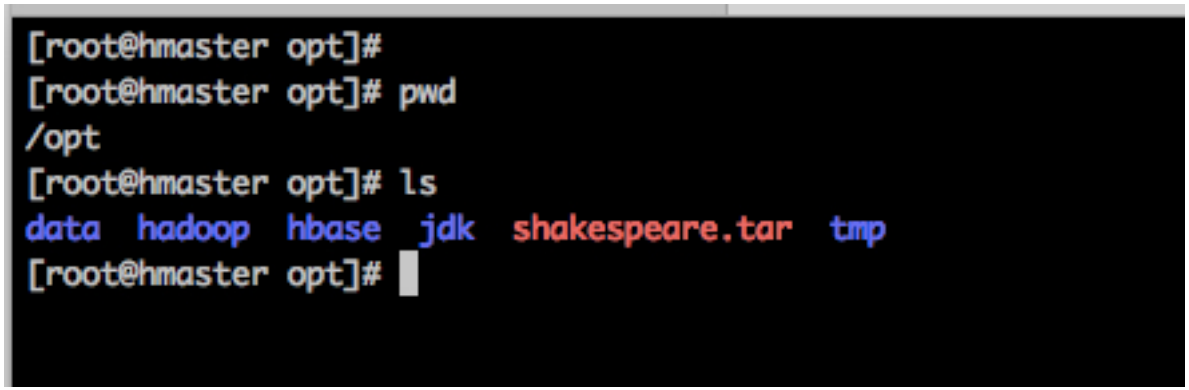
Note that the directory structure in HDFS has nothing to do with the directory structure of the local filesystem; they are completely separate namespaces.

Step 2: Uploading Files

Besides browsing the existing filesystem, another important thing you can do with FsShell is to upload new data into HDFS.

1. Change directories to the directory containing the sample data we will be using in the course. Or select any folder that contains some files.

$ cd /opt/



If you perform a 'regular' ls command in this directory, you will see a few files, including one named shakespeare.tar.gz.

Unzip shakespeare.tar.gz by running:

$ tar -xvf shakespeare.tar

This creates a directory named shakespeare/ containing several files on your local filesystem. If you have used this VM for a previous course, we recommend that you unzip the tar file again to ensure you have the file required for this lab.

Change to the shakespeare directory and view the files:

$ cd shakespeare
$ ls

You will see a few files, one of which is named glossary.

Insert this file into HDFS:
$ hdfs dfs -mkdir -p /user/training/glossary
$ hdfs dfs -put glossary /user/training/glossary

This copies the local glossary file into the remote HDFS directory named /user/training/

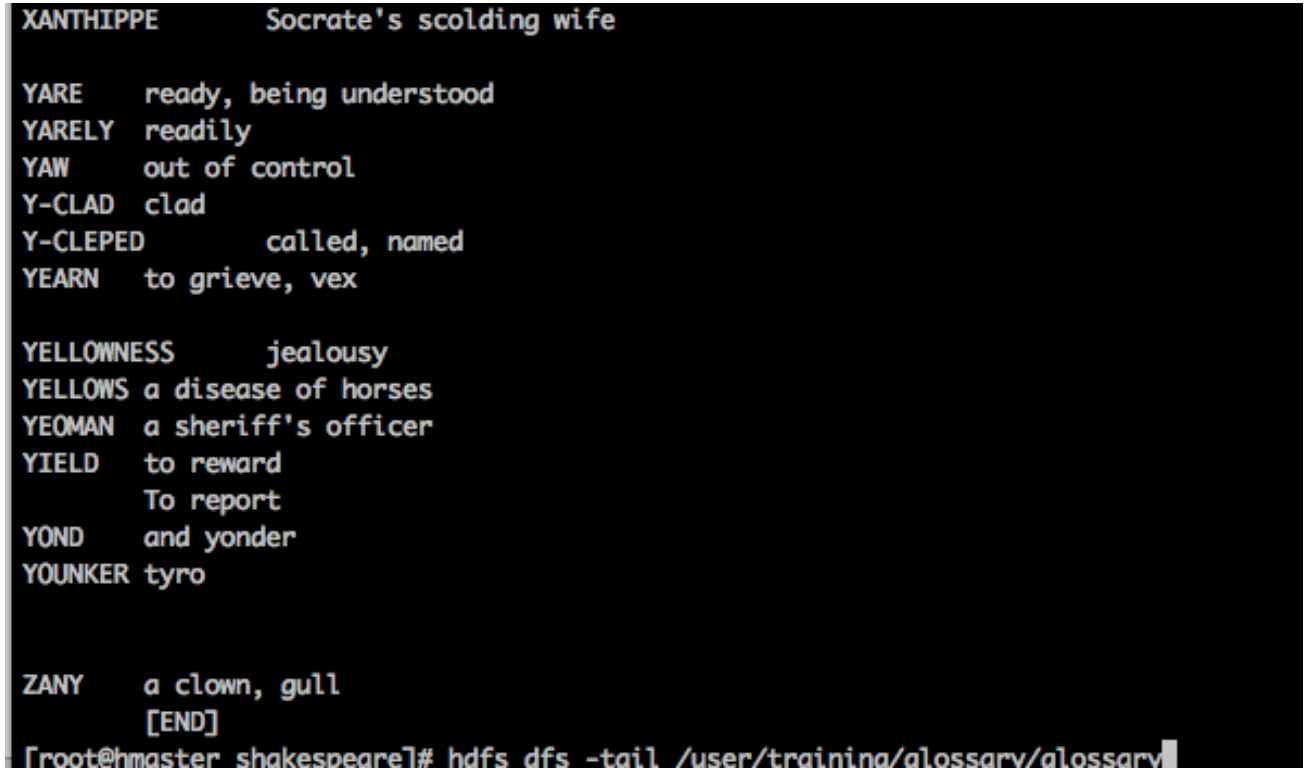
List the contents of your HDFS home directory now:

$ hdfs dfs -ls /user/training

You should see an entry for the glossary file.

Enter:

$ hdfs dfs -tail /user/training/glossary/glossary



This prints the last 1KB of the glossary to your terminal. This command is handy for viewing the output of MapReduce programs. Very often, an individual output file of a MapReduce program is very large, making it inconvenient to view the entire file in the terminal.

Other Commands
There are several other commands associated with the FsShell subsystem which let you perform most common filesystem manipulations: rm, rm -r (recursive rm), mv, cp, mkdir, etc.

1. In the terminal window, enter:

$ hdfs dfs

You see a help message describing all the commands associated with this subsystem.

Bonus Exercise: Advanced FsShell Commands
1. In the terminal window, enter:

$ hdfs dfs -help

This will give a more complete description of each FsShell command and the arguments for it.

2. Enter:

$ hdfs dfs -cat /user/training/glossary/glossary | tail -n 50

This prints the last 50 lines of the glossary to your terminal. This command is handy for viewing the output of MapReduce programs. Very often, an individual output file of a MapReduce program is very large, making it inconvenient to view the entire file in the terminal. For this reason, it's often a good idea to pipe the output of the fs -cat command into head, tail, more, or less.

---------- Lab Ends Here ----------

7. Pseudo Distributed Hbase Cluster – 60 Minutes

This lab depends on the Lab – Confugure Hadoop Cluster.

Each HBase daemon (HMaster, HRegionServer, and ZooKeeper) runs as a separate process.

By default, unless you configure the `hbase.rootdir` property , your data is still stored in */tmp/*.

Stop HBase if it is running

Configure HBase.

Edit the conf/*hbase-site.xml* configuration. First, add the following property which directs HBase to run in distributed mode, with one JVM instance per daemon.

```
<property>
  <name>hbase.cluster.distributed</name>
  <value>true</value>
</property>
<property>
  <name>hbase.rootdir</name>
  <value>hdfs://hmaster.master.com:9000/hbase</value>
</property>
```

Skip the following installation if Already done before else execute the following commands to install the ssh server.
#yum install openssh-server openssh-clients

#systemctl start sshd
OSTech

Start HBase.

Use the *bin/start-hbase.sh* command to start HBase. If your system is configured correctly, the `jps` command should show the HMaster and HRegionServer processes running.

```
[root@hmaster hbase]# bin/start-hbase.sh
127.0.0.1: running zookeeper, logging to /opt/hbase/bin/../logs/hbase-root-zookeeper-hmaster.master.com.out
running master, logging to /opt/hbase/bin/../logs/hbase--master-hmaster.master.com.out
hmaster.master.com: running regionserver, logging to /opt/hbase/bin/../logs/hbase-root-regionserver-hmaster.master
.com.out
[root@hmaster hbase]# jps
3876 DataNode
10293 HMaster
3753 NameNode
10538 Jps
10426 HRegionServer
10188 HQuorumPeer
4076 SecondaryNameNode
[root@hmaster hbase]#
```

Check the HBase directory

Open a terminal, change directory /opt/hadoop

```
# hdfs dfs -ls /hbase
```

```
[root@hmaster hadoop]# bin/hadoop fs -ls /hbase
Found 12 items
drwxr-xr-x   - root supergroup          0 2020-11-27 15:44 /hbase/.hbck
drwxr-xr-x   - root supergroup          0 2020-11-27 15:44 /hbase/.tmp
drwxr-xr-x   - root supergroup          0 2020-11-27 15:44 /hbase/MasterData
drwxr-xr-x   - root supergroup          0 2020-11-27 15:44 /hbase/WALs
drwxr-xr-x   - root supergroup          0 2020-11-27 15:44 /hbase/archive
drwxr-xr-x   - root supergroup          0 2020-11-27 15:44 /hbase/corrupt
drwxr-xr-x   - root supergroup          0 2020-11-27 15:44 /hbase/data
-rw-r--r--   3 root supergroup         42 2020-11-27 15:44 /hbase/hbase.id
-rw-r--r--   3 root supergroup          7 2020-11-27 15:44 /hbase/hbase.version
drwxr-xr-x   - root supergroup          0 2020-11-27 15:44 /hbase/mobdir
drwxr-xr-x   - root supergroup          0 2020-11-27 15:44 /hbase/oldWALs
drwx--x--x   - root supergroup          0 2020-11-27 15:44 /hbase/staging
[root@hmaster hadoop]#
```

Create a table and populate it with data.

```
#bin/hbase shell
```

create 'test', 'cf'
list 'test'
put 'test', 'row1', 'cf:a', 'value1'
put 'test', 'row2', 'cf:b', 'value2'
put 'test', 'row3', 'cf:c', 'value3'
scan 'test'

```
Version 2.3.3, r3e4bf4bee3a08b25591b9c22fea0518686a7e834, Wed Oct 28 06:36:25 UTC 2020
Took 0.0042 seconds
hbase(main):001:0> create 'test', 'cf'
Created table test
Took 2.7289 seconds
=> Hbase::Table - test
hbase(main):002:0> list 'test'
TABLE
test
1 row(s)
Took 0.0771 seconds
=> ["test"]
hbase(main):003:0> put 'test', 'row1', 'cf:a', 'value1'
Took 0.5316 seconds
hbase(main):004:0> put 'test', 'row2', 'cf:b', 'value2'
Took 0.0135 seconds
hbase(main):005:0> put 'test', 'row3', 'cf:c', 'value3'
Took 0.0138 seconds
hbase(main):006:0> scan 'test'
ROW                           COLUMN+CELL
 row1                         column=cf:a, timestamp=2020-11-27T16:00:27.341, value=value1
 row2                         column=cf:b, timestamp=2020-11-27T16:00:32.395, value=value2
 row3                         column=cf:c, timestamp=2020-11-27T16:00:37.782, value=value3
3 row(s)
Took 0.1571 seconds
hbase(main):007:0>
```

---------- Lab Ends Here ----------

OSTech

Procedure: Stop HBase

1. In the same way that the *bin/start-hbase.sh* script is provided to conveniently start all HBase daemons, the *bin/stop-hbase.sh*script stops them.

```
$ ./bin/stop-hbase.sh
stopping hbase...................
$
```

4. After issuing the command, it can take several minutes for the processes to shut down. Use the `jps` to be sure that the HMaster and HRegionServer processes are shut down.

The above has shown you how to start and stop a standalone instance of HBase. In the next sections we give a quick overview of other modes of hbase deploy.

https://hbase.apache.org/book.html#quickstart

8. Using the Developer JAVA API - 60 Minutes

In this exercise you will use the HBase Java API to programmatically perform CRUD operations using - Put, Get, and Scan methods.

Requirements:

- Mention the zookeeper details in hbase-site.xml.
- Start Hbase Pseudo cluster.
- Use eclipse to write code.

Use Case: Create a table and insert records in the table.

Start the eclipse and create a maven project with the following project details. Use simple maven project type.

```
<groupId>com.ostech</groupId>
<artifactId>LearningHbase</artifactId>
<version>0.0.1-SNAPSHOT</version>
```

Update the pom.xml with the following details. All jars dependencies are being mention in this file.

```
<project xmlns="http://maven.apache.org/POM/4.0.0"
    xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
    xsi:schemaLocation="http://maven.apache.org/POM/4.0.0
https://maven.apache.org/xsd/maven-4.0.0.xsd">
    <modelVersion>4.0.0</modelVersion>
    <groupId>com.ostech</groupId>
    <artifactId>LearningHbase</artifactId>
    <version>0.0.1-SNAPSHOT</version>
    <properties>
        <maven.compiler.source>1.8</maven.compiler.source>
        <maven.compiler.target>1.8</maven.compiler.target>
        <project.build.sourceEncoding>UTF-
8</project.build.sourceEncoding>
        <junit.version>4.8.1</junit.version>
        <hadoop.version>2.7.2</hadoop.version>
        <hbase.version>2.3.3</hbase.version>
    </properties>
    <dependencies>

        <!-- Hadoop -->
        <dependency>
            <groupId>org.apache.hadoop</groupId>
            <artifactId>hadoop-hdfs</artifactId>
            <version>${hadoop.version}</version>
```

```
            <exclusions>
                <exclusion>
                    <groupId>javax.servlet</groupId>
                    <artifactId>*</artifactId>
                </exclusion>
            </exclusions>
        </dependency>
        <dependency>
            <groupId>org.apache.hadoop</groupId>
            <artifactId>hadoop-client</artifactId>
            <version>${hadoop.version}</version>
            <exclusions>
                <exclusion>
                    <groupId>javax.servlet</groupId>
                    <artifactId>*</artifactId>
                </exclusion>
            </exclusions>
        </dependency>

        <!-- hbase version -->
        <dependency>
            <groupId>org.apache.hbase</groupId>
            <artifactId>hbase-client</artifactId>
            <version>${hbase.version}</version>
```

```
        </dependency>


    </dependencies>
</project>
```

Create a package with the following class in it.

```
package com.ostech.hbase;

import java.io.IOException;

import org.apache.hadoop.conf.Configuration;
import org.apache.hadoop.hbase.HBaseConfiguration;
import org.apache.hadoop.hbase.TableName;
import org.apache.hadoop.hbase.client.Admin;
import org.apache.hadoop.hbase.client.ColumnFamilyDescriptorBuilder;
import org.apache.hadoop.hbase.client.Connection;
import org.apache.hadoop.hbase.client.ConnectionFactory;
import org.apache.hadoop.hbase.client.Put;
import org.apache.hadoop.hbase.client.Table;
import org.apache.hadoop.hbase.client.TableDescriptor;
import org.apache.hadoop.hbase.client.TableDescriptorBuilder;
```

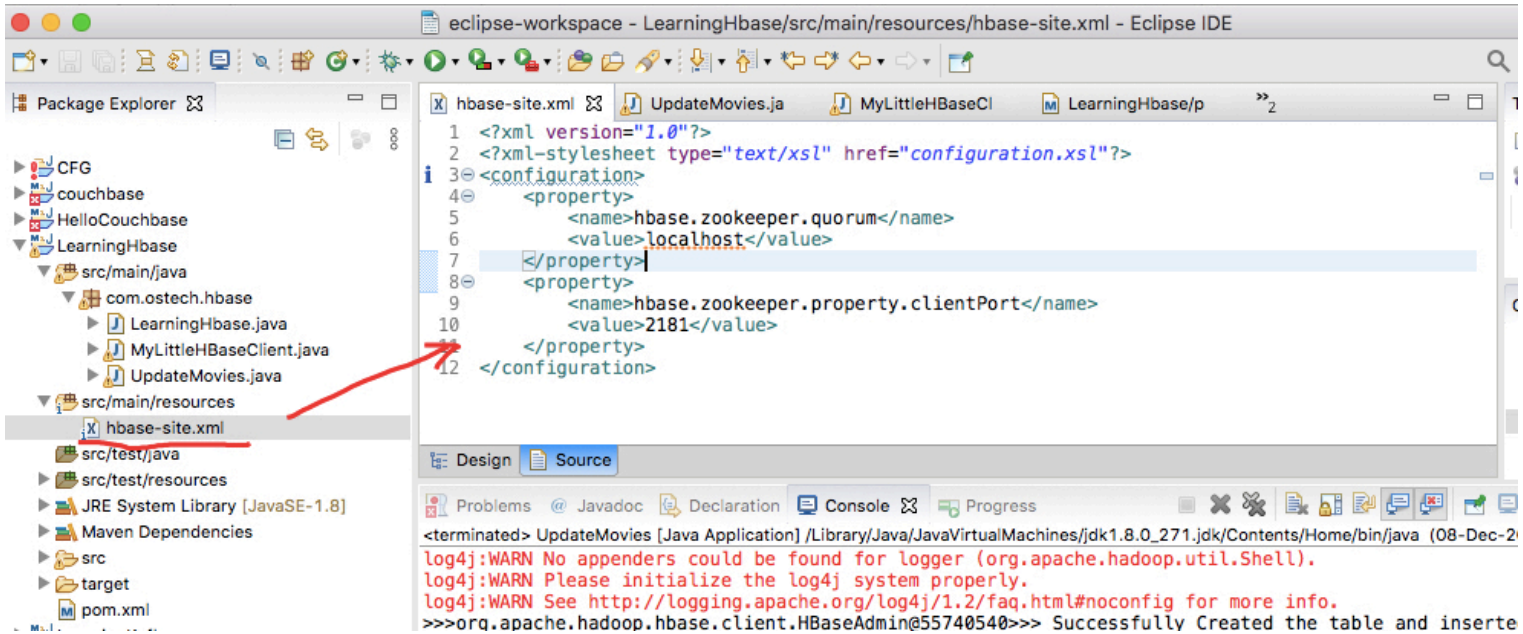
```
import org.apache.hadoop.hbase.util.Bytes;

public class UpdateMovies {

    public static void main(String[] args) throws IOException {
        // TODO Auto-generated method stub
        Configuration config = HBaseConfiguration.create();
        String path = UpdateMovies.class
                .getClassLoader()
                .getResource("hbase-site.xml")
                .getPath();

         Connection connection =
ConnectionFactory.createConnection(config);
        try {
            Admin admin = connection.getAdmin() ;

            System.out.println(">>>" + admin);
             createTable(admin);
             putRow(connection.getTable(TABLE_NAME));
             System.out.println(">>> Successfully Created the table and
inserted a row " + admin);

        }catch (Exception e) {
```

```
            e.printStackTrace();
        }

    }

        private static final TableName TABLE_NAME =
TableName.valueOf("mymovietbl");
        private static final byte[] CF_NAME = Bytes.toBytes("movie");
        private static final byte[] QUALIFIER = Bytes.toBytes("mname");
        private static final byte[] ROW_ID = Bytes.toBytes("row01");

        public static void createTable(final Admin admin) throws
IOException {
            if(!admin.tableExists(TABLE_NAME)) {
                TableDescriptor desc =
TableDescriptorBuilder.newBuilder(TABLE_NAME)

.setColumnFamily(ColumnFamilyDescriptorBuilder.of(CF_NAME))
                        .build();
                admin.createTable(desc);
            }
        }
```

```
        public static void putRow(final Table table) throws IOException
{
            table.put(new Put(ROW_ID).addColumn(CF_NAME, QUALIFIER,
Bytes.toBytes("Hello, World!")));
        }
}
```

In the above class, we have mention `hbase-site.xml` in the configuration to fetch the hbase cluster details for creating a table.

Copy the following hbase-site.xml content in the the following location:


eclipse-workspace - LearningHbase/src/main/resources/hbase-site.xml - Eclipse IDE
Package Explorer
CFG
couchbase
HelloCouchbase
LearningHbase
src/main/java
com.ostech.hbase
LearningHbase.java
MyLittleHBaseClient.java
UpdateMovies.java
src/main/resources
hbase-site.xml
src/test/java
src/test/resources
JRE System Library [JavaSE-1.8]
Maven Dependencies
src
target
pom.xml
hbase-site.xml
UpdateMovies.ja
MyLittleHBaseCl
LearningHbase/p
<?xml version="1.0"?>
<?xml-stylesheet type="text/xsl" href="configuration.xsl"?>
<configuration>
<property>
<name>hbase.zookeeper.quorum</name>
<value>localhost</value>
</property>
<property>
<name>hbase.zookeeper.property.clientPort</name>
<value>2181</value>
</property>
</configuration>
Design
Source
Problems
Javadoc
Declaration
Console
Progress
<terminated> UpdateMovies [Java Application] /Library/Java/JavaVirtualMachines/jdk1.8.0_271.jdk/Contents/Home/bin/java (08-Dec-2
log4j:WARN No appenders could be found for logger (org.apache.hadoop.util.Shell).
log4j:WARN Please initialize the log4j system properly.
log4j:WARN See http://logging.apache.org/log4j/1.2/faq.html#noconfig for more info.
>>>org.apache.hadoop.hbase.client.HBaseAdmin@55740540>>> Successfully Created the table and inserte

```
<?xml version="1.0"?>
<?xml-stylesheet type="text/xsl" href="configuration.xsl"?>
<configuration>
    <property>
        <name>hbase.zookeeper.quorum</name>
        <value>localhost</value>
    </property>
    <property>
        <name>hbase.zookeeper.property.clientPort</name>
        <value>2181</value>
    </property>
</configuration>
```

Execute the main program.

You can verify the result from the hbase shell. Start a hbase console.
#hbase shell
Determine the table in the shell – mymovietbl

#list
#scan 'mymovietbl'

```
hbase(main):007:0* list
TABLE
mymovietbl
test
2 row(s)
Took 0.0565 seconds
=> ["mymovietbl", "test"]
hbase(main):008:0> scan 'mymovietbl'
ROW                                   COLUMN+CELL
 row01                                column=movie:mname, timestamp=2020-12-08T08:11:19.876, value=Hello, World!
1 row(s)
Took 0.1123 seconds
hbase(main):009:0>
```

As shown above, the table; mymovietbl was created and a row with row01 was inserted.

In the next program, we will perform the followings:

- Insert 3 movies record using Put.
- Fetch a record using – Get (Row Key - nungshi)
- Fetch all records using scanner.

Create the following class.

```
package co.ostech.hbase;

import java.io.IOException;
import java.util.ArrayList;

import org.apache.hadoop.conf.Configuration;
import org.apache.hadoop.fs.Path;
import org.apache.hadoop.hbase.CompareOperator;
import org.apache.hadoop.hbase.HBaseConfiguration;
import org.apache.hadoop.hbase.TableName;
import org.apache.hadoop.hbase.client.Connection;
import org.apache.hadoop.hbase.client.ConnectionFactory;
import org.apache.hadoop.hbase.client.Get;
import org.apache.hadoop.hbase.client.Put;
import org.apache.hadoop.hbase.client.Result;
import org.apache.hadoop.hbase.client.ResultScanner;
import org.apache.hadoop.hbase.client.Scan;
import org.apache.hadoop.hbase.client.Table;
import org.apache.hadoop.hbase.filter.Filter;
import org.apache.hadoop.hbase.filter.FilterList;
import org.apache.hadoop.hbase.filter.SingleColumnValueFilter;
```

```
import org.apache.hadoop.hbase.util.Bytes;

public class MyLittleHBaseClient {
    private static String table_name = "mymovietbl";
    public static void main(String[] args) throws IOException {
        Configuration config = HBaseConfiguration.create();
        String path =
UpdateMovies.class.getClassLoader().getResource("hbase-
site.xml").getPath();
        config.addResource(new Path(path));
        Connection connection =
ConnectionFactory.createConnection(config);

        try {

            Table table =
connection.getTable(TableName.valueOf(table_name));
            UpdateInfo(table);
            //findMoviesByLanguage(table);
            //findMoviesByTwoLanguage(table);
        } finally {
            connection.close();
        }
    }
```

```
    public static void UpdateInfo(Table table) throws IOException {

        try {

            Put p = new Put(Bytes.toBytes("braveheart"));

    p.addColumn(Bytes.toBytes("movie"),Bytes.toBytes("mname"),Bytes.toBy
tes("Braveheart"));

    p.addColumn(Bytes.toBytes("movie"),Bytes.toBytes("language"),Bytes.t
oBytes("English"));

            table.put(p);

            Put p1 = new Put(Bytes.toBytes("kasam"));

    p1.addColumn(Bytes.toBytes("movie"),Bytes.toBytes("mname"),Bytes.toB
ytes("Kasam"));

    p1.addColumn(Bytes.toBytes("movie"),Bytes.toBytes("language"),Bytes.
toBytes("Hindi"));

            table.put(p1);
```

```
            Put p2 = new Put(Bytes.toBytes("nungshi"));

    p2.addColumn(Bytes.toBytes("movie"),Bytes.toBytes("mname"),Bytes.toB
ytes("Nungshi"));

    p2.addColumn(Bytes.toBytes("movie"),Bytes.toBytes("language"),Bytes.
toBytes("Manipuri"));

            table.put(p2);


            Get g = new Get(Bytes.toBytes("nungshi"));
            Result r = table.get(g);
            byte[] value = r.getValue(Bytes.toBytes("movie"),
Bytes.toBytes("mname"));

            // If we convert the value bytes, we should get back 'Some
Value', the
            // value we inserted at this location.
            String valueStr = Bytes.toString(value);
            System.out.println("Get Information for Row Key – nungshi " +
valueStr);
```

```
            // Sometimes, you won't know the row you're looking for. In
this case, you
            // use a Scanner. This will give you cursor-like interface to
the contents
            // of the table. To set up a Scanner, do like you did above
making a Put
            // and a Get, create a Scan. Adorn it with column names, etc.
            Scan s = new Scan();
            s.addColumn(Bytes.toBytes("movie"), Bytes.toBytes("mname"));
            ResultScanner scanner = table.getScanner(s);
            try {
                // Scanners return Result instances.
                // Now, for the actual iteration. One way is to use a
while loop like so:
                for (Result rr = scanner.next(); rr != null; rr =
scanner.next()) {
                    // print out the row we found and the columns we were
looking for
                    System.out.println("Found row: " + rr);
                }

                // The other approach is to use a foreach loop. Scanners
are iterable!
                // for (Result rr : scanner) {
```

```
                // System.out.println("Found row: " + rr);
                // }
            } finally {
                // Make sure you close your scanners when you are done!
                // Thats why we have it inside a try/finally clause
                scanner.close();
            }

            // Close your table and cluster connection.
        } finally {
            if (table != null)
                table.close();
        }
    }

    public static void findMoviesByLanguage(Table table) throws
IOException {

        byte[] FAMILY_BYTES =  Bytes.toBytes("movie");  // Column Family
Name
        byte[] COLUMN_BYTES =  Bytes.toBytes("language"); // Column
        SingleColumnValueFilter f1 = new
SingleColumnValueFilter(FAMILY_BYTES,COLUMN_BYTES,
CompareOperator.EQUAL, Bytes.toBytes("Manipuri"));
```

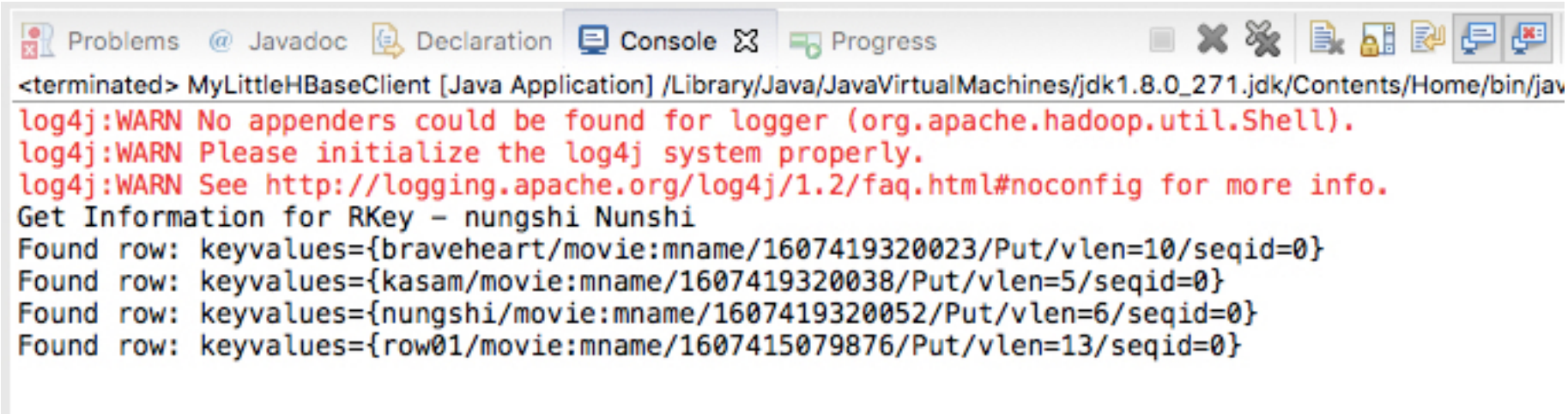
```
        f1.setFilterIfMissing(true);

        ArrayList<Filter> filters = new ArrayList<Filter>();
        filters.add(f1);

        FilterList filterList = new
FilterList(FilterList.Operator.MUST_PASS_ONE, filters);

        Scan s = new Scan();
        s.setFilter(filterList);
        s.addColumn(Bytes.toBytes("movie"), Bytes.toBytes("mname"));
        s.addColumn(Bytes.toBytes("movie"), Bytes.toBytes("language"));

        ResultScanner scanner = table.getScanner(s);
        for (Result rr = scanner.next(); rr != null; rr =
scanner.next()) {
            // print out the row we found and the columns we were looking
for
            System.out.println("Language: " +
              Bytes.toString(rr.getValue(Bytes.toBytes("movie"),
Bytes.toBytes("language")))
              + "\t" +
Bytes.toString(rr.getValue(Bytes.toBytes("movie"),
Bytes.toBytes("mname"))) );
```

```
        }


    }

public static void findMoviesByTwoLanguage(Table table) throws
IOException {

        byte[] FAMILY_BYTES =  Bytes.toBytes("movie");
        byte[] COLUMN_BYTES =  Bytes.toBytes("language");
        SingleColumnValueFilter f1 = new
SingleColumnValueFilter(FAMILY_BYTES,COLUMN_BYTES,
CompareOperator.EQUAL, Bytes.toBytes("Manipuri"));
        f1.setFilterIfMissing(true);

        SingleColumnValueFilter f2 = new
SingleColumnValueFilter(FAMILY_BYTES,COLUMN_BYTES,
CompareOperator.EQUAL, Bytes.toBytes("English"));
        f2.setFilterIfMissing(true);

        ArrayList<Filter> filters = new ArrayList<Filter>();
        filters.add(f1);
        filters.add(f2);
```

```
        FilterList filterList = new
FilterList(FilterList.Operator.MUST_PASS_ONE, filters);

        Scan s = new Scan();
        s.setFilter(filterList);
        s.addColumn(Bytes.toBytes("movie"), Bytes.toBytes("mname"));
        s.addColumn(Bytes.toBytes("movie"), Bytes.toBytes("language"));

        ResultScanner scanner = table.getScanner(s);
        for (Result rr = scanner.next(); rr != null; rr =
scanner.next()) {
            // print out the row we found and the columns we were looking
for
            System.out.println("Language: " +
              Bytes.toString(rr.getValue(Bytes.toBytes("movie"),
Bytes.toBytes("language")))
                + "\t" +
Bytes.toString(rr.getValue(Bytes.toBytes("movie"),
Bytes.toBytes("mname"))) );

        }
```

```
    }
}
```

Execute the program:



As shown above, the first row is the result of the record being fetch using key – nungshi.
The last four rows are fetch using the scanner.

Case 1:

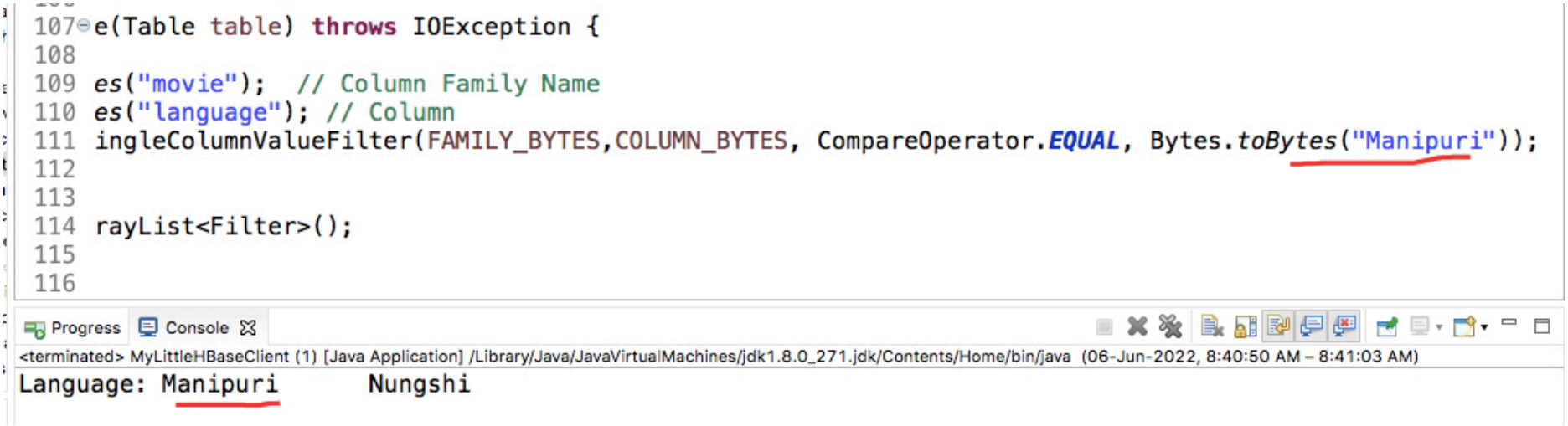
Comment and uncomment as shown below in the main method.
It will apply a single filter to fetch the record with the language of type ‘Manipuri’

MyLittleHBaseClient.java

```
29          config.addResource(new Path(path));
30          Connection connection = ConnectionFactory.createConnection(config);
31
32          try {
33
34              Table table = connection.getTable(TableName.valueOf(table_name));
35              //UpdateInfo(table);
36              findMoviesByLanguage(table);
37              //findMoviesByTwoLanguage(table);
38          } finally {
39              connection.close();
40          }
41      }
42
43      public static void UpdateInfo(Table table) throws IOException {
```

Execute the program.

As shown below. Its filter by the Manipuri.



Next apply Two filters i.e English and Manipuri Language.

Comment as shown below:

MyLittleHBaseClient.java

```
31
32          try {
33
34              Table table = connection.getTable(TableName.valueOf(table_name));
35              //UpdateInfo(table);
36              // findMoviesByLanguage(table);
37              findMoviesByTwoLanguage(table);
38          } finally {
39              connection.close();
40          }
41      }
42
43      public static void UpdateInfo(Table table) throws IOException {
44
45          try {
```

Execute the program.

```
140 ingleColumnValueFilter(FAMILY_BYTES,COLUMN_BYTES, CompareOperator.EQUAL, Bytes.toBytes("Manipuri"));
141
142
143 ingleColumnValueFilter(FAMILY_BYTES,COLUMN_BYTES, CompareOperator.EQUAL, Bytes.toBytes("English"));
144
145
146 rayList<Filter>();
147
148
```

Task:

Try adding a new record and execute scanner to fetch that record.

---------- Lab Ends Here ----------

9. HBase Filters – 30 Minutes

In this exercise you will use the HBase Java API to programmatically scan rows in a hbase table and apply Filter on it.

With HBase scan and filters, it verify every row in the mymovietbl table and find those rows that match the following criteria:

1. Language is Manipuri
2. Language is Manipuri or English

It displays the movie name, and language for those rows that match the above criterias.

Use the Java Project created in the Lab- Using the Developer API

1) Add the following method in the class file, `MyLittleHBaseClient.java` and invoke from the main method. Comment all others method invocation from the main method. – Filter by One Column Value Only.

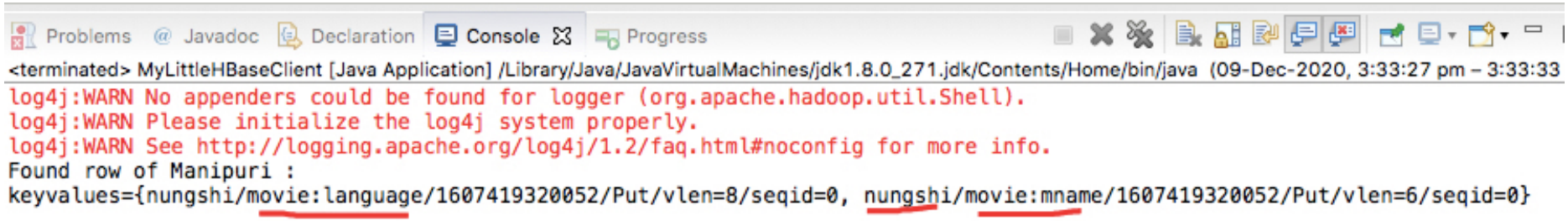
```
public static void findMoviesByLanguage(Table table) throws IOException
{

        byte[] FAMILY_BYTES =  Bytes.toBytes("movie");  // Column Family
Name
        byte[] COLUMN_BYTES =  Bytes.toBytes("language"); // Column
```

```
        SingleColumnValueFilter f1 = new
SingleColumnValueFilter(FAMILY_BYTES,COLUMN_BYTES,
CompareOperator.EQUAL, Bytes.toBytes("Manipuri"));
        f1.setFilterIfMissing(true);

        ArrayList<Filter> filters = new ArrayList<Filter>();
        filters.add(f1);

        FilterList filterList = new
FilterList(FilterList.Operator.MUST_PASS_ONE, filters);

        Scan s = new Scan();
        s.setFilter(filterList);
        s.addColumn(Bytes.toBytes("movie"), Bytes.toBytes("mname"));
        s.addColumn(Bytes.toBytes("movie"), Bytes.toBytes("language"));

        ResultScanner scanner = table.getScanner(s);
        for (Result rr = scanner.next(); rr != null; rr =
scanner.next()) {
            // print out the row we found and the columns we were looking
for
            System.out.println("Language: " +
              Bytes.toString(rr.getValue(Bytes.toBytes("movie"),
Bytes.toBytes("language")))
```

```
                + "\t" +
Bytes.toString(rr.getValue(Bytes.toBytes("movie"),
Bytes.toBytes("mname"))) );

        }


    }
```

```
23
24  public class MyLittleHBaseClient {
25      private static String table_name = "mymovietbl";
26      public static void main(String[] args) throws IOException {
27          Configuration config = HBaseConfiguration.create();
28          String path = UpdateMovies.class.getClassLoader().getResource("hbase-site.xml").getPath();
29          config.addResource(new Path(path));
30          Connection connection = ConnectionFactory.createConnection(config);
31
32          try {
33
34              Table table = connection.getTable(TableName.valueOf(table_name));
35              //UpdateInfo(table);
36               findMoviesByLanguage(table);
37               //findMoviesByTwoLanguage(table);
38          } finally {
39              connection.close();
40          }
41      }
42
```

Execute the class.



You will get only the Manipuri movie as shown above.

Now let us filter the English and Manipuri movies i.e applying multiples filters and will select specific columns only.

Add the following method.

```
public static void findMoviesByTwoLanguage(Table table) throws
IOException {

        byte[] FAMILY_BYTES =  Bytes.toBytes("movie");
        byte[] COLUMN_BYTES =  Bytes.toBytes("language");
       SingleColumnValueFilter f1 = new
SingleColumnValueFilter(FAMILY_BYTES,COLUMN_BYTES,
CompareOperator.EQUAL, Bytes.toBytes("Manipuri"));
```

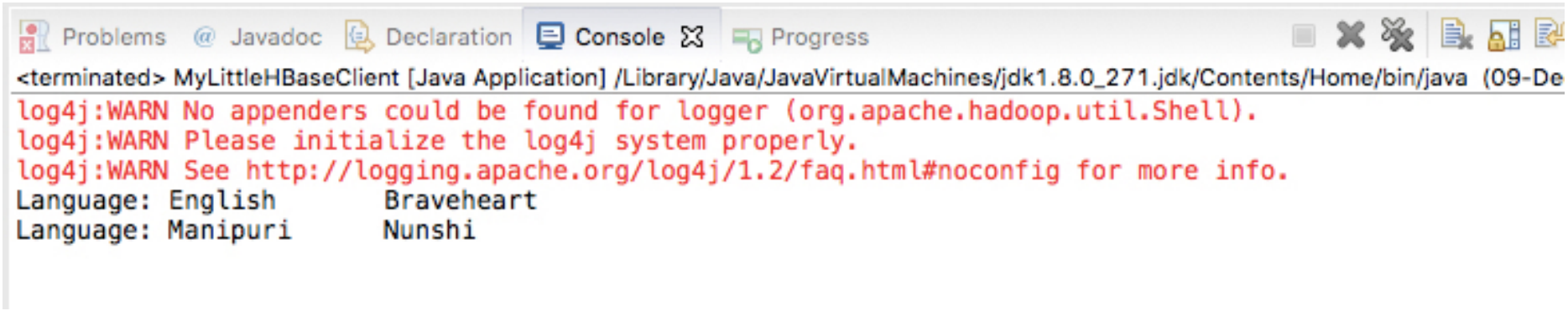
```
        f1.setFilterIfMissing(true);

        SingleColumnValueFilter f2 = new
SingleColumnValueFilter(FAMILY_BYTES,COLUMN_BYTES,
CompareOperator.EQUAL, Bytes.toBytes("English"));
        f2.setFilterIfMissing(true);

        ArrayList<Filter> filters = new ArrayList<Filter>();
        filters.add(f1);
        filters.add(f2);

        FilterList filterList = new
FilterList(FilterList.Operator.MUST_PASS_ONE, filters);

        Scan s = new Scan();
        s.setFilter(filterList);
        s.addColumn(Bytes.toBytes("movie"), Bytes.toBytes("mname"));
        s.addColumn(Bytes.toBytes("movie"), Bytes.toBytes("language"));

        ResultScanner scanner = table.getScanner(s);
        for (Result rr = scanner.next(); rr != null; rr =
scanner.next()) {
            // print out the row we found and the columns we were looking
for
```

```
            System.out.println("Language: " +
              Bytes.toString(rr.getValue(Bytes.toBytes("movie"),
Bytes.toBytes("language")))
              + "\t" +
Bytes.toString(rr.getValue(Bytes.toBytes("movie"),
Bytes.toBytes("mname"))) );

        }


    }
```

Now invoke the above method by commenting earlier methods.

```
public class MyLittleHBaseClient {
    private static String table_name = "mymovietbl";
    public static void main(String[] args) throws IOException {
        Configuration config = HBaseConfiguration.create();
        String path = UpdateMovies.class.getClassLoader().getResource("hbase-site.xml").getPath();
        config.addResource(new Path(path));
        Connection connection = ConnectionFactory.createConnection(config);

        try {

            Table table = connection.getTable(TableName.valueOf(table_name));
             //UpdateInfo(table);
             //findMoviesByLanguage(table);
              findMoviesByTwoLanguage(table);
        } finally {
            connection.close();
        }
    }

```

Execute the main class:



As shown above, two filters were applied and only the movies of English and Manipuri were displayed.

---------- Lab Ends here ----------

10. Built-in TSV Bulk Loader – 60 Minutes

ImportTsv is a utility that will load data in TSV format into HBase. It has two distinct usages:

- loading data from TSV format in HDFS into HBase via Puts,
- and preparing StoreFiles to be loaded via the `completebulkload`.

In this lab, we are going to perform the second option.

Start HDFS cluster.
Start hbase cluster

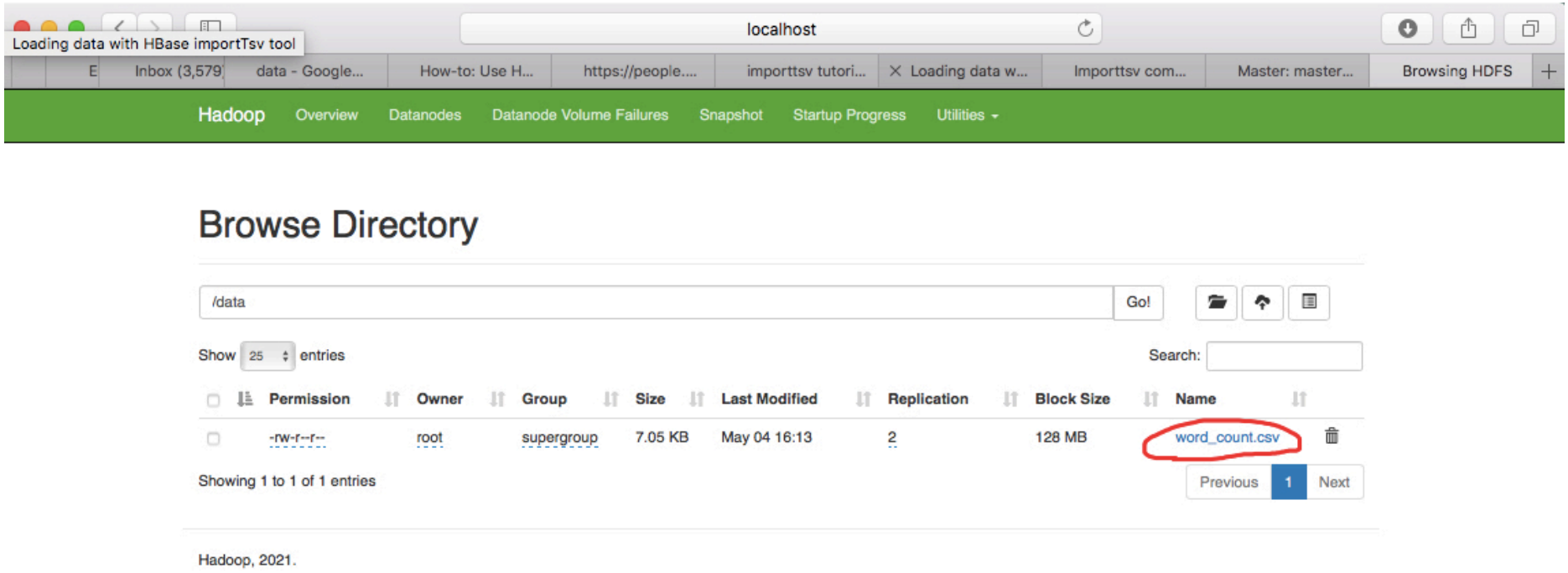
HBase ships with a MR job that can read a delimiter-separated values file and output directly into an HBase table or create HFiles for bulk loading. Here we are going to:

1. Get the sample data and upload it to HDFS. - word_count.csv
2. Run the ImportTsv job to transform the file into multiple HFiles according to a pre-configured table.
3. Prepare and load the files in HBase.

Now, upload the file to HDFS:
Create /data directory to upload the word_count.csv file to it.
#hdfs dfs -mkdir /data

```
#hdfs dfs -put word_count.csv /data
```

```
You can verify the file using the HDFS browser:
Access the HDSF web UI :http://localhost:9870/
Utilities -> Browse the file system -> /data
```



First you need to design the table. To keep things simple, call it “wordcount” — the row keys will be the words themselves and the only column will contain the count, in a family that we’ll call “count”.

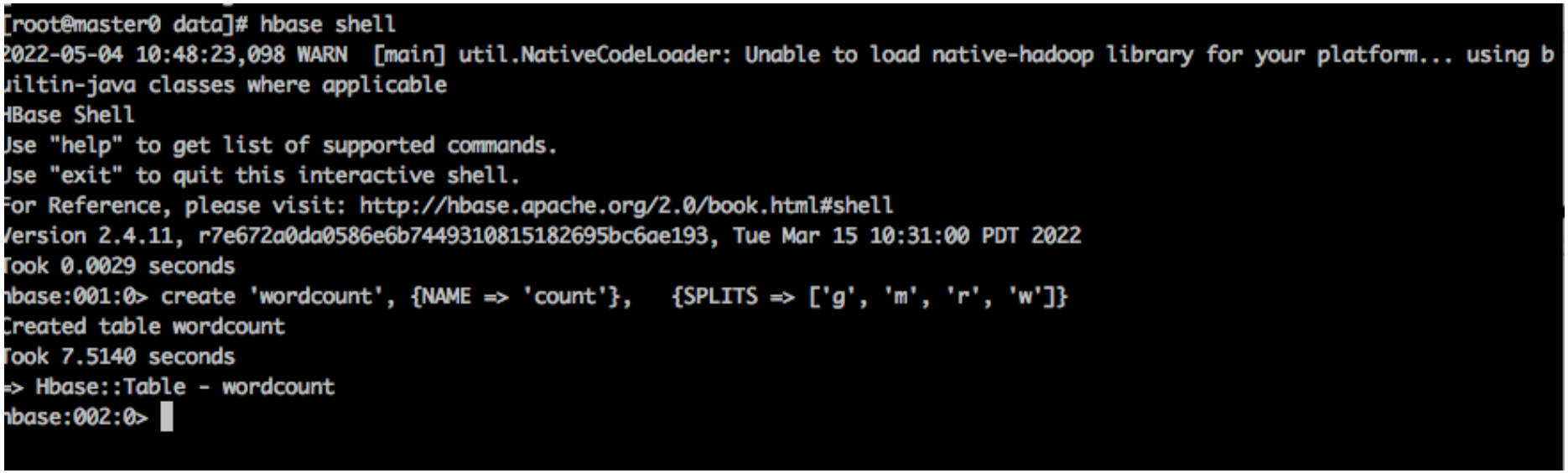
OSTech

The best practice when creating a table is to split it according to the row key distribution but for this example we'll just create five regions with split points spread evenly across the key space. Open the hbase shell:

hbase shell

And run the following command to create the table:

#create 'wordcount', {NAME => 'count'}, {SPLITS => ['g', 'm', 'r', 'w']}



Verify the table:

http://localhost:16010/tablesDetailed.jsp

Click on Table Details → Wordcount

word
coun
t

'wordcount', {NAME => 'count', BLOOMFILTER => 'ROW', IN_MEMORY => 'false', VERSIONS => '1', KEEP_DELETED_CELLS => 'FALSE', DATA_BLOCK_ENCODING => 'NONE', COMPRESSION => 'NONE', TTL => 'FOREVER', MIN_VERSIONS => '0', BLOCKCACHE => 'true', BLOCKSIZE => '65536', REPLICATION_SCOPE => '0'}

Determine the no of Regions: 5

Table Regions

Base Stats | Localities | Compactions

Name(5)	Region Server	ReadRequests (0)	WriteRequests (0)	StorefileSize (0 MB)	Num.Storefiles (0)	MemSize (0 MB)	Start Key	End Key	Region State
wordcount,,1651661388861.66b297fc6f59b59262b694088fa15913.	master0.hp.com:16030	0	0	0 MB	0	0 MB		g	OPEN
wordcount,g,1651661388861.186db9219698cb4756d6a2371e9b7f37.	master0.hp.com:16030	0	0	0 MB	0	0 MB	g	m	OPEN
wordcount,m,1651661388861.12c9c699e640d152ac628367e5e428b9.	master0.hp.com:16030	0	0	0 MB	0	0 MB	m	r	OPEN
wordcount,r,1651661388861.37c4b40731617b8dd8005e4a805c6407.	master0.hp.com:16030	0	0	0 MB	0	0 MB	r	w	OPEN
wordcount,w,1651661388861.d1bade65ab2f1ba4e788f087d0a1b0ab.	master0.hp.com:16030	0	0	0 MB	0	0 MB	w		OPEN

Invoking the HBase jar on the command line with the “hadoop” script will show a list of available tools.

hbase org.apache.hadoop.hbase.mapreduce.ImportTsv

The one we want is called importtsv and has the following usage:
(format : hbase – Import Util – Columns - table – Importing file)
Open a New terminal and execute the following to generate StoreFiles for bulk-loading:

hbase org.apache.hadoop.hbase.mapreduce.ImportTsv -Dimporttsv.separator=, -Dimporttsv.bulk.output=hdfs://master0.hp.com:9000/data/hfile_temp -Dimporttsv.columns="HBASE_ROW_KEY,count:thevalue" wordcount hdfs://master0.hp.com:9000/data/word_count.csv

```
                Reduce output records=722
                Spilled Records=1444
                Shuffled Maps =1
                Failed Shuffles=0
                Merged Map outputs=1
                GC time elapsed (ms)=115
                CPU time spent (ms)=3920
                Physical memory (bytes) snapshot=635961344
                Virtual memory (bytes) snapshot=6834982912
                Total committed heap usage (bytes)=442687488
        ImportTsv
                Bad Lines=0
        Shuffle Errors
                BAD_ID=0
                CONNECTION=0
                IO_ERROR=0
                WRONG_LENGTH=0
                WRONG_MAP=0
                WRONG_REDUCE=0
        File Input Format Counters
                Bytes Read=7217
        File Output Format Counters
                Bytes Written=36864
[root@master0 data]#  hbase org.apache.hadoop.hbase.mapreduce.ImportTsv -Dimporttsv.separator=, -Dimporttsv.bulk.outp
ut=/data/hfile_temp -Dimporttsv.columns="HBASE_ROW_KEY, count:thevalue" wordcount /data/word_count.csv
```

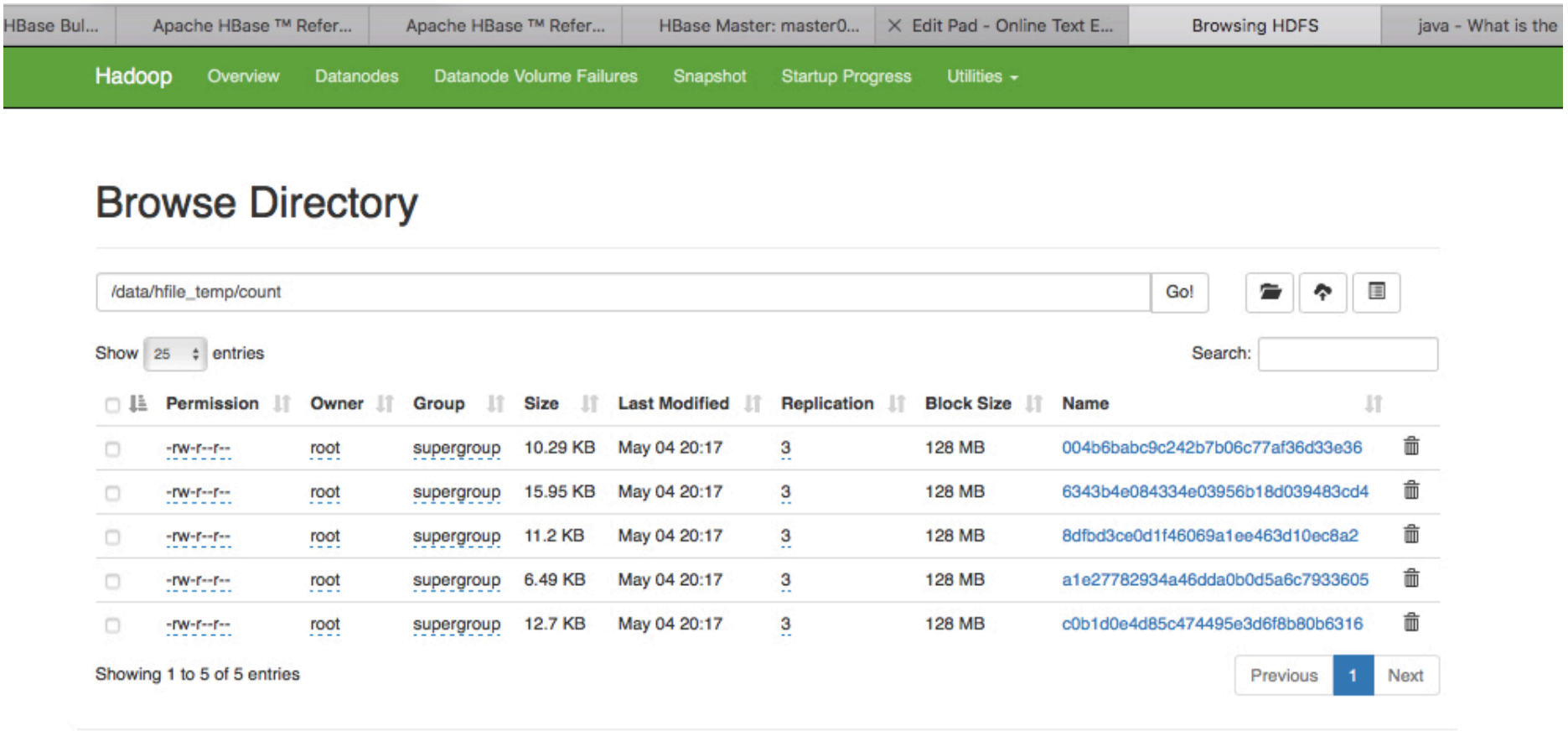
Here’s a rundown of the different configuration elements:

- **-Dimporttsv.separator=,** specifies that the separator is a comma.

- **-Dimporttsv.bulk.output=output** is a relative path to where the HFiles will be written. the files will be in /data/. Skipping this option will make the job write directly to HBase.
- **-Dimporttsv.columns=HBASE_ROW_KEY,count** is a list of all the columns contained in this file. The row key needs to be identified using the all-caps HBASE_ROW_KEY string; otherwise it won't start the job. (I decided to use the qualifier "count" but it could be anything else.)

  The job should complete within a minute, given the small input size. Note that five Reducers are running, one per region. Here's the result on HDFS:

  Verify the files using the HDFS Browser:



Using hdfs command.
#hdfs dfs -ls -R /data/hfile_temp

```
drwxr-xr-x   - root supergroup          0 2022-05-04 14:47 /data/hfile_temp/count
[root@master0 hadoop]# hdfs dfs -ls -R /data/hfile_temp
2022-05-04 14:56:39,225 INFO Configuration.deprecation: No unit for dfs.client.datanode-restart.timeout(30) assuming SECONDS
-rw-r--r--   3 root supergroup          0 2022-05-04 14:47 /data/hfile_temp/_SUCCESS
drwxr-xr-x   - root supergroup          0 2022-05-04 14:47 /data/hfile_temp/count
-rw-r--r--   3 root supergroup      10539 2022-05-04 14:47 /data/hfile_temp/count/004b6babc9c242b7b06c77af36d33e36
-rw-r--r--   3 root supergroup      16336 2022-05-04 14:47 /data/hfile_temp/count/6343b4e084334e03956b18d039483cd4
-rw-r--r--   3 root supergroup      11465 2022-05-04 14:47 /data/hfile_temp/count/8dfbd3ce0d1f46069a1ee463d10ec8a2
-rw-r--r--   3 root supergroup       6646 2022-05-04 14:47 /data/hfile_temp/count/a1e27782934a46dda0b0d5a6c7933605
-rw-r--r--   3 root supergroup      13009 2022-05-04 14:47 /data/hfile_temp/count/c0b1d0e4d85c474495e3d6f8b80b6316
[root@master0 hadoop]#
```

For the final step, we need to use the completebulkload tool to point to where the files are and which tables we are loading to:

```
#hbase org.apache.hadoop.hbase.mapreduce.LoadIncrementalHFiles
hdfs://master0.hp.com:9000/data/hfile_temp wordcount
```

- Argument : First – the Output folder specified in the importTSV output parameter.
- Second : Name of the Table.

```
2022-05-04 15:00:20,824 INFO  [ReadOnlyZKClient-127.0.0.1:2181@0x5be6e01c-SendThread(127.0.0.1:2181)] zookeeper.ClientCnxn: Opening socket con
nection to server localhost/127.0.0.1:2181. Will not attempt to authenticate using SASL (unknown error)
2022-05-04 15:00:20,859 INFO  [ReadOnlyZKClient-127.0.0.1:2181@0x5be6e01c-SendThread(127.0.0.1:2181)] zookeeper.ClientCnxn: Socket connection
established, initiating session, client: /127.0.0.1:59950, server: localhost/127.0.0.1:2181
2022-05-04 15:00:20,912 INFO  [ReadOnlyZKClient-127.0.0.1:2181@0x5be6e01c-SendThread(127.0.0.1:2181)] zookeeper.ClientCnxn: Session establishm
ent complete on server localhost/127.0.0.1:2181, sessionid = 0x100000e926e004c, negotiated timeout = 90000
2022-05-04 15:00:36,880 WARN  [main] tool.LoadIncrementalHFiles: Skipping non-directory hdfs://master0.hp.com:9000/data/hfile_temp/_SUCCESS
2022-05-04 15:00:38,516 INFO  [main] metrics.MetricRegistries: Loaded MetricRegistries class org.apache.hadoop.hbase.metrics.impl.MetricRegist
riesImpl
2022-05-04 15:00:41,203 INFO  [LoadIncrementalHFiles-0] tool.LoadIncrementalHFiles: Trying to load hfile=hdfs://master0.hp.com:9000/data/hfile
_temp/count/004b6babc9c242b7b06c77af36d33e36 first=Optional[m] last=Optional[quickstart]
2022-05-04 15:00:41,215 INFO  [LoadIncrementalHFiles-1] tool.LoadIncrementalHFiles: Trying to load hfile=hdfs://master0.hp.com:9000/data/hfile
_temp/count/6343b4e084334e03956b18d039483cd4 first=Optional[a] last=Optional[from]
2022-05-04 15:00:41,304 INFO  [LoadIncrementalHFiles-1] tool.LoadIncrementalHFiles: Trying to load hfile=hdfs://master0.hp.com:9000/data/hfile
_temp/count/8dfbd3ce0d1f46069a1ee463d10ec8a2 first=Optional[g] last=Optional[lot]
2022-05-04 15:00:41,367 INFO  [LoadIncrementalHFiles-1] tool.LoadIncrementalHFiles: Trying to load hfile=hdfs://master0.hp.com:9000/data/hfile
_temp/count/c0b1d0e4d85c474495e3d6f8b80b6316 first=Optional[r] last=Optional[vms]
2022-05-04 15:00:41,375 INFO  [LoadIncrementalHFiles-0] tool.LoadIncrementalHFiles: Trying to load hfile=hdfs://master0.hp.com:9000/data/hfile
_temp/count/a1e27782934a46dda0b0d5a6c7933605 first=Optional[w] last=Optional[yourself]
2022-05-04 15:00:42,504 INFO  [main] client.ConnectionImplementation: Closing master protocol: MasterService
2022-05-04 15:00:42,628 INFO  [ReadOnlyZKClient-127.0.0.1:2181@0x5be6e01c-EventThread] zookeeper.ClientCnxn: EventThread shut down for session
: 0x100000e926e004c
2022-05-04 15:00:42,628 INFO  [ReadOnlyZKClient-127.0.0.1:2181@0x5be6e01c] zookeeper.ZooKeeper: Session: 0x100000e926e004c closed
[root@master0 data]# hbase org.apache.hadoop.hbase.mapreduce.LoadIncrementalHFiles hdfs://master0.hp.com:9000/data/hfile_temp wordcount
```

Going back into the HBase shell, you can run the count command that will show you how many rows were loaded. If you forgot to chown, the command will hang.

```
# scan 'wordcount'
```

```
 words                          column=count:thevalue, timestamp=2022-05-04T13:58:35.806, value=3
 work                           column=count:thevalue, timestamp=2022-05-04T13:58:35.806, value=1
 workload                       column=count:thevalue, timestamp=2022-05-04T13:58:35.806, value=2
 works                          column=count:thevalue, timestamp=2022-05-04T13:58:35.806, value=2
 write                          column=count:thevalue, timestamp=2022-05-04T13:58:35.806, value=11
 writeheavy                     column=count:thevalue, timestamp=2022-05-04T13:58:35.806, value=1
 writing                        column=count:thevalue, timestamp=2022-05-04T13:58:35.806, value=1
 written                        column=count:thevalue, timestamp=2022-05-04T13:58:35.806, value=3
 wrong                          column=count:thevalue, timestamp=2022-05-04T13:58:35.806, value=1
 wrote                          column=count:thevalue, timestamp=2022-05-04T13:58:35.806, value=1
 you                            column=count:thevalue, timestamp=2022-05-04T13:58:35.806, value=43
 youll                          column=count:thevalue, timestamp=2022-05-04T13:58:35.806, value=2
 your                           column=count:thevalue, timestamp=2022-05-04T13:58:35.806, value=24
 youre                          column=count:thevalue, timestamp=2022-05-04T13:58:35.806, value=2
 yourself                       column=count:thevalue, timestamp=2022-05-04T13:58:35.806, value=1
722 row(s)
Took 3.3041 seconds
hbase:008:0> scan 'wordcount'
```

You can verify the table regions details from the web UI:

http://localhost:16010/master-status

Home -> Table (User Tables) → Wordcount → Table regions:

Table Regions

Base Stats | Localities | Compactions

Name(5)	Region Server	ReadRequests (722)	WriteRequests (5)	StorefileSize (5 MB)	Num.Storefiles (5)	MemSize (0 MB)	Start Key	End Key	Region State
wordcount,,1651672640769.0b49ac33cea190195653c14bbf52bcb2.	master0.hp.com:16030	246	1	1 MB	1	0 MB		g	OPEN
wordcount,g,1651672640769.e502c2ac2a17e8f917f2ab4ebbf18533.	master0.hp.com:16030	137	1	1 MB	1	0 MB	g	m	OPEN
wordcount,m,1651672640769.07977d3dee2bc8bf9e11510211ec5dc4.	master0.hp.com:16030	121	1	1 MB	1	0 MB	m	r	OPEN
wordcount,r,1651672640769.9bb89b702a505d94d460baf6264fec5e.	master0.hp.com:16030	180	1	1 MB	1	0 MB	r	w	OPEN
wordcount,w,1651672640769.ce6fad66a0aa9ac9718233fb93c7785c.	master0.hp.com:16030	38	1	1 MB	1	0 MB	w		OPEN

As you can view from above, all the Memsize values are Zero. This means the records are not gone through the normal process.

Insert a record using CLI and verify the record:

```
# put 'wordcount' , 'henry' , 'count:thevalue' , 15
# scan 'wordcount'
```

After that verify the Table Regions:

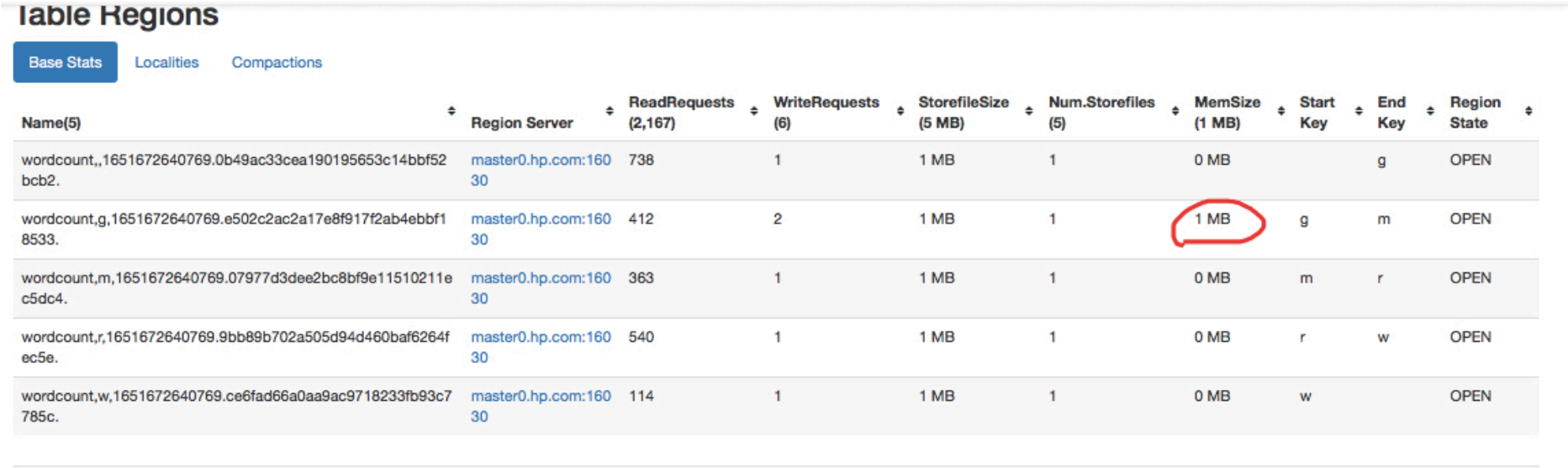


Table Regions

Base Stats | Localities | Compactions

Name(5)	Region Server	ReadRequests (2,167)	WriteRequests (6)	StorefileSize (5 MB)	Num.Storefiles (5)	MemSize (1 MB)	Start Key	End Key	Region State
wordcount,,1651672640769.0b49ac33cea190195653c14bbf52bcb2.	master0.hp.com:16030	738	1	1 MB	1	0 MB		g	OPEN
wordcount,g,1651672640769.e502c2ac2a17e8f917f2ab4ebbf18533.	master0.hp.com:16030	412	2	1 MB	1	1 MB	g	m	OPEN
wordcount,m,1651672640769.07977d3dee2bc8bf9e11510211ec5dc4.	master0.hp.com:16030	363	1	1 MB	1	0 MB	m	r	OPEN
wordcount,r,1651672640769.9bb89b702a505d94d460baf6264fec5e.	master0.hp.com:16030	540	1	1 MB	1	0 MB	r	w	OPEN
wordcount,w,1651672640769.ce6fad66a0aa9ac9718233fb93c7785c.	master0.hp.com:16030	114	1	1 MB	1	0 MB	w		OPEN

As you can see, the memtable get populated.

---------- Lab Ends Here ----------

11. Hbase Spark Integration – Spark-Hbase connector – 90 Minutes

Compatible: Hbase 2.4 and Spark – 3.0.2

Stop the Hbase cluster and revert to hbase standalone mode. Refer the Annexure (Standalone Mode)

We will perform the following in this lab in a standalone Hbase installation:

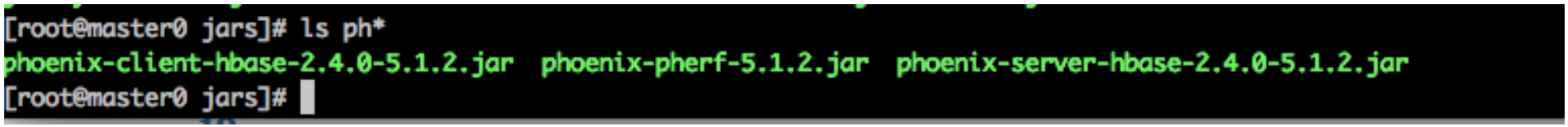
- Configure Spark to intergrate with Hbase.
- Retrieve records from Hbase table using Spark and create dataframe.
- Store spark dataframe in Hbase table.

Extract hbase connector and then copy the hbase* jar from the lib folder to the spark home jars folder.

#cd /Software/hbase-connectors-1.0.1-SNAPSHOT/lib
#cp hbase-* /opt/spark/jars/

```
[root@master0 lib]# ls hbase*
hbase-annotations-2.2.4-tests.jar     hbase-kafka-proxy-1.0.1-SNAPSHOT.jar  hbase-shaded-netty-2.2.1.jar
hbase-client-2.2.4.jar                hbase-metrics-2.2.4.jar               hbase-shaded-protobuf-2.1.0.jar
hbase-common-2.2.4.jar                hbase-metrics-api-2.2.4.jar           hbase-shaded-protobuf-2.2.1.jar
hbase-hadoop-compat-2.2.4.jar         hbase-protocol-2.2.4.jar              hbase-spark-1.0.1-SNAPSHOT.jar
hbase-hadoop2-compat-2.2.4.jar        hbase-protocol-shaded-2.2.4.jar       hbase-spark-protocol-1.0.1-SNAPSHOT.jar
hbase-kafka-model-1.0.1-SNAPSHOT.jar  hbase-shaded-miscellaneous-2.2.1.jar  hbase-spark-protocol-shaded-1.0.1-SNAPSHOT.jar
[root@master0 lib]# pwd
/Software/hbase-connectors-1.0.1-SNAPSHOT/lib
[root@master0 lib]#
```

Extract Phoenix to a folder and copy the phoenix related jar to the spark jar folder. (Assuming that the phoenix is extracted in /opt/phoenix folder)
Copy Phoenix jar, it contains all the dependencies jar required for interaction with spark and hbase.
#cd /opt/phoenix
#cp phoenix-* /opt/spark/jars/



Prepare HBase table with data

Run the following commands in HBase shell to prepare a sample table that will be used in the following sections.

hbase shell

```
create 'Person', 'Name', 'Address'
put 'Person', '1', 'Name:First', 'Raymond'
put 'Person', '1', 'Name:Last', 'Tang'
put 'Person', '1', 'Address:Country', 'Australia'
put 'Person', '1', 'Address:State', 'VIC'
```

```
put 'Person', '2', 'Name:First', 'Dnomyar'
put 'Person', '2', 'Name:Last', 'Gnat'
put 'Person', '2', 'Address:Country', 'USA'
```

```
put 'Person', '2', 'Address:State', 'CA'
```

The table returns the following result when scanning:

```
scan 'Person'
```

Case 1 : Fetch records from a Hbase and Store in a file using Spark.

Open a terminal and start a spark shell.

#spark-shell

Create DataFrame with the following commands. Execute all the commands in spark shell.

1) First import the required classes:

```
import org.apache.hadoop.hbase.spark.HBaseContext
import org.apache.hadoop.hbase.HBaseConfiguration
```

2) Create HBase configurations

```
val conf = HBaseConfiguration.create()
```

```
conf.set("hbase.zookeeper.quorum", "127.0.0.1:2181")
```

3) Create HBase context

```
// Instantiate HBaseContext that will be used by the following code
new HBaseContext(spark.sparkContext, conf)
```

4) Create DataFrame

```
val hbaseDF = (spark.read.format("org.apache.hadoop.hbase.spark")
 .option("hbase.columns.mapping",
   "rowKey STRING :key," +
   "firstName STRING Name:First, lastName STRING Name:Last," +
   "country STRING Address:Country, state STRING Address:State"
 )
 .option("hbase.table", "Person")
 ).load()
```

The columns mapping matches with the definition in the steps above.

5) Show DataFrame schema

```
scala> hbaseDF.schema
```

6) Show data

```
hbaseDF.show()
```

```
scala> val hbaseDF = (spark.read.format("org.apache.hadoop.hbase.spark")
     |   .option("hbase.columns.mapping",
     |     "rowKey STRING :key," +
     |     "firstName STRING Name:First, lastName STRING Name:Last," +
     |     "country STRING Address:Country, state STRING Address:State"
     |   )
     |   .option("hbase.table", "Person")
     |   ).load()
rowKey STRING :key,firstName STRING Name:First, lastName STRING Name:Last,country STRING Address:Country, state STRING
 Address:State
hbaseDF: org.apache.spark.sql.DataFrame = [lastName: string, country: string ... 3 more fields]

scala> hbaseDF.schema
res2: org.apache.spark.sql.types.StructType = StructType(StructField(lastName,StringType,true), StructField(country,St
ringType,true), StructField(state,StringType,true), StructField(firstName,StringType,true), StructField(rowKey,StringT
ype,true))

scala> hbaseDF.show()
+--------+---------+-----+---------+------+
|lastName|  country|state|firstName|rowKey|
+--------+---------+-----+---------+------+
|    Tang|Australia|  VIC|  Raymond|     1|
|    Gnat|      USA|   CA|  Dnomyar|     2|
+--------+---------+-----+---------+------+
```

Untill now, we've successfully loaded data from HBase in Spark 3.0.1.

You can also write into HBase from Spark too

OSTech

Case 2 : Fetching records from a file/from Memory dataframe and insert into Hbase using Spark.

In this example we want to store personal data in an HBase table. We want to store name as string, email address as string, birth date as date and height as a floating point number. The contact information (email) is stored in the `c` column family and personal information (birth date, height) is stored in the `p` column family. The key in HBase table will be the `name` attribute.

	Spark	HBase
Type/Table	`Person`	`person`
Name	`name: String`	`key`
Email address	`email: String`	`c:email`
Birth date	`birthDate: Date`	`p:birthDate`
Height	`height: Float`	`p:height`

Create HBase table – Using hbase shell.

Use the following command to create the HBase table:

```
shell> create 'person', 'p', 'c'
```

Insert data – In Spark-shell.

Use the following spark code in spark-shell to insert data into our HBase table:

```
val sql = spark.sqlContext

import java.sql.Date

case class Person(name: String,
                  email: String,
                  birthDate: Date,
                  height: Float)

var personDS = Seq(
 Person("alice", "alice@alice.com", Date.valueOf("2000-01-01"), 4.5f),
 Person("bob", "bob@bob.com", Date.valueOf("2001-10-17"), 5.1f)
).toDS
```

```
scala> val sql = spark.sqlContext
sql: org.apache.spark.sql.SQLContext = org.apache.spark.sql.SQLContext@1cc784c9

scala> import java.sql.Date
import java.sql.Date

scala>

scala> case class Person(name: String,
     |                   email: String,
     |                   birthDate: Date,
     |                   height: Float)
defined class Person

scala>

scala> var personDS = Seq(
     |  Person("alice", "alice@alice.com", Date.valueOf("2000-01-01"), 4.5f),
     |  Person("bob", "bob@bob.com", Date.valueOf("2001-10-17"), 5.1f)
     | ).toDS
personDS: org.apache.spark.sql.Dataset[Person] = [name: string, email: string ... 2 more fields]

scala>
```

Save the datframe to a Hbase table.

```
personDS.write.format("org.apache.hadoop.hbase.spark")
 .option("hbase.columns.mapping",
   "name STRING :key, email STRING c:email, " +
     "birthDate DATE p:birthDate, height FLOAT p:height")
 .option("hbase.table", "person")
 .option("hbase.spark.use.hbasecontext", false)
 .save()
```

The previously inserted data can be tested with a simple scan: On the Hbase shell.

```
shell> scan ‘person’
```

```
Use "exit" to quit this interactive shell.
For Reference, please visit: http://hbase.apache.org/2.0/book.html#shell
Version 2.4.11, r7e672a0da0586e6b7449310815182695bc6ae193, Tue Mar 15 10:31:00 PDT 2022
Took 0.0065 seconds
hbase:001:0> scan 'person'
ROW                           COLUMN+CELL
 alice                        column=c:email, timestamp=2022-05-06T15:58:19.976, value=alice@alice.com
 alice                        column=p:birthDate, timestamp=2022-05-06T15:58:19.976, value=\x00\x00\x00\xDCj\xCF\xAC\
                              x00
 alice                        column=p:height, timestamp=2022-05-06T15:58:19.976, value=@\x90\x00\x00
 bob                          column=c:email, timestamp=2022-05-06T15:58:19.990, value=bob@bob.com
 bob                          column=p:birthDate, timestamp=2022-05-06T15:58:19.990, value=\x00\x00\x00\xE9\x97\xF5\x
                              10\x00
 bob                          column=p:height, timestamp=2022-05-06T15:58:19.990, value=@\xA333
2 row(s)
```

You have successfully inserted record to hbase table using Spark Dataframe.

----------Lab Ends Here ----------

12. Install MySQL on a CentOS 7 Server

```
wget https://dev.mysql.com/get/mysql80-community-release-el7-3.noarch.rpm

rpm -Uvh mysql80-community-release-el7-3.noarch.rpm

yum install mysql-server
```

Start MySQL and Check its Status

1. Once you have MySQL ready on CentOS 7, it does not automatically start right after the installation. Therefore, you need to start it manually through the following command:

   ```
   systemctl start mysqld
   ```

2. You will get no response once MySQL starts so to check if it is working properly, use the command below:

   ```
   systemctl status mysqld
   ```

When installing MySQL on CentOS 7, a temporary root password is generated. Issue the command below to see it:
`grep 'password' /var/log/mysqld.log`

In order to change it, follow these steps:

1. Firstly, run the following command:

   ```
   sudo mysql_secure_installation
   ```

password: Root123.

CLI

OSTech

```
#mysql –u root –p
```

############ Lab Ends Here ----------
Source: https://www.hostinger.in/tutorials/how-to-install-mysql-on-centos-7

13. Errors:

ERROR: org.apache.hadoop.hbase.PleaseHoldException: Master is initializing
Scenario: Unable to create table using the hbase shell.

Solution : clean /opt/hbase/tmp and restart hbase server else try the following commands.

Install zookeeper use it library to access hbase's zookeeper to remove the following entry.
tar -xvf apache-zookeeper* -c /opt
cd /opt/apache-zookeeper*
#bin/zkCli.sh
[zk: localhost:2181(CONNECTED) 3] get /hbase/meta-region-server
?master:16000?m l5???PBUF

master0.hp.com?}?????0

#delete /hbase/meta-region-server

Restart the hbase.

Error: Unable to access from Laptop in docker.

```
127.0.0.1  Henrys-MacBook-Air.local localhost
255.255.255.255 broadcasthost
::1             localhost
127.0.0.1  couchbase0 master0 master0.hp.com
# Added by Docker Desktop
# To allow the same kube context to work on the host and the container:
127.0.0.1 kubernetes.docker.internal
# End of section
```

Ensure to enter the hostname of the Hbase Node in the Laptop /etc/hosts

Error
ERROR: org.apache.hadoop.hbase.PleaseHoldException: Master is initializing
at org.apache.hadoop.hbase.master.HMaster.checkInitialized(HMaster.java:2829)
at org.apache.hadoop.hbase.master.HMaster.createTable(HMaster.java:2085)
at
org.apache.hadoop.hbase.master.MasterRpcServices.createTable(MasterRpcServices.java:706)

at
org.apache.hadoop.hbase.shaded.protobuf.generated.MasterProtos$MasterService$2.callBlocking
Method(MasterProtos.java)
at org.apache.hadoop.hbase.ipc.RpcServer.call(RpcServer.java:392)
at org.apache.hadoop.hbase.ipc.CallRunner.run(CallRunner.java:133)
at org.apache.hadoop.hbase.ipc.RpcExecutor$Handler.run(RpcExecutor.java:359)
at org.apache.hadoop.hbase.ipc.RpcExecutor$Handler.run(RpcExecutor.java:339)

Solution: Clean the zookeeper directory.

https://stackoverflow.com/questions/17038957/org-apache-hadoop-hbase-pleaseholdexception-master-is-initializing

Error: java.io.IOException: cannot get log writer

2022-06-04 02:32:15,842 INFO [master/master0:16000:becomeActiveMaster] wal.AbstractFSWAL: Closed WAL: AsyncFSWAL master0.hp.com%2C16000%2C1654309909250:(num 1654309935646)
2022-06-04 02:32:15,917 ERROR [master/master0:16000:becomeActiveMaster] master.HMaster: Failed to become active master
java.io.IOException: cannot get log writer
at org.apache.hadoop.hbase.wal.AsyncFSWALProvider.createAsyncWriter(AsyncFSWALProvider.java:128)
at org.apache.hadoop.hbase.regionserver.wal.AsyncFSWAL.createWriterInstance(AsyncFSWAL.java:689)
at org.apache.hadoop.hbase.regionserver.wal.AsyncFSWAL.createWriterInstance(AsyncFSWAL.java:130)
at org.apache.hadoop.hbase.regionserver.wal.AbstractFSWAL.rollWriter(AbstractFSWAL.java:841)
at org.apache.hadoop.hbase.regionserver.wal.AbstractFSWAL.rollWriter(AbstractFSWAL.java:548)
at org.apache.hadoop.hbase.regionserver.wal.AbstractFSWAL.init(AbstractFSWAL.java:489)
at org.apache.hadoop.hbase.wal.AbstractFSWALProvider.getWAL(AbstractFSWALProvider.java:160)
at org.apache.hadoop.hbase.wal.AbstractFSWALProvider.getWAL(AbstractFSWALProvider.java:62)
at org.apache.hadoop.hbase.wal.WALFactory.getWAL(WALFactory.java:296)

at org.apache.hadoop.hbase.master.region.MasterRegion.createWAL(MasterRegion.java:187)
at org.apache.hadoop.hbase.master.region.MasterRegion.bootstrap(MasterRegion.java:207)

Solution : hbase.unsafe.stream.capability.enforce to false in base-site.xml

ERROR: KeeperErrorCode = NoNode for /hbase/master
Solution: Check the log. Determine the Namenode port no etc.

14. Annexure:

Hbase & YARN – with Mounted Folder and Extra port.

#docker run --name hbase0 --hostname hmaster.master.com --privileged --network spark-net -v /Users/henrypotsangbam/Documents/Docker:/opt -p 2181:2181 -p 16000:16000 -p 16010:16010 -p 16002:16002 -p 16012:16012 -p 16020:16020 -p 16030:16030 -p 8080:8080 -p 8085:8085 -p 8088:8088 -p 9870:9870 -p 9864:9864 -p 9000:9000 -p 9001:9002 -p 9003:9003 -p 9004:9004 -p 9005:9005 -i -t centos:7 /usr/sbin/init

#docker run --name monitor --hostname monitor --privileged --network spark-net -v /Users/henrypotsangbam/Documents/Docker:/opt -p 9090:9090 -p 9010:9010 -p 3000:3000 -i -t centos:7 /usr/sbin/init

15. Annexure (Standalone Mode)

Take a back up of hbase-site.xml. Location : /opt/hbase/conf

Replace the above file with the following content.

```
<?xml version="1.0"?>
<?xml-stylesheet type="text/xsl" href="configuration.xsl"?>
<configuration>
 <property>
  <name>hbase.rootdir</name>
  <value>file:///opt/data/hbase</value>
 </property>
 <property>
  <name>hbase.zookeeper.property.dataDir</name>
  <value>/opt/data/zookeeper</value>
 </property>
<property>
  <name>hbase.unsafe.stream.capability.enforce</name>
  <value>false</value>
 </property>
</configuration>
```

Start the hbase standalone cluster: bin/start-hbase.sh