HBase findings and trials

<u>Different ways to use HBase</u>:

HBase is an open source NoSQL distributed database, we don't need a specific schema for the storageFrom our investigation on materials present on HBase, there are different ways we could use HBase in our project: We can

1. Use it locally, by installing it.
2. We can use the AWS EMR HBase service.
3. We can use the plain EC2 instance and install HBase on top of it.

Using Hbase Locally:

For local installation we have below options:

- Standalone mode installation (No dependency on Hadoop system)
- Pseudo-Distributed mode installation (Single node Hadoop system + HBase installation)
- Fully Distributed mode installation (MultinodeHadoop environment + HBase installation)

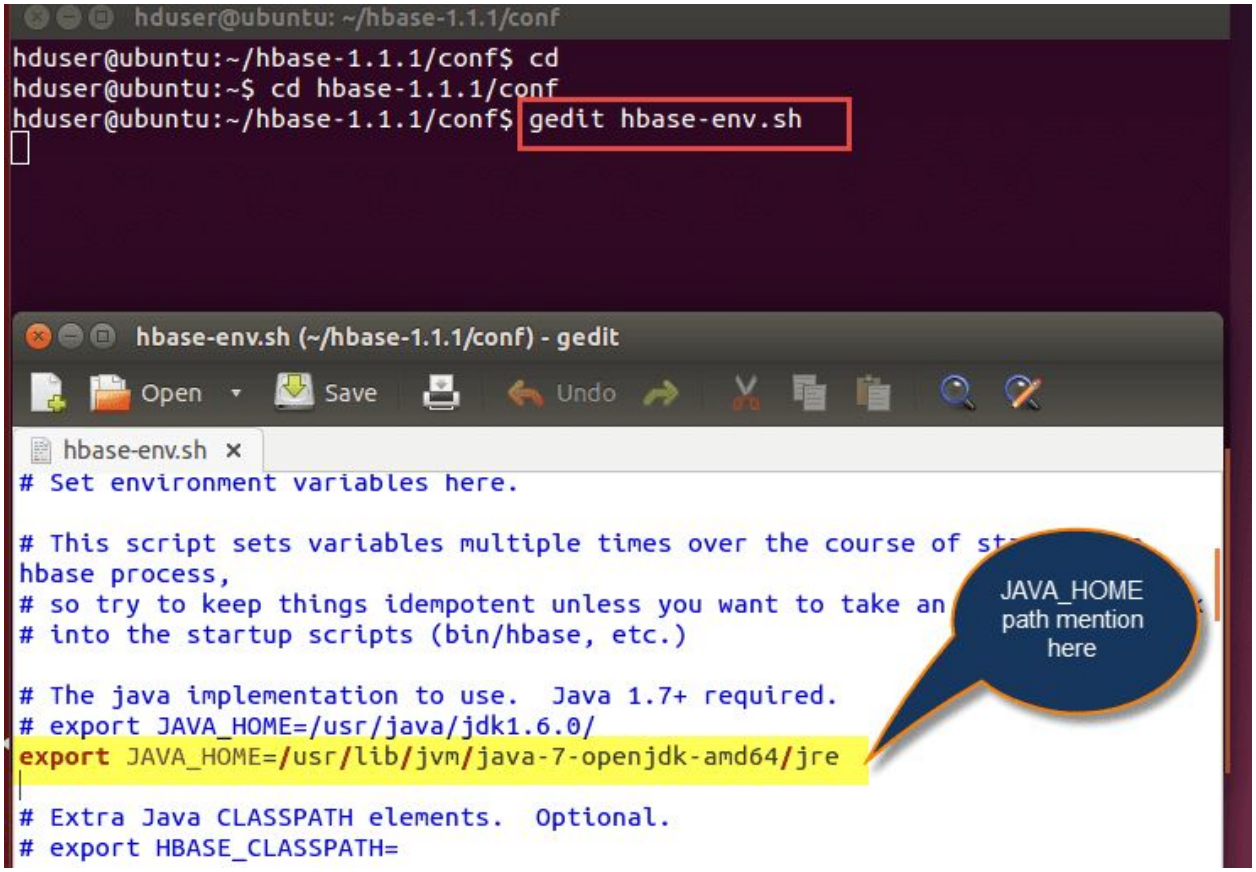
We tried the Standalone installation mode, which require us to follow below steps:

Installation is performed on Ubuntu with Hadoop already installed.

Step 1) Place hbase-x.x.x-bin.tar.gz in /home/hduser

Step 2) Unzip by executing command $tar -xvf hbase-x.x.x-bin.tar.gz**.** It will unzip the contents, and it will create hbase-x.x.x in the location /home/hduser

Step 3) Open hbase-env.sh as below and mention JAVA_HOME path in the location.



Step 4) Open ~/.bashrc file and mention HBASE_HOME path as shown in below

```
export HBASE_HOME=/home/hduser/hbase-x.x.x export PATH=
$PATH:$HBASE_HOME/bin
```

```
hduser@ubuntu:~/hbase-1.1.1/conf$ cd
hduser@ubuntu:~$ cd hbase-1.1.1/conf
hduser@ubuntu:~/hbase-1.1.1/conf$ gedit hbase-env.sh
hduser@ubuntu:~/hbase-1.1.1/conf$ gedit ~./bashrc
hduser@ubuntu:~/hbase-1.1.1/conf$ gedit ~/.bashrc
```

.bashrc (~) - gedit

Open Save Undo

.bashrc

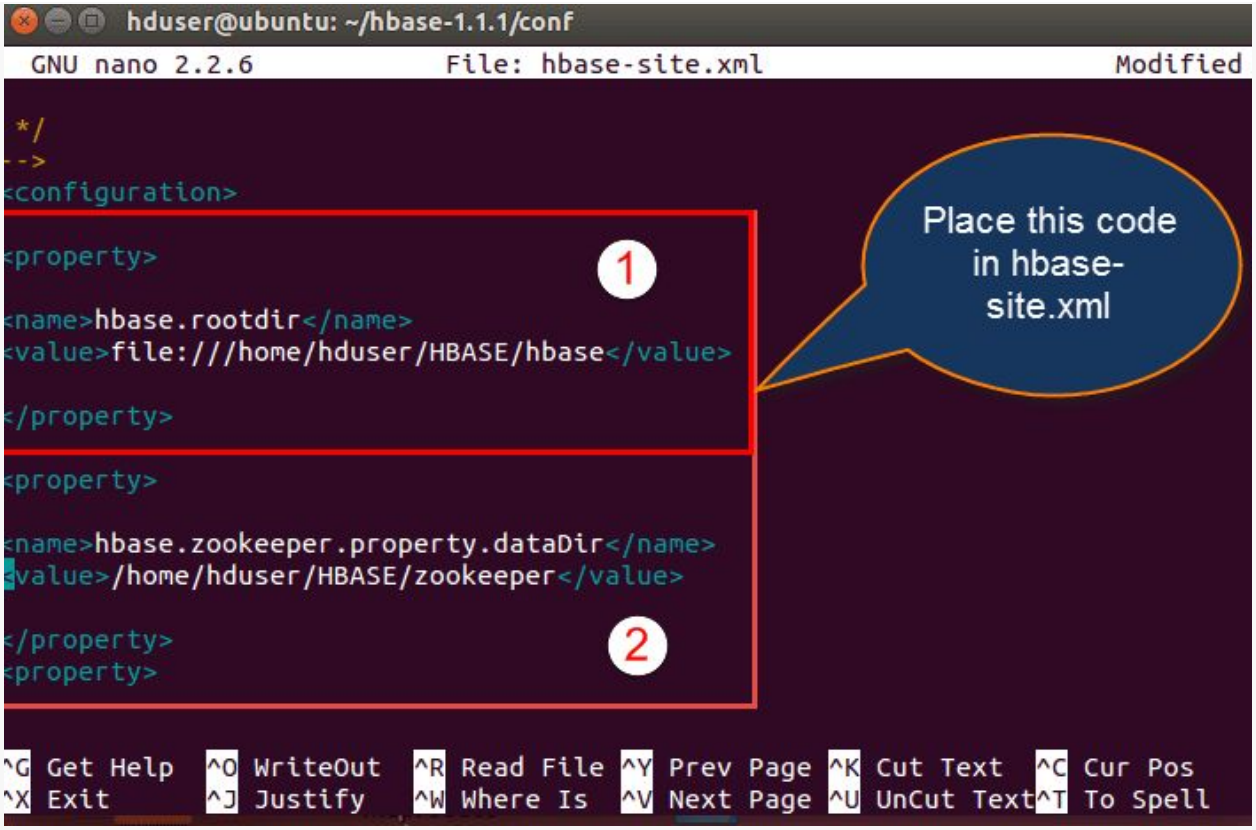
```
#export HADOOP_COMMON_LIB_NATIVE_DIR=${HADOOP_PREFIX}/lib/na
#export HADOOP_OPTS="-Djava.library.path=$HADOOP_PREFIX/lib"


export HADOOP_COMMON_LIB_NATIVE_DIR=$HADOOP_HOME/lib/native
export HADOOP_OPTS="-Djava.library.path=$HADOOP_HOME/lib"
export HADOOP_INSTALL=$HADOOP_HOME



export HIVE_HOME=/home/hduser/apache-hiv
export PATH=$PATH:$HIVE_HOME/bin



# HBASE PATH
export HBASE_HOME=/home/hduser/hbase-1.1.1
export PATH=$PATH:$HBASE_HOME/bin
```

HBASE_HOME
path setting

Step 5) Open hbase-site.xml and place the following properties inside the file

ubuntu$ gedit hbase-site.xml(code as below)

```
<property>
<name>hbase.rootdir</name>
<value>file:///home/hduser/HBASE/hbase</value>
</property>
<property>
<name>hbase.zookeeper.property.dataDir</name>
<value>/home/hduser/HBASE/zookeeper</value>
</property>
```



Here we are placing two properties

- One for HBase root directory and
- Second one for the data directory corresponds to ZooKeeper.

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

Step 6) Open hosts file present in /etc. location and mention the IPs as shown in below.

```
hduser@ubuntu: /etc
hduser@ubuntu:~/hbase-1.1.1/conf$ cd
hduser@ubuntu:~$ cd hbase-1.1.1/conf
hduser@ubuntu:~/hbase-1.1.1/conf$ gedit hbase-env.sh
hduser@ubuntu:~/hbase-1.1.1/conf$ gedit ~./bashrc
hduser@ubuntu:~/hbase-1.1.1/conf$ gedit ~/.bashrc
hduser@ubuntu:~/hbase-1.1.1/conf$ cd
hduser@ubuntu:~$ cd /etc
hduser@ubuntu:/etc$ nano hosts
hduser@ubuntu:/etc$ gedit hosts
```

```
*hosts [Read-Only] (/etc) - gedit
Open  Save  Undo
*hosts
127.0.0.1       localhost
127.0.0.1       ubuntu

# The following lines are desirable for IPv6 capable hosts
::1     ip6-localhost ip6-loopback
fe00::0 ip6-localnet
ff00::0 ip6-mcastprefix
ff02::1 ip6-allnodes
ff02::2 ip6-allrouters
```

Step 7) Now Run Start-hbase.sh in hbase-x.x.x/bin location as shown below.

And we can check by jps command to see whether HMaster is running or not.

```
hduser@ubuntu:~/hbase-1.1.1/bin$ start-hbase.sh
starting master, logging to /home/hduser/hbase-1.1.1/logs/hbase-hduser-master-ub
untu.out
hduser@ubuntu:~/hbase-1.1.1/bin$ jps
3597 HMaster
3665 Jps
```

HMaster Daemon

Step8) HBase shell can start by using "hbase shell" and it will enter into interactive shell mode as shown in the screenshot below. Once it enters shell mode, we can perform all types of commands.

```
rahul@rahul-inspiron-7586:~/hduser/hbase-2.2.6/bin$ ./start-hbase.sh
running master, logging to /home/rahul/hduser/hbase-2.2.6/bin/../logs/hbase-rahul-master-rahul-inspiron-7586.out
rahul@rahul-inspiron-7586:~/hduser/hbase-2.2.6/bin$ jps
5296 RemoteMavenServer36
14536 Jps
3689 Main
rahul@rahul-inspiron-7586:~/hduser/hbase-2.2.6/bin$
```

```
ubuntu: ~/hbase-1.1.1/bin
hduser@ubuntu:~/hbase-1.1.1/bin$ hbase shell
SLF4J: Class path contains multiple SLF4J bindings.
SLF4J: Found binding in [jar:file:/home/hduser/hbase-1.1.1/
SLF4J: Found binding in [jar:file:/home/hduser/hadoop-2.2.0
SLF4J: See http://www.slf4j.org/codes.html#multiple_binding
SLF4J: Actual binding is of type [org.slf4j.impl.Log4jLogge
2015-09-11 17:01:42,907 WARN  [main] util.NativeCodeLoader:
HBase Shell; enter 'help<RETURN>' for list of supported com
Type "exit<RETURN>" to leave the HBase Shell
Version 1.1.1, rd0a115a7267f54e01c72c603ec53e91ec418292f, T

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

As it is seen above the standalone mode works fine, but when we try to access it at: http://localhost:60010/ we can't see the page. We also checked the zookeeper on the port 2181, which is the default mode for the zookeeper installation but it didn't work.

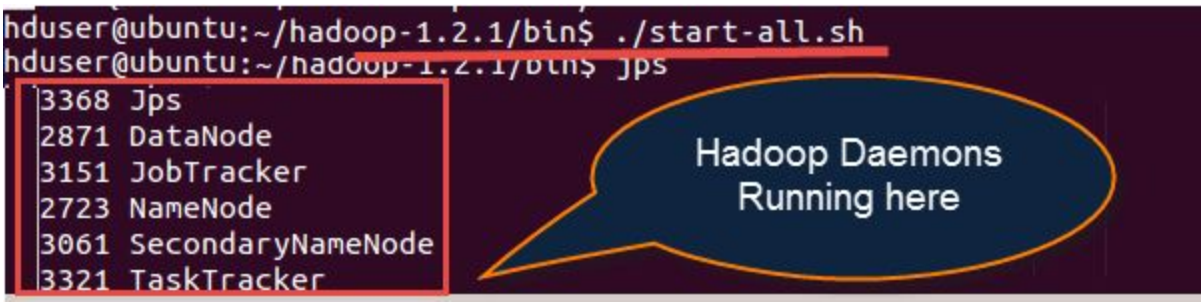
So we assumed it may work in pseudo distributed mode:

We follow the steps similar to we did in case of standalone installation but changing the distributed property in HBase-site.xml as **true**.

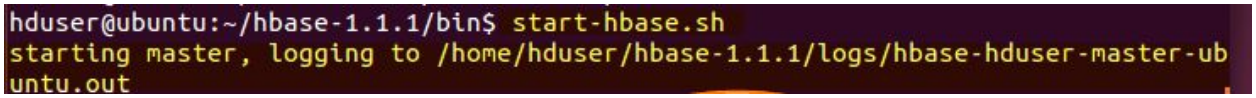
```
<property>
<name>hbase.cluster.distributed</name>
<value>true</value>
</property>
```

Now we point to our hadoop installation and start Hadoop daemons first and after that start HBase daemons as shown below:

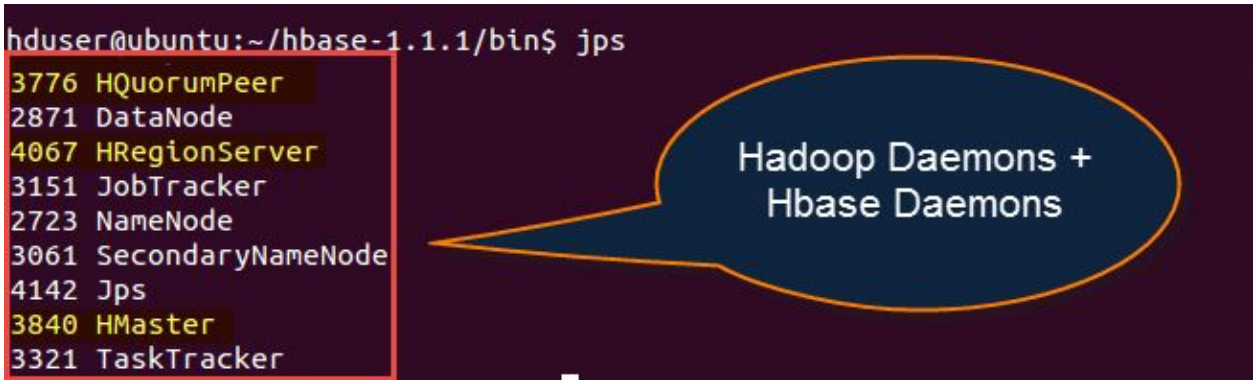
Here first you have to start Hadoop daemons by using**"./start-all.sh"** command as below.



After starting Hbase daemons by hbase-start.sh



Checking **jps**



As we see the all the above steps worked perfectly fine but still we were not able to expose the hbase via zookeeper on port 2181, we followed different articles on stack overflow, blogs but didn't find something meaningful that could solve this issue. While following the above steps we encountered many errors but were able to solve them and reach the above steps at the end.

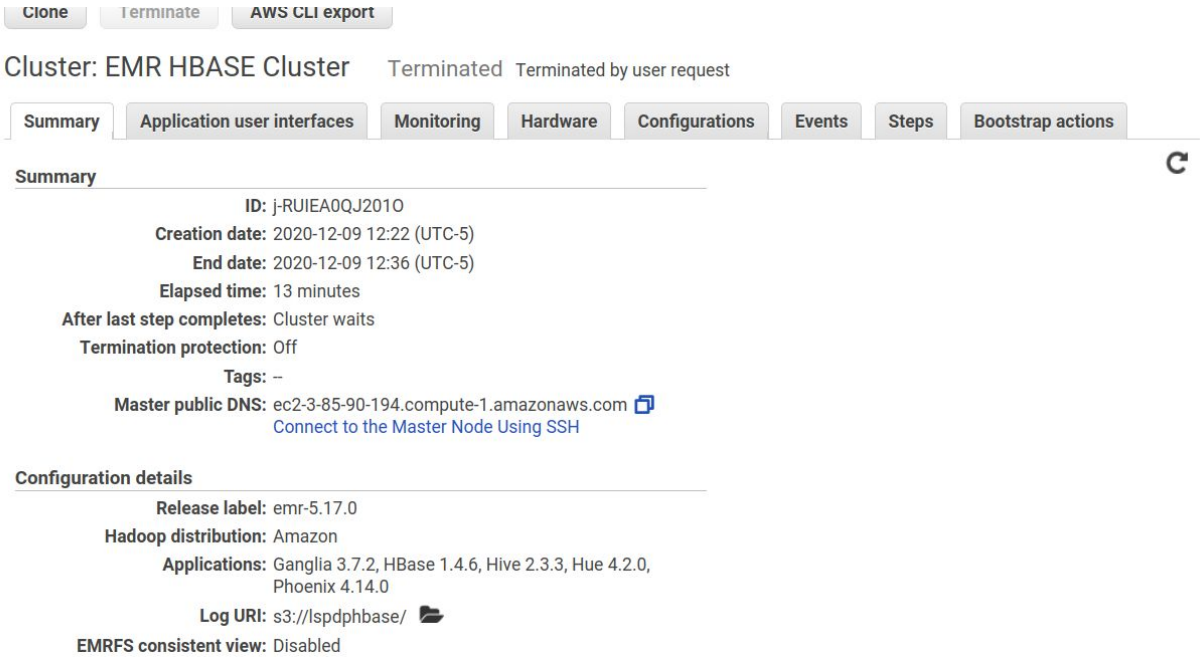
We also tried following steps mentioned in the blog:
https://www.tutorialspoint.com/hbase/hbase_installation.htm

But we encountered many errors and were not able to expose the port and access it from our java application.

After not being able to make the above steps work, we thought of leveraging the managed HBase service from AWS which comes with AWS EMR:

We followed the blog: https://aws.amazon.com/emr/features/hbase/ and tried following the configuration from the UI as mentioned in the post: https://docs.aws.amazon.com/emr/latest/ReleaseGuide/emr-hbase-configure.html

We created an EMR cluster with HBase installed as seen below:



And tried accessing the cluster on http://*master-public-dns-name*:16010/ for the HBase service. https://docs.aws.amazon.com/emr/latest/ManagementGuide/emr-web-interfaces.html

But here as well we were unable to access the HBase via port 16010 or zookeeper 2181. Working on this has already consumed a significant portion of time hence we dropped the plan to further get hbase working, we could have installed it on a plain EC2 instance and get it working but as the local installation didn't work and we were not able to expose the port **2181** or **16010.**

Hence as a suggestion from the professor we opted for broadcasting the HBase data during the MapReduce jobs.