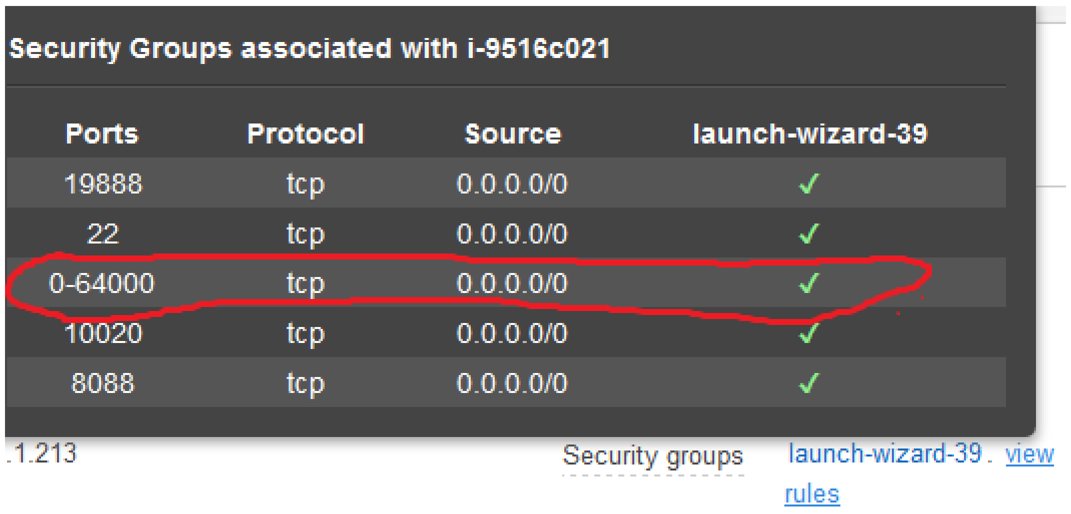
Neil Yao 10/28/2016

# Install Multi-Nodes Hadoop Cluster on AWS

**1. Create a new node of at least medium size. It is possible, but I do not recommend trying to reconfigure existing node into a new cluster for several reasons.**

a) When creating a node I recommend changing the default 8G hard drive to at least 32G so that you do not run out of space easily.

b) Change your security group setting to open firewall access. Rather than figure out all individual port, you can set 0-64000 range opening up all ports (not the  most secure setting in the long term)



Security Groups associated with i-9516c021

| Ports | Protocol | Source | launch-wizard-39 |
|---|---|---|---|
| 19888 | tcp | 0.0.0.0/0 | ✓ |
| 22 | tcp | 0.0.0.0/0 | ✓ |
| 0-64000 | tcp | 0.0.0.0/0 | ✓ |
| 10020 | tcp | 0.0.0.0/0 | ✓ |
| 8088 | tcp | 0.0.0.0/0 | ✓ |

c) Finally, right click on the Master node and choose “create more like this” to create 2 more nodes with same settings. If you configure the master first, all of the settings (drive size, security group, etc.) will be automatically copied to new instances.

**2. Connect to the master and set up Hadoop. Do not attempt to repeat these steps on workers yet – you will only need to set up Hadoop once.**

a. Install ant to list java processes

```
sudo yum install ant
```

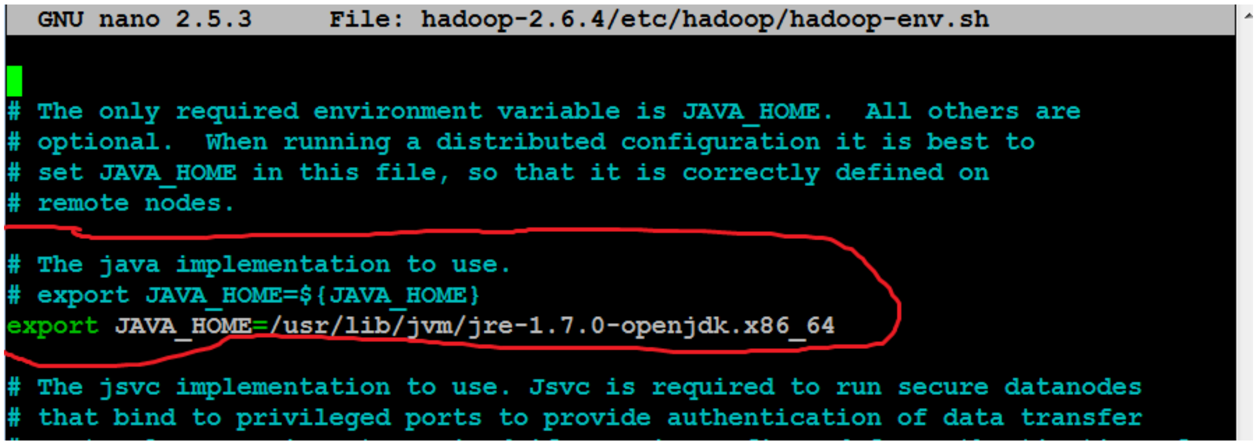
b. Download and unpack Hadoop

```
wget http://rasinsrv07.cstcis.cti.depaul.edu/CSC555/hadoop-2.6.4.tar.gz
tar xzf hadoop-2.6.4.tar.gz
```

c. Edit conf/Hadoop-env.sh to add the JAVA_HOME configuration:

```
nano hadoop-2.6.4/etc/hadoop/hadoop-env.sh
```

*export JAVA_HOME=/usr/lib/jvm/jre-1.7.0-openjdk.x86_64/*

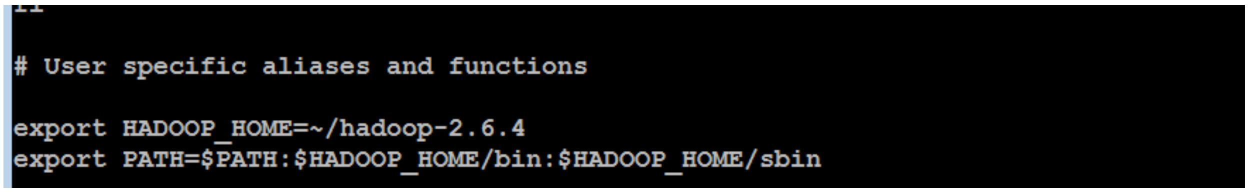


```
  GNU nano 2.5.3      File: hadoop-2.6.4/etc/hadoop/hadoop-env.sh


# The only required environment variable is JAVA_HOME.  All others are
# optional.  When running a distributed configuration it is best to
# set JAVA_HOME in this file, so that it is correctly defined on
# remote nodes.

# The java implementation to use.
# export JAVA_HOME=${JAVA_HOME}
export JAVA_HOME=/usr/lib/jvm/jre-1.7.0-openjdk.x86_64

# The jsvc implementation to use. Jsvc is required to run secure datanodes
# that bind to privileged ports to provide authentication of data transfer
```

d. Modify .bashrc file to add these two lines:

```
nano ~/.bashrc
```

export HADOOP_HOME=~/hadoop-2.6.4
export PATH=$PATH:$HADOOP_HOME/bin:$HADOOP_HOME/sbin



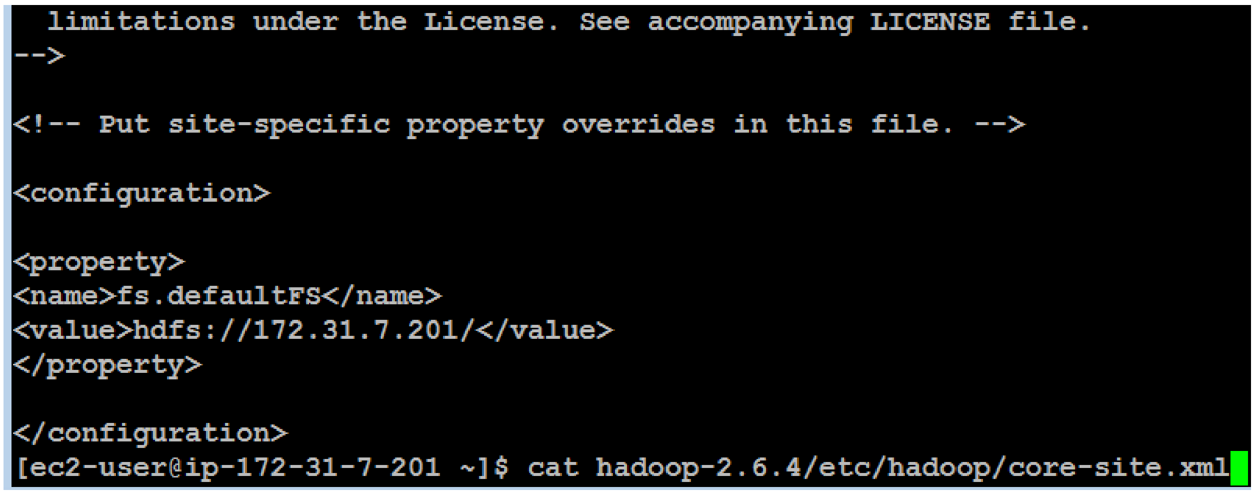
```
# User specific aliases and functions

export HADOOP_HOME=~/hadoop-2.6.4
export PATH=$PATH:$HADOOP_HOME/bin:$HADOOP_HOME/sbin
```

e. To immediately refresh the settings (that will be automatic on next login), run

```
source ~/.bashrc
```

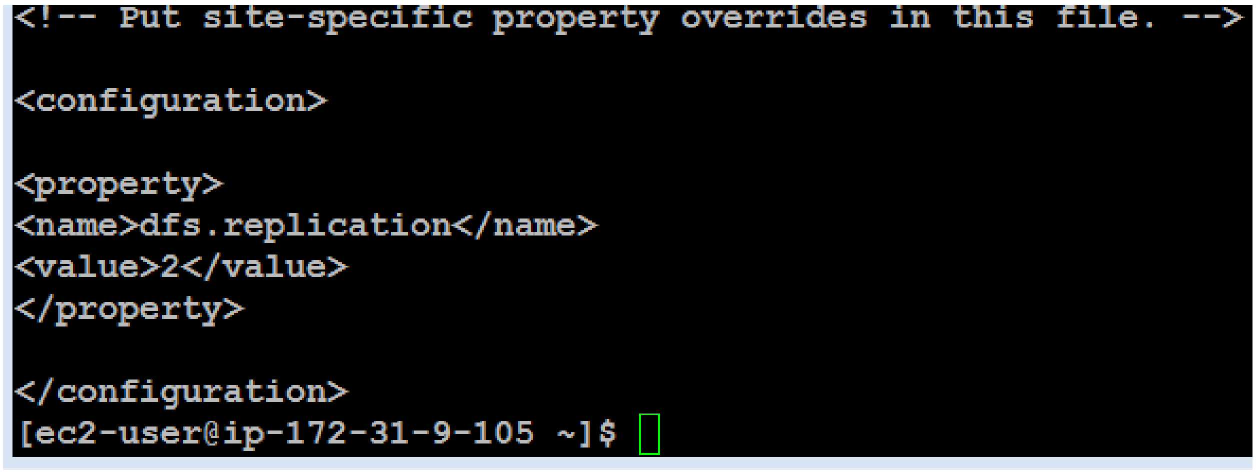
f. Configure core-site.xml, adding the PrivateIP (Private NOT public) of the master.

```
nano hadoop-2.6.4/etc/hadoop/core-site.xml
```



```
  limitations under the License. See accompanying LICENSE file.
-->

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

<configuration>

<property>
<name>fs.defaultFS</name>
<value>hdfs://172.31.7.201/</value>
</property>

</configuration>
[ec2-user@ip-172-31-7-201 ~]$ cat hadoop-2.6.4/etc/hadoop/core-site.xml
```

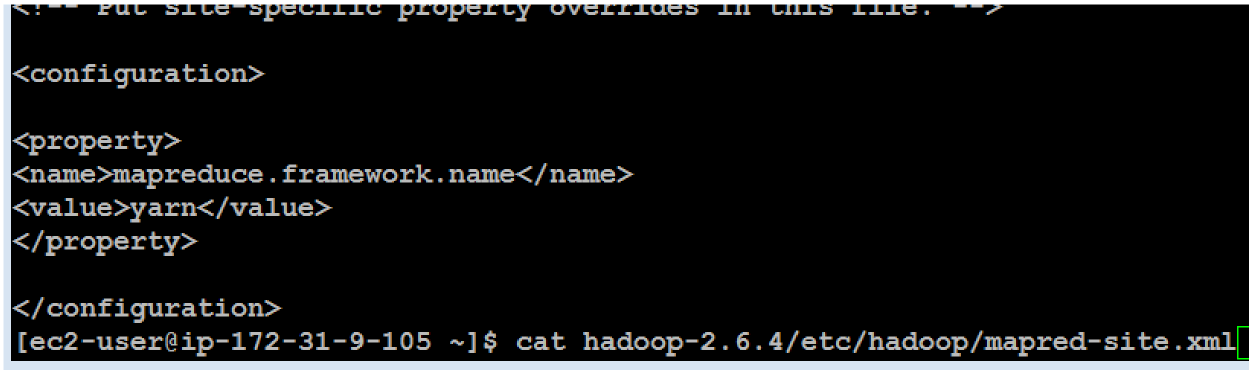
g. Configure hdfs-site and set replication factor to 2.



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

<configuration>

<property>
<name>dfs.replication</name>
<value>2</value>
</property>

</configuration>
[ec2-user@ip-172-31-9-105 ~]$
```

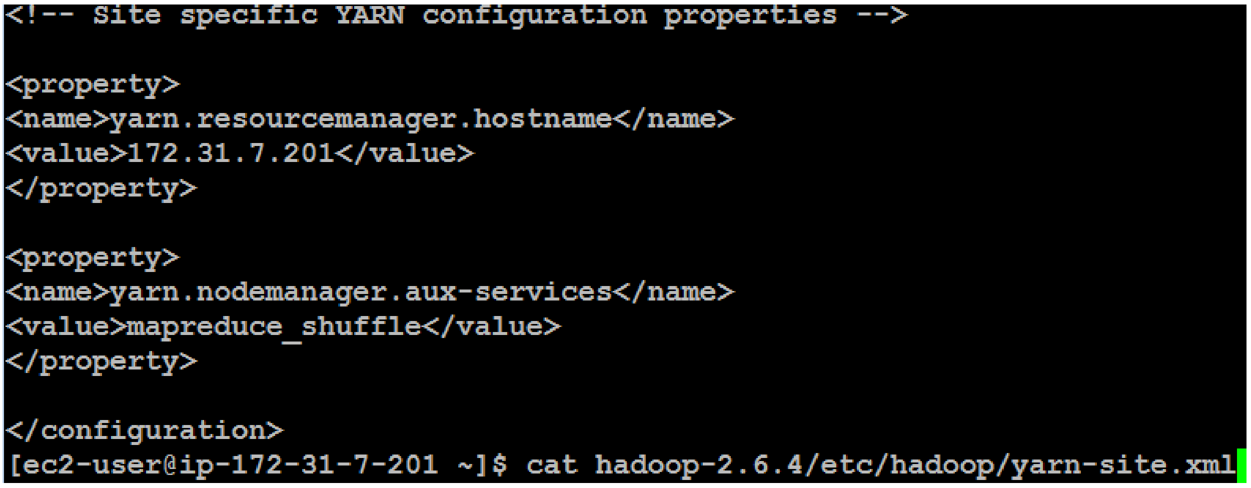
h. Configure mapred-site.

```
cp hadoop-2.6.4/etc/hadoop/mapred-site.xml.template hadoop-2.6.4/etc/hadoop/mapred-site.xml and then configure mapred-site.xml
nano hadoop-2.6.4/etc/hadoop/mapred-site.xml
```



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

<configuration>

<property>
<name>mapreduce.framework.name</name>
<value>yarn</value>
</property>

</configuration>
[ec2-user@ip-172-31-9-105 ~]$ cat hadoop-2.6.4/etc/hadoop/mapred-site.xml
```

i. Configure yarn-site.xml (once again, use PrivateIP of the master)



```
<!-- Site specific YARN configuration properties -->

<property>
<name>yarn.resourcemanager.hostname</name>
<value>172.31.7.201</value>
</property>

<property>
<name>yarn.nodemanager.aux-services</name>
<value>mapreduce_shuffle</value>
</property>

</configuration>
[ec2-user@ip-172-31-7-201 ~]$ cat hadoop-2.6.4/etc/hadoop/yarn-site.xml
```

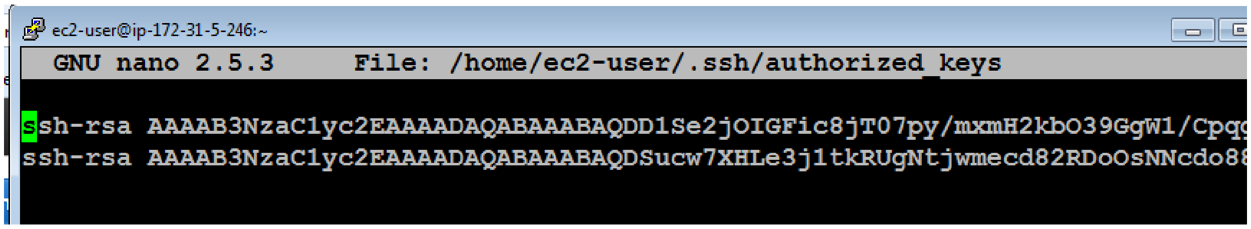
j. Finally, edit the slaves file and list your 3 nodes (master and 2 workers) using Private IPs

```
[ec2-user@ip-172-31-7-201 ~]$ cat hadoop-2.6.4/etc/hadoop/slaves
172.31.7.201
172.31.5.246
172.31.11.50
```

Use **nano hadoop-2.6.4/etc/hadoop/slaves** to add the privates IPs

**3. Now, we will pack up and move Hadoop to the workers. All you need to do is to generate and then copy the public key to the worker nodes to achieve passwordless access across your cluster.**

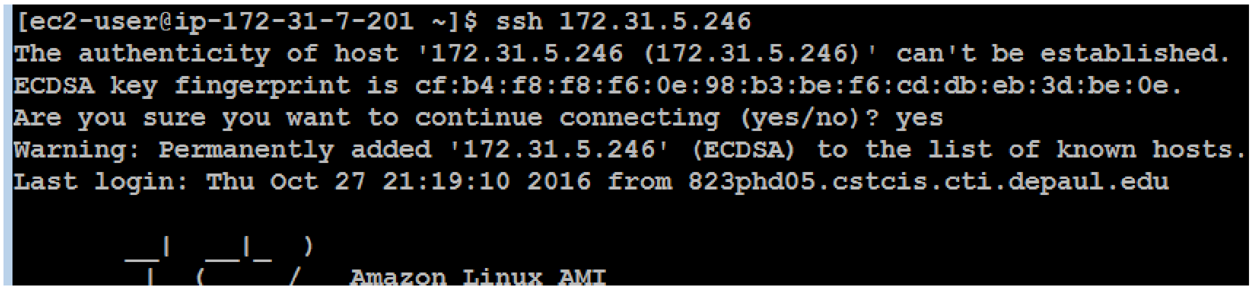
a. Run `ssh-keygen -t rsa` (and enter empty values for the passphrase) on the master node. That will generate .ssh/id_rsa and .ssh/id_rsa.pub (private and public key). You now need to manually copy the .ssh/id_rsa.pub and append it to ~/.ssh/authorized_keys **on each worker. Not on the master.** Keep in mind that this is a single-line public key and accidentally introducing a line break would break it. For example: Note that this is NOT the master, but one of the workers (ip-172-31-5-246). The first public key is the .pem Amazon half and the 2nd public key is the master's public key copied in as one line.



```
ec2-user@ip-172-31-5-246:~
  GNU nano 2.5.3       File: /home/ec2-user/.ssh/authorized_keys

ssh-rsa AAAAB3NzaC1yc2EAAAADAQABAAABAQDD1Se2jOIGFic8jT07py/mxmH2kbO39GgW1/Cpq
ssh-rsa AAAAB3NzaC1yc2EAAAADAQABAAABAQDSucw7XHLe3j1tkRUgNtjwmecd82RDoOsNNcdo8
```

b. You can add the public key of the master to the master by running this command:
`cat ~/.ssh/id_rsa.pub >> ~/.ssh/authorized_keys`

c. Make sure that you can ssh to all of the nodes from the master node (by running ssh 54.186.221.92, where the IP address is your worker node) from the master and ensuring that you were able to login. You can exit after successful ssh connection by typing exit (the command prompt will tell you which machine you are connected to, e.g., ec2-user@ip-172-31-37-113). Here's me ssh-ing from master to worker.



```
[ec2-user@ip-172-31-7-201 ~]$ ssh 172.31.5.246
The authenticity of host '172.31.5.246 (172.31.5.246)' can't be established.
ECDSA key fingerprint is cf:b4:f8:f8:f6:0e:98:b3:be:f6:cd:db:eb:3d:be:0e.
Are you sure you want to continue connecting (yes/no)? yes
Warning: Permanently added '172.31.5.246' (ECDSA) to the list of known hosts.
Last login: Thu Oct 27 21:19:10 2016 from 823phd05.cstcis.cti.depaul.edu

       __|  __|_  )
       _|  (     /   Amazon Linux AMI
```

Once you are have verified that you can ssh from the master node to every cluster member including the master itself (ssh localhost), you are going to return to the master node (exit until your prompt shows the IP address of the master node) and pack up the contents of the hadoop directory there. Make sure your Hadoop installation is configured correctly (because from now on, you will have 3 copies of the Hadoop directory and all changes may need to be applied in 3 places).

**cd** (go to root home directory, i.e. /home/ec2-user/)

(pack up the entire Hadoop directory into a single file for transfer. You can optionally compress the file with gzip)

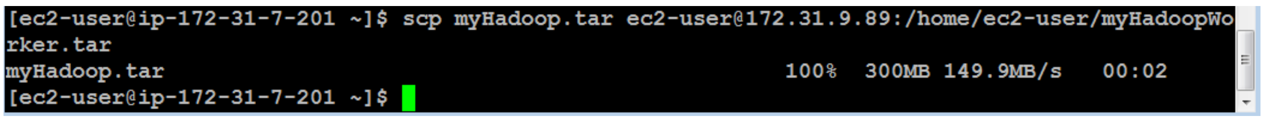
```
tar cvf myHadoop.tar hadoop-2.6.4
```

**ls -al myHadoop.tar** (to verify that the .tar file had been created)

Now, you need to copy the myHadoop.tar file to every non-master node in the cluster. If you had successfully setup public-private key access in the previous step, this command (for each worker node) will do that:

(copies the myHadoop.tar file from the current node to a remote node into a file called myHadoopWorker.tar**. Don't forget to replace the IP address with that your worker nodes**. By the way, since you are on the Amazon EC2 network, either Public or Private IP will work just fine.)

```
scp myHadoop.tar ec2-user@54.187.63.189:/home/ec2-user/myHadoopWorker.tar
```



```
[ec2-user@ip-172-31-7-201 ~]$ scp myHadoop.tar ec2-user@172.31.9.89:/home/ec2-user/myHadoopWo
rker.tar
myHadoop.tar                                              100%  300MB 149.9MB/s   00:02
[ec2-user@ip-172-31-7-201 ~]$
```

Once the tar file containing your Hadoop installation from master node has been copied to each worker node, you need to login to each non-master node and unpack the .tar file.

Run the following command (on each worker node, not on the master) to untar the hadoop file. We are purposely using a different tar archive name (i.e., **myHadoopWorker.tar**), so if you get "file not found" error, that means you are running this command on the master node or have not successfully copied myHadoopWorker.tar file.

```
tar xvf myHadoopWorker.tar
```

Once you are done, run this on the master:

```
hadoop namenode –format
start-dfs.sh
start-yarn.sh
mr-jobhistory-daemon.sh start historyserver
```

Verify if everything is running: **jps**

You should verify that the cluster is running by pointing your browser to the link below.

**http://[insert-the-public-ip-of-master]:50070/**

Make sure that the cluster is operational (you can see the 3 nodes under Datanodes tab). Submit a screenshot of your cluster status view.

## Install Pig

Download and install Pig:

```
cd
wget http://rasinsrv07.cstcis.cti.depaul.edu/CSC555/pig-0.15.0.tar.gz
gunzip pig-0.15.0.tar.gz
tar xvf pig-0.15.0.tar
```

Set the environment variables (this can also be placed in ~/.bashrc to make it permanent)

```
export PIG_HOME=/home/ec2-user/pig-0.15.0
export PATH=$PATH:$PIG_HOME/bin
```

Now run pig (and use the pig home variable we set earlier):

```
cd $PIG_HOME
bin/pig
```

## Install Hive

Download and install Hive:

```
cd
```

(this command is there to make sure you start from home directory, on the same level as where hadoop is located)

```
wget http://rasinsrv07.cstcis.cti.depaul.edu/CSC555/apache-hive-2.0.1-bin.tar.gz
gunzip apache-hive-2.0.1-bin.tar.gz
tar xvf apache-hive-2.0.1-bin.tar
```

set the environment variables (can be automated by recording these lines in ~/.bashrc or ~/.bash_profile). If you don't, you have to set these variables every time you use Hive.

```
export HIVE_HOME=/home/ec2-user/apache-hive-2.0.1-bin
export PATH=$HIVE_HOME/bin:$PATH
```

```
$HADOOP_HOME/bin/hadoop fs -mkdir /tmp
$HADOOP_HOME/bin/hadoop fs -mkdir /user/hive/warehouse
```

(if you get an error here, it means that /user/hive does not exist yet. Fix that by running `$HADOOP_HOME/bin/hadoop fs -mkdir -p /user/hive/warehouse` instead)

```
$HADOOP_HOME/bin/hadoop fs -chmod g+w /tmp
$HADOOP_HOME/bin/hadoop fs -chmod g+w /user/hive/warehouse
```

Run hive (from the hive directory because of the first command below):

```
cd $HIVE_HOME
$HIVE_HOME/bin/schematool -initSchema -dbType derby
```

(NOTE: This command initializes the database metastore. If you need to restart/reformat or see errors related to meta store, run `rm -rf metastore_db/` and then repeat the above initSchema command)

```
bin/hive
```

## Install HBase

Download and install HBase

```
wget http://rasinsrv07.cstcis.cti.depaul.edu/CSC555/hbase-0.90.3.tar.gz
gunzip hbase-0.90.3.tar.gz
tar xvf hbase-0.90.3.tar
cd hbase-0.90.3
```

(Start HBase service, there is a corresponding stop service and this assumes Hadoop home is set)

```
bin/start-hbase.sh
```

(Open the HBase shell – at this point jps should show HMaster)

bin/hbase shell

# Install Spark