**CLUSTER SETUP AND AMI CREATION**

I first created an instance using the Amazon Linux AMI to build my own custom AMI caller “**linux-hadoop-madiraju”** with Hadoop in it. Since certain settings remain common throughout all nodes, I installed Hadoop and changed settings which are common for all namenodes and datanodes. The steps to build this AMI are as given below

1. I first chose the launch instance option from the EC2 dashboard to create a new instance.
2. When asked to choose a AMI for the instance, I chose the Amazon Linux AMI.
3. I then selected the t2.micro instance type, set storage size to 8GB and configured my own security group.
4. The instance was then reviewed and launched after which I downloaded my SSH Private key.
5. Used putty to ssh into my newly created instance using my ec2 username and public dns. The ssh private key was converted into a .pem file using puttygen.
6. I was now able to access my instance via ssh.
7. Installed java 8 using the command **“sudo yum install java-1.8.0-openjdk-src.x86\_64”.**
8. Downloaded the Hadoop2.7.4 tarball binary file from Hadoop.apache.org using the command **“wget <http://apache.claz.org/hadoop/common/hadoop-2.7.4/hadoop-2.7.4.tar.gz>”**.
9. Extracted all files from Hadoop-2.7.4.tar.gz into /usr/local and moved all files to a folder called Hadoop.
10. Set the .bashrc file with the following settings:

**export JAVA\_HOME=/usr**

**export PATH=$PATH:$JAVA\_HOME/bin**

**export HADOOP\_HOME=/usr/local/hadoop**

**export PATH=$PATH:$HADOOP\_HOME/bin**

**export HADOOP\_CONF\_DIR=/usr/local/hadoop/etc/Hadoop**

1. Loaded the above variables with ” . ~/.bashrc”.
2. Updated the hadoop-env.sh file by setting JAVA\_HOME=/usr.
3. Changed the core-site.xml by adding the following settings:

**<configuration>**

**<property>**

**<name>fs.defaultFS</name>**

**<value>hdfs://ec2-34-229-227-202.compute-1.amazonaws.com:9000</value>**

**</property>**

**</configuration>**

1. All settings up to the above stage remains common for all nodes. We might need to change the values of dns in core-site.xml if the namenode DNS changes.
2. I saved an image with the above settings by right clicking on the instance and choosing save image. I gave it the name **linux-hadoop-madiraju.**
3. Launched 4 instances using the linux-hadoop-madiraju keeping one instance as namenode and rest of them as datanode.
4. The namenode was setup as follows:

* Updated the /etc/hosts file by including the IP and DNS of the namenode and all the datanodes. Removed the 127.0.0.1 localhost line from the file.
* Modified hdfs-site.xml by adding the following lines:

**<configuration>**

**<property>**

**<name>dfs.replication</name>**

**<value>3</value>**

**</property>**

**<property>**

**<name>dfs.namenode.name.dir</name>**

**<value>file:///usr/local/hadoop/hadoop\_data/hdfs/namenode</value>**

**</property>**

**</configuration>**

* Created the hadoop data directory in HADOOP\_HOME with the command

**sudo mkdir -p $HADOOP\_HOME/hadoop\_data/hdfs/namnode**

* Created a file named masters in $HADOOP\_CONF\_DIR directory with the command

echo “namenode\_hostname” || cat >> $HADOOP\_CONF\_DIR/masters.

* Added datanode hosts to $HADOOP\_CONF\_DIR/slaves

1. The datanodes were setup as follows:

* Modified each datanodes hdfs-site.xml by adding the following lines:

**<configuration>**

**<property>**

**<name>dfs.replication</name>**

**<value>3</value>**

**</property>**

**<property>**

**<name>dfs.datanode.data.dir</name>**

**<value>file:///usr/local/hadoop/hadoop\_data/hdfs/datanode</value>**

**</property>**

**</configuration>**

* Created a data directory across each data node using “**sudo mkdir -p $HADOOP\_CONFIG\_DIR/hadoop\_data/hdfs/datanode”.**
* Copied the core-site.xml file from namenode to all the datanodes
* Updated HADOOP\_HOME ownership from root to user.

1. To setup password less ssh I created a config file with the following lines and added it to /home/ec2-user/ssh in all the nodes:

**Host NameNode**

**HostName ec2-34-229-227-202.compute-1.amazonaws.com**

**User ec2-user**

**IdentityFile ~/.ssh/Hadoop-PA.pem**

**Host DataNode1**

**HostName ec2-54-90-209-201.compute-1.amazonaws.com**

**User ec2-user**

**IdentityFile ~/.ssh/Hadoop-PA.pem**

**Host DataNode2**

**HostName ec2-34-229-242-12.compute-1.amazonaws.com**

**User ec2-user**

**IdentityFile ~/.ssh/Hadoop-PA.pem**

**Host DataNode3**

**HostName ec2-54-211-38-107.compute-1.amazonaws.com**

**User ec2-user**

**IdentityFile ~/.ssh/Hadoop-PA.pem**

The file includes the DNS of all namenode and datanodes along with the username and path to the ssh private key file.

Generated key files on namenode and copied it to authorized\_keys folder in ssh across all nodes.

1. Formatted HDFS by using the command **“hdfs namenode -format”.**
2. Started the DFS service using $HADOOP\_HOME/sbin/start-dfs.sh.
3. Browsed the HDFS site using <http://namenode_dns:50070> .
4. Started YARN on namenode

**$HADOOP\_HOME/sbin/start-yarn.sh**

**$HADOOP\_HOME/sbin/mr-jobhistory-daemon.sh start historyserver**

1. Installed ant across all nodes.
2. Ran jsp on all nodes to check if the appropriate java processes are running.
3. I copied the input file states from local to hdfs with the command “hdfs dfs -put / states.
4. I first compiled my java program, created the jar file and ran the jar using the below commands:

**hadoop com.sun.tools.javac.Main FileName.java**

**jar cf WordCount5.jar WordCount\*.class**

**hadoop jar FileName.jar FileName /states /output**