## **Aim:** To install Openstack cloud and installing hdfs and checking fault tolerance.

## **Procedure:**

Installed devstack.

* We needed to add a user to install DevStack.

$ adduser stack

* It will need to have sudo privileges:

$ apt-get install sudo -y || yum install -y sudo  
$ echo "stack ALL=(ALL) NOPASSWD: ALL" >> /etc/sudoers

* **Logged out** and **logged in** as that user.
* Downloaded Devstack

$ sudo apt-get install git -y || sudo yum install -y git  
$ git clone https://git.openstack.org/openstack-dev/devstack  
$ cd devstack

* Ran Devstack. Now to configure stack.sh. DevStack includes a sample in devstack/samples/local.conf.

local.conf will look something like this:

[[local|localrc]]  
FLOATING\_RANGE=192.168.1.224/27  
FIXED\_RANGE=10.11.12.0/24  
FIXED\_NETWORK\_SIZE=256  
FLAT\_INTERFACE=eth0  
ADMIN\_PASSWORD=supersecret  
DATABASE\_PASSWORD=iheartdatabases  
RABBIT\_PASSWORD=flopsymopsy  
SERVICE\_PASSWORD=iheartksl  
SERVICE\_TOKEN=xyzpdqlazydog

* Ran DevStack:

$ ./stack.sh

**HDFS installation:**

The approach adopted during the installation involved configuring single data nodes on each VM initially. Thereafter the same were reconfigured into a multi-node cluster through installation of namenodes and resource manager. The important steps of the installation are elucidated below:

### 3.1 Pre-install Steps

3.1.1 Installed Oracle Java version 1.7.0.15 and set paths in the .bashrc file.

3.1.2 Created a dedicated Hadoop user hduser and a group hadoop.

$ sudo addgroup hadoop

$ sudo adduser --ingroup hadoop hduser

3.1.3 Switched to newly cretaed hduser and configured SSH which included generation of keys and enabling access to local machine. After this we were able to SSH to localhost.

$ su - hduser

$ ssh-keygen -t rsa -P ""

## 

## 

### 3.2 Installing Hadoop on a Single Node

3.2.1 Configuration: After getting Hadoop package and unzipping it, move into etc/hadoop. It contains various XML and \*.sh files, some of which need to be configured. We also configured .bashrc to set some of the environment variables. We configured the following:

hadoop-env.shconfigured JAVA\_HOME environment variable.

core-site.xml configured temporary directory and default file system.

mapred-site.xml configured host and port for map-reduce jobtracker.

hdfs-site.xml configured parameters for replication.

.bashrc set $HADOOP\_HOME and $JAVA\_HOME

3.2.2 Set up Name Node: Formatted the node with hdfs file system for a single name node.

$bin/hdfs namenode -format

3.2.3 Start up Name Node: The various service deamons of the system are started.

$sbin/start-dfs.sh

Services can also be stopped if so required.

$sbin/stop-dfs.sh

The various services of datanode and resource manager of Hadoop system can betested by invoking example jar files provided in share folder before migration to multi-node installation.

### 3.3 Installing Hadoop on a Multi Node Cluster.

Architecture of Cluster

Four nodes were used by us for creating the Hadoop cluster.

Node-1 : - master host – working as Primary name node

Node-2 :- slave1 host - working as Secondary name node , Resource node

Node -3 & Node – 4 :- working as Data nodes.

Hadoop Multinode Installation Steps

In brief the following steps will be performed

1. • Create User

2. • Install and configure keyless SSH login (From Masters to Slaves and between Masters)

3. • Install Java

4. • Setup Environment

5. • Install Hadoop

6. • Setup Hadoop Directories

7. • Set up Configuration Files

8. • Start Hadoop Processes

9. • Test with Word Count

Step – 1 :- User groups created in single node hadoop installation were used in the process.

We used 4 nodes(/VMs) for our purposes.

Aliases for all nodes were created in /etc/hosts/ file .

master <ip-of-master>

slave1 <ip-of-slave>

slave2 <ip-of-slave>

slave3 <ip-of-slave>

Step – 2 :- For easier access to slave nodes, passwordless login was enabled for the master

node.

ssh-copy-id ~/.ssh/id\_rsa.pub hduser@master

where ssh-copy-id - command to copy ssh id , ~/.ssh/id\_rsa.pub – the file where ssh rsa key is

stored , hduser – hadoop user created for installation , master – alias for master node

same is run for all slaves ( in our case 4 slaves were used)

Step – 3 :- Environment variables were set up for installation purposes.

export HADOOP\_HOME=/usr/local/hadoop

export HADOOP\_MAPRED\_HOME=$HADOOP\_HOME

export HADOOP\_COMMON\_HOME=$HADOOP\_HOME

export HADOOP\_HDFS\_HOME=$HADOOP\_HOME

export YARN\_HOME=$HADOOP\_HOME

export HADOOP\_CONF\_DIR=$HADOOP\_HOME/etc/hadoop

export YARN\_CONF\_DIR=$HADOOP\_HOME/etc/hadoop

export JAVA\_HOME=/usr/bin/java

Step – 4 :- Hadoop directories were setup.

On NameNode and Resource Manager

mkdir -p /usr/local/hadoop/custom

mkdir -p /usr/local/hadoop/custom/hdfs/namenode

mkdir -p /usr/local/hadoop/custom/hdfs/namenode2-checkpoint

On DataNode and Node Manager

mkdir -p /usr/local/hadoop/custom

mkdir -p /usr/local/hadoop/custom/hdfs/datanode

mkdir -p /usr/local/hadoop/custom/yarn/nm-local-dir

mkdir -p /usr/local/hadoop/custom/yarn/nm-log-dir/userlogs

Step – 4 :- Hadoop configuration files were initialized to proper values-

This included the following files -

**$HADOOP\_HOME/etc/hadoop/core-site.xml**:

default file system port id is setup here.

**$HADOOP\_HOME/etc/hadoop/hdfs-site.xml:**

dfs file node properties are set here (default directories etc.)

**$HADOOP\_HOME/etc/hadoop/yarn-site.xml:**

Resource manager properties and yarn properties are initialized here.

**$HADOOP\_HOME/etc/hadoop/mapred-site.xml:**

Job scheduler properties (ports, address etc.) and mapreduce properties are set here.

Step – 5 :- These configured files are transferred to the slaves using scp.

cp \*site.xml $HADOOP\_CONF\_DIR/

cp hadoop-config.sh $HADOOP\_HOME/libexec/

cp yarn-env.sh $HADOOP\_CONF\_DIR/

scp \*site.xml slave1:$HADOOP\_CONF\_DIR/

scp hadoop-config.sh slave1:$HADOOP\_HOME/libexec/

(Same thing copied to other slaves too)

Step – 6 :- Slave node names added to slaves file

vi $HADOOP\_CONF\_DIR/slaves

slave1

slave2

slave3

Step – 7:- On resource node (slave 1) hdfs directories were created and permissions modified

$HADOOP\_HOME/bin/hdfs dfs -mkdir -p /mr-history/tmp

$HADOOP\_HOME/bin/hdfs dfs -chmod -R 1777 /mr-history/tmp

$HADOOP\_HOME/bin/hdfs dfs -mkdir -p /mr-history/done

$HADOOP\_HOME/bin/hdfs dfs -chmod -R 1777 /mr-history/done

$HADOOP\_HOME/bin/hdfs dfs -chown -R hadoopuser:hadoopgrp /mr-history

Step - 8 :- Primary NameNode file system was formatted(On master node )

$HADOOP\_HOME/bin/hdfs namenode -format

Step – 9:- Main processes started on various nodes -

Master Node

**Primary Namenode -**

$/usr/local/hadoop/sbin/hadoop-daemon.sh –config $HADOOP\_CONF\_DIR start namenode

Resource Node

**Secondary Namenode** -

$/usr/local/hadoop/sbin/hadoop-daemon.sh –config $HADOOP\_CONF\_DIR start secondarynamenode Resource Manager/

usr/local/hadoop/sbin/yarn-daemon.sh –config $HADOOP\_CONF\_DIR start resourcemanager

**Job History Server -**

/usr/local/hadoop/sbin/yarn-daemon.sh –config $HADOOP\_CONF\_DIR start resourcemanager

Data Nodes

**Data Node-**

$/usr/local/hadoop/sbin/hadoop-daemon.sh –config $HADOOP\_CONF\_DIR start datanode

**Node Manager-**

/usr/local/hadoop/sbin/yarn-daemon.sh –config $HADOOP\_CONF\_DIR start nodemanager

All nodes were checked for the processes running using $jps.

Step – 10 :- Testing Hadoop

We tested by mapreduce app wordcount. Created a test file 'wctest' and uploaded it to HDFS file system.

$HADOOP\_HOME/bin/hdfs dfs -copyFromLocal /var/tmp/test/wctest /wctest

$HADOOP\_HOME/bin/hadoop jar $HADOOP\_HOME/share/hadoop/mapreduce/hadoopmapreduce-

examples-2.4.0.jar wordcount /wctest /result/\*

Step – 11 :- Shutting down Hadoop

HDFS processes -

Master node

/usr/local/hadoop/sbin/hadoop-daemon.sh –config $HADOOP\_CONF\_DIR stop namenode

Resource node

/usr/local/hadoop/sbin/hadoop-daemon.sh –config $HADOOP\_CONF\_DIR stop secondarynamenode

Data node

/usr/local/hadoop/sbin/hadoop-daemon.sh –config $HADOOP\_CONF\_DIR stop datanode

Yarn Processes -

/usr/local/hadoop/sbin/yarn-daemon.sh –config $HADOOP\_CONF\_DIR stop resourcemanager

Job History Server -

usr/local/hadoop/sbin/mr-jobhistory-daemon.sh –config $HADOOP\_CONF\_DIR stop historyserver

References:

1) <http://www.michael-noll.com/tutorials/running-hadoop-on-ubuntu-linux-single-node-cluster/>

2) http://www.michael-noll.com/tutorials/running-hadoop-on-ubuntu-linux-multi-node-cluster/

3) http://docs.openstack.org/developer/devstack/guides/single-vm.html