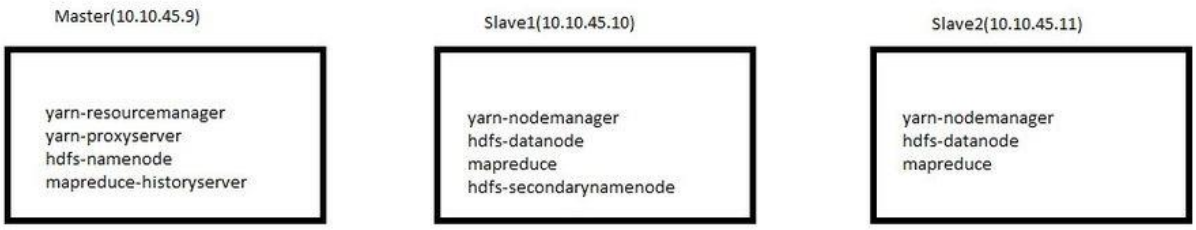
Introduction:

With a physical machine with 20 Gb memory, we can allocate 3 VMs to set up a complete Hadoop cluster with multiple mapReduce nodes and multiple data nodes for HDFS.

After setting up development environment in local machine using single vm, we can set up local testing environment for hadoop with controlled version. During the setting up process, we need to download hadoop. We can download the version that we need to gain version control. Here we use hadoop-2.9.2 as an example.(Also, we use Windows as the underlying environment in the example. It's pretty similar under MAC OS environment.)

On top of the hadoop cluster environment, we can install Oozie, Spark, etc..

The topological distribution is like the following picture:



1 VM Cluster Setup in local machine using vagrant

1.1 on Windows, set up vagrant-proxyconf

a) install rubyinstaller
https://rubyinstaller.org/downloads/

b) download file vagrant-proxyconf-1.5.2.gem
https://rubygems.org/downloads/vagrant-proxyconf-1.5.2.gem

c) install vagrant-proxyconf-1.5.2.gem

```
gem install vagrant-proxyconf-1.5.2.gem
```

d) add the content to $HOME/.vagrant.d/Vagrantfile (or to a project specific Vagrantfile):

```
Vagrant.configure("2") do |config|
```

```
if Vagrant.has_plugin?("vagrant-proxyconf")
config.proxy.http = "http://username:password@http-
yourCompany.com.au:8080/"
config.proxy.https = "http://username:password@http-
yourCompany.com.au:8080/"
config.proxy.no_proxy =
"localhost,192.168.99.100,artifactory.D.corp.yourCompany.com"
end
# ... other stuff
end
```

1.2 set up master, slave1, slave2 with ip addresses

a) Vagrantfile for master

```
vagrantNode = { :host => "masterT", :ip => "10.10.45.9", :box =>
"bento/ubuntu-18.04", :gui => false}


varDomain = "hadoopVagrantTest"

puts "#{vagrantNode[:host]}.#{varDomain}"
puts "#{vagrantNode[:box]}"

Vagrant.configure("2") do |config|

config.vm.box = vagrantNode[:box]
config.vm.network "private_network", ip: vagrantNode[:ip], :netmask =>
"255.255.255.0"
config.vm.hostname = "#{vagrantNode[:host]}.#{varDomain}"

#
config.vm.provider "virtualbox" do |vb|
vb.name = vagrantNode[:host].to_s
vb.gui = vagrantNode[:gui]
vb.memory = "4096"
vb.cpus = 2
end
#

config.vm.provision "shell", inline: <<-SHELL
# Install JDK8
apt update
apt install -y openjdk-8-jdk
apt install -y gradle
# Make gradle to use BUD2 Artifactor
mkdir -p /home/vagrant/.gradle
cat > /home/vagrant/.gradle/gradle.properties << EOF
org.gradle.daemon=true
artifactory_contextUrl=https://artifactory.D.yourCompany.com/artifactory
artifactory_pluginsUrl=https://artifactory.D.yourCompany.com/artifactory/p
lugins-release
artifactory_user=dummy
artifactory_password=dummy
```

```
EOF
SHELL
end
```

b) Vagrantfile for slave1

```
vagrantNode = { :host => "slave1T", :ip => "10.10.45.10", :box =>
"bento/ubuntu-18.04", :gui => false}

varDomain = "hadoopVagrantTest"

puts "#{vagrantNode[:host]}.#{varDomain}"
puts "#{vagrantNode[:box]}"

Vagrant.configure("2") do |config|

config.vm.box = vagrantNode[:box]
config.vm.network "private_network", ip: vagrantNode[:ip], :netmask =>
"255.255.255.0"
config.vm.hostname = "#{vagrantNode[:host]}.#{varDomain}"

#
config.vm.provider "virtualbox" do |vb|
vb.name = vagrantNode[:host].to_s
vb.gui = vagrantNode[:gui]
vb.memory = "6144"
vb.cpus = 2
end
#

config.vm.provision "shell", inline: <<-SHELL
# Install JDK8
apt update
apt install -y openjdk-8-jdk
apt install -y gradle
# Make gradle to use BUD2 Artifactor
mkdir -p /home/vagrant/.gradle
cat > /home/vagrant/.gradle/gradle.properties << EOF
org.gradle.daemon=true
artifactory_contextUrl=https://artifactory.D.corp.yourCompany.com/artifact
ory
artifactory_pluginsUrl=https://artifactory.D.corp.yourCompany.com/artifact
ory/plugins-release
artifactory_user=dummy
artifactory_password=dummy
EOF
SHELL
end
```

c) Vagrantfile for slave2

```
vagrantNode = { :host => "slave2T", :ip => "10.10.45.11", :box =>
"bento/ubuntu-18.04", :gui => false}

varDomain = "hadoopVagrantTest"

puts "#{vagrantNode[:host]}.#{varDomain}"
puts "#{vagrantNode[:box]}"

Vagrant.configure("2") do |config|

config.vm.box = vagrantNode[:box]
config.vm.network "private_network", ip: vagrantNode[:ip], :netmask =>
"255.255.255.0"
config.vm.hostname = "#{vagrantNode[:host]}.#{varDomain}"

#
config.vm.provider "virtualbox" do |vb|
vb.name = vagrantNode[:host].to_s
vb.gui = vagrantNode[:gui]
vb.memory = "6144"
vb.cpus = 2
end
#

config.vm.provision "shell", inline: <<-SHELL
# Install JDK8
apt update
apt install -y openjdk-8-jdk
apt install -y gradle
# Make gradle to use BUD2 Artifactor
mkdir -p /home/vagrant/.gradle
cat > /home/vagrant/.gradle/gradle.properties << EOF
org.gradle.daemon=true
artifactory_contextUrl=https://artifactory.D.corp.yourCompany.com/artifact
ory
artifactory_pluginsUrl=https://artifactory.D.corp.yourCompany.com/artifact
ory/plugins-release
artifactory_user=dummy
artifactory_password=dummy
EOF
SHELL
end
```

Finally, we use `vagrant up` to start these 3 VMs

1.3 generate a ssh key pair and deploy the same key pair to all 3 VMs

so that from master, we can scp configure files to slave1 and slave2 to save time

for master

```
# step 1
ssh-keygen -t rsa
# Press enter for each line

# step2
```

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

# step 3
chmod og-wx ~/.ssh/authorized_keys
```

```
copy the following files from master to slave1 and slave2 in current
user's .ssh folder
```
```
~/.ssh/id_rsa
~/.ssh/id_rsa.pub
```

```
for both slave 1 and slave 2
```sh
cat ~/.ssh/id_rsa.pub >> ~/.ssh/authorized_keys
chmod og-wx ~/.ssh/authorized_keys
```

copy the following files from master to slave1 and slave2 in current user's .ssh folder

```
~/.ssh/id_rsa
~/.ssh/id_rsa.pub
```

for both slave 1 and slave 2

```
cat ~/.ssh/id_rsa.pub >> ~/.ssh/authorized_keys
chmod og-wx ~/.ssh/authorized_keys
```

Edit /etc/hosts files for all 3 VMs(master, slave1 and slave2)

```
sudo nano /etc/hosts
```

add lines and delete lines for /etc/hosts (master, slave1 and slave2)

```
add the following lines:
```
```
10.10.45.9 masterT
10.10.45.9 10.10.45.9
10.10.45.9 masterT.hadoopVagrantTest
10.10.45.10 slave1T
10.10.45.10 10.10.45.10
10.10.45.10 slave1T.hadoopVagrantTest
10.10.45.11 slave2T
10.10.45.11 10.10.45.11
10.10.45.11 slave2T.hadoopVagrantTest
```

```
delete the 127.0.1.1(or 127.0.0.1) entry that points to your host name
```
```

```
127.0.0.1 localhost
127.0.1.1 masterT.hadoopVagrantTest masterT
```

add JAVA_HOME to /etc/environment

```
# add the following line to  /etc/environment by `sudo nano
/etc/environment`
JAVA_HOME = /usr/lib/jvm/java-8-openjdk-amd64/jre
```

2 install Hadoop(version control)

2.1 For each VM(master, slave1 and slave2), install hadoop

We can install download any version of hadoop. Here, we use hadoop-2.9.2 as an example.

a) unzip hadoop and add users

```
# Under Windows, we can put hadoop-2.9.2.tar.gz in the same directory as
Vagrantfile.
sudo mv hadoop-2.9.2.tar.gz /etc
cd /etc
sudo tar -zxvf hadoop-2.9.2.tar.gz
sudo rm hadoop-2.9.2.tar.gz
sudo mv hadoop-2.9.2 hadoop


sudo adduser hdfs
# the passwd also use hdfs
sudo adduser yarn
# the passwd also use yarn
sudo adduser mapred
# the passwd also use mapred
sudo usermod -aG sudo hdfs
sudo usermod -aG sudo yarn
sudo usermod -aG sudo mapred
sudo usermod -aG sudo vagrant

# create group hadoop and add vagrant to this group
sudo groupadd hadoop
sudo usermod -a -G hadoop vagrant
```

b) add environment vars

.bashrc

```
add the following lines to .bashrc by `sudo nano ~/.bashrc`

export JAVA_HOME=/usr/lib/jvm/java-8-openjdk-amd64/jre
```

```
export HADOOP_HOME=/etc/hadoop
export HADOOP_YARN_HOME=$HADOOP_HOME
export HADOOP_HDFS_HOME=$HADOOP_HOME
export HADOOP_COMMON_HOME=$HADOOP_HOME
export HADOOP_MAPRED_HOME=$HADOOP_HOME
export HADOOP_CONF_DIR=$HADOOP_HOME/etc/hadoop
export PATH=$PATH:/etc/hadoop/bin/
```

hadoop-env.sh

```
# add the line to the file hadoop-env.sh by `nano
/etc/hadoop/etc/hadoop/hadoop-env.sh`

export JAVA_HOME=/usr/lib/jvm/java-8-openjdk-amd64/jre
```

reload environment vars for bash

```
source ~/.bashrc
```

2.2 set up hadoop configuration files on master

During this step, we first configre files on master VM, then use scp copy these configuration files to slave1 and slave2

core-site.xml(master VM)

```
# add the line to the file core-site.xml by `sudo nano
/etc/hadoop/etc/hadoop/core-site.xml`
<property>
<name>fs.defaultFS</name>
<value>hdfs://10.10.45.9:8020</value>
</property>
<property>
<name>io.compression.codecs</name>
<value>org.apache.hadoop.io.compress.DefaultCodec,org.apache.hadoop.io.com
press.GzipCodec,org.apache.hadoop.io.compress.BZip2Codec,org.apache.hadoop
.io.compress.SnappyCodec</value>
</property>
<property>
<name>hadoop.proxyuser.mapred.groups</name>
<value>*</value>
</property>
<property>
<name>hadoop.proxyuser.mapred.hosts</name>
<value>*</value>
</property>
```

core-site.xml(master VM)

```
# add the line to the file core-site.xml by `sudo nano
/etc/hadoop/etc/hadoop/core-site.xml`

<property>
<name>fs.defaultFS</name>
<value>hdfs://10.10.45.9:8020</value>
</property>
<property>
<name>io.compression.codecs</name>
<value>org.apache.hadoop.io.compress.DefaultCodec,org.apache.hadoop.io.com
press.GzipCodec,org.apache.hadoop.io.compress.BZip2Codec,org.apache.hadoop
.io.compress.SnappyCodec</value>
</property>
<property>
<name>hadoop.proxyuser.mapred.groups</name>
<value>*</value>
</property>
<property>
<name>hadoop.proxyuser.mapred.hosts</name>
<value>*</value>
</property>
```

hdfs-site.xml(master VM)

```
# add the line to the file hdfs-site.xml by `sudo nano
/etc/hadoop/etc/hadoop/hdfs-site.xml`

<property>
<name>dfs.permissions.superusergroup</name>
<value>hadoop</value>
</property>
<property>
<name>dfs.namenode.name.dir</name>
<value>file:///data/1/dfs/nn</value>
</property>
<property>
<name>dfs.datanode.data.dir</name>
<value>file:///data/1/dfs/dn</value>
</property>
<property>
<name>dfs.namenode.http-address</name>
<value>10.10.45.9:50070</value>
<description>
The address and the base port on which the dfs NameNode Web UI will
listen.
</description>
</property>
<property>
<name>dfs.webhdfs.enabled</name>
<value>true</value>
</property>
<property>
<name>dfs.namenode.datanode.registration.ip-hostname-check</name>
```

```
<value>false</value>
</property>
```

mapred-site.xml(master VM)

```
# add the line to the file mapred-site.xml by `sudo mv mapred-
site.xml.template mapred-site.xml`;`sudo nano
/etc/hadoop/etc/hadoop/mapred-site.xml`

<property>
<name>mapreduce.framework.name</name>
<value>yarn</value>
</property>
<property>
<name>mapreduce.jobhistory.address</name>
<value>10.10.45.9:10020</value>
</property>
<property>
<name>mapreduce.jobhistory.webapp.address</name>
<value>10.10.45.9:19888</value>
</property>
<property>
<name>yarn.app.mapreduce.am.staging-dir</name>
<value>/user</value>
</property>
<property>
<name>mapreduce.application.classpath</name>
<value>$HADOOP_HOME/share/hadoop/mapreduce/*,$HADOOP_HOME/share/hadoop/map
reduce/lib/*,$HADOOP_HOME/share/hadoop/common/*,$HADOOP_HOME/share/hadoop/
common/lib/*,$HADOOP_HOME/share/hadoop/yarn/*,$HADOOP_HOME/share/hadoop/ya
rn/lib/*,$HADOOP_HOME/share/hadoop/hdfs/*,$HADOOP_HOME/share/hadoop/hdfs/l
ib/*</value>
</property>
```

yarn-site.xml(master VM)

```
# add the line to the file yarn-site.xml by `sudo nano
/etc/hadoop/etc/hadoop/yarn-site.xml`

<property>
<name>yarn.resourcemanager.hostname</name>
<value>10.10.45.9</value>
</property>
<property>
<name>yarn.resourcemanager.resource-tracker.address</name>
<value>10.10.45.9:8031</value>
</property>
<property>
<name>yarn.resourcemanager.address</name>
<value>10.10.45.9:8032</value>
</property>
<property>
<name>yarn.resourcemanager.scheduler.address</name>
<value>10.10.45.9:8030</value>
</property>
```

```
<property>
<name>yarn.resourcemanager.admin.address</name>
<value>10.10.45.9:8033</value>
</property>
<property>
<name>yarn.resourcemanager.webapp.address</name>
<value>10.10.45.9:8088</value>
</property>
<property>
<description>Classpath for typical applications.</description>
<name>yarn.application.classpath</name>
<value>
$HADOOP_CONF_DIR,
$HADOOP_COMMON_HOME/*,$HADOOP_COMMON_HOME/lib/*,
$HADOOP_HDFS_HOME/*,$HADOOP_HDFS_HOME/lib/*,
$HADOOP_MAPRED_HOME/*,$HADOOP_MAPRED_HOME/lib/*,
$HADOOP_YARN_HOME/*,$HADOOP_YARN_HOME/lib/*
</value>
</property>
<property>
<name>yarn.nodemanager.aux-services</name>
<value>mapreduce_shuffle</value>
</property>
<property>
<name>yarn.nodemanager.aux-services.mapreduce_shuffle.class</name>
<value>org.apache.hadoop.mapred.ShuffleHandler</value>
</property>
<property>
<name>yarn.nodemanager.local-dirs</name>
<value>file:///data/1/yarn/local</value>
</property>
<property>
<name>yarn.nodemanager.log-dirs</name>
<value>file:///data/1/yarn/logs</value>
</property>
<property>
<name>yarn.log.aggregation.enable</name>
<value>true</value>
</property>
<property>
<description>Where to aggregate logs</description>
<name>yarn.nodemanager.remote-app-log-dir</name>
<value>hdfs://10.10.45.9:8020/var/log/hadoop-yarn/apps</value>
</property>
<property>
<name>yarn.nodemanager.resource.memory-mb</name>
<value>3072</value>
</property>
<property>
<name>yarn.scheduler.minimum-allocation-mb</name>
<value>2048</value>
</property>
<property>
<name>yarn.nodemanager.vmem-pmem-ratio</name>
<value>2.1</value>
</property>
<property>
<name>yarn.web-proxy.address</name>
<value>10.10.45.9:9046</value>
</property>
```

masters(master VM)(this file contains the addr of secondary name node)

```
# add the line to the file masters by `sudo nano
/etc/hadoop/etc/hadoop/masters`

10.10.45.10
```

2.3 use scp to copy hadoop configuration files from master to slave1 and slave2

On both Slave1 and Slave2

```
sudo chmod 777 -R /etc/hadoop/etc/hadoop
```

On master

```
scp -r /etc/hadoop/etc/hadoop/* 10.10.45.10:/etc/hadoop/etc/hadoop
scp -r /etc/hadoop/etc/hadoop/* 10.10.45.11:/etc/hadoop/etc/hadoop
```

On Slave1 and Slave2 (after copying, revoke write right)

```
sudo chmod 755 -R /etc/hadoop
On Slave1 and Slave2 (after copying, revoke write right)
```

2.4 create folders for name node and data node

On Master

```
sudo mkdir -p /data/1/dfs/nn
sudo chown -R hdfs:hdfs /data/1/dfs/nn
sudo chmod 777 /data/1/dfs/nn

sudo mkdir -p /data/1/yarn/logs
sudo chown -R hdfs:hdfs /data/1/yarn/logs
sudo chmod 777 /data/1/yarn/logs

sudo mkdir -p /data/1/yarn/local
sudo chown -R hdfs:hdfs /data/1/yarn/local
sudo chmod 777 /data/1/yarn/local
```

```
sudo mkdir -p /etc/hadoop/logs
sudo chown -R hdfs:hdfs /etc/hadoop/logs
sudo chmod 777 /etc/hadoop/logs

sudo mkdir -p /var/log/hadoop-yarn/apps
sudo chown -R yarn:yarn /var/log/hadoop-yarn/apps
```

On Slave1 and Slave2

```
sudo mkdir -p /data/1/dfs/dn
sudo chown -R vagrant:hadoop /data/1/dfs/dn
sudo chmod 777 /data/1/dfs/dn

sudo mkdir -p /etc/hadoop/logs/
sudo chown -R vagrant:hadoop /etc/hadoop/logs/
sudo chmod 777 /etc/hadoop/logs/

sudo mkdir -p /data/1/yarn/local
sudo chown -R vagrant:hadoop /data/1/yarn/local
sudo chmod 777 /data/1/yarn/local

sudo mkdir -p /data/1/yarn/logs
sudo chown -R vagrant:hadoop /data/1/yarn/logs
sudo chmod 777 /data/1/yarn/logs

sudo mkdir -p /var/log/hadoop-yarn/apps
sudo chown -R yarn:yarn /var/log/hadoop-yarn/apps
```

On Master(format hdfs)

```
sudo -u hdfs /etc/hadoop/bin/hdfs namenode -format
```

on hdfs

```
# Create /tmp dir -- on hdfs

sudo -u hdfs hadoop fs -mkdir /tmp
sudo -u hdfs hadoop fs -chmod -R 1777 /tmp


# Create user directory --Only for the first time
sudo -u hdfs hadoop fs -mkdir /user
sudo -u hdfs hadoop fs -mkdir /user/vagrant
sudo -u hdfs hadoop fs -chown vagrant:vagrant /user/vagrant

# For Job History -- On HDFS
sudo -u hdfs hadoop fs -mkdir -p /user/history
sudo -u hdfs hadoop fs -chmod -R 1777 /user/history
sudo -u hdfs hadoop fs -chown mapred:hadoop /user/history

# For YARN logs -- On HDFS
sudo -u hdfs hadoop fs -mkdir -p /var/log/hadoop-yarn
```

```
sudo -u hdfs hadoop fs -mkdir -p /var/log/hadoop-yarn/apps

sudo -u hdfs hadoop fs -chown yarn:mapred /var/log/hadoop-yarn
sudo -u hdfs hadoop fs -chown yarn:mapred /var/log/hadoop-yarn/apps
```

2.5 Start HDFS and other services(name node, secondarynamenode, 2 datanodes, 2 mapReduce nodes)

On Slave1

```
/etc/hadoop/sbin/hadoop-daemon.sh start secondarynamenode
/etc/hadoop/sbin/yarn-daemon.sh start nodemanager
/etc/hadoop/sbin/hadoop-daemon.sh start datanode
jps
```

On Slave2

```
/etc/hadoop/sbin/yarn-daemon.sh start nodemanager
/etc/hadoop/sbin/hadoop-daemon.sh start datanode
jps
```

On Master

```
/etc/hadoop/sbin/start-yarn.sh
/etc/hadoop/sbin/yarn-daemon.sh stop resourcemanager
/etc/hadoop/sbin/yarn-daemon.sh start resourcemanager
sudo /etc/hadoop/sbin/hadoop-daemon.sh start namenode
/etc/hadoop/sbin/mr-jobhistory-daemon.sh start historyserver
/etc/hadoop/sbin/yarn-daemon.sh start proxyserver
jps
```

then navigate to the following addr:

```
# name node
http://10.10.45.9:50070

# application
http://10.10.45.9:8088

# history server
http://10.10.45.9:19888
```

3 run a simple Hadoop MapReduce job on the Hadoop cluster

```
sudo -u vagrant hadoop fs -mkdir -p input
sudo -u vagrant hadoop fs -ls /user/vagrant
# then we see the directory /user/hdfs/input

# download sample map-reduce source files
wget http://salsahpc.indiana.edu/tutorial/source_code/Hadoop-WordCount.zip
unzip Hadoop-WordCount.zip
sudo -u vagrant hadoop fs -put Hadoop-WordCount/input/
/user/vagrant/input/
sudo -u vagrant hadoop fs -ls /user/vagrant/input/input
sudo -u vagrant hadoop fs -cat
/user/vagrant/input/input/Word_Count_input.txt

# remove the former result if there is any
sudo -u vagrant hadoop fs -rm -r output/mapreduce/WordCount
#
sudo -u vagrant hadoop jar Hadoop-WordCount/wordcount.jar WordCount
/user/vagrant/input/input output/mapreduce/WordCount
# see the result
sudo -u vagrant hadoop fs -ls output/mapreduce/WordCount
sudo -u vagrant hadoop fs -cat output/mapreduce/WordCount/part-r-00000
```

4 debug tips

4.1 name node or data node bugs

error msg:

```
java.io.IOException: Incompatible clusterIDs in /data/1/dfs/dn: namenode
clusterID = CID-de0c0a97-a3f1-4247-9d0e-9ce3146150d3; datanode clusterID =
CID-3b3b4a71-d013-4ad9-a80d-46829e976fde
```

solve method:

```
# on master
sudo rm -r /data/1/dfs/nn/*
sudo chown -R hdfs:hdfs /data/1/dfs/nn/current/
sudo chown -R hdfs:hdfs /data/1/dfs/nn/current/VERSION
sudo chown -R hdfs:hdfs /data/1/dfs/nn/*
sudo chown -R hdfs:hdfs /data/1/dfs
```

```
sudo chmod 777 /data/1/dfs/nn/
sudo chmod 777 /data/1/dfs/nn/current/
sudo chmod 777 /data/1/dfs/nn/current/VERSION
sudo chmod 777 /data/1/dfs

sudo chown hdfs:hdfs -R /data/1/dfs/nn/current/

# on slaves
sudo rm -r /data/1/dfs/dn/*

# on master
sudo rm -r /data/1/dfs/nn/*

# on slaves
sudo rm -r /data/1/dfs/dn/*

# on master
sudo -u hdfs /etc/hadoop/bin/hdfs namenode -format

hdfs dfsadmin -report
```

4.2 see logs

see logs

```
/etc/hadoop/logs
```