CREAR UN CLUSTER HADOOP

Instalarse VirtualBox y CentOS (version 7.9.2009 x86-64)

Crear MV Linux

. Nombre: nodo1
. Version: Red Hat 64-bit
. RAM: 4gb
. Disco: Crear disco virtual VDI de 60gb (reservado dinamicamente)
. Despues cambiar memoria de video de 16 MB a 32 MB en config
. Para meterle el ISO: configuracion – almacenamiento – controlador: IDE – Unidad Optica: pinchar dibujo – seleccionar un archivo de disco optico virtual

Crear CentOS

Seleccionar Idioma español – Dispositivos de almacenamientos basicos – nombre del host: nodo1 – gnome gui – crear usuario en la instalacion y ponerle como administrador

Guest additions

- sudo yum -y install kernel-header kernel-devel gcc
- sudo yum update kernel
- reiniciamos
- Dispositivos -> insertar imagen de las Guest Additions

Hadoop, java…

- Descargar hadoop en hadoop.apache.org (3.2.2) .binary
- meter la carpeta en /opt
- darle permisos: sudo chown hadoop /opt/hadoop-3.2.2

PARA JAVA:
- su - root
- descargar rpm x64 de 8u331
- ejecutar rpm -ivh jdk-8u331-linux-x64.rpm
- su - root y alternatives – config java (tiene que aparecer la que hemos instalado) darle al 3
- javac para comprobar

Variables de entorno

- en usuario hadoop ir a /home/hadoop y hacer vi .bashrc ahi introducir:
  export HADOOP_HOME=/opt/hadoop
  export PATH=$PATH:$HADOOP_HOME/bin:$HADOOP_HOME/sbin

- para actualizar (. ./.bashrc)
- alternatives --config java
- abrir .bashrc e introducir lo siguiente antes del PATH:
- export JAVA_HOME=/usr/java/jdk1.8.0_331-amd64 (hasta antes del bin)

  Para los hosts

  ifconfig (suponiendo que estemos conectados) nos mostrará nuestra IP en la interfaz en la que estamos conectados.

  - gedit /etc/hosts, añadamos la línea <IP> nodo1: 10.0.2.15 nodo1.

Ejemplo de uso de hadoop

Para ir abriendo boca, nos dirigimos a /opt/hadoop/share/hadoop/mapreduce/, y vamos a pasar los archivos xml a una carpeta temporal:

- mkdir /tmp/entrada
- cp /opt/hadoop/etc/hadoop/*.xml /tmp/entrada
- hadoop jar hadoop-mapreduce-examples-3.3.2.jar grep /tmp/entrada /tmp/salida 'kms[a-z]+'

En /tmp/salida aparecen los sospechosos habituales: _SUCCESS y part-r-00000

SSH

- ssh-keygen en la carpeta raíz del usuario (/home/hadoop).
- cd .ssh
- cp id_rsa.pub authorized_keys.
- ssh nodo1 para conectarnos a nosotros mismos y añadirnos a la lista de known_hosts.

FICHEROS XML

- Ir a /opt/hadoop-3.2.2/etc/hadoop

vi core-site.xml y añadir dentro de configuration

```
<configuration>

<property>
```

```
  <name>fs.defaultFS</name>
    <value>hdfs://nodo1:9000</value>
    </property>
 </configuration>
```

vi hdfs-site.xml y añadir dentro de configuration

```
<configuration>
    <property>
        <name>dfs.replication</name>
        <value>1</value>
    </property>
    <property>
        <name>dfs.namenode.name.dir</name>
        <value>/datos/namenode</value>
    </property>
    <property>
        <name>dfs.datanode.data.dir</name>
        <value>/datos/datanode</value>
    </property>
<configuration>
```

Formateo

vamos a root con su - root

- cd /
- sudo mkdir /datos
- sudo mkdir /datos/datanode
- sudo mkdir /datos/namenode
- sudo chown -R hadoop:hadoop datos

formatear con hdfs namenode -format,

iniciar hdfs con start-dfs.sh

vamos a /opt/hadoop-3.2.3/hadoop/sbin

jps para ver los procesos java

FICHEROS HDFS: -

crear directorio: hdfs dfs -mkdir /temporal1

copiar archivo: hdfs dfs -cp /temporal/prueba.txt /temporal1/prueba.txt

pasarlo del hdfs al local: hdfs dfs -get /temporal1/prueba.txt /home/hadoop/test.txt

pasarlo del local al hdfs: hdfs dfs -put /home/hadoop/test.txt /temporal1/prueba.txt

Snapshots

informe: hdfs dfsadmin -report

estado actual del sistema de ficheros hdfs fsck / (con opciones como -files, -blocks y -locations)

topologia de nodos: hdfs dfsadmin -printTopology

ficheros: hdfs dfsadmin -listOpenFiles

Proceso para hacer una snapshot

tenemos f1.txt

hdfs dfs -put f1.txt /datos

hdfs dfsadmin -allowSnapshot /datos

hdfs dfs -createSnapshot /datos snap1

Si borramos f1 se podra recuperar con un ctrl+c ctrl+v

Ejemplo snapshots

1 - echo Ejemplo de Snapshot > /tmp/f1.txt

2 - hdfs dfs -mkdir /datos4

3 - hdfs dfs -put /tmp/f1.txt /datos4

4 - hdfs dfsadmin -allowSnapshot /datos4

5 - hdfs dfs -createSnapshot /datos4 s1

6 - hdfs dfs -rm /datos4/f1.txt

7 - hdfs dfs -cp /datos4/.snapshot/s1/f1.txt /datos4/ -- Este es el método.

YARN

Modificar el archivo mapred-site.xml (/otp/hadoop-3.2.3/etc/hadoop)

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

- Copiar el resultado de hadoop classpath

Modificar el archivo yarn-site.xml

```
<configuration>
    <property>
        <name>yarn.resourcemanager.hostname</name>
        <value>nodo1</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.application.classpath</name>
        <value>
            LO COPIADO DE HADOOP CLASSPATH
        </value>
    </property>
</configuration>
```

Despues de esto ejecutar start-dfs.sh y start-yarn.sh

- Interfaz hadoop en nodo1:8088
- Interfaz hdfs en nodo1:9870

ejecutar el comando mapred --daemon start historyserver para ver el historial

- Resumiendo, a partir de ahora cada vez que iniciemos: start-dfs.sh, start-yarn.sh, mapred --daemon start historyserver.
- Y para pararlo: stop-dfs.sh, stop-yarn.sh, mapred --daemon stop historyserver.

CLONAR NODO

- Cambiar la red. Añadir un adaptador y ponerle red interna (intnet)
- No marcar nada menos en MAC: generar nuevas direcciones

CONFIGURAR RED EN LOS NODOS

- Cambiar el nombre de la maquina: vamos a /etc/hostname y ponemos el nodo en cuestión
- network: vamos a /etc/sysconfig/network y escribimos lo siguiente:

```
NETWORKING=yes
HOSTNAME=nodo2
```

- En /etc/hosts tendremos que tener lo siguiente (en todos lo mismo):

```
127.0.0.1     localhost localhost.localdomain localhost4
localhost4.localdomain4

::1           localhost localhost.localdomain localhost6
localhost6.localdomain6

192.168.0.101 nodo1

192.168.0.102 nodo2

192.168.0.103 nodo3
```

Tambien ir red y darle una ip, mascara de red (255.255.255.0) y puerta de enlace (192.168.0.1 en el caso de nodo 1)

Comprobar que funciona la red con ping y la ip

- SSH
    1. para cada nodo hay que eliminar todo el contenido de /home/hadoop/.ssh
    2. conectarse a cada nodo desde el 1 y hacer ssh-keygen
    3. En el nodo1 Ejecutar cp id_rsa.pub authorized_keys
    4. Ejecutar scp authorized_keys nodo2:/home/hadoop/.ssh
    5. En el nodo2 hacer cat id_rsa.pub >> authorized_keys para que se añada al final del fichero. Esto lo pasamos al nodo3 con el mismo comando de arriba, y este authorized_keys se pasa tanto al 1 como al 2, ya que es el final.
    6. en cada nodo hacer chmod 600 authorized_keys

- Configurar nodos:
    - Borrar /datos/namenode y /datos/datanode/current en los nodos 2 y 3
    - Borrar /datos/datanode en nodo1
    - en hdfs-site.xml cambiar dfs.replication de 1 a 2 (Porque ahora tenemos + de un nodo)
    - Se puede pasar a los otros nodos con scp hdfs-site.xml /nodo2:/opt/hadoop-3.2.3/etc/hadoop/hdfs-site.xml
    - en workers poner los nombres de los otros nodos:
      nodo2
      nodo3

- Desactivar cortafuegos: En los 3 nodos
    1. sudo systemctl stop firewalld
    2. sudo systemctl disable firewalld
    3. hdfs namenode -format

Ejemplo:

- hdfs dfs -mkdir /practicas
- hdfs dfs -put cite75_99.txt /practicas/cite75_99.txt
- crear MyJob.java con el contenido del pdf
- export HADOOP_CLASSPATH=usr/java/jdk1.8.0_331-amd64/lib/tools.jar
- hadoop com.sun.tools.javac.Main MyJob.java
- jar cvf MyJob.jar My*
- hadoop jar MyJob.jar MyJob /practicas/cite75_99.txt /resultado4
-

GESTION DEL CLUSTER

- yarn application: muestra las aplicaciones activas.
- yarn container: muestra los contenedores que han ejecutado las aplicaciones.
- yarn node: muestra información de los nodos con los que estamos trabajando.
- yarn applicationattempt: muestra los intentos de ejecución de cada aplicación.

- Cada una tiene una serie de opciones muy interesantes. Por ejemplo, -list, que muestra una lista.
- Mencionar que solo se mostrarán datos activos; es decir, aplicaciones que se están ejecutando en el momento concreto en que se usa el comando.
- Matar a un job con yarn application -kill <id>

YARN SCHEDULER

- `mapred queue -list` muestra las colas del scheduler
- modificar capacity-scheduler.xml

```
<property>
    <name>yarn.scheduler.capacity.root.queues</name>
    <value>default,prod,desa</value>
    <description>
        lo que ponga aquí que no nos interesa.
    </description>
</property>
```

- Ejemplo con varias colas:

```
 <property>
    <name>yarn.scheduler.capacity.root.queues</name>
    <value>default,prod,desa</value>
    <description>
      The queues at the this level (root is the root
queue).
    </description>
  </property>

  <property>

<name>yarn.scheduler.capacity.root.default.capacity</name>
    <value>30</value>
    <description>Default queue target
capacity.</description>
  </property>

  <property>
    <name>yarn.scheduler.capacity.root.prod.capacity</name>
    <value>50</value>
    <description>Default queue target
capacity.</description>
  </property>

  <property>
    <name>yarn.scheduler.capacity.root.desa.capacity</name>
    <value>20</value>
```

```
      <description>Default queue target
    capacity.</description>
    </property>
```

- Para actualizar las colas `yarn rmadmin -refreshQueues`.

Subcolas

ejemplo de subcolas

```
<property>

      <name>yarn.scheduler.capacity.root.queues</name>
      <value>default,prod,desa</value>
      <description>
        The queues at the this level (root is the root
    queue).
      </description>
    </property>

    <property>
      <name>yarn.scheduler.capacity.root.prod.queues</name>
      <value>warehouse,batch</value>
      <description>
        The queues at the this level (root is the root
    queue).
      </description>
    </property>

    <property>

    <name>yarn.scheduler.capacity.root.default.capacity</name>
      <value>30</value>
      <description>Default queue target
    capacity.</description>
    </property>

    <property>
      <name>yarn.scheduler.capacity.root.prod.capacity</name>
      <value>50</value>
      <description>Default queue target
    capacity.</description>
    </property>

    <property>
```

```
<name>yarn.scheduler.capacity.root.prod.warehouse.capacity<
/name>
    <value>40</value>
    <description>Default queue target
capacity.</description>
  </property>

  <property>

<name>yarn.scheduler.capacity.root.prod.batch.capacity</nam
e>
    <value>60</value>
    <description>Default queue target
capacity.</description>
  </property>

  <property>
    <name>yarn.scheduler.capacity.root.desa.capacity</name>
    <value>20</value>
    <description>Default queue target
capacity.</description>
  </property>
```

Debido a que hay colas que vamos a actualizar y que funcionan, en vez de refrescar las colas hay que reiniciar el cluster

para probar las maquinas hadoop jar /opt/hadoop/share/hadoop/mapreduce/hadoop-mapreduce-examples-3.3.2.jar wordcount -Dmapred.job.queue.name=warehouse /practicas/cite75_99.txt /salida10

Ejemplo

```
<property>
    <name>yarn.scheduler.capacity.root.queues</name>
    <value>default,rrhh,marketing,ventas</value>
    <description>Lo que ponga aquí.</description>
</property>

<property>

<name>yarn.scheduler.capacity.root.default.capacity</name>
    <value>10</value>
    <description>Capacidad de la cola default</description>
</property>

<property>
```

```
    <name>yarn.scheduler.capacity.root.rrhh.capacity</name>
    <value>50</value>
    <description>Capacidad de la cola rrhh</description>
</property>

<property>

<name>yarn.scheduler.capacity.root.marketing.capacity</name
>
    <value>20</value>
    <description>Capacidad de la cola
marketing</description>
</property>

<property>

<name>yarn.scheduler.capacity.root.ventas.capacity</name>
    <value>20</value>
    <description>Capacidad de la cola ventas</description>
</property>
```

HIVE

Para descargarlo, vamos a apache.mirror.iph.net/hive y descargamos el tar.gz
Para descomprimirlo vamos a /opt/hadoop y hacemos:

- tar xvf /home/hadoop/Descargas/apache-hive-3.1.3.bin.tar.gz
- en el.bashrc, añadimos `export HIVE_HOME=/opt/hadoop/hive`
- y en el path $HIVE_HOME/bin
- Vamos a /opt/hadoop-3.2.3/hive/conf
- Copiar hive-site.xml quitando el template y añadir:

```
<property>

    <name>system:java.io.tmpdir</name>

    <value>/tmp/hive/java</value>

  </property>

  <property>

    <name>system:user.name</name>

    <value>${user.name}</value>

  </property>
```

- Lo mismo con hive-env.sh

```
export HADOOP_HOME=/opt/hadoop
```

```
export HIVE_CONF_DIR=/opt/hadoop/hive/conf
```

Quitar tambien los templates de los siguientes archivos:

```
cp hive-exec-log4j2.properties.template
hive-exec-log4j2.properties

cp hive-log4j2.properties.template hive-log4j2.properties

cp beeline-log4j2.properties.template
beeline-log4j2.properties
```

HDFS

`hdfs dfs -mkdir /tmp`, y darle los permisos `hdfs dfs -chmod g+w /tmp`.

`hdfs dfs -mkdir -p /user/hive/warehouse`, y darle los permisos `hdfs dfs -chmod g+w /user/hive/warehouse`

Ahora Creamos una carpeta /opt/hadoop/hive/bbdd, y dentro lanzamos el comando schematool -dbType derby -initSchema, que inicializa el schema en la carpeta correspondiente de la base de datos por defecto.

No se por que, pero aqui me han dado errrores que no le habia dado a nadie, asi que pongo aqui la solucion

he tenido una cadena de errores bastante interesante al intentar ejecutar el comando "hive --service schematool -dbType derby -initSchema"

Lo paso por aqui por si a alguien le ocurre alguno de estos errores

- 1º error: No encontraba el comando. Eso es porque habia que actualizar las variables de entorno con . ./.bashrc
- 2º error: Classpath contains multiple SLF4J bindings. Para solucionar esto he tenido que borrar el archivo log4j-slf4j-impl-2.17.1.jar de la carpeta lib
- 3º error: Exception in thread "main" java.lang.NoSuchMethodError: com.google.common.base.Preconfitions.checkArgument(ZLjava/lang/String;Ljava/lang/Object;)V Para esto lo que he hecho ha sido borrar el archivo guava-19.0.jar de la carpeta lib de hive y he copiado el de hadoop a donde estaba el de hive con este comando: "cp /opt/hadoop-3.2.3/share/hadoop/hdfs/lib/guava-27.0-jre.jar /opt/hadoop-3.2.3/hive/lib/guava-27.0-jre.jar"
- 4º error: Exception in thread "main" java.lang.RuntimeException: com.ctc.wstx.exc.WstxParsingException: Illegal character entity: expansion character (code 0x8 at [row,col,system-id]: [3225,96,"file:/opt/hadoop-3.2.3/hive/conf/hive-site.xml"]. Para arreglarlo hay que ir al hive-site.xml y borrar ese caracter de la linea 3225

Prueba de hive

Esto ya lo conocia, pero veremos algunos comandos basicos (casi como sql) que se usan en hive

```
- create database ejemplo;
- use ejemplo;
- create table if not exists t1 (name string);
- insert into t1 values ('mi nombre');
- create table t2 (codigo integer);
- insert into t2 values(10);
```

1. Tablas internas

• Comprobar si hay bases de datos

show databases;

• Nos conectamos a la Base de Datos de ejemplo

use ejemplo;

• Crear las siguientes tablas

CREATE TABLE IF NOT EXISTS empleados_internal

(

name string,

work_place ARRAY<string>,

sex_age STRUCT<sex:string,age:int>,

skills_score MAP<string,int>,

depart_title MAP<STRING,ARRAY<STRING>>

)

COMMENT 'This is an internal table'

ROW FORMAT DELIMITED

FIELDS TERMINATED BY '|'

COLLECTION ITEMS TERMINATED BY ','

MAP KEYS TERMINATED BY ':'

• Lo cargamos con los datos del fichero empleados.txt que teneis en los recursos del curso.

LOAD DATA LOCAL INPATH '/home/curso/Downloads/empleados.txt'

OVERWRITE INTO TABLE empleados_internal;

• Comprobar que existe en el directorio warehouse de HIVE, dentro de la base de datos ejemplo. También lo podemos ver con HDFS

hdfs dfs -ls /user/hive/warehouse/ejemplo.db

2. Tablas externas

• Creamos ahora una tabla externa. Hemos de asegurarnos de que tenemos el directorio /ejemplo, ya que es donde se van a quedar los datos.

CREATE EXTERNAL TABLE IF NOT EXISTS empleados_external

(name string, work_place ARRAY<string>, sex_age STRUCT<sex:string,age:int>, skills_score MAP<string,int>, depart_title MAP<STRING,ARRAY<STRING>>)

COMMENT 'This is an external table'

ROW FORMAT DELIMITED

FIELDS TERMINATED BY '|'

COLLECTION ITEMS TERMINATED BY ','

MAP KEYS TERMINATED BY ':'

LOCATION '/ejemplo/empleados;

• Lo cargamos con los mismos datos

jdbc:hive2://localhost:10000> LOAD DATA LOCAL INPATH '/home/curso/Desktop/empleados.txt' OVERWRITE INTO TABLE empleados_external;

• Probamos que estén las filas

jdbc:hive2://localhost:10000> select * from empleados_external;

• Comprobar que existen el directorio datos

• Hacer alguna SELECT por ejemplo para buscar al empleado “Lucy”

• Borrar la dos tablas

• Comprobar que ha borrado la interna, pero los datos de la externa permanecen.

Beeline y hiveServer2

ir a hive-site.xml y modificar hive.server2.enable.doAs a “false”

arrancamos hiverserver desde el directorio bbdd con hiverserver2 &

nos conctamos a beeline con !connect jdb:hive2://localhost:10000

NO FUNCIONA

HUE

Para descargarlo descargamos la version 4.1 en gethue.com/hue-4-1-is-out y descargamos el tgz

Descomprimirlo con tar xvf hue-4.1.0

Hay que ejecutar los siguientes comandos:

yum install libffi-devel

yum install gmp-devel

yum install python-devel mysql-devel

yum install ant gcc gcc-c++ rsync krb5-devel mysql openssl-devel cyrus-sasl-devel cyrus-sasl-gssapi sqlite-devel openldap-devel python-simplejson

yum install libtidy libxml2-devel libxslt-devel

yum install python-devel python-simplejson python-setuptools

yum install maven

Para compilar todo, hay que ejecutar el siguiente comando en el directo de hue-4.1.0

```
PREFIX=/opt/hadoop make install
```

1- Configuración de HUE

• Accedemos al directorio de configuración de HUE /opt/hadoop/hue/desktop/conf

• Dentro, debemos tener el fichero hue.ini y configurar los valores para:

- HDFS
- YARN
- HIVE

•• La parte de HDFS debe quedar similar a la siguiente

```
# Settings to configure your Hadoop cluster.

###########################################################
####

############

[hadoop]

# Configuration for HDFS NameNode

#
-----------------------------------------------------------
--------

[[hdfs_clusters]]

# HA support by using HttpFs

[[[default]]]

# Enter the filesystem uri

fs_defaultfs=hdfs://nodo1:9000

# NameNode logical name.

Apasoft Training

## logical_name=

# Use WebHdfs/HttpFs as the communication mechanism.

# Domain should be the NameNode or HttpFs host.

# Default port is 14000 for HttpFs.

webhdfs_url=http://nodo1:50070/webhdfs/v1
```

• La parte de YARN debe ser similar a la siguiente

```
[[yarn_clusters]]

[[[default]]]

# Enter the host on which you are running the
ResourceManager

resourcemanager_host=nodo1

# The port where the ResourceManager IPC listens on

resourcemanager_port=8032
```

```
# Whether to submit jobs to this cluster

submit_to=True

# Resource Manager logical name (required for HA)

## logical_name=

# Change this if your YARN cluster is Kerberos-secured

## security_enabled=false

# URL of the ResourceManager API

## resourcemanager_api_url=http://nodo1:8088

# URL of the ProxyServer API

## proxy_api_url=http://nodo1:8088

# URL of the HistoryServer API

## history_server_api_url=http://nodo1:19888Se hace con el
siguiente

comando.
```

• Y por último, la parte de HIVE debe poner lo siguiente

```
[beeswax]

# Host where HiveServer2 is running.

# If Kerberos security is enabled, use fully-qualified
domain name (FQDN).

hive_server_host=nodo1

# Port where HiveServer2 Thrift server runs on.

hive_server_port=10000

# Hive configuration directory, where hive-site.xml is
located

hive_conf_dir=/opt/hadoop/hive/conf
```

• Por último, y muy importante, debemos activar WEBHDFS en nuestro cluster, lo que permite hacer llamadas vía HTTP al cluster.

• Modificamos el fichero hdfs-site y añadimos la siguiente propiedad

```
<property>
```

```
<name>dfs.webhdfs.enabled</name>

<value>true</value>

</property>
```

• Y en el fichero core-site.xml añadimos la siguiente propiedad

```
<property>

<name>hadoop.proxyuser.hue.hosts</name>

<value>*</value>

</property>

<property>

<name>hadoop.proxyuser.hue.groups</name>

<value>*</value>

</property>
```

• Paramos el cluster

• Copiamos los ficheros de configuración al resto de nodos

• Arrancamos el cluster

2- Arrancar y probar HUE

• Para arrancar el cluster debemos ejecutar el siguiente comando

/opt/hadoop/hue/build/env/bin/supervisor -d

• Si todos va bien debemos tener el proceso funcionando

ps -ef | grep supervisor

• Ahora abrimos el firefox y nos conectamos por el puerto 8888

• En la primera pantalla se nos pide establecer un usuario para la herramienta, en este caso lo llamo admin, puedes poner el nombre que quieras

NO ME DEJA ACCEDER Y NO LO PUEDO ARREGLAR ASI QUE PASO A LO SIGUIENTE

SQOOP

Descarga e instalación

Descargamos la version 1.4.6

- cd /opt/hadoop/

- tar xvf /home/hadoop/Descargas/sqoop-1.4.6.bin__hadoop-2.0.4-alpha.tar.gz

- mv sqoop-1.4.6.bin__hadoop-2.0.4-alpha/ sqoop

editamos .bashrc y añadimos el sqoop_home y el path

```
export SQOOP_HOME=/opt/hadoop/sqoop

export PATH=$PATH:$HADOOP_HOME/bin:$HADOOP_HOME/sbin:$SQOOP_HOME/bin
```

• Ahora nos vamos al directorio de configuración de Sqoop

cd /opt/hadoop/sqoop/conf

• Copiamos la plantilla

cp sqoop-env-template.sh sqoop-env.sh

• Editamos el fichero sqoop-env.sh

• Debe quedar algo parecido a lo siguiente. Debe apuntar a nuestros directorios de productos de HADOOP y HIVE

```
#Set path to where bin/hadoop is available

export HADOOP_COMMON_HOME=/opt/hadoop

#Set path to where hadoop-*-core.jar is available

export HADOOP_MAPRED_HOME=/opt/hadoop/

#set the path to where bin/hbase is available

#export HBASE_HOME=

#Set the path to where bin/hive is available

export HIVE_HOME=/opt/hadoop/hive

#Set the path for where zookeper config dir is
```

• Probamos que funciona

sqoop-version

el comando sqoop-import –connect jdbc:oracle:thin:@servidor:1521:xe –username hr –password HR –table DEPARTMENTS –tarjet-dire /datos –as-textfile NO ME FUNCIONA Y NO LOGRO QUE FUNCIONE POR LO QUE PASARÉ A ZOOKEEPER

ZOOKEEPER

Instalación y configuración

Descargamos ZooKeeper de la página zookeeper.apache.org
• Lo descomprimimos en /opt/hadoop
tar xvf /home/hadoop/Descargas/zookeeperXXXXX

• Lo cambiamos de nombre para manejarlo de forma más sencilla
mv zookeeperXXX zoo

• Lo copiamos al nodo2 y nodo3
scp -r zoo nodo2:/opt/hadoop
scp -r zoo nodo3:/opt/hadoop
• Configuramos el fichero /home/hadoop/.bashrc para incluir las líneas de ZooKeeper. Y lo debemos copiar a el resto de nodos donde vamos a tener funcionando ZooKeeper
• Debería quedar algo parecido a lo siguiente:

```
export HADOOP_HOME=/opt/hadoop
export JAVA_HOME=/usr/java/jdk1.8.0_151
export HIVE_HOME=/opt/hadoop/hive
export SQOOP_HOME=/opt/hadoop/sqoop
export ZOOKEEPER_HOME=/opt/hadoop/zoo
export
PATH=$PATH:$HADOOP_HOME/bin:$HADOOP_HOME/sbin:$SQOOP_HOME/b
in:$HIVE_HOME/bin:$ZOOKEEPER_HOME/bin
```

• Si entramos de nuevo en un terminal tendremos ya cargadas las variables.
• Nos situamos en el directorio de configuración
cd /opt/hadooop/zoo/conf
• Copiamos el fichero zoo_sample.cfg como zoo.cfg
cp zoo_sample.cfg zoo.cfg
• Creamos el siguiente contenido dentro del fichero

```
# the directory where the snapshot is stored.
# do not use /tmp for storage, /tmp here is just
# example sakes.
dataDir=/datos/zoo
# the port at which the clients will connect
clientPort=2181
```

```
# the maximum number of client connections.
# increase this if you need to handle more clients
#maxClientCnxns=60
#
# Be sure to read the maintenance section of the
# administrator guide before turning on autopurge.
#
#
http://zookeeper.apache.org/doc/current/zookeeperAdmin.html
#sc_maintenance
#
# The number of snapshots to retain in dataDir
#autopurge.snapRetainCount=3
# Purge task interval in hours
# Set to "0" to disable auto purge feature
#autopurge.purgeInterval=1
server.1=nodo1:2888:3888
server.2=nodo2:2888:3888
server.3=nodo3:2888:3888
```

• Creamos en los 3 nodos el directorio de trabajo
mkdir /datos/zoo
• Creamos en el directorio un fichero denominado “myid” que tiene que contener el número de servidor dentro de ZooKeeper.
• Por ejemplo en el nodo1:
echo 1 > /datos/zoo/myid
• Y en el nodo2 y el nodo3 ponemos 2 y 3.
• Ejecutamos el siguiente comando en los 3 nodos
zkCli.sh start
• Comprobamos con “jps” que tenemos el proceso QuorumPeerMain funcionando
• Podemos preguntar el estado y si es leader o follower
zkServer.sh status

Trabajar con el cliente de Zookeeper

ZooKeeper
accederemos con zkClient.sh
zkClient.sh

• Escribimos help para ver la ayuda disponible en el cliente
• Comprobamos si hay algún znode en la estructura jerárquica. Debe aparecer vacío, solo con el nodo “zookeeper” predefinido
ls /
[zookeeper]
• Creamos un znode, con algún valor
create /m1 v1

• Comprobarmos el resultado
get /m1

• Nos vamos al nodo2, accedemos al cliente zkClient y comprobamos que tenemos el znode m1
• Lo borramos desde el nodo2
delete /m1
• Vamos al nodo1 y comprobamos que ha desaparecido
ls /

Configurar y arrancar HDFS en alta disponibilidad

Debemos incorporar las siguientes propiedades en nuestros ficheros.
• CORE-SITE.XML

```
<property>
<name>fs.defaultFS</name>
<value>hdfs://ha-cluster</value>
</property>
```

• HDFS-SITE.XML

```
<name>dfs.nameservices</name>
<value>ha-cluster</value>
</property>
<property>
<name>dfs.ha.namenodes.ha-cluster</name>
<value>nodo1,nodo2</value>
</property>
<property>
<name>dfs.permissions</name>
<value>false</value>
</property>
<property>
<name>dfs.namenode.rpc-address.ha-cluster.nodo1</name>
<value>nodo1:9000</value>
</property>
<property>
<name>dfs.namenode.rpc-address.ha-cluster.nodo2</name>
<value>nodo2:9000</value>
</property>
<property>
<name>dfs.namenode.http-address.ha-cluster.nodo1</name>
<value>nodo1:50070</value>
</property>
<property>
<name>dfs.namenode.http-address.ha-cluster.nodo2</name>
<value>nodo2:50070</value>
</property>
```

```
<property>
<name>dfs.namenode.shared.edits.dir</name>
<value>qjournal://nodo3:8485;nodo2:8485;nodo1:8485/ha-clust
er</value>
</property>
<property>
<name>dfs.journalnode.edits.dir</name>
<value>/datos/jn</value>
</property>
<property>
<name>dfs.client.failover.proxy.provider.ha-cluster</name>
<value>org.apache.hadoop.hdfs.server.namenode.ha.Configured
FailoverProxyProvider
</value>
</property>
<property>
<name>dfs.ha.automatic-failover.enabled</name>
<value>true</value>
</property>
<property>
<name>ha.zookeeper.quorum</name>
<value>nodo1:2181,nodo2:2181,nodo3:2181</value>
</property>
<property>
<name>dfs.ha.fencing.methods</name>
<value>sshfence</value>
</property>
<property>
<name>dfs.ha.fencing.ssh.private-key-files</name>
<value>/home/hadoop/.ssh/id_rsa</value>
</property>
```

Terminar la configuración y arrancar el cluster
• Parar todo el cluster si lo tenemos arrancado
• Borrar los directorios de /datos SOLO SI QUEREMOS CREAR EL CLUSTER DESDE CERO.
• Nos debemos asegurar que hemos copiado los ficheros de configuración al resto de nodos
• Primero nos aseguramos de que tenemos los 3 servidores zookeeper funcionando
zkServer.sh status

• Arrancar en los tres nodos el Journal
hadoop-daemon.sh start journalnode

• Vamos al nodo1. Creamos de nuevo el HDFS del cluster. SOLO SI QUEREMOS CREAR EL CLUSTER DESDE CERO. De lo contrario no ejecutamos nada

hdfs namenode -format
• Arrancamos el namenode
hadoop-daemon.sh start namenode

• Vamos al nodo2. Ejecutamos la sincronización con el namenode del nodo1
namenode -bootstrapStandby

• Una vez terminado satisfctoriamente arrancamos el namenode. De esa forma ya tenemos el standby funcionando
hadoop-daemon.sh start namenode

• Comprobamos con jps que tenemos todos los procesos funcionando.
• Volvemos al nodo1
• En los dos nodos Preparamos y arrancamos el zkController
hdfs zkfc –formatZK
hadoop-daemon.sh start zkfc

• Comprobamos que tenemos todos los procesos funcionando, con jps
• Paramos todo el cluster stop-dfs.sh
• Arrancamos de nuevo el cluster para comprobar que todo arranca start-dfs.sh
• Arrancamos la Web Admin del nodo1
• Arrancamos el Web Admin del nodo2. Vemos que está en standby
• También lo podemos ver desde línea de comandos:
$hdfs haadmin -getServiceState nodo1
active
$ hdfs haadmin -getServiceState nodo2
standby
$ hdfs haadmin -getAllServiceState
nodo1:9000 active
nodo2:9000 standby
• Podemos hacer un failover manual
• Por ejemplo, para pasar al nodo1 al nodo2
$hdfs haadmin -failover nodo1 nodo2
Failover to NameNode at nodo2/192.168.56.105:9000 successful
$ hdfs haadmin -getAllServiceState
nodo1:9000 standby
nodo2:9000 active
• Podemos comprobarlo en la Web Admin

SPARK

Antes que nada, debido a que parece que tiene unos conflictos con el Zookeeper, y despues de un intento fallido de cambiar todos los archivos y los procesos que realicé para Zookeeper, he decidido volver a una snapshot anterior que tenia antes de Zookeeper.

- Para descargar Spark vamos a spark.apache.org y descargamos la verison mas conveniente. Descargaremos la version without Apache hadoop
- Se hace el tar correspondiente en /otp/hadoop y se cambia el nombre a spark
- vamos a .bashrc y añadimos en el path
  /opt/hadoop/spark/bin:/opt/hadoop/spark/sbin
  tambien añadimos export SPARK_DIST_CLASSPATH=$(hadoop classpath)
- Vamos a /opt/hadoop/spark y ejecutamos spark-shell
  No voy a apuntar aqui la prueba de spark y python porque ya he hecho varias practicas asi y es espacio a lo tonto.

  para configurarlo como hdfs:
- vamos a .bashrc y añadimos export SPARK_HOME=/opt/hadoop/spark y export HADOOP_CONF_DIR=$HADOOP_HOME/etc/hadoop
  Ahora se podrá usar el spark-shell con archivos hdfs

  Ejemplo Spark Scala con Yarn
  spark-submit –class org.apache.spark.examples.SparkPi –master yarn –deploy-mode cluster –name “aplil” /opt/hadoop-3.2.3/spark/examples/jars/spark-examples_2.11-2.3.0.jar 5

  si vamos al puerto 8088 en firefox se verá el estado de la aplicacion y los logs de los resultados

  Ejemplo Spark Python con Yarn
  spark-submit –master yarn –deploy-mode cluster –name “ContarPalabras” ContarPalabras.py /Practicas/quijote.txt /salida_spark_wc

  si vamos al puerto 8088 en firefox se verá el estado de la aplicacion y los logs de los resultados

Spark Standalone

Descargaremos la version Pre-built for apache hadoop
Se descomprime en otro directorio que el anterior, por ejemplo en /home/hadoop/spark_standalone

vamos a su - root y cambiamos el nombre de la carpeta
en /home/hadoop/spark_standalone/spark hacemos sbin/start-master.sh
y vamos a localhost:8080 donde pondrá info

ahora arrancamos el esclavo con sbin/start-slave.sh nodo1:7077
y en el localhost:8080 se verán los workers
Podemos realizar consultas con spark-shell perfectamente con maestros y esclavos

HBASE

Lo instalamos en /opt/hadoop/hbase con tar como siempre

Editamos `/opt/hadoop/hbase/conf/hbase-site.xml`,

```
<property>

      <name>hbase.cluster.distributed</name>
         <value>false</value>
       </property>
       <property>
         <name>hbase.tmp.dir</name>
         <value>./tmp</value>
       </property>
       <property>
         <name>hbase.rootdir</name>
         <value>file:///opt/hadoop/hbase/data/hbase</value>
       </property>
       <property>
         <name>hbase.zookeeper.property.dataDir</name>
         <value>/opt/hadoop/hbase/data/hbase</value>
       </property>
       <property>
         <name>hbase.unsafe.stream.capability.enforce</name>
         <value>false</value>
       </property>
```

vamos a bin y ejecutamos `./start-hbase.sh`

ademas tenemos que añadir en .bashrc HBASE_HOME al path

Se prueban comandos de Hbase, como

- create ‘t1’, ‘cf1’
- describe ‘t1’
- create ‘empleados’, ‘personal’, ‘trabajo’
- put ‘empleados’, ‘personal:nombre’, ‘Sergio’
- put ‘empleados’, ‘1’,’personal:nombre’,’Sergio’
- scan ‘empleados’
- scan ‘empleados’, {COLUMNS=>’personal:nombre’}
- create ‘empleado1’, {NAME => ‘familia1’, VERSIONS=>3}

Cluster Pseudo-distribuido

- Modificar el archivo hbase-site.xml

```
<property>

<name>hbase.cluster.distributed</name>

  <value>true</value>
</property>

<property>
  <name>hbase.rootdir</name>
  <value>hdfs://nodo1:9000/hbase</value>
</property>

<property>
  <name>hbase.zookeeper.quorum</name>
  <value>nodo1,nodo2,nodo3</value>
</property>
```

- Modificar el archivo hbase-env.sh

```
# Tell HBase whether it should manage it's own instance of
ZooKeeper or not.
export HBASE_MANAGES_ZK=false
export HBASE_DISABLE_HADOOP_CLASSPATH_LOOKUP=true
```

Hacemos start-hbase.sh pero no funciona, asi que dejo hbase aca

AMBARI

Para descargarlo, tenemos que descargar el repositorio de Ambari en el Server Host Ambari

vamos a /etc/yum.repos.d

escribimos su - root y vamos a ese directorio otra vez

alli hacemos wget y el enlace del repositorio, pero no funciona. Asi que veré los videos para entenderlo pero esto se queda asi.