

Prácticas BigData

1. Instalación de Hadoop. Modo StandAlone

- Arrancamos la máquina virtual facilitada por el profesor o la que hayamos construido nosotros

1.1. Crear usuario hadoop

- Si no usamos la máquina del curso debemos crear un usuario para hadoop.
- Accedemos como ROOT al sistema.
- Ejecutamos el siguiente comando para crear el usuario

```
useradd hadoop
```

- Le ponemos contraseña

```
passwd hadoop
```

- Debe haber creado un directorio denominado /home/hadoop

1.2. Instalar JDK de Oracle

- Seguimos como ROOT
- Descargamos el RPM de Java de la página de Oracle. Siempre es más fácil de instalar.
- NOTA: la versión puede no cuadrar con la existente en el momento de hacer este manual.

Java SE Development Kit 8 Downloads

Thank you for downloading this release of the Java™ Platform, Standard Edition Development Kit (JDK™). The JDK is a development environment for building applications, applets, and components using the Java programming language.

The JDK includes tools useful for developing and testing programs written in the Java programming language and running on the Java platform.

See also:

- [Java Developer Newsletter](#): From your Oracle account, select **Subscriptions**, expand **Technology**, and subscribe to **Java**.
- [Java Developer Day hands-on workshops \(free\) and other events](#)
- [Java Magazine](#)

JDK 8u151 checksum
JDK 8u152 checksum

Java SE Development Kit 8u151

You must accept the [Oracle Binary Code License Agreement for Java SE](#) to download this software.
Thank you for accepting the Oracle Binary Code License Agreement for Java SE; you may now download this software.

Product / File Description	File Size	Download
Linux ARM 32 Hard Float ABI	77.9 MB	jdk-8u151-linux-arm32-vfp-hflt.tar.gz
Linux ARM 64 Hard Float ABI	74.85 MB	jdk-8u151-linux-arm64-vfp-hflt.tar.gz
Linux x86	168.95 MB	jdk-8u151-linux-i586.rpm
Linux x86	183.73 MB	jdk-8u151-linux-i586.tar.gz
Linux x64	166.1 MB	jdk-8u151-linux-x64.rpm
Linux x64	180.95 MB	jdk-8u151-linux-x64.tar.gz
macOS	247.06 MB	jdk-8u151-macosx-x64.dmg
Solaris SPARC 64-bit	449.96 MB	jdk-8u151-solaris-sparcv9.tar.gz

- Instalamos JDK mediante RPM o cualquier otro de los mecanismos del Sistema operativo con el que estemos trabajando. El que usamos durante el curso es un CENTOS.

```
rpm -ivh jdkXXXXXX.rpm
```

- Debemos asegurarnos de que usa el JDK que hemos descargado para que no tengamos problemas.
- Si disponemos de distintas versiones podemos utilizar el siguiente comando. Debemos seleccionar la versión que hemos descargado. Seguramente existen otras que vienen con el propio CENTOS.

alternatives --config java

Hay 3 programas que proporcionan 'java'.

Selección Comando

```
*+ 1        /usr/lib/jvm/java-1.7.0-openjdk-1.7.0.75-2.5.4.2.el7_0.x86_64/jre/bin/java
2        /usr/java/jdk1.8.0_45/jre/bin/java
3        /usr/java/jdk1.8.0_45/bin/java
```

www.apasoft-training.com

apasoft.training@gmail.com

Presione Intro para mantener la selección actual[+], o escriba el número de la selección: 3

- Comprobamos que podemos acceder a JAVA y a sus comandos

javac -version

javac 1.8.0_45

java -version

java version "1.8.0_45"

Java(TM) SE Runtime Environment (build 1.8.0_45-b14)

Java HotSpot(TM) 64-Bit Server VM (build 25.45-b02, mixed mode)

1.3. Configurar las variables de entorno para JAVA

- Accedemos como usuario HADOOP
- Para que no haya problemas durante el curso debemos configurar el entorno.
- Configuramos las variables de entorno dentro del usuario que estemos utilizando.
- Debemos poner esas variables en algún fichero que las cargue cuando el usuario “hadoop” acceda al sistema.
- En este caso vamos a usar el fichero “**.bashrc**” (Con punto al principio. Recordemos que los ficheros que empiezan con punto en Linux son ocultos y no se ven con un simple “ls”. Hay que hacer un “ls -a”)
- Se encuentra en el directorio del usuario “**/home/hadoop**”, pero podemos usar cualquier otro fichero que permita cargar las variables al inicio.
- Incorporamos el acceso a JAVA. (En realidad, ponerlo en el PATH seguramente no hace falta porque lo ha hecho el instalador, pero siempre es mejor indicarlo)

```
export JAVA_HOME=/usr/java/jdkXXXXX
export PATH=$PATH:$JAVA_HOME/bin
```

1.4. Descargar e instalar hadoop

- Accedemos como usuario ROOT
- Con el Firefox, vamos a la página de Hadoop y descargamos el software. En el momento de hacer esta documentación la última versión estable es la 2.9
- **NOTA IMPORTANTE: el 15 de Diciembre se ha liberado la versión 3, pero este curso está basado en la 2. La instalación y otros componentes cambian. Al menos durante un tiempo y hasta que se estabilice es preferible seguir usando la 2. Estamos trabajando para añadir al curso los cambios de la 3 y hacer un anexo con los mismos.**
- Lo debe haber dejado en /root/Descargas (o Downloads si tenéis el entorno en inglés).
- Vamos a hacer la instalación en el directorio /opt
- Copiamos el software de hadoop a /opt.

```
cp hadoop-XXXX.tar /opt
```

- Accedemos a /opt

```
cd /opt
```

- Desempaquetamos el software

```
tar xvf hadoopXXX-bin.tar
```

- Esto debe haber creado un directorio denominado hadoop-XXXXXX.
- Para hacer más sencillo el trabajo lo cambiamos de nombre a “hadoop”

```
mv hadoop-XXXX hadoop
```

- Comprobamos si existen los ficheros desempaquetados en el directorio

```
ls -l /opt/hadoop
total 341504
drwxr-xr-x. 2 hadoop hadoop 194 nov 14 00:28 bin
drwxr-xr-x. 3 hadoop hadoop 20 nov 14 00:28 etc
drwxr-xr-x. 2 hadoop hadoop 106 nov 14 00:28 include
drwxr-xr-x. 3 hadoop hadoop 20 nov 14 00:28 lib
drwxr-xr-x. 2 hadoop hadoop 239 nov 14 00:28 libexec
-rw-r--r--. 1 hadoop hadoop 106210 nov 14 00:28 LICENSE.txt
drwxrwxr-x. 2 hadoop hadoop 4096 ene 4 18:28 logs
-rw-r--r--. 1 hadoop hadoop 15915 nov 14 00:28 NOTICE.txt
-rw-r--r--. 1 hadoop hadoop 1366 nov 14 00:28 README.txt
drwxr-xr-x. 3 hadoop hadoop 4096 dic 27 17:03 sbin
drwxr-xr-x. 4 hadoop hadoop 31 nov 14 00:28 share
```

- Cambiamos los permisos para que pertenezcan al usuario “hadoop”, que es con el que vamos a trabajar.

```
cd /opt  
chown -R hadoop:hadoop hadoop
```

1.5. Configurar las variables de HADOOP y comprobar que todo funciona

- Salimos como usuario “root” y accedemos como usuario HADOOP
- Configuramos en el fichero “/home/hadoop/.bashrc” para las variables de acceso a Hadoop. Incluimos las siguientes

```
export HADOOP_HOME=/opt/hadoop
export PATH=$PATH:$HADOOP_HOME/bin
export HADOOP_CONF_DIR=$HADOOP_HOME/etc/hadoop
```

- Para probarlo, salimos de la sesión y volvemos a entrar, o bien ejecutamos el siguiente comando (debemos dejar un espacio en blanco entre los dos puntos).

```
. ./bashrc
```

- Ejecutar “hadoop -h” para comprobar si accedemos correctamente

```
hadoop -h
Usage: hadoop [--config confdir] COMMAND
    where COMMAND is one of:
    fs          run a generic filesystem user client
    version     print the version
    jar <jar>   run a jar file
    checknative [-a|-h] check native hadoop and compression libraries availability
    distcp <srcurl> <desturl> copy file or directories recursively
    archive -archiveName NAME -p <parent path> <src>* <dest> create a hadoop archive
    classpath   prints the class path needed to get the
    credential  interact with credential providers
               Hadoop jar and the required libraries
    daemonlog   get/set the log level for each daemon
    trace       view and modify Hadoop tracing settings
    or
    CLASSNAME   run the class named CLASSNAME

Most commands print help when invoked w/o parameters.
```

- Comprobamos la versión utilizada (seguramente no será ya igual a la vuestra, en este caso usamos la última de la 2, que es la 2.9)

```
hadoop version
Hadoop 2.9.0
Subversion      https://git-wip-us.apache.org/repos/asf/hadoop.git -r
756ebc8394e473ac25feac05fa493f6d612e6c50
Compiled by arsuresh on 2017-11-13T23:15Z
Compiled with protoc 2.5.0
From source with checksum 0a76a9a32a5257331741f8d5932f183
```

This command was run using /opt/hadoop/share/hadoop/common/hadoop-common-2.9.0.jar

- Vamos a realizar un pequeño ejemplo en modo standalone. Esto nos permite comprobar que todo funciona correctamente.
- **NOTA IMPORTANTE: recordad que tenéis que cambiar la versión en los ficheros. En este ejemplo estamos usando la 2.9. Debéis adaptarlo a lo que tengáis vosotros.**

- Nos situamos en /opt/hadoop

```
cd /opt/hadoop
```

- Creamos un directorio en /tmp

```
mkdir /tmp/input
```

- Copiamos todos los ficheros XML que hay en el directorio /opt/hadoop/etc/hadoop

```
cp etc/hadoop/*.xml /tmp/input/
```

- Ejecutamos el siguiente comando que busca todos los ficheros de /tmp/input que tengan el texto “dfs” y luego tenga un carácter de la “a” a la “z” y deja el resultado en el directorio /tmp/output”. Funciona de forma parecida al grep de linux

- **NOTA:** en siguientes capítulos veremos con más detalle el comando hadoop. Por ahora solo es necesario saber que lanza un proceso de tipo MapReduce de Hadoop

```
hadoop jar share/hadoop/mapreduce/hadoop-mapreduce-examples-2.9.jar grep /tmp/input /tmp/output 'dfs[a-z.]+'
```

- Vemos lo que ha dejado en /tmp/output

```
ls -l /tmp/output/
total 4
-rw-r--r--. 1 hadoop hadoop 404 ene 6 15:00 part-r-00000
-rw-r--r--. 1 hadoop hadoop 0 ene 6 15:00 _SUCCESS
```

- Este programa genera un fichero denominado “part-r-0000” con el resultado del comando.

- Debe contener algo parecido a lo siguiente

```
cat /tmp/output/part-r-00000
2 dfs.namenode.http
2 dfs.namenode.rpc
1 dfsadmin
1 dfs.server.namenode.ha.
1 dfs.replication
1 dfs.permissions
1 dfs.nameservices
1 dfs.namenode.shared.edits.dir
```


1	dfs.namenode.name.dir
1	dfs.namenode.checkpoint.dir
1	dfs.journalnode.edits.dir
1	dfs.ha.namenodes.ha
1	dfs.ha.fencing.ssh.private
1	dfs.ha.fencing.methods
1	dfs.ha.automatic
1	dfs.datanode.data.dir
1	dfs.client.failover.proxy.provider.ha

2. Configurar SSH

- Para poder trabajar con hadoop, incluso aunque tengamos un solo nodo, debemos configurar la conectividad SSH.
- Luego también debemos hacerlo para el resto de nodo
- Entramos como usuario HADOOP
- Configuramos la conectividad SSH del nodo1.
- Creamos las claves con el comando ssh-keygen

```
ssh-keygen
Generating public/private rsa key pair.
Enter file in which to save the key (/home/hadoop/.ssh/id_rsa):
Created directory '/home/hadoop/.ssh'.
Enter passphrase (empty for no passphrase):
Enter same passphrase again:
Your identification has been saved in /home/hadoop/.ssh/id_rsa.
Your public key has been saved in /home/hadoop/.ssh/id_rsa.pub.
The key fingerprint is:
6e:3e:59:be:9b:8e:1f:59:ea:a5:dd:87:8d:31:2d:82 hadoop@localhost.localdomain
The key's randomart image is:
+--[ RSA 2048 ]-----+
|           |
|           |
|           |
|           |
|  S  .. . |
| . E+ . + . |
|  o++ .. B |
|  ooo.* .o o|
|  o+Oo. .. |
+-----+
```

- Esto habrá creado un directorio denominado /home/hadoop/.ssh (si no existía ya) y habrá creado dos ficheros con las clave pública y la privada

```
ls -l .ssh
total 16
-rw----- 1 hadoop hadoop 1675 dic 26 18:45 id_rsa
-rw-r--r-- 1 hadoop hadoop 394 dic 26 18:45 id_rsa.pub
```

- El fichero id_rsa.pub debe contener algo parecido a lo siguiente:

```
cat id_rsa.pub
```

www.apasoft-training.com

apasoft.training@gmail.com

```
ssh-rsa
AAAAB3NzaC1yc2EAAAADAQABAAQDA5JBldH7BzK2/+wV1UzYxvyMPLN2
Et7Ql5aOXyW6aC7kW3L2XqQ+9KAQW7ZCdt5+69qZp8HuV+oNTONISLvVLfXoEwQ0
odzTFI7LPNWXkNWuuOr5GxejKW5Xgld/J6BKKeQu6ocnQhyfEw/ZtDEj55WZtPnnBmz
uwuw0djnf9EttMZZSW3LwApuTiqG58voLy3yQHvE2AN6SiFGLh7/qUwQJP41ISvOXRty
V2oOrS7wBjVA9ow3FFI1qg9ONVmlzn8MpdXyvU8B1zE82RZv5piALIAGJgwHV8hO+v
T+4YKLgH9cW7TU1IFVxWYM+cMy1yBL7Df4hWIA5SazDrjf
hadoop@nodo1
```

- Creamos el fichero `authorized_keys` que necesitamos para luego pasar la clave pública al resto de nodos. En principio debemos copiarlo al resto de los nodos, algo que haremos posteriormente.

```
cd .ssh
cp id_rsa.pub authorized_keys
```

- Comprobamos el acceso a través de `ssh` en el propio `nodo1`, ya que por ahora vamos a trabajar en modo local. No debe pedirnos contraseña de acceso.

```
ssh nodo1
The authenticity of host 'nodo1 (::1)' can't be established.
ECDSA key fingerprint is 3d:94:76:5e:20:c4:b7:c1:98:91:bb:db:fb:e2:01:ea.
Are you sure you want to continue connecting (yes/no)? yes
Warning: Permanently added nodo1 (ECDSA) to the list of known hosts.
Last login: Sun Apr 19 10:33:40 2015
```

- Para salir de la conexión `ssh` y volver a la Shell anterior ponemos “exit”

3. Instalación pseudo-distribuida

- Accedemos como usuario ROOT
- Creamos un directorio denominado /datos, donde crearemos nuestros directorios de datos

```
mkdir /datos
```

- Le damos permisos para que pueda ser usado por el usuario hadoop

```
chown hadoop:hadoop /datos
```

- Salimos de la sesión de ROOT y nos logamos como HADOOP
- Acceder al directorio “/opt/hadoop/etc/hadoop” de hadoop
- Modificamos el fichero “core-site.xml para que quede de la siguiente manera. Debemos asegurarnos de no poner localhost, o de lo contrario no funcionará cuando pongamos un cluster real

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

- Modificamos el hdfs-site.xml para que quede de la siguiente forma. Como solo tenemos un nodo, debemos dejarlo con un valor de replicación de 1.
- También le indicamos como se va a llamar el directorio para los metadatos y el directorio para los datos.

```
<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>
```

- Vamos a crear los directorios para el sistema de ficheros HDFS, tanto para el namenode como para el datanode (no se tienen porque llamar así, pero para el curso queda más claro). Debemos crearlos con el mismo nombre que pusimos en el fichero de configuración.
- No es necesario crearlos porque los crea automáticamente, pero de ese modo nos aseguramos de que no tenemos problemas de permisos

```
mkdir /datos/namenode
```

```
mkdir /datos/datanode
```

- Formateamos el sistema de ficheros que acabamos de crear

```
hdfs namenode -format
```

```
18/01/06 16:29:46 INFO namenode.NameNode: STARTUP_MSG:
```

```
/******
```

```
STARTUP_MSG: Starting NameNode
```

```
STARTUP_MSG: host = nodo1/192.168.56.101
```

```
STARTUP_MSG: args = [-format]
```

```
STARTUP_MSG: version = 2.9.0
```

```
STARTUP_MSG: classpath =
```

```
/opt/hadoop/etc/hadoop:/opt/hadoop/share/hadoop/common/lib/nimbus-jose-jwt-3.9.jar:/opt/hadoop/share/hadoop/common/lib/java-xmlbuilder-0.4.jar:/opt/hadoop/share/hadoop/common/lib/commons-configuration-1.6.jar:/opt/hadoop/share/hadoop/common/lib/commons-cli-1.2.jar:/opt/hadoop/share/hadoop/common/lib/commons-net-3.1.jar:/opt/hadoop/share/hadoop/common/lib/jersey-core-1.9.jar:/opt/hadoop/share/hadoop/common/lib/guava-11.0.2.jar:/opt/hadoop/share/hadoop/common/lib/gson-2.2.4.jar:/opt/hadoop/share/hadoop/common/lib/jackson-core-asl-1.9.13.jar:/opt/hadoop/share/hadoop/common/lib/log4j-1.2.17.jar:/opt/hadoop/share/hadoop/common/lib/woodstox-core-5.0.3.jar:/opt/hadoop/share/hadoop/
```

```
....
```

```
...
```

```
..
```

- Si todos es correcto debemos tener el directorio /datos/namenode que debe tener otro subdirectorio denominado “current”

```
ls -l /datos/namenode/
```

```
total 0
```

```
drwxrwxr-x. 2 hadoop hadoop 112 ene 6 16:57 current
```

- Arrancamos los procesos de HDFS. Debe arrancar el NAMENODE, el SECONDARY NAMENODE y el DATANODE

start-dfs.sh

Starting namenodes on [localhost]

localhost: starting namenode, logging to /opt/hadoop/logs/hadoop-hadoop-namenode-nodo1.out

nodo1: starting datanode, logging to /opt/hadoop/logs/hadoop-hadoop-datanode-nodo1.out

Starting secondary namenodes [0.0.0.0]

0.0.0.0: starting secondarynamenode, logging to /opt/hadoop/logs/hadoop-hadoop-secondarynamenode-nodo1.out

- Comprobamos con el comando jps los procesos que se están ejecutando

jps

22962 DataNode

23187 SecondaryNameNode

23395 Jps

22837 NameNode

- También podemos comprobarlo con ps, para ver los procesos java

ps -ef | grep java

```
hadoop  20694      1  2 16:46 ?        00:00:06 /usr/java/jdk1.8.0_151/bin/java -
Dproc_datanode      -Xmx1000m      -Djava.net.preferIPv4Stack=true      -
Dhadoop.log.dir=/opt/hadoop/logs      -Dhadoop.log.file=hadoop.log      -
Dhadoop.home.dir=/opt/hadoop      -Dhadoop.id.str=hadoop      -
Dhadoop.root.logger=INFO,console      -Djava.library.path=/opt/hadoop/lib/native      -
Dhadoop.policy.file=hadoop-policy.xml      -Djava.net.preferIPv4Stack=true      -
Djava.net.preferIPv4Stack=true      -Djava.net.preferIPv4Stack=true      -
Dhadoop.log.dir=/opt/hadoop/logs -Dhadoop.log.file=hadoop-hadoop-datanode-nodo1.log -
Dhadoop.home.dir=/opt/hadoop -Dhadoop.id.str=hadoop -Dhadoop.root.logger=INFO,RFA -
Djava.library.path=/opt/hadoop/lib/native -Dhadoop.policy.file=hadoop-policy.xml -
Djava.net.preferIPv4Stack=true -server -Dhadoop.security.logger=ERROR,RFA -
Dhadoop.security.logger=ERROR,RFA -Dhadoop.security.logger=ERROR,RFA -
Dhadoop.security.logger=INFO,RFA org.apache.hadoop.hdfs.server.datanode.DataNode

hadoop  20885      1  2 16:46 ?        00:00:05 /usr/java/jdk1.8.0_151/bin/java -
Dproc_secondarynamenode -Xmx1000m      -Djava.net.preferIPv4Stack=true      -
Dhadoop.log.dir=/opt/hadoop/logs      -Dhadoop.log.file=hadoop.log      -
Dhadoop.home.dir=/opt/hadoop      -Dhadoop.id.str=hadoop      -
Dhadoop.root.logger=INFO,console      -Djava.library.path=/opt/hadoop/lib/native      -
Dhadoop.policy.file=hadoop-policy.xml      -Djava.net.preferIPv4Stack=true      -
Djava.net.preferIPv4Stack=true      -Djava.net.preferIPv4Stack=true      -
Dhadoop.log.dir=/opt/hadoop/logs -Dhadoop.log.file=hadoop-hadoop-secondarynamenode-
nodo1.log      -Dhadoop.home.dir=/opt/hadoop      -Dhadoop.id.str=hadoop      -
Dhadoop.root.logger=INFO,RFA      -Djava.library.path=/opt/hadoop/lib/native      -
Dhadoop.policy.file=hadoop-policy.xml      -Djava.net.preferIPv4Stack=true      -
Dhadoop.security.logger=INFO,RFA      -Dhdfs.audit.logger=INFO,NullAppender      -
Dhadoop.security.logger=INFO,RFA      -Dhdfs.audit.logger=INFO,NullAppender      -
Dhadoop.security.logger=INFO,RFA      -Dhdfs.audit.logger=INFO,NullAppender      -
Dhadoop.security.logger=INFO,RFA
org.apache.hadoop.hdfs.server.namenode.SecondaryNameNode

hadoop  21140 26575  0 16:51 pts/0   00:00:00 grep --color=auto java
....
...
...
```

- Debe haber creado el directorio /datos/datanode que también tiene que tener un directorio llamado current.

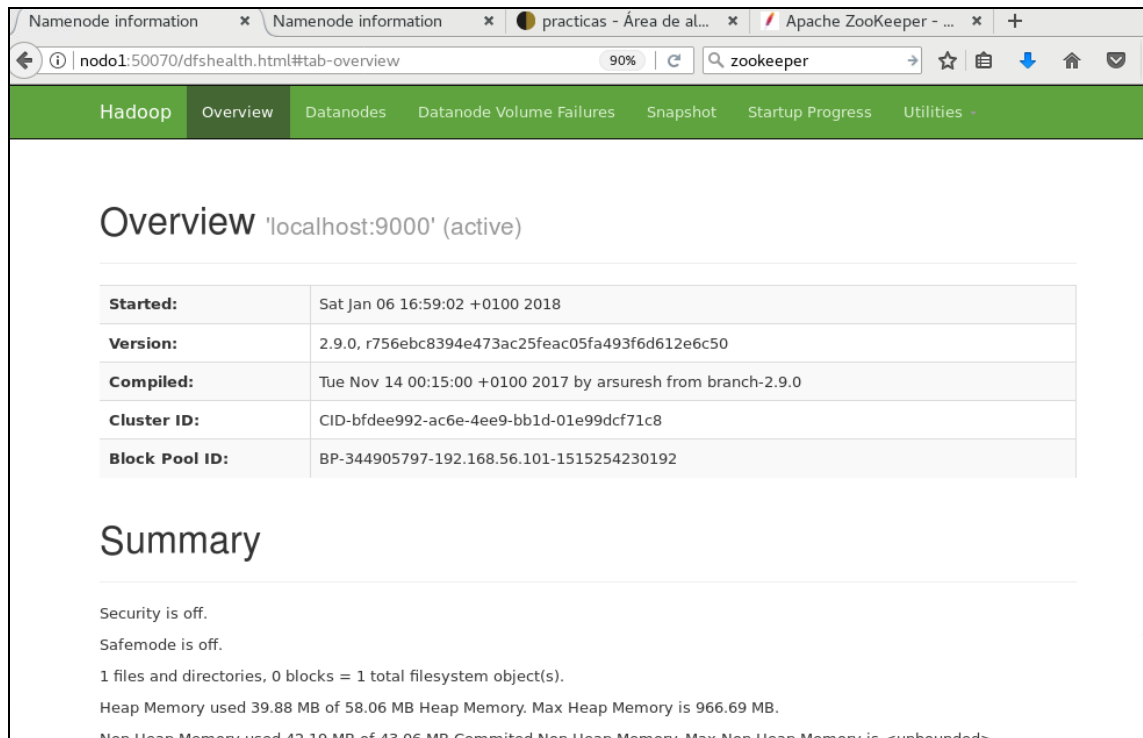
ls -l /datos/datanode

total 4

drwxrwxr-x. 3 hadoop hadoop 70 ene 6 16:59 current

-rw-rw-r--. 1 hadoop hadoop 11 ene 6 16:59 in_use.lock

- Podemos acceder a la web de Administración para ver el resultado



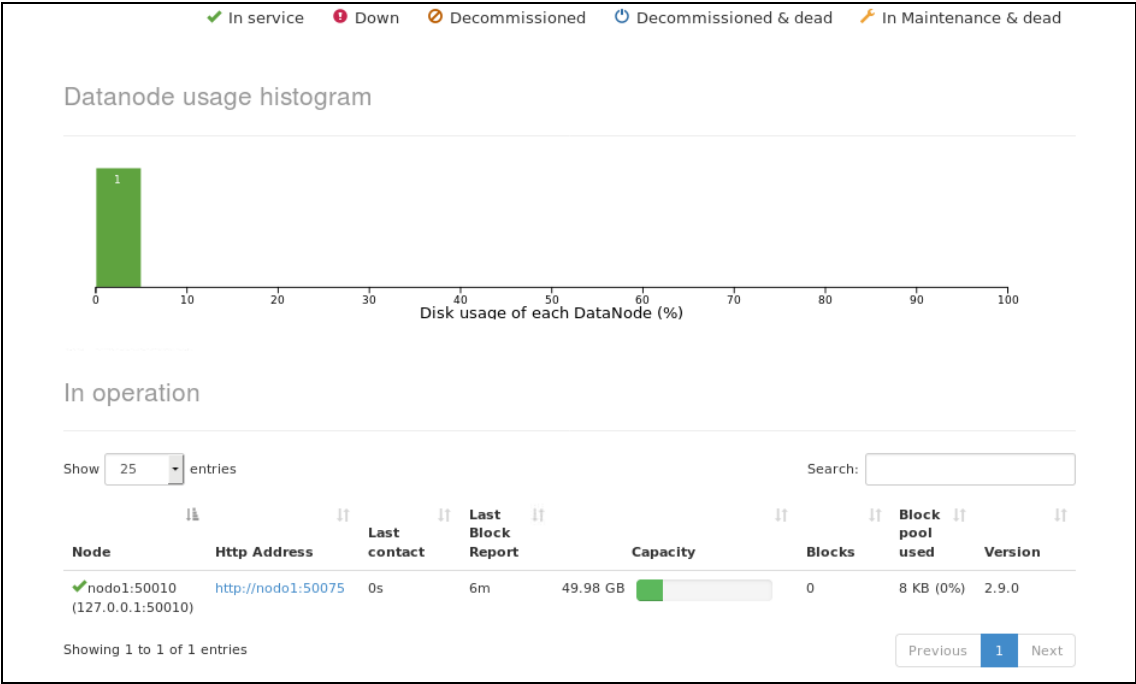
The screenshot shows a web browser window with the URL `node1:50070/dfshealth.html#tab-overview`. The page title is "Overview 'localhost:9000' (active)". The page contains a table with the following information:

Started:	Sat Jan 06 16:59:02 +0100 2018
Version:	2.9.0, r756ebc8394e473ac25feac05fa493f6d612e6c50
Compiled:	Tue Nov 14 00:15:00 +0100 2017 by arsureh from branch-2.9.0
Cluster ID:	CID-bfdee992-ac6e-4ee9-bb1d-01e99dcf71c8
Block Pool ID:	BP-344905797-192.168.56.101-1515254230192

Below the table, there is a "Summary" section with the following text:

Security is off.
 Safemode is off.
 1 files and directories, 0 blocks = 1 total filesystem object(s).
 Heap Memory used 39.88 MB of 58.06 MB Heap Memory. Max Heap Memory is 966.69 MB.
 Non Heap Memory used 42.19 MB of 43.06 MB Committed Non Heap Memory. Max Non Heap Memory is <unbounded>

- Podemos comprobar el datanode



●

4. Prácticas con HDFS

4.1. Comando `hdfs dfs`

- Ejecutar el comando “`hdfs dfs`”. Este comando permite trabajar con los ficheros de HDFS.
- Casi todas las opciones son similares a los comandos “Linux”

`hdfs dfs`

Usage: `hadoop fs [generic options]`

```
[-appendToFile <localsrc> ... <dst>]
[-cat [-ignoreCrc] <src> ...]
[-checksum <src> ...]
[-chgrp [-R] GROUP PATH...]
[-chmod [-R] <MODE[,MODE]... | OCTALMODE> PATH...]
[-chown [-R] [OWNER][:[GROUP]] PATH...]
[-copyFromLocal [-f] [-p] [-l] [-d] <localsrc> ... <dst>]
[-copyToLocal [-f] [-p] [-ignoreCrc] [-crc] <src> ... <localdst>]
[-count [-q] [-h] [-v] [-t <storage type>]] [-u] [-x] <path> ...]
[-cp [-f] [-p | -p[topax]] [-d] <src> ... <dst>]
[-createSnapshot <snapshotDir> [<snapshotName>]]
[-deleteSnapshot <snapshotDir> <snapshotName>]
[-df [-h] [<path> ...]]
[-du [-s] [-h] [-x] <path> ...]
[-expunge]
[-find <path> ... <expression> ...]
[-get [-f] [-p] [-ignoreCrc] [-crc] <src> ... <localdst>]
[-getfacl [-R] <path>]
[-getfattr [-R] {-n name | -d} [-e en] <path>]
[-getmerge [-nl] [-skip-empty-file] <src> <localdst>]
[-help [cmd ...]]
[-ls [-C] [-d] [-h] [-q] [-R] [-t] [-S] [-r] [-u] [<path> ...]]
[-mkdir [-p] <path> ...]
[-moveFromLocal <localsrc> ... <dst>]
[-moveToLocal <src> <localdst>]
[-mv <src> ... <dst>]
[-put [-f] [-p] [-l] [-d] <localsrc> ... <dst>]
[-renameSnapshot <snapshotDir> <oldName> <newName>]
[-rm [-f] [-r|-R] [-skipTrash] [-safely] <src> ...]
```

```
[-rmdir [--ignore-fail-on-non-empty] <dir> ...]
[-setfacl [-R] [{-b|-k} {-m|-x <acl_spec>} <path>][[--set <acl_spec> <path>]]
[-setfattr {-n name [-v value] | -x name} <path>]
[-setrep [-R] [-w] <rep> <path> ...]
[-stat [format] <path> ...]
[-tail [-f] <file>]
[-test -[defsz] <path>]
[-text [-ignoreCrc] <src> ...]
[-touchz <path> ...]
[-truncate [-w] <length> <path> ...]
[-usage [cmd ...]]
```

Generic options supported are:

```
-conf <configuration file>    specify an application configuration file
-D <property=value>           define a value for a given property
-fs <file:///hdfs://namenode:port> specify default filesystem URL to use, overrides
'fs.defaultFS' property from configurations.
-jt <local|resourcemanager:port> specify a ResourceManager
-files <file1,...>            specify a comma-separated list of files to be copied to the
map reduce cluster
-libjars <jar1,...>           specify a comma-separated list of jar files to be included
in the classpath
-archives <archive1,...>      specify a comma-separated list of archives to be
unarchived on the compute machines
```

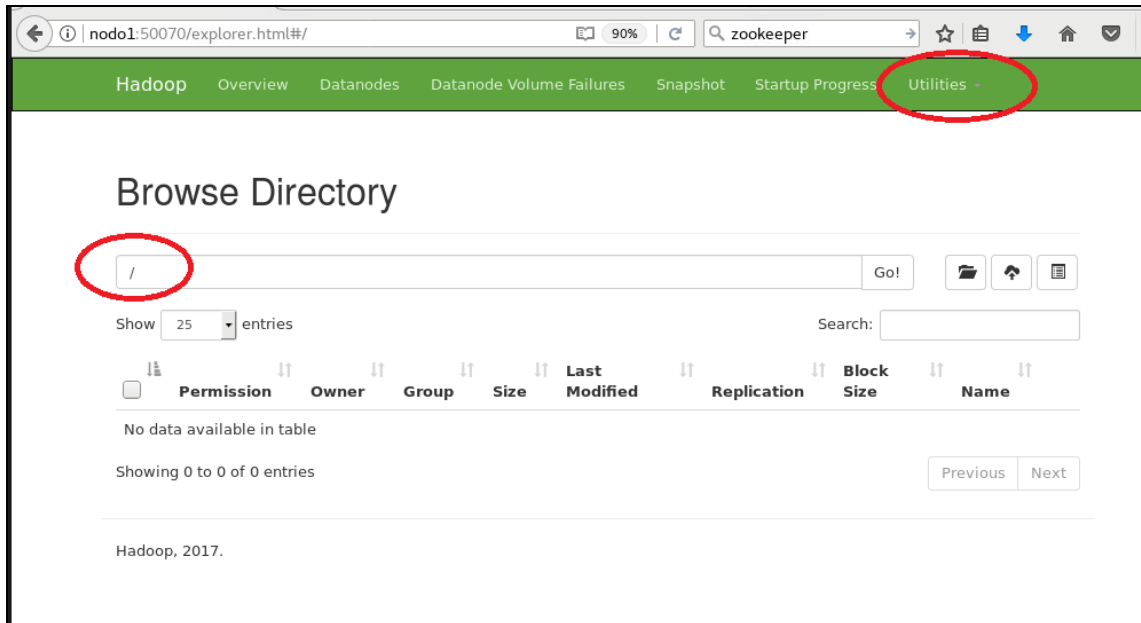
The general command line syntax is:

```
command [genericOptions] [commandOptions]
```

- Vamos a ver el contenido de nuestro HDFS. En principio debe estar vacío

```
hdfs dfs -ls /
```

- También podemos ver que está vacío desde la web de administración en el menú Utilities → Browse the File System



- Vamos a crear un nuevo directorio

```
hdfs dfs -mkdir /datos
```

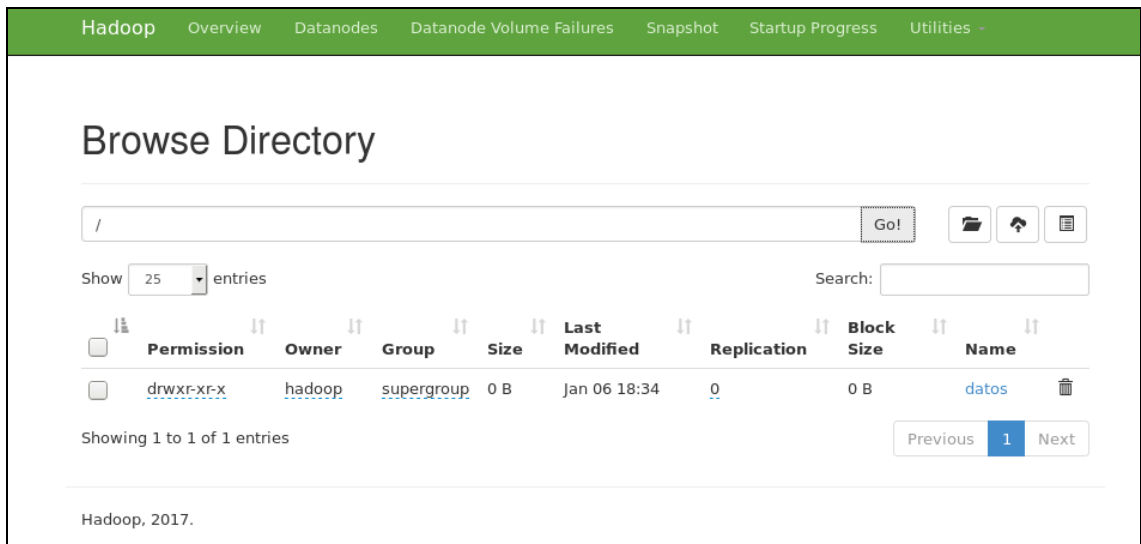
- Comprobar que existe

```
hdfs dfs -ls /
```

Found 1 items

```
drwxr-xr-x - hadoop supergroup 0 2018-01-06 18:31 /datos
```

- Podemos verlo en la página WEB



- Creamos un fichero en el directorio /tmp con alguna frase

```
echo "Esto es una prueba" >/tmp/prueba.txt
```

- Copiarlo al HDFS, en concreto al directorio /datos. Usamos el comando "put"

```
hdfs dfs -put /tmp/prueba.txt /datos
```

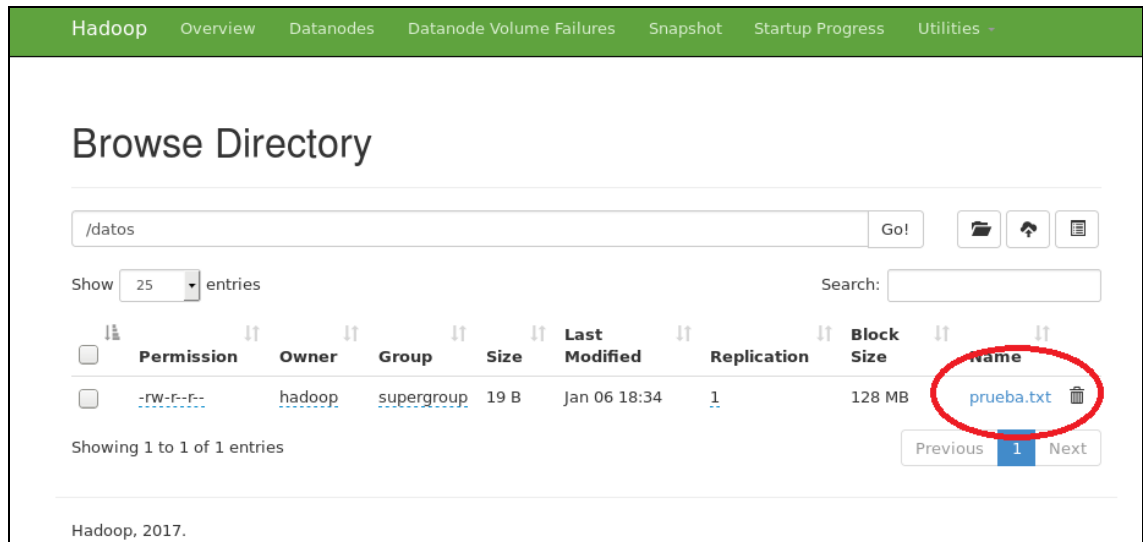
- Comprobar su existencia

```
hdfs dfs -ls /datos
```

Found 1 items




```
-rw-r--r--  1 hadoop supergroup      19 2018-01-06 18:34 /datos/prueba.txt
```

- También podemos verlo en la página web. Podemos comprobar el tipo de replicación que tiene y el tamaño correspondiente.




Hadoop Overview Datanodes Datanode Volume Failures Snapshot Startup Progress Utilities -

Browse Directory

/datos Go!   

Show 25 entries Search:

<input type="checkbox"/>	Permission	Owner	Group	Size	Last Modified	Replication	Block Size	Name
<input type="checkbox"/>	-rw-r--r--	hadoop	supergroup	19 B	Jan 06 18:34	1	128 MB	prueba.txt 

Showing 1 to 1 of 1 entries Previous 1 Next

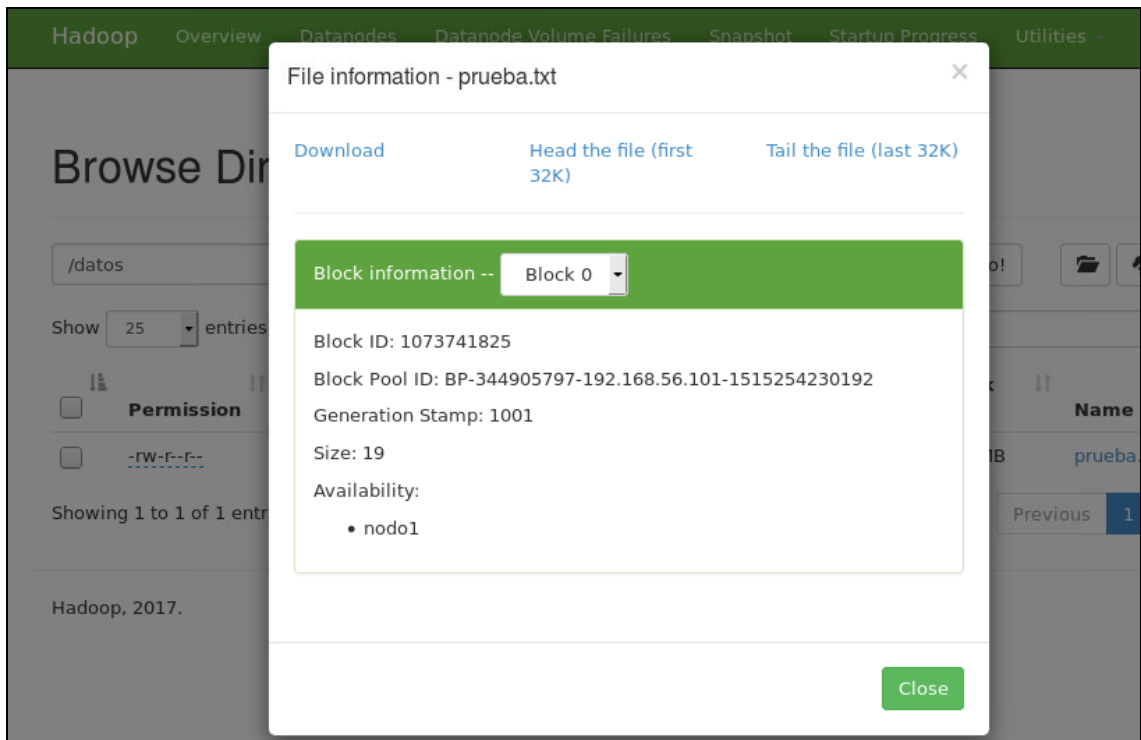
Hadoop, 2017.

- Visualizar su contenido

```
hdfs dfs -cat /datos/prueba.txt
```

Esto es una prueba

- Vamos a comprobar lo que ha creado a nivel de HDFS
- Vamos a la página WEB y pulsamos en el nombre del fichero.
- Debe aparecer algo parecido a lo siguiente



- Vemos que solo ha creado un bloque, ya que el BLOCK SIZE por defecto de HDFS es 128M y por lo tanto nuestro pequeño fichero solo genera uno.
- Además, nos dice el BLOCK_ID y también los nodos donde ha creado las réplicas. Como tenemos un replication de 1, solo aparece el nodo1. Cuando veamos la parte del cluster completo veremos más nodos
- Volvemos al sistema operativo y nos vamos al directorio siguiente. Evidentemente el subdirectorio BP-XXXX será distinto en tu caso. Se corresponde con el Block Pool ID que genera de forma automática Hadoop.

```
/datos/datanode/current/BP-344905797-192.168.56.101-1515254230192/current/finalized
```

- Dentro de este subdirectorio, Hadoop irá creando una estructura de subdirectorios donde albergará los bloques de datos, con el formato subdirN/subdirN, en este caso subdir0/subdir0.
- Entramos en él.

```
cd subdir0/
cd subdir0/
ls -l
total 8
-rw-rw-r--. 1 hadoop hadoop 19 ene  6 18:34 blk_1073741825
-rw-rw-r--. 1 hadoop hadoop 11 ene  6 18:34 blk_1073741825_1001.meta
```

- Podemos comprobar que hay dos ficheros con el mismo BLOCK_ID que aparece en la página WEB.
 - Uno contiene los datos

- El otro contiene metadatos
- Podemos comprobarlo si visualizamos el contenido

```
cat blk_1073741825
```

Esto es una prueba

- Evidentemente, cuando tengamos ficheros muy grandes o que no sean texto, esto no es de ninguna utilidad. Solo lo hacemos para entender bien HDFS.
- Vamos a crear otro ejemplo con un fichero grande
- Lanzamos este comando para generar un fichero de 1G en /tmp, llamado fic_grande.dat, lleno de ceros (el comando dd de Linux permite hacer esto entre otras muchas cosas)

```
dd if=/dev/zero of=/tmp/fic_grande.dat bs=1024 count=1000000
```

1000000+0 registros leídos

1000000+0 registros escritos

1024000000 bytes (1,0 GB) copiados, 5,1067 s, 201 MB/s

- Lo subimos al directorio /datos de nuestro HDFS

```
hdfs dfs -put /tmp/fic_grande.dat /datos
```

- Podemos comprobar en la página web que ha creado múltiples bloques de 128MB

The screenshot shows the HDFS web interface. A modal window titled 'File information - fic_grande.dat' is open. It displays file details: Block ID: 107374182, Block Pool ID: BP-34, Generation Stamp: 1, Size: 134217728, and Availability: nodo1. A dropdown menu for 'Block 0' is open, showing a list of blocks from Block 0 to Block 7. A red arrow points from the file name 'fic_grande.dat' in the background file list to the 'Block 0' dropdown menu.

- Si comprobamos de nuevo el directorio subdir0 podemos ver los bloques correspondientes

```
ls -l
total 1007852
-rw-rw-r--. 1 hadoop hadoop    19 ene  6 18:34 blk_1073741825
-rw-rw-r--. 1 hadoop hadoop    11 ene  6 18:34 blk_1073741825_1001.meta
-rw-rw-r--. 1 hadoop hadoop 134217728 ene  6 18:59 blk_1073741826
-rw-rw-r--. 1 hadoop hadoop 1048583 ene  6 18:59 blk_1073741826_1002.meta
-rw-rw-r--. 1 hadoop hadoop 134217728 ene  6 18:59 blk_1073741827
-rw-rw-r--. 1 hadoop hadoop 1048583 ene  6 18:59 blk_1073741827_1003.meta
-rw-rw-r--. 1 hadoop hadoop 134217728 ene  6 18:59 blk_1073741828
-rw-rw-r--. 1 hadoop hadoop 1048583 ene  6 18:59 blk_1073741828_1004.meta
-rw-rw-r--. 1 hadoop hadoop 134217728 ene  6 18:59 blk_1073741829
-rw-rw-r--. 1 hadoop hadoop 1048583 ene  6 18:59 blk_1073741829_1005.meta
-rw-rw-r--. 1 hadoop hadoop 134217728 ene  6 18:59 blk_1073741830
-rw-rw-r--. 1 hadoop hadoop 1048583 ene  6 18:59 blk_1073741830_1006.meta
-rw-rw-r--. 1 hadoop hadoop 134217728 ene  6 18:59 blk_1073741831
-rw-rw-r--. 1 hadoop hadoop 1048583 ene  6 18:59 blk_1073741831_1007.meta
-rw-rw-r--. 1 hadoop hadoop 134217728 ene  6 18:59 blk_1073741832
-rw-rw-r--. 1 hadoop hadoop 1048583 ene  6 18:59 blk_1073741832_1008.meta
-rw-rw-r--. 1 hadoop hadoop 84475904 ene  6 19:00 blk_1073741833
-rw-rw-r--. 1 hadoop hadoop 659975 ene  6 19:00 blk_1073741833_1009.meta
```

- Vamos a crear otro directorio llamado “practicass”

```
hdfs dfs -mkdir /practicass
```

- Copiamos prueba.txt desde datos a prácticas

```
hdfs dfs -cp /datos/prueba.txt /practicass/prueba.txt
```

- Comprobamos el contenido

```
hdfs dfs -ls /practicass
Found 1 items
-rw-r--r-- 1 hadoop supergroup    19 2018-01-06 19:08 /practicass/prueba.txt
```

- Borramos el fichero

```
hdfs dfs -rm /practicass/prueba.txt
Deleted /practicass/prueba.txt
```

- Vemos que los comandos son muy parecidos a Linux

4.2. Nuestro primer proceso Hadoop

- Vamos a ejecutar nuestro primer trabajo hadoop. Luego veremos con más detalle esto.
- Hadoop tiene una serie de ejemplos que se encuentran en el fichero siguiente (recordad el número de versión)

```
/opt/hadoop/share/hadoop/mapreduce/hadoop-mapreduce-examples-2.9.0.jar
```

- Para lanzar un proceso hadoop Map Reduce usamos el comando

```
hadoop jar librería.jar proceso
```

- En este caso, si queremos ver los programas que hay en ese “jar” ponemos lo siguiente, sin poner el comando final

```
hadoop jar /opt/hadoop/share/hadoop/mapreduce/hadoop-mapreduce-examples-2.9.0.jar
```

An example program must be given as the first argument.

Valid program names are:

aggregatewordcount: An Aggregate based map/reduce program that counts the words in the input files.

aggregatewordhist: An Aggregate based map/reduce program that computes the histogram of the words in the input files.

bbp: A map/reduce program that uses Bailey-Borwein-Plouffe to compute exact digits of Pi.

dbcount: An example job that count the pageview counts from a database.

distbbp: A map/reduce program that uses a BBP-type formula to compute exact bits of Pi.

grep: A map/reduce program that counts the matches of a regex in the input.

join: A job that effects a join over sorted, equally partitioned datasets

multifilewc: A job that counts words from several files.

pentomino: A map/reduce tile laying program to find solutions to pentomino problems.

pi: A map/reduce program that estimates Pi using a quasi-Monte Carlo method.

randomtextwriter: A map/reduce program that writes 10GB of random textual data per node.

randomwriter: A map/reduce program that writes 10GB of random data per node.

secondarysort: An example defining a secondary sort to the reduce.

sort: A map/reduce program that sorts the data written by the random writer.

sudoku: A sudoku solver.

teragen: Generate data for the terasort

terasort: Run the terasort

teravalidate: Checking results of terasort

wordcount: A map/reduce program that counts the words in the input files.

wordmean: A map/reduce program that counts the average length of the words in the input files.

wordmedian: A map/reduce program that counts the median length of the words in the input files.

wordstandarddeviation: A map/reduce program that counts the standard deviation of the length of the words in the input files.

- Vemos que hay un comando llamado “wordcount”.
- Permite contar las palabras que hay en uno o varios ficheros.
- Creamos un par de ficheros con palabras (algunas repetidas) y lo guardamos en ese directorio

```
hdfs dfs -put /tmp/palabras.txt /practicass
hdfs dfs -put /tmp/palabras1.txt /practicass
```

- Lanzamos el comando

```
hadoop jar /opt/hadoop/share/hadoop/mapreduce/hadoop-mapreduce-examples-2.9.0.jar wordcount /practicass /salida1
```

INFO mapreduce.Job: Counters: 38

File System Counters

```
FILE: Number of bytes read=812740
FILE: Number of bytes written=1578775
FILE: Number of read operations=0
FILE: Number of large read operations=0
FILE: Number of write operations=0
HDFS: Number of bytes read=211
HDFS: Number of bytes written=74
HDFS: Number of read operations=25
HDFS: Number of large read operations=0
HDFS: Number of write operations=5
```

Map-Reduce Framework

```
Map input records=2
Map output records=16
Map output bytes=147
Map output materialized bytes=191
Input split bytes=219
Combine input records=16
Combine output records=16
Reduce input groups=10
Reduce shuffle bytes=191
Reduce input records=16
Reduce output records=10
Spilled Records=32
Shuffled Maps =2
```

Failed Shuffles=0
Merged Map outputs=2
GC time elapsed (ms)=131
CPU time spent (ms)=0
Physical memory (bytes) snapshot=0
Virtual memory (bytes) snapshot=0
Total committed heap usage (bytes)=549138432
Shuffle Errors
BAD_ID=0
CONNECTION=0
IO_ERROR=0
WRONG_LENGTH=0
WRONG_MAP=0
WRONG_REDUCE=0
File Input Format Counters
Bytes Read=84
File Output Format Counters
Bytes Written=74

- Podemos ver el contenido del directorio

hdfs dfs -ls /salida
Found 2 items
-rw-r--r-- 1 hadoop supergroup 0 2015-04-20 07:52 /salida/_SUCCESS
-rw-r--r-- 1 hadoop supergroup 74 2015-04-20 07:52 /salida/part-r-00000
[hadoop@localhost ~]\$ hadoop fs -cat /salida/part-r-00000
Esto 1
con 2
el 2
es 2
esto 1
fichero 2
primer 1
prueba 2
segundo 1
una 2

5. Administración de HDFS

- Realizar un report del sistema HDFS

hdfs dfsadmin -report

Configured Capacity: 50940104704 (47.44 GB)
 Present Capacity: 44860829696 (41.78 GB)
 DFS Remaining: 44859387904 (41.78 GB)
 DFS Used: 1441792 (1.38 MB)
 DFS Used%: 0.00%
 Under replicated blocks: 0
 Blocks with corrupt replicas: 0
 Missing blocks: 0

----- Live datanodes (1):

Name: 127.0.0.1:50010 (localhost)
 Hostname: localhost
 Decommission Status : Normal
 Configured Capacity: 50940104704 (47.44 GB)
 DFS Used: 1441792 (1.38 MB)
 Non DFS Used: 6079275008 (5.66 GB)
 DFS Remaining: 44859387904 (41.78 GB)
 DFS Used%: 0.00%
 DFS Remaining%: 88.06%
 Configured Cache Capacity: 0 (0 B)
 Cache Used: 0 (0 B)
 Cache Remaining: 0 (0 B)
 Cache Used%: 100.00%
 Cache Remaining%: 0.00%
 Xceivers: 1
 Last contact: Mon Apr 20 08:17:15 CEST 2015

- Comprobar con hdfs fsck el estado del sistema de ficheros

hdfs fsck /

Connecting to namenode via http://localhost:50070
 FSCK started by hadoop (auth:SIMPLE) from /127.0.0.1 for path / at Mon Apr 20 08:19:57 CEST 2015

```
.....Status: HEALTHY
Total size: 1380389 B
Total dirs: 6
Total files: 6
Total symlinks: 0
Total blocks (validated): 5 (avg. block size 276077 B)
Minimally replicated blocks: 5 (100.0 %)
Over-replicated blocks: 0 (0.0 %)
Under-replicated blocks: 0 (0.0 %)
Mis-replicated blocks: 0 (0.0 %)
Default replication factor: 1
Average block replication: 1.0
Corrupt blocks: 0
Missing replicas: 0 (0.0 %)
Number of data-nodes: 1
Number of racks: 1
FSCK ended at Mon Apr 20 08:19:57 CEST 2015 in 14 milliseconds

The filesystem under path '/' is HEALTHY
```

- También Podemos comprobar el estado de un determinado directorio, por ejemplo de prácticas

```
hdfs fsck /practicass
Connecting to namenode via
http://localhost:50070/fsck?ugi=hadoop&path=%2Fpracticass
FSCK started by hadoop (auth:SIMPLE) from /127.0.0.1 for path /practicass at Sat
Jan 06 19:55:19 CET 2018
...Status: HEALTHY
Total size: 2647821 B
Total dirs: 2
Total files: 3
Total symlinks: 0
Total blocks (validated): 2 (avg. block size 1323910 B)
Minimally replicated blocks: 2 (100.0 %)
Over-replicated blocks: 0 (0.0 %)
Under-replicated blocks: 0 (0.0 %)
Mis-replicated blocks: 0 (0.0 %)
Default replication factor: 1
```

```
Average block replication: 1.0
Corrupt blocks: 0
Missing replicas: 0 (0.0 %)
Number of data-nodes: 1
Number of racks: 1
FCK ended at Sat Jan 06 19:55:19 CET 2018 in 7 milliseconds
```

- Podemos ver la topología que tenemos en este momento

```
hdfs dfsadmin -printTopology
```

```
Rack: /default-rack
```

```
127.0.0.1:50010 (localhost)
```

- Podemos comprobar si hay algún fichero abierto

```
hdfs dfsadmin -listOpenFiles
```

Client Host	Client Name	Open File Path
-------------	-------------	----------------

-

6. Snapshots

- Creamos un pequeño fichero en /tmp

```
echo Ejemplo de Snapshot > /tmp/f1.txt
```

- Creamos un directorio HDFS para probar

```
hadoop fs -mkdir /datos4
```

- Subimos el fichero pequeño que hayamos creado

```
hdfs dfs -put f1.txt /datos4
```

- Ejecutamos un fsck sobre el fichero. Podemos preguntar información de bloques, ficheros, nodos donde se encuentra, etc... una opción muy interesante.

```
hdfs fsck /datos4/f1.txt -files -blocks -locations
```

```
Connecting to namenode via
http://localhost:50070/fsck?ugi=hadoop&files=1&blocks=1&locations=1&path=
%2Fdatos4%2Ff1.txt
```

```
FSCK started by hadoop (auth:SIMPLE) from /127.0.0.1 for path /datos4/f1.txt
at Sat Jan 06 20:16:02 CET 2018
```

```
/datos4/f1.txt 20 bytes, 1 block(s): OK
```

```
0. BP-344905797-192.168.56.101-1515254230192:blk_1073741837_1013
len=20 Live_repl=1 [DatanodeInfoWithStorage[127.0.0.1:50010,DS-173cc83b-
694a-425e-ad0f-c4c86352e2f6,DISK]]
```

```
Status: HEALTHY
```

```
Total size: 20 B
```

```
Total dirs: 0
```

```
Total files: 1
```

```
Total symlinks: 0
```

```
Total blocks (validated): 1 (avg. block size 20 B)
```

```
Minimally replicated blocks: 1 (100.0 %)
```

```
Over-replicated blocks: 0 (0.0 %)
```

```
Under-replicated blocks: 0 (0.0 %)
```

```
Mis-replicated blocks: 0 (0.0 %)
```

```
Default replication factor: 1
```

```
Average block replication: 1.0
```

```
Corrupt blocks: 0
```

```
Missing replicas: 0 (0.0 %)
```

```
Number of data-nodes: 1
```

```
Number of racks: 1
```

FSCK ended at Sat Jan 06 20:16:02 CET 2018 in 1 milliseconds
The filesystem under path '/datos4/f1.txt' is HEALTHY

- Buscamos el fichero en el Sistema de ficheros del Linux a partir de su número de bloque

```
find /datos/datanode -name "blk_1073741837"  
/datos/datanode/current/BP-344905797-192.168.56.101-  
1515254230192/current/finalized/subdir0/subdir0/blk_1073741837
```

- Habilitamos los snapshot sobre el directorio /datos4

```
hdfs dfsadmin -allowSnapshot /datos4  
Allowing snapshot on /datos4 succeeded
```

- Creamos un snapshot llamado "s1" en el directorio

```
hdfs dfs -createSnapshot /datos4 s1  
Created snapshot /datos4/.snapshot/s1
```

- Comprobamos que se ha creado satisfactoriamente

```
hdfs dfs -ls /datos4/.snapshot  
Found 1 items  
drwxr-xr-x - hadoop supergroup 0 2018-01-06 20:20 /datos4/.snapshot/s1
```

- Si hacemos un ls, en principio debe tener lo mismo que su directorio asociado

```
hdfs dfs -ls /datos4/.snapshot/s1  
Found 1 items  
-rw-r--r-- 1 hadoop supergroup 20 2018-01-06 20:15  
/datos4/.snapshot/s1/f1.txt
```

- Vamos a borrar el fichero f1.txt

```
hdfs dfs -rm /datos4/f1.txt  
Deleted /datos4/f1.txt
```

- Podemos comprobar que ya no existe

```
hdfs dfs -ls /datos4
```

- Sin embargo, con Snapshots, es muy fácil recuperarlo, sencillamente lo copiamos de nuevo a su sitio original

```
hadoop fs -cp /datos4/.snapshot/s1/f1.txt /datos4/  
  
hdfs dfs -ls /datos4  
Found 1 items  
-rw-r--r-- 1 hadoop supergroup 20 2018-01-06 20:25 /datos4/f1.txt
```

7. YARN EN ENTORNOS PSEUDO-DISTRIBUIDO

- Copiamos el fichero “mapred-site.xml.template” a “mapred-site.xml”
- Ponemos la siguiente propiedad.

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

- Y en el fichero yarn-site.xml ponemos las siguientes:

```
<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>
```

- Arrancamos HDFS si no lo está
- Arrancamos los servicios YARN

```
start-yarn.sh
```

- Comprobamos con jps que tenemos los dos procesos de YARN
- Arrancamos HDFS si no lo está
- Arrancamos los servicios YARN
- Start-yarn.sh
- En el caso de tener hadoop versión 2 ejecutamos el comando

```
/mr-jobhistory-daemon.sh starthistoryserver
```

- En el caso de tener hadoop 3 ejecutamos el comando

```
mapred --daemon start historyserver
```

- Comprobamos con “jps” que tenemos los procesos lanzados

-
- Arrancamos HDFS si no lo está
- Arrancamos los servicios YARN

```
Start-yarn.sh
```

- Comprobamos con jps que está funcionando

```
jps
1137 Jps
22962 DataNode
23187 SecondaryNameNode
419 ResourceManager
22837 NameNode
541 NodeManager
```


- Arrancamos el servicio que permite guardar el histórico de los Jobs lanzados

```
mr-jobhistory-daemon.sh start historyserver
starting historyserver, logging to /opt/hadoop/logs/mapred-hadoop-historyserver-
nodo1.out
```

- Comprobamos con jps que está funcionando

```
jps
22962 DataNode
23187 SecondaryNameNode
419 ResourceManager
1475 JobHistoryServer
22837 NameNode
1515 Jps
541 NodeManager
```

- Accedemos a la página WEB de Administración para comprobar que está funcionando.



All Applications

Cluster

- About
- Nodes
- Node Labels
- Applications
- NEW
- NEW SAVING
- SUBMITTED
- ACCEPTED
- RUNNING
- FINISHED
- FAILED
- KILLED
- Scheduler

Tools

Cluster Metrics

Apps Submitted	Apps Pending	Apps Running	Apps Completed	Containers Running	Memory Used
0	0	0	0	0	0 B

Cluster Nodes Metrics

Active Nodes	Decommissioning Nodes	Decommissioned Nodes	Lost Nodes
1	0	0	0

Scheduler Metrics


Scheduler Type	Scheduling Resource Type	Minimum Allocation
Capacity Scheduler	[MEMORY]	<memory:1024, vCores:1>

Show 20 entries

ID	User	Name	Application Type	Queue	Application Priority	StartTime	FinishTime	State	FinalStatus	Running Containers
No data available in table										

Showing 0 to 0 of 0 entries

- Comprobamos en Nodes que tenemos el nodo activo



Nodes of the cluster

Cluster

- About
- Nodes
- Node Labels
- Applications
- NEW
- NEW SAVING
- SUBMITTED
- ACCEPTED
- RUNNING
- FINISHED
- FAILED
- KILLED
- Scheduler

Tools

Cluster Metrics

Apps Submitted	Apps Pending	Apps Running	Apps Completed	Containers Running	Memory Used	Memory Total	Memory Reserved	VCores Used
0	0	0	0	0	0 B	8 GB	0 B	0

Cluster Nodes Metrics

Active Nodes	Decommissioning Nodes	Decommissioned Nodes	Lost Nodes	Unhealthy Nodes	Rebooted
1	0	0	0	0	0

Scheduler Metrics

Scheduler Type	Scheduling Resource Type	Minimum Allocation	Maximum Allocation	Maximum VCore
Capacity Scheduler	[MEMORY]	<memory:1024, vCores:1>	<memory:8192, vCores:4>	0

Show 20 entries

Node Labels	Rack	Node State	Node Address	Node HTTP Address	Last health-update	Health-report	Containers	Mem Used	Mem Avail
/default-rack	RUNNING	nodo1:42534	nodo1:8042	sáb ene 06 22:11:26 +0100 2018		0	0 B	8 GB	

8. MapReduce

- Vamos a subir al directorio prácticas un fichero denominado “quijote.txt” que contiene el Quijote. Lo tienes disponible en los recursos de las prácticas. Lo más sencillo es que lo descargues desde la propia máquina virtual

```
hdfs dfs -put /home/hadoop/Descargas/quijote.txt /practicass
```

- NOTA IMPORTANTE:** Aquellos que estáis usando **Hadoop 3**, es posible que el siguiente ejemplo no funcione correctamente. En ese caso tenemos que añadir al fichero yarn-site.xml el siguiente contenido. Por supuesto adaptarlo a vuestro HADOOP_PATH, es decir, yo tengo el directorio hadoop3, pero hay que poner el que tengáis vosotros

```
<property>
<name>yarn.application.classpath</name>
<value>
    /opt/hadoop3/hadoop/etc/hadoop,
    /opt/hadoop3/share/hadoop/common/*,
    /opt/hadoop3/share/hadoop/common/lib/*,
    /opt/hadoop3/share/hadoop/hdfs/*,
    /opt/hadoop3/share/hadoop/hdfs/lib/*,
    /opt/hadoop3/share/hadoop/mapreduce/*,
    /opt/hadoop3/share/hadoop/mapreduce/lib/*,
    /opt/hadoop3/share/hadoop/yarn/*,
    /opt/hadoop3/share/hadoop/yarn/lib/*
</value>
</property>
```

- NOTA IMPORTANTE**
- Es posible que dependiendo de la máquina, a alguno de vosotros os arroje este error de memoria virtual (código 143), algo parecido a lo siguiente

```
Current usage: 449 MB of 1 GB physical memory used; 2.6 GB of 2.1 GB
virtual memory used. Killing container
```

- En ese caso, hay que poner la siguiente propiedad dentro del yarn-site.xml que “engaña” a hadoop. Evidentemente, lo ideal sería poder disponer de toda esa memoria, pero con esto podéis ejecutar el programa

```
<property>
<name>
    yarn.nodemanager.vmem-check-enabled
</name>
<value>>false</value>
<description>
    Whether virtual memory limits will be enforced for containers.
</description>
```

</property>

- Lanzamos el wordcount contra el fichero. Indicamos el directorio de salida donde dejar el resultado, en este caso en /practicas/resultado (siempre en HDFS)

```
hadoop jar /opt/hadoop/share/hadoop/mapreduce/hadoop-mapreduce-examples-2.9.0.jar wordcount /practicas/quijote.txt /practicas/resultado
```

```
8/01/06 19:29:24 INFO Configuration.deprecation: session.id is deprecated.
Instead, use dfs.metrics.session-id
```

```
18/01/06 19:29:24 INFO jvm.JvmMetrics: Initializing JVM Metrics with
processName=JobTracker, sessionId=
```

```
18/01/06 19:29:26 INFO input.FileInputFormat: Total input files to process : 1
```

```
18/01/06 19:29:27 INFO mapreduce.JobSubmitter: number of splits:1
```

```
18/01/06 19:29:28 INFO mapreduce.JobSubmitter: Submitting tokens for job:
job_local382862986_0001
```

```
18/01/06 19:29:28 INFO mapreduce.Job: The url to track the job:
http://localhost:8080/
```

```
18/01/06 19:29:28 INFO mapreduce.Job: Running job:
job_local382862986_0001
```

```
18/01/06 19:29:28 INFO mapred.LocalJobRunner: OutputCommitter set in
config null
```

```
18/01/06 19:29:28 INFO output.FileOutputCommitter: File Output Committer
Algorithm version is 1
```

```
18/01/06 19:29:28 INFO output.FileOutputCommitter: FileOutputCommitter
skip cleanup _temporary folders under output directory:false, ignore cleanup
failures: false
```

```
18/01/06 19:29:28 INFO mapred.LocalJobRunner: OutputCommitter is
org.apache.hadoop.mapreduce.lib.output.FileOutputCommitter
```

```
.....
```

```
.....
```

```
.....
```

```
8/01/06 19:29:35 INFO mapreduce.Job: Job job_local382862986_0001
completed successfully
```

```
18/01/06 19:29:35 INFO mapreduce.Job: Counters: 35
```

```
File System Counters
```

```
FILE: Number of bytes read=1818006
```

```
FILE: Number of bytes written=3374967
```

```
FILE: Number of read operations=0
```

```
FILE: Number of large read operations=0
```

```
FILE: Number of write operations=0
```

```
HDFS: Number of bytes read=4397854
```

www.apasoft-training.com

apasoft.training@gmail.com

HDFS: Number of bytes written=448894

HDFS: Number of read operations=13

HDFS: Number of large read operations=0

HDFS: Number of write operations=4

Map-Reduce Framework

Map input records=37861

Map output records=384260

Map output bytes=3688599

Map output materialized bytes=605509

Input split bytes=108

Combine input records=384260

Combine output records=40059

Reduce input groups=40059

Reduce shuffle bytes=605509

Reduce input records=40059

Reduce output records=40059

Spilled Records=80118

Shuffled Maps =1

Failed Shuffles=0

Merged Map outputs=1

GC time elapsed (ms)=100

Total committed heap usage (bytes)=331489280

Shuffle Errors

BAD_ID=0

CONNECTION=0

IO_ERROR=0

WRONG_LENGTH=0

WRONG_MAP=0

WRONG_REDUCE=0

File Input Format Counters

Bytes Read=2198927

File Output Format Counters

Bytes Written=448894

- Vemos que nos hace un resumen del resultado
- Podemos ver el contenido del directorio

hdfs dfs -ls /practicass/resultado

Found 2 items

```
-rw-r--r--      1 hadoop supergroup          0 2018-01-06 19:29
/practicass/resultado/_SUCCESS
-rw-r--r--      1 hadoop supergroup    448894 2018-01-06 19:29
/practicass/resultado/part-r-00000
```

- Podemos traerlo desde HDFS al Linux con el comando “get” y lo dejamos en /tmp con otro nombre

```
hdfs dfs -get /practicass/resultado/part-r-00000 /tmp/palabras_quijote.txt
```

Con “vi” podemos ver el contenido

```
Mal      1
"Al      1
"Cuando  2
"Cuidados      1
"De      2
"Defects, "    1
"Desnudo      1
"Dijo      1
"Dime      1
"Don      1
"Donde     1
"Dulcinea      1
"El      2
"Esta     1
"Harto     1
"Iglesia,    1
"Information  1
"Más      2
"No      5
"Nunca     1
"Plain     2
"Project    5
"Que      1
"Quien     1
"Right     1
"Salta     1
```

```
"Sancho 1
"Si 3
"Tened 1
"Toda 1
"Vengan 1
"Vete, 1
"/tmp/palabras_quijote.txt" 40059L, 448894C
```

- Accedemos a la WEB de Administración de YARN.
- Si seleccionamos la opción “Applications” podemos ver la aplicación que acabamos de lanzar

The screenshot shows the Hadoop YARN web interface at localhost:8088/cluster/apps. The 'Applications' menu item is highlighted in the left sidebar. A red arrow points to the application entry 'application_1515272962334_0001' in the table. The application's state is 'FINISHED' and 'SUCCEEDED'.

ID	User	Name	Application Type	Queue	Application Priority	StartTime	FinishTime	State	FinalStatus
application_1515272962334_0001	hadoop	word count	MAPREDUCE	default	0	Sat Jan 6 22:31:42 +0100 2018	Sat Jan 6 22:32:23 +0100 2018	FINISHED	SUCCEEDED

- A la derecha de la aplicación, si pulsamos sobre “history”, podremos ver el detalle completo de la aplicación

The screenshot shows the Hadoop YARN web interface at localhost:8088/cluster/apps. The 'History' link is highlighted in the bottom right corner of the application entry.

ID	User	Name	Application Type	Queue	Application Priority	StartTime	FinishTime	State	FinalStatus	Running Containers	Allocated CPU VCoers	Allocated Memory MB	Reserved CPU VCoers	Reserved Memory MB	% of Queue	% of Cluster	Progress	Tracking	Black Node
application_1515272962334_0001	hadoop	word count	MAPREDUCE	default	0	Sat Jan 6 22:31:42 +0100 2018	Sat Jan 6 22:32:23 +0100 2018	FINISHED	SUCCEEDED	N/A	N/A	N/A	N/A	N/A	0,0	0,0		History	0

- Podemos ver información muy valiosa
-

TIEMPO TRANSCURRIDO

NODO DEL APPLICATION MASTER

MAPPERS Y REDUCERS

- Seleccionando un mapper o un reducer podemos acceder a su información: nodo en el que se ha ejecutado, etc...

Show 20 entries

Attempt	State	Status	Node	Logs	Start Time	Finish Time	Elapsed Time
attempt_1515272962334_0001_m_000000_0	SUCCEEDED	map	/default-rack/nodo1:8042	logs	Sat Jan 6 22:32:00 +0100 2018	Sat Jan 6 22:32:11 +0100 2018	11sec

Showing 1 to 1 of 1 entries

9. Crear un programa MapReduce

- Vamos a crear un programa desde Cero. En concreto el programa “wordcount” que ya hemos usado a lo largo del curso en alguna ocasión, pero le vamos a cambiar de nombre y compilar....
- Primero configuramos el entorno, para poder acceder a las librerías de compilación de Java.

```
export HADOOP_CLASSPATH=$JAVA_HOME/lib/tools.jar
```

- Creamos el programa con el vi o cualquier otro editor. Lo llamamos ContarPalabras.java. También lo tenéis en los recursos de la lección.

```
import java.io.IOException;
import java.util.StringTokenizer;
import org.apache.hadoop.conf.Configuration;
import org.apache.hadoop.fs.Path;
```



```
import org.apache.hadoop.io.IntWritable;
import org.apache.hadoop.io.Text;
import org.apache.hadoop.mapreduce.Job;
import org.apache.hadoop.mapreduce.Mapper;
import org.apache.hadoop.mapreduce.Reducer;
import org.apache.hadoop.mapreduce.lib.input.FileInputFormat;
import org.apache.hadoop.mapreduce.lib.output.FileOutputFormat;
import org.apache.hadoop.util.GenericOptionsParser;

/**
 * <p>Contamos el número de palabras que aparecen en un documento usando
 * MapReduce. El código tiene un mapper,
 * reducer, y el programa principal.</p>
 *
 */

public class ContarPalabras {
    /**
     * <p>
     * El mapper extiende el nterface org.apache.hadoop.mapreduce.Mapper. Al
     * ejecutar Hadoop ,
     * se recibe cada línea del fichero de entrada como input
     * La función map devuelve por cada palabra (word) un (word,1) como salida. </p>
     */

    public static class TokenizerMapper
        extends Mapper<Object, Text, Text, IntWritable>{

        private final static IntWritable one = new IntWritable(1);
        private Text word = new Text();

        public void map(Object key, Text value, Context context
            ) throws IOException, InterruptedException {
            StringTokenizer itr = new StringTokenizer(value.toString());
            while (itr.hasMoreTokens()) {
                word.set(itr.nextToken());
                context.write(word, one);
            }
        }
    }
}
```

```

    }

    /*
    La función reduce recibe todos los valores que tienen la misma clave como
    entrada y devuelve la clave y el número de
    ocurrencias como salida
    */

    public static class IntSumReducer
    extends Reducer<Text,IntWritable,Text,IntWritable> {
    private IntWritable result = new IntWritable();
    public void reduce(Text key, Iterable<IntWritable> values,
    Context context
    ) throws IOException, InterruptedException {
    int sum = 0;
    for (IntWritable val : values) {
    sum += val.get();
    }
    result.set(sum);
    context.write(key, result);
    }
    }

    /**
    * La entrada es cualquier fichero
    */

    public static void main(String[] args) throws Exception {
    Configuration conf = new Configuration();
    String[] otherArgs = new GenericOptionsParser(conf,
    args).getRemainingArgs();
    if (otherArgs.length != 2) {
    System.err.println("Uso: ContarPalabras <in> <out>");
    System.exit(2);
    }
    Job job = Job.getInstance(conf, "word count");
    job.setJarByClass(ContarPalabras.class);
    job.setMapperClass(TokenizerMapper.class);

```

```

/**** Dejarlo tal cual ****/
//job.setCombinerClass(IntSumReducer.class);

job.setReducerClass(IntSumReducer.class);
job.setOutputKeyClass(Text.class);
job.setOutputValueClass(IntWritable.class);
FileInputFormat.addInputPath(job, new Path(otherArgs[0]));
FileOutputFormat.setOutputPath(job, new Path(otherArgs[1]));
System.exit(job.waitForCompletion(true) ? 0 : 1);
}
}

```

- Compilamos

```
hadoop com.sun.tools.javac.Main ContarPalabras.java
```

- Nos habrá generado 3 ficheros .class, uno para la clase principal, otro para el Mapper y otro para el Reducer

```

ls -l *.class
-rw-rw-r--. 1 hadoop hadoop 1846 ene  6 23:48 ContarPalabras.class
-rw-rw-r--.  1  hadoop  hadoop  1754  ene      6  23:48
ContarPalabras$IntSumReducer.class
-rw-rw-r--.  1  hadoop  hadoop  1751  ene      6  23:48
ContarPalabras$TokenizerMapper.class

```

- Generamos un JAR, denominado mi_librería con las tres clases

```
jar cf mi_libreria.jar Contar*.class
```

- Ejecutamos. Si todavía tenemos el fichero quijote.txt en el directorio /practicass lo usamos. Si no lo volvemos a subir

```

hadoop jar mi_libreria.jar ContarPalabras /practicass/quijote.txt
/resultado3

18/01/06 23:52:13 INFO client.RMProxy: Connecting to ResourceManager at
localhost/127.0.0.1:8032
18/01/06 23:52:14 INFO input.FileInputFormat: Total input files to process : 1
18/01/06 23:52:15 INFO mapreduce.JobSubmitter: number of splits:1
18/01/06 23:52:16 INFO Configuration.deprecation:
yarn.resourcemanager.system-metrics-publisher.enabled is deprecated. Instead,
use yarn.system-metrics-publisher.enabled
18/01/06 23:52:17 INFO mapreduce.JobSubmitter: Submitting tokens for job:
job_1515272962334_0003

```

```
18/01/06 23:52:17 INFO impl.YarnClientImpl: Submitted application
application_1515272962334_0003
18/01/06 23:52:17 INFO mapreduce.Job: The url to track the job:
http://nodo1:8088/proxy/application_1515272962334_0003/
18/01/06 23:52:17 INFO mapreduce.Job: Running job:
job_1515272962334_0003
.....
```

- Comprobamos el directorio de salida del HDFS “resultado3” y comprobamos el contenido

```
hdfs dfs -ls /resultado3
```

```
Found 2 items
```

```
-rw-r--r--      1  hadoop  supergroup              0  2018-01-06 23:52
/resultado3/_SUCCESS
-rw-r--r--      1  hadoop  supergroup    448894  2018-01-06 23:52 /resultado3/part-r-
00000
```

- También podemos ver la aplicación en la página WEB
-

Cluster Metrics											
Apps Submitted		Apps Pending		Apps Running		Apps Completed		Containers Running		Memory Used	
2		0		0		2		0		0 B	
Cluster Nodes Metrics											
Active Nodes			Decommissioning Nodes			Decommissioned Nodes			Lost Nodes		
1			0			0			0		
Scheduler Metrics											
Scheduler Type		Scheduling Resource Type				Minimum Allocation					
Capacity Scheduler		[MEMORY]				<memory:1024, vCores:1>				<m	
Show 20 entries											
ID	User	Name	Application Type	Queue	Application Priority	StartTime	FinishTime	State	FinalStatus		
application_1515272962334_0003	hadoop	word count	MAPREDUCE	default	0	Sat Jan 6 23:52:17 +0100 2018	Sat Jan 6 23:52:54 +0100 2018	FINISHED	SUCCEEDED		
application_1515272962334_0001	hadoop	word count	MAPREDUCE	default	0	Sat Jan 6 22:31:42 +0100 2018	Sat Jan 6 22:32:23 +0100 2018	FINISHED	SUCCEEDED		
Showing 1 to 2 of 2 entries											

10. Otro programa. Analizar un fichero de Log.

- Vamos a analizar un fichero de log de una página WEB, para determinar el tamaño máximo, media y mínimo de los ficheros que se solicitan.
- Copiamos el fichero "**log1.log**" que está en los recursos del curso a la carpeta prácticas de HDFS.
- Tiene un formato parecido a este. Cada línea indica la llamada y el tamaño devuelto

```
199.72.81.55 - - [01/Jul/1995:00:00:01 -0400] "GET /history/apollo/ HTTP/1.0"
200 6245
unicomp6.unicomp.net - - [01/Jul/1995:00:00:06 -0400] "GET
/shuttle/countdown/ HTTP/1.0" 200 3985
199.120.110.21 - - [01/Jul/1995:00:00:09 -0400] "GET /shuttle/missions/sts-
73/mission-sts-73.html HTTP/1.0" 200 4085
burger.letters.com - - [01/Jul/1995:00:00:11 -0400] "GET
/shuttle/countdown/liftoff.html HTTP/1.0" 304 0
199.120.110.21 - - [01/Jul/1995:00:00:11 -0400] "GET /shuttle/missions/sts-
73/sts-73-patch-small.gif HTTP/1.0" 200 4179
burger.letters.com - - [01/Jul/1995:00:00:12 -0400] "GET /images/NASA-
logosmall.gif HTTP/1.0" 304 0
burger.letters.com - - [01/Jul/1995:00:00:12 -0400] "GET
/shuttle/countdown/video/livevideo.gif HTTP/1.0" 200 0
205.212.115.106 - - [01/Jul/1995:00:00:12 -0400] "GET
/shuttle/countdown/countdown.html HTTP/1.0" 200 3985
d104.aa.net - - [01/Jul/1995:00:00:13 -0400] "GET /shuttle/countdown/
HTTP/1.0" 200 3985
```

- Creamos el siguiente programa, y lo llamamos AnalizarLog.java

```
import java.io.IOException;
import java.util.Iterator;
import java.util.regex.Matcher;
import java.util.regex.Pattern;

import org.apache.hadoop.conf.Configuration;
import org.apache.hadoop.conf.Configured;
import org.apache.hadoop.fs.Path;
import org.apache.hadoop.io.IntWritable;
import org.apache.hadoop.io.Text;
import org.apache.hadoop.mapreduce.Job;
import org.apache.hadoop.mapreduce.Mapper;
```

```
import org.apache.hadoop.mapreduce.Reducer;
import org.apache.hadoop.mapreduce.lib.input.FileInputFormat;
import org.apache.hadoop.mapreduce.lib.output.FileOutputFormat;
import org.apache.hadoop.util.Tool;
import org.apache.hadoop.util.ToolRunner;

/**
 * Calcula la media, máximo y mínimo de los ficheros de un servidor Web
 */

public class AnalizarLog extends Configured implements Tool {

    public static void main(String[] args) throws Exception {
        int res = ToolRunner.run(new Configuration(),
            new AnalizarLog (), args);
        System.exit(res);
    }

    @Override
    public int run(String[] args) throws Exception {

        if (args.length != 2) {
            System.err.println("Usage:                    <input_path>
<output_path>");
            System.exit(-1);
        }
        /* input parameters */
        String inputPath = args[0];
        String outputPath = args[1];

        Job job = Job.getInstance(getConf(),
            "WebLogMessageSizeAggregator");
        job.setJarByClass(AnalizarLog.class);
        job.setMapperClass(AMapper.class);
        job.setReducerClass(AReducer.class);
        job.setNumReduceTasks(1);
```

```

        job.setOutputKeyClass(Text.class);
        job.setOutputValueClass(IntWritable.class);
        FileInputFormat.setInputPaths(job, new Path(inputPath));
        FileOutputFormat.setOutputPath(job, new Path(outputPath));

        int exitStatus = job.waitForCompletion(true) ? 0 : 1;
        return exitStatus;
    }

    public static class AMapper extends Mapper<Object, Text, Text,
IntWritable> {
        public static final Pattern httplogPattern = Pattern
            .compile("([^\s]+) - - \[(.+)\] \"([^\s]+) ([^\s]*)
HTTP/[^\s]+\\" [^\s]+ ([0-9]+)");

        public void map(Object key, Text value, Context context)
            throws IOException, InterruptedException {
            Matcher matcher =
httplogPattern.matcher(value.toString());
            if (matcher.matches()) {
                int size = Integer.parseInt(matcher.group(5));
                context.write(new Text("msgSize"), new
IntWritable(size));
            }
        }
    }

    public static class AReducer extends
        Reducer<Text, IntWritable, Text, IntWritable> {
        public void reduce(Text key, Iterable<IntWritable> values,
            Context context) throws IOException,
InterruptedException {
            double tot = 0;
            int count = 0;
            int min = Integer.MAX_VALUE;

```

```

        int max = 0;
        Iterator<IntWritable> iterator = values.iterator();
        while (iterator.hasNext()) {
            int value = iterator.next().get();
            tot = tot + value;
            count++;
            if (value < min) {
                min = value;
            }
            if (value > max) {
                max = value;
            }
        }
        context.write(new Text("Mean"), new IntWritable((int) tot
/ count));

        context.write(new Text("Max"), new IntWritable(max));
        context.write(new Text("Min"), new IntWritable(min));
    }
}
}

```

- Lo compilamos

```
hadoop com.sun.tools.javac.Main AnalizarLog.java
```

- Añadimos a la librería JAR creada en el ejemplo anterior las clases

```
jar uf mi_libreria.jar Analizar*.class
```

- Ejecutamos para ver el resultado

```
hadoop jar mi_libreria.jar AnalizarLog /practicass/log1.log /resultado_log
```

```
INFO client.RMProxy: Connecting to ResourceManager at
localhost/127.0.0.1:8032
```

```
18/01/07 01:21:10 INFO input.FileInputFormat: Total input files to process : 1
```

```
18/01/07 01:21:11 INFO mapreduce.JobSubmitter: number of splits:2
```

```
18/01/07 01:21:12 INFO Configuration.deprecation:
yarn.resourcemanager.system-metrics-publisher.enabled is deprecated. Instead,
use yarn.system-metrics-publisher.enabled
```

```
18/01/07 01:21:13 INFO mapreduce.JobSubmitter: Submitting tokens for job:
job_1515272962334_0004
```



```
18/01/07 01:21:16 INFO impl.YarnClientImpl: Submitted application
application_1515272962334_0004
18/01/07 01:21:16 INFO mapreduce.Job: The url to track the job:
http://nodo1:8088/proxy/application_1515272962334_0004/
18/01/07 01:21:16 INFO mapreduce.Job: Running job:
job_1515272962334_0004
18/01/07 01:21:35 INFO mapreduce.Job: Job job_1515272962334_0004
running in uber mode : false
18/01/07 01:21:35 INFO mapreduce.Job: map 0% reduce 0%
....
....
....
```

- Visualizamos el contenido del fichero de salida

```
hdfs dfs -ls /resultado_log
Found 2 items
-rw-r--r-- 1 hadoop supergroup 0 2018-01-07 01:22
/resultado_log/_SUCCESS
-rw-r--r-- 1 hadoop supergroup 28 2018-01-07 01:22 /resultado_log/part-r-
00000

hdfs dfs -cat /resultado_log/part-r-00000
Mean 1150
Max 6823936
Min 0
```

- También podemos ver el trabajo realizado a través de la consola WEB

11. Lanzar procesos con Python

- Vamos a probar el ejemplo incluido en el vídeo de explicación, es decir el programa wordocount pero en Python
- Con vi o cualquier otro editor, creamos el siguiente programa Python y lo llamamos "pymap.py"
- El programa va extrayendo las palabras del fichero y añadiendo un 1 a cada una de ellas, siguiendo el patrón map reduce

```
#!/usr/bin/env python

import sys

for line in sys.stdin:
    line = line.strip()
    words = line.split()
    for word in words:
        print '%s\t1' % word
```

- Ahora creamos el programa para el reduce, que permite realizar la suma total de palabras. Lo llamamos pyreduce.py

```
#!/usr/bin/env python

from operator import itemgetter
import sys

last_word = None
last_count = 0
cur_word = None

for line in sys.stdin:
    line = line.strip()

    cur_word, count = line.split('\t', 1)

    count = int(count)

    if last_word == cur_word:
```

```

        last_count += count
    else:
        if last_word:
            print '%s\t%s' % (last_word, last_count)
            last_count = count
            last_word = cur_word

    if last_word == cur_word:
        print '%s\t%s' % (last_word, last_count)

```

- Lanzamos el proceso a través de hadoop streaming

```

hadoop jar /opt/hadoop/share/hadoop/tools/lib/hadoop-streaming-2.9.0.jar
-file pypmap.py -mapper pypmap.py -file pyreduce.py -reducer pyreduce.py -
input /practicass/quijote.txt -output /resultado4

```

18/01/07 10:08:12 WARN streaming.StreamJob: -file option is deprecated, please use generic option -files instead.

packageJobJar: [pypmap.py, pyreduce.py, /tmp/hadoop-unjar2186090198010276252/] [] /tmp/streamjob8257554939186511413.jar tmpDir=null

18/01/07 10:08:15 INFO client.RMPProxy: Connecting to ResourceManager at localhost/127.0.0.1:8032

18/01/07 10:08:15 INFO client.RMPProxy: Connecting to ResourceManager at localhost/127.0.0.1:8032

18/01/07 10:08:19 INFO mapred.FileInputFormat: Total input files to process : 1

18/01/07 10:08:20 INFO mapreduce.JobSubmitter: number of splits:2

18/01/07 10:08:21 INFO Configuration.deprecation: yarn.resourcemanager.system-metrics-publisher.enabled is deprecated. Instead, use yarn.system-metrics-publisher.enabled

18/01/07 10:08:22 INFO mapreduce.JobSubmitter: Submitting tokens for job: job_1515272962334_0006

18/01/07 10:08:24 INFO impl.YarnClientImpl: Submitted application application_1515272962334_0006

18/01/07 10:08:24 INFO mapreduce.Job: The url to track the job: http://nodo1:8088/proxy/application_1515272962334_0006/

18/01/07 10:08:24 INFO mapreduce.Job: Running job: job_1515272962334_0006

.....

....

- Vemos que genera una salida similar al programa hecho en Java

- También podemos ver en la página Web que el tipo de programa lanzado es Map Reduce

Labels

Annotations

View

SAVING

MITTED

DELETED

NING

SHED

ED

ED

UI

Cluster Metrics

Apps Submitted

Apps Pending

Apps Running

Apps Completed

Containers Running

4

0

0

4

0

Cluster Nodes Metrics

Active Nodes

Decommissioning Nodes

Decommissioned Nodes

1

0

0

Scheduler Metrics

Scheduler Type

Scheduling Resource Type

Minimum Resource

Capacity Scheduler

[MEMORY]

<memory:1024, vCores:1>

Show 20 entries

ID

User

Name

Application Type

Queue

Application Priority

application_1515272962334_0005

hadoop

streamjob9196474600754661752.jar

MAPREDUCE

default

0

application_1515272962334_0004

hadoop

WebLogMessageSizeAggregator

MAPREDUCE

default

0

application_1515272962334_0003

hadoop

word count

MAPREDUCE

default

0

- En el capítulo del cluster veremos algunos ejemplos más de Python y otros entornos de Streaming

12. Montar un cluster real

- Paramos el cluster y el nodo1
- Clonar las máquinas tal y como se indican en el vídeo. En principio, en este curso probamos con 3 nodos, pero si tienes suficiente hardware puedes probar con más.
- En el nodo 1 modificamos el fichero “/opt/hadoop/etc/hadoop/slaves” para añadir los nodos que vamos a crear y que van a ser los esclavos. Debemos asegurarnos de que el nodo1 no está, porque va a ser el maestro. Por ejemplo
 - Nodo2
 - Nodo3
 -
- Vamos a generar de nuevo todo el cluster, por lo tanto vamos a gtener que limpiar todo.
- En cada nodo, debemos asegurarnos de:
 - Poner el nombre correcto a la máquina: nodo2, nodo3, etc...
 - En el caso de CENTOS 6 se hace con el comando

host nodo2

 - Y modificando el nombre en el fichero /etc/sysconfig/network
 - En el caso de CENTOS 7 se hace con el comando

hostnamectl set-hostname nodo2
 - Configurar correctamente la dirección IP, según se indica en el vídeo
 - El /etc/hosts debe ser igual en todos los nodos
 - Copiar el fichero “/home/hadoop/.bashrc” del nodo1 al resto de nodos para asegurarnos de que tenemos corresectamente las variables de arranque
 - Generar clave pública y privada con el comando “ssh-keygen” en cada nodo
 - Copiar la clave (/home/hadoop/.ssh/authorized_keys) del nodo1 al resto de nodos para poder tener conectividad ssh.
 - Si queremos poder trabajar desde todos los nodos debemos copiar la clave de todos
 - Copiar hadoop en cda nodo al /opt/hadoop y ponerle los permisos adecuados para el usuario “hadoop”
 - Limpiamos el directorio “/datos” en cada nodo. Lo podemos hacer con el comando

```
rm -rf /datos/*
```

- Cambiamos en el hdfs-site.xml el valor de replicación a 3:

```
<property>
  <name>dfs.replication</name>
  <value>3</value>
</property>
<property>
```

- Nos aseguramos de que en el core-site.xml esté identificado el nodo1 (No puede ser localhost, fallaría)
- **IMPORTANTE:** copiamos el directorio /opt/hadoop/etc/hadoop del nodo1 al resto de nodos, para asegurarnos de que los ficheros de configuración son iguales y por tanto son correctos. Cada vez que cambiemos la configuración debemos copiarlo al resto de nodos.
- Arrancamos las máquinas
- Antes de arrancar el cluster, volvemos a crear el HDFS

hdfs namenode -format

```
8/01/07 10:30:47 INFO util.GSet: capacity   = 2^15 = 32768 entries
18/01/07 10:30:47 INFO namenode.FSImage: Allocated new BlockPoolId: BP-130915252-192.168.56.101-1515317447407
18/01/07 10:30:47 INFO common.Storage: Storage directory /datos/namenode has been successfully formatted.
18/01/07 10:30:47 INFO namenode.FSImageFormatProtobuf: Saving image file /datos/namenode/current/fsimage.ckpt_00000000000000000000 using no compression
18/01/07 10:30:47 INFO namenode.FSImageFormatProtobuf: Image file /datos/namenode/current/fsimage.ckpt_00000000000000000000 of size 323 bytes saved in 0 seconds.
18/01/07 10:30:47 INFO namenode.NNStorageRetentionManager: Going to retain 1 images with txid >= 0
18/01/07 10:30:47 INFO namenode.NameNode: SHUTDOWN_MSG:
/*****
SHUTDOWN_MSG: Shutting down NameNode at nodo1/192.168.56.101
*****/
```

- Probamos a arrancar todo el cluster, primero la parte de HDFS. En este caso yo estoy haciendo el ejemplo con 7 nodos, 1 maestro y 6 esclavos

start-dfs.sh

```
Starting namenodes on [localhost]
localhost: starting namenode, logging to /opt/hadoop/logs/hadoop-hadoop-namenode-nodo1.out
```

```
nodo2: starting datanode, logging to /opt/hadoop/logs/hadoop-hadoop-
datanode-nodo2.out
nodo6: starting datanode, logging to /opt/hadoop/logs/hadoop-hadoop-
datanode-nodo6.out
nodo7: starting datanode, logging to /opt/hadoop/logs/hadoop-hadoop-
datanode-nodo7.out
nodo5: starting datanode, logging to /opt/hadoop/logs/hadoop-hadoop-
datanode-nodo5.out
nodo4: starting datanode, logging to /opt/hadoop/logs/hadoop-hadoop-
datanode-nodo4.out
nodo3: starting datanode, logging to /opt/hadoop/logs/hadoop-hadoop-
datanode-nodo3.out
Starting secondary namenodes [0.0.0.0]
0.0.0.0: starting secondarynamenode, logging to /opt/hadoop/logs/hadoop-
hadoop-secondarynamenode-nodo1.out
```

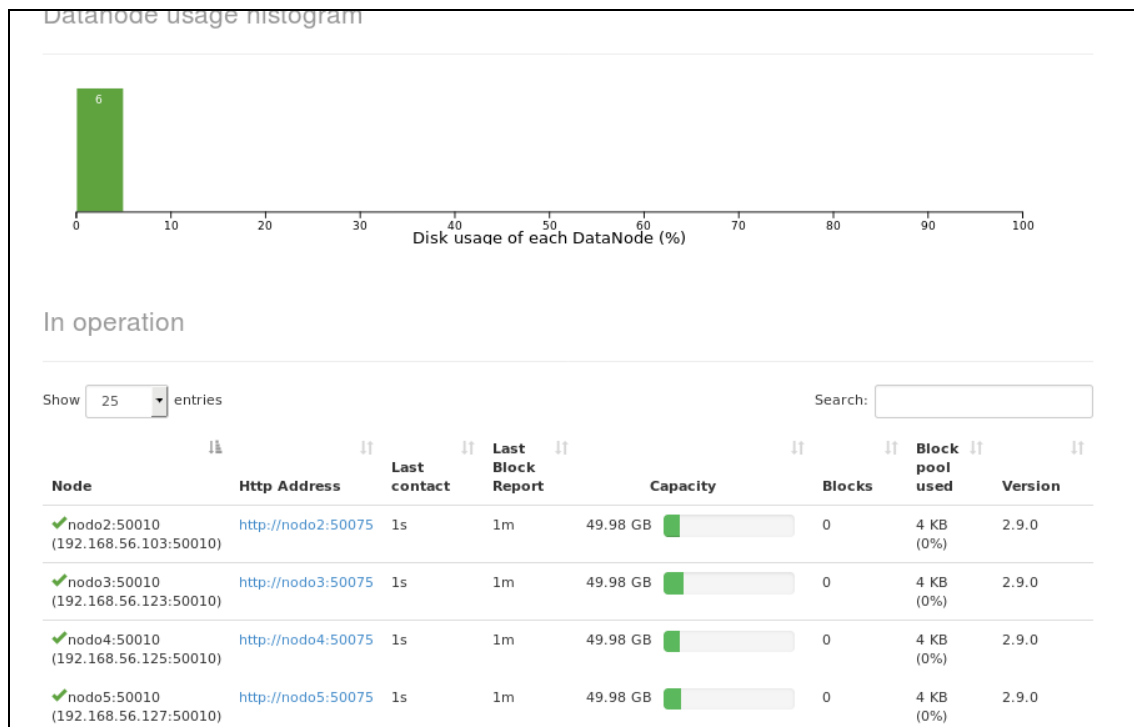
- En el maestro debemos tener estos procesos. Solo los procesos maestros

```
jps
21928 NameNode
22314 Jps
22157 SecondaryNameNode
```

- En cada uno de los esclavos solo debemos tener el datanode

```
jps
3834 Jps
3723 DataNode
```

- En el admin web deben aparecer todos los nodos que hemos creado



- Arrancamos ahora el YARN.

```
start-yarn.sh
```

- En este caso, los procesos del maestro deben ser

```
jps
28833 NameNode
29538 Jps
29268 ResourceManager
29062 SecondaryNameNode
```

- Y los de los esclavos

```
jps
6373 Jps
6086 DataNode
6232 NodeManager
```

- En la Web de administración de YARN deben a parecer los nodos

Cluster

About

Nodes

Node Labels

Applications

NEW

NEW SAVING

SUBMITTED

ACCEPTED

RUNNING

FINISHED

FAILED

KILLED

Scheduler

Tools

node1:8088/cluster/nodes

90%

zookeeper

☆

📁

⬇️

🏠

🔍

Cluster Metrics

Apps Submitted	Apps Pending	Apps Running	Apps Completed	Containers Running	Memory Used	Memory Total	Memory Reserved	VCores Used	VCores Total
0	0	0	0	0	0 B	48 GB	0 B	0	48

Cluster Nodes Metrics

Active Nodes	Decommissioning Nodes	Decommissioned Nodes	Lost Nodes	Unhealthy Nodes	Rebooted Nodes
6	0	0	0	0	0

Scheduler Metrics

Scheduler Type	Scheduling Resource Type	Minimum Allocation	Maximum Allocation	Maximum Clusters
Capacity Scheduler	[MEMORY]	<memory:1024, vCores:1>	<memory:8192, vCores:4>	0

Show 20 entries

Search:

Node Labels	Rack	Node State	Node Address	Node HTTP Address	Last health-update	Health-report	Containers	Mem Used	Mem Avail	VCores Used
/default-rack	RUNNING	nodo2:37058	nodo2:8042	dom ene 07 12:48:58 +0100 2018	0	0 B	8 GB	0		
/default-rack	RUNNING	nodo4:39696	nodo4:8042	dom ene 07 12:48:57 +0100 2018	0	0 B	8 GB	0		
/default-rack	RUNNING	nodo5:42850	nodo5:8042	dom ene 07 12:48:57 +0100 2018	0	0 B	8 GB	0		

- Por último, en el nodo1, arrancamos el History Server para poder controlar los procesos de Mapper y reducer

```
mapred --daemon start historyserver
```

```
starting historyserver, logging to /opt/hadoop/logs/mapred-hadoop-historyserver-nodo1.out
```

```
jps
```

```
28833 NameNode
```

```
29907 ResourceManager
```

```
29062 SecondaryNameNode
```

```
2794 JobHistoryServer
```

```
2844 Jps
```

13. Trabajar con el cluster

- Podemos comprobar el cluster con `dfsadmin -report`
- Aparece información sobre cada nodo

hdfs dfsadmin -report

Configured Capacity: 321965260800 (299.85 GB)

Present Capacity: 281433378816 (262.11 GB)

DFS Remaining: 281433329664 (262.11 GB)

DFS Used: 49152 (48 KB)

DFS Used%: 0.00%

Under replicated blocks: 0

Blocks with corrupt replicas: 0

Missing blocks: 0

Missing blocks (with replication factor 1): 0

Pending deletion blocks: 0

Live datanodes (6):

Name: 192.168.56.103:50010 (nodo2)

Hostname: nodo2

Decommission Status : Normal

Configured Capacity: 53660876800 (49.98 GB)

DFS Used: 8192 (8 KB)

Non DFS Used: 6442340352 (6.00 GB)

DFS Remaining: 47218528256 (43.98 GB)

DFS Used%: 0.00%

DFS Remaining%: 87.99%

Configured Cache Capacity: 0 (0 B)

Cache Used: 0 (0 B)

Cache Remaining: 0 (0 B)

Cache Used%: 100.00%

Cache Remaining%: 0.00%

Xceivers: 1

Last contact: Sun Jan 07 12:53:28 CET 2018

Last Block Report: Sun Jan 07 12:41:43 CET 2018

Name: 192.168.56.123:50010 (nodo3)
 Hostname: nodo3
 Decommission Status : Normal
 Configured Capacity: 53660876800 (49.98 GB)
 DFS Used: 8192 (8 KB)
 Non DFS Used: 7820476416 (7.28 GB)
 DFS Remaining: 45840392192 (42.69 GB)
 DFS Used%: 0.00%
 DFS Remaining%: 85.43%
 Configured Cache Capacity: 0 (0 B)
 Cache Used: 0 (0 B)
 Cache Remaining: 0 (0 B)
 Cache Used%: 100.00%
 Cache Remaining%: 0.00%
 Xceivers: 1
 Last contact: Sun Jan 07 12:53:28 CET 2018
 Last Block Report: Sun Jan 07 12:41:43 CET 2018

Name: 192.168.56.125:50010 (nodo4)
 Hostname: nodo4
 Decommission Status : Normal
 Configured Capacity: 53660876800 (49.98 GB)
 DFS Used: 8192 (8 KB)
 Non DFS Used: 6591770624 (6.14 GB)
 DFS Remaining: 47069097984 (43.84 GB)
 DFS Used%: 0.00%
 DFS Remaining%: 87.72%
 Configured Cache Capacity: 0 (0 B)
 Cache Used: 0 (0 B)
 Cache Remaining: 0 (0 B)
 Cache Used%: 100.00%
 Cache Remaining%: 0.00%
 Xceivers: 1
 Last contact: Sun Jan 07 12:53:28 CET 2018
 Last Block Report: Sun Jan 07 12:41:43 CET 2018

14. Probar el rendimiento del cluster

- Podemos probar también el rendimiento del Cluster antes de empezar a trabajar con él.
- Existe un comando denominado TestDFIO en la librería “hadoop-mapreduce-client-jobclient-2.9.0-tests.jar” que permite indicar el número de ficheros y el tamaño de cada uno par que nos haga un benchmark.
 - Se encuentra en el directorio /opt/hadoop/share/hadoop/mapreduce
- Por ejemplo, para probar 10 ficheros de 100MG cada uno

hadoop jar hadoop-mapreduce-client-jobclient-2.9.0-tests.jar TestDFSIO -write -nrFiles 10 -fileSize 100

- El resultado será similar al siguiente

```
8/01/07 13:14:02 INFO fs.TestDFSIO: ----- TestDFSIO ----- : write
18/01/07 13:14:02 INFO fs.TestDFSIO:          Date & time: Sun Jan 07 13:14:02 CET 2018
18/01/07 13:14:02 INFO fs.TestDFSIO:          Number of files: 10
18/01/07 13:14:02 INFO fs.TestDFSIO: Total MBytes processed: 1000
18/01/07 13:14:02 INFO fs.TestDFSIO:          Throughput mb/sec: 12,22
18/01/07 13:14:02 INFO fs.TestDFSIO: Average IO rate mb/sec: 12,71
18/01/07 13:14:02 INFO fs.TestDFSIO: IO rate std deviation: 2,56
18/01/07 13:14:02 INFO fs.TestDFSIO: Test exec time sec: 71,13
```

- También lo Podemos lanzar en modo “read” para ver el rendimiento en lectura

15. Lanzar un proceso MapReduce contra el cluster

- Vamos a usar un fichero de ejemplo de patentes en Estados Unidos, llamado "cite75_99.txt". Lo tienes disponible en los recursos del capítulo
- Lo subimos a la carpeta prácticas de HDFS

hdfs dfs -put acite75_99.txt /practicass

- Creamos un programa para trabajar con él. Le vamos a llamar "MyJob.java". La idea es agrupar las patentes por el código principal
- Ponemos el siguiente contenido

```
import java.io.IOException;
import java.util.Iterator;

import org.apache.hadoop.conf.Configuration;
import org.apache.hadoop.conf.Configured;
import org.apache.hadoop.fs.Path;
import org.apache.hadoop.io.Text;
import org.apache.hadoop.io.LongWritable;
import org.apache.hadoop.mapreduce.Job;
import org.apache.hadoop.mapreduce.Mapper;
import org.apache.hadoop.mapreduce.Reducer;
import org.apache.hadoop.mapreduce.lib.input.FileInputFormat;
import org.apache.hadoop.mapreduce.lib.input.TextInputFormat;
import org.apache.hadoop.mapreduce.lib.output.FileOutputFormat;
import org.apache.hadoop.mapreduce.lib.output.TextOutputFormat;
import org.apache.hadoop.util.Tool;
import org.apache.hadoop.util.ToolRunner;

public class MyJob extends Configured implements Tool {

    public static class MapClass extends Mapper<LongWritable, Text, Text, Text> {

        public void map(LongWritable key, Text value, Context context) throws IOException,
        InterruptedException {

            String[] citation = value.toString().split(",");
            context.write(new Text(citation[1]), new Text(citation[0]));
        }
    }

    public static class Reduce extends Reducer<Text, Text, Text, Text> {
```

```

    public void reduce(Text key, Iterable<Text> values, Context context) throws IOException,
        InterruptedException {

        String csv = "";
        for (Text val:values) {
            if (csv.length() > 0) csv += ",";
            csv += val.toString();
        }

        context.write(key, new Text(csv));
    }
}

public int run(String[] args) throws Exception {
    Configuration conf = getConf();

    Job job = new Job(conf, "MyJob");
    job.setJarByClass(MyJob.class);

    Path in = new Path(args[0]);
    Path out = new Path(args[1]);
    FileInputFormat.setInputPaths(job, in);
    FileOutputFormat.setOutputPath(job, out);

    job.setMapperClass(MapClass.class);
    job.setReducerClass(Reduce.class);

    job.setInputFormatClass(TextInputFormat.class);
    job.setOutputFormatClass(TextOutputFormat.class);
    job.setOutputKeyClass(Text.class);
    job.setOutputValueClass(Text.class);

    System.exit(job.waitForCompletion(true)?0:1);

    return 0;
}

public static void main(String[] args) throws Exception {
    int res = ToolRunner.run(new Configuration(), new MyJob(), args);
}

```

```
        System.exit(res);
    }
}
```

- Exportamos la librería para localizarla en el CLASSPATH

```
export HADOOP_CLASSPATH=$JAVA_HOME/lib/tools.jar
```

- Compilamos

```
hadoop com.sun.tools.javac.Main MyJob.java
```

- Creamos el JAR correspondiente

```
jar cvf MyJob.jar My*
```

manifiesto agregado

agregando: MyJob.class(entrada = 2028) (salida = 967)(desinflado 52%)

agregando: Myjob.java(entrada = 2657) (salida = 739)(desinflado 72%)

agregando: MyJob\$MapClass.class(entrada = 1490) (salida = 585)(desinflado 60%)

agregando: MyJob\$Reduce.class(entrada = 1811) (salida = 786)(desinflado 56%)

- Ejecutamos

```
hadoop jar MyJob.jar MyJob /practicac/cite75_99.txt /resultado7
```

- Comprobamos el resultado dejado en el directorio “resultado7”

```
hdfs dfs -cat /resultado7/part-r-00000
```

- Podemos ver en la página de Admon de YARN donde se ha ejecutado cada uno de los procesos, mapper y reducer en su caso

16. Lanzar un proceso en streaming con comandos Shell de Linux


- En este caso, vamos a usar comandos de la Shell de Linux para hacer de Mapper y Reducer, simulando el programa que hemos hecho en el punto anterior.
- Lo hacemos con la librería de Streaming
- Le pasamos el fichero y en el mapper extraemos solo los datos del campo 2 con el comando “cut” y eliminamos duplicados con el reducer y el comando “uniq”.
- NOTA: recuerda adaptar el número de tu versión de hadoop al fichero streaming

```
hadoop jar /opt/hadoop/share/hadoop/tools/lib/hadoop-streaming-2.9.0.jar -input /practicass/cite75_99.txt -output /resultado8 -mapper 'cut -f 2 -d ,' -reducer 'uniq'
```

```
packageJobJar:          [/tmp/hadoop-unjar2997682931744806206/] []
/tmp/streamjob7022129202320877187.jar tmpDir=null

18/01/07 15:08:26 INFO client.RMProxy: Connecting to ResourceManager at
nodo1/192.168.56.101:8032
18/01/07 15:08:26 INFO client.RMProxy: Connecting to ResourceManager at
nodo1/192.168.56.101:8032
18/01/07 15:08:27 INFO mapred.FileInputFormat: Total input files to process : 1
18/01/07 15:08:27 INFO net.NetworkTopology: Adding a new node: /default-
rack/192.168.56.132:50010
18/01/07 15:08:27 INFO net.NetworkTopology: Adding a new node: /default-
rack/192.168.56.123:50010
18/01/07 15:08:27 INFO net.NetworkTopology: Adding a new node: /default-
rack/192.168.56.103:50010
18/01/07 15:08:27 INFO net.NetworkTopology: Adding a new node: /default-
rack/192.168.56.125:50010
18/01/07 15:08:28 INFO mapreduce.JobSubmitter: number of splits:2
18/01/07 15:08:29 INFO Configuration.deprecation: yarn.resourcemanager.system-
metrics-publisher.enabled is deprecated. Instead, use yarn.system-metrics-
publisher.enabled
18/01/07 15:08:29 INFO mapreduce.JobSubmitter: Submitting tokens for job:
job_1515325737805_0006
18/01/07 15:08:30 INFO impl.YarnClientImpl: Submitted application
application_1515325737805_0006
18/01/07 15:08:30 INFO mapreduce.Job: The url to track the job:
http://nodo1:8088/proxy/application_1515325737805_0006/
18/01/07 15:08:30 INFO mapreduce.Job: Running job: job_1515325737805_0006
```

- Podemos verlo en la WEB de administración



Cluster

- About
- Nodes
- Node Labels
- Applications
- NEW
- NEW SAVING
- SUBMITTED
- ACCEPTED
- RUNNING
- FINISHED
- FAILED
- KILLED
- Scheduler

Tools

Cluster Metrics

Apps Submitted	Apps Pending	Apps Running	Apps Completed	Containers Running
3	0	0	3	0

Cluster Nodes Metrics

Active Nodes	Decommissioning Nodes	Decommissioned Nodes
6	0	0

Scheduler Metrics

Scheduler Type	Scheduling Resource Type	Minimum Resource
Capacity Scheduler	[MEMORY]	<memory:1024, vCores:1

Show 20 entries

ID	User	Name	Application Type	Queue	Application Priority
application_1515325737805_0006	hadoop	streamjob7022129202320877187.jar	MAPREDUCE	default	0
application_1515325737805_0005	hadoop	MyJob	MAPREDUCE	default	0

- Si seleccionamos History, podemos ver el número de mappers y reducers

on

- Job Overview
- Job Configuration
- Job Tasks

Job Name: streamjob7022129202320877187.jar

User Name: hadoop

Queue: default

State: SUCCEEDED

Uberized: false

Submitted: Sun Jan 07 15:08:30 CET 2018

Started: Sun Jan 07 15:08:39 CET 2018

Finished: Sun Jan 07 15:10:41 CET 2018

Elapsed: 2mins, 1sec

Diagnostics:

Average Map Time: 1mins, 17sec

Average Shuffle Time: 13sec

Average Merge Time: 0sec

Average Reduce Time: 25sec

ApplicationMaster

Attempt Number	Start Time	Node	Log
1	Sun Jan 07 15:08:32 CET 2018	nodo7:8042	logs

Task Type	Total	Complete
Map	2	2
Reduce	1	1

Attempt Type	Failed	Killed	Successful
Maps	0	0	2
Reduces	0	0	1

- Otro ejemplo. Si solo queremos contar ocurrencias. No necesitamos Reducer y le decimos que solo hay mapper, y ejecutamos el mapper con el comando "wc -l"

```
hadoop jar /opt/hadoop/share/hadoop/tools/lib/hadoop-streaming-2.9.0.jar -D mapred.reduce.tasks=0 -input /practicass/cite75_99.txt -output /resultado9 -mapper 'wc -l'
```

```
packageJobJar: [/tmp/hadoop-unjar2795715086677832161/] []
/tmp/streamjob6020154308636157887.jar tmpDir=null
```

```
18/01/07 15:16:47 INFO client.RMPProxy: Connecting to ResourceManager at nodo1/192.168.56.101:8032
```

```
18/01/07 15:16:48 INFO client.RMPProxy: Connecting to ResourceManager at
nodo1/192.168.56.101:8032
18/01/07 15:16:49 INFO mapred.FileInputFormat: Total input files to process : 1
18/01/07 15:16:49 INFO net.NetworkTopology: Adding a new node: /default-
rack/192.168.56.123:50010
18/01/07 15:16:49 INFO net.NetworkTopology: Adding a new node: /default-
rack/192.168.56.103:50010
18/01/07 15:16:49 INFO net.NetworkTopology: Adding a new node: /default-
rack/192.168.56.132:50010
18/01/07 15:16:49 INFO net.NetworkTopology: Adding a new node: /default-
rack/192.168.56.125:50010
18/01/07 15:16:50 INFO mapreduce.JobSubmitter: number of splits:2
18/01/07 15:16:50 INFO Configuration.deprecation: mapred.reduce.tasks is
deprecated. Instead, use mapreduce.job.reduces
```

- Podemos ver el resultado en el directorio /resultado9
- Como le hemos quitado el reducer y tenemos dos bloques en el fichero (recordemos que cada bloque es de 128Mb y nuestro fichero ocupa 200MB), nos aparecen dos ficheros con el número de líneas en cada bloque

hdfs dfs -ls /resultado9

Found 3 items

```
-rw-r--r--      3 hadoop supergroup          0 2018-01-07 15:17
/resultado9/_SUCCESS
-rw-r--r--      3 hadoop supergroup          9 2018-01-07 15:17 /resultado9/part-
00000
-rw-r--r--      3 hadoop supergroup          9 2018-01-07 15:17 /resultado9/part-
00001
```

hdfs dfs -cat /resultado9/part-00001

8264198

hdfs dfs -cat /resultado9/part-00000

8258241

- Como siempre, podemos ver en la página Web el resultado

17. Lanzar un proceso en Streaming con Python

- En este caso vamos a crear un programa Python que recupere de forma aleatoria parte de las filas del fichero.
- Le llamamos “rand.py” por ejemplo

```
#!/usr/bin/env python
import sys, random
for line in sys.stdin:
    if (random.randint(1,100) <= int(sys.argv[1])):

        print line.strip()
```

- Lo ejecutamos con un reducer de 1

```
hadoop jar /opt/hadoop/share/hadoop/tools/lib/hadoop-streaming-2.9.0.jar -D
mapred.job.reducers=1 -input /practicac/cite75_99.txt -output /resultado11 -
mapper 'rand.py' -file rand.py
```

```
packageJobJar: [random.py, /tmp/hadoop-unjar2385850152370180720/] []
/tmp/streamjob2584922200311003370.jar tmpDir=null

18/01/07 15:26:33 INFO client.RMProxy: Connecting to ResourceManger at
nodo1/192.168.56.101:8032
18/01/07 15:26:33 INFO client.RMProxy: Connecting to ResourceManger at
nodo1/192.168.56.101:8032
18/01/07 15:26:35 INFO mapred.FileInputFormat: Total input files to process : 1
18/01/07 15:26:35 INFO net.NetworkTopology: Adding a new node: /default-
rack/192.168.56.123:50010
18/01/07 15:26:35 INFO net.NetworkTopology: Adding a new node: /default-
rack/192.168.56.103:50010
18/01/07 15:26:35 INFO net.NetworkTopology: Adding a new node: /default-
rack/192.168.56.132:50010
18/01/07 15:26:35 INFO net.NetworkTopology: Adding a new node: /default-
rack/192.168.56.125:50010
18/01/07 15:26:35 INFO mapreduce.JobSubmitter: number of splits:2
18/01/07 15:26:36 INFO Configuration.deprecation: yarn.resourcemanager.system-
metrics-publisher.enabled is deprecated. Instead, use yarn.system-metrics-
publisher.enabled
18/01/07 15:26:36 INFO mapreduce.JobSubmitter: Submitting tokens for job:
job_1515325737805_0010
18/01/07 15:26:37 INFO impl.YarnClientImpl: Submitted application
application_1515325737805_0010
18/01/07 15:26:37 INFO mapreduce.Job: The url to track the job:
http://nodo1:8088/proxy/application_1515325737805_0010/
18/01/07 15:26:37 INFO mapreduce.Job: Running job: job_1515325737805_0010
.....
....
```


18. Comando YARN

- Vamos a ver como trabajar con el comando YARN
- Mantenemos dos terminales abiertos
- En el primero lanzamos un proyecto mapreduce.
 - Si es en versiones anteriores a la 2.9 tenemos que lanzarlo de esta manera. Es debido a que en la 2.9 se ha modificado la propiedad queue.name por queuename

```
hadoop jar /opt/hadoop/share/hadoop/mapreduce/hadoop-mapreduce-examples-2.9.0.jar wordcount -Dmapred.job.queue.name=warehouse /practicascite75_99.txt /salida20
```

- En la 2.9 lo hacemos así

```
hadoop jar /opt/hadoop/share/hadoop/mapreduce/hadoop-mapreduce-examples-2.9.0.jar wordcount -Dmapred.job.queue.name=warehouse /practicascite75_99.txt /salida20
```

- Vamos a comprobar que se ha lanzado

yarn application -list

18/01/07 20:57:28 INFO client.RMPProxy: Connecting to ResourceManager at nodo1/192.168.56.101:8032

Total number of applications (application-types: [], states: [SUBMITTED, ACCEPTED, RUNNING] and tags: []):1

Queue	Application-Id State	Application-Name Final-State	Application-Type Progress	User Tracking-URL
application_1515354119376_0002	hadoop default	word count ACCEPTED	MAPREDUCE UNDEFINED	0%
N/A				

- Comprobamos en que nodos se está ejecutando

yarn node -list

18/01/07 20:57:38 INFO client.RMPProxy: Connecting to ResourceManager at nodo1/192.168.56.101:8032

Total Nodes:6

Node-Id	Node-State	Node-Http-Address	Number-of-Running-Containers
nodo7:34788	RUNNING	nodo7:8042	0
nodo3:40979	RUNNING	nodo3:8042	0
nodo4:36792	RUNNING	nodo4:8042	0
nodo6:33922	RUNNING	nodo6:8042	0
nodo2:36376	RUNNING	nodo2:8042	3
nodo5:37517	RUNNING	nodo5:8042	0

- Comprobamos el estado completo de la aplicación

yarn application -status application_1515354119376_0002

18/01/07 20:57:50 INFO client.RMPProxy: Connecting to ResourceManager at nodo1/192.168.56.101:8032

Application Report :

```

Application-Id : application_1515354119376_0002
Application-Name : word count
Application-Type : MAPREDUCE
User : hadoop
Queue : default
Application Priority : 0
Start-Time : 1515355041442
Finish-Time : 0
Progress : 5%
State : RUNNING
Final-State : UNDEFINED
Tracking-URL : http://nodo2:34944
RPC Port : 41274
AM Host : nodo2
Aggregate Resource Allocation : 91429 MB-seconds, 59 vcore-seconds
Aggregate Resource Preempted : 0 MB-seconds, 0 vcore-seconds
Log Aggregation Status : DISABLED
Diagnostics :
Unmanaged Application : false
Application Node Label Expression : <Not set>
AM container Node Label Expression : <DEFAULT_PARTITION>
TimeoutType : LIFETIME      ExpiryTime : UNLIMITED      RemainingTime
: -1seconds
    
```

- Comprobamos los Application Attempts:

```

yarn applicationattempt -list application_1515354119376_0002
18/01/07 20:58:50 INFO client.RMPProxy: Connecting to ResourceManager at
nodo1/192.168.56.101:8032
Total number of application attempts :1
      ApplicationAttempt-Id      State      AM-Container-Id
Tracking-URL
appattempt_1515354119376_0002_000001      RUNNING
      container_1515354119376_0002_01_000001
      http://nodo1:8088/proxy/application_1515354119376_0002/
    
```

- Vemos los contenedores asociados

```

yarn container -list appattempt_1515354119376_0002_000001
18/01/07 20:59:13 INFO client.RMPProxy: Connecting to ResourceManager at
nodo1/192.168.56.101:8032
Total number of containers :2
      Container-Id      Start Time      Finish Time      State
Host      Node Http Address      LOG-URL
    
```

```
container_1515354119376_0002_01_000004 dom ene 07 20:58:51 +0100 2018
N/A RUNNING nodo2:36376 http://nodo2:8042
http://nodo2:8042/node/containerlogs/container_1515354119376_0002_01_000
004/hadoop

container_1515354119376_0002_01_000001 dom ene 07 20:57:21 +0100 2018
N/A RUNNING nodo2:36376 http://nodo2:8042
http://nodo2:8042/node/containerlogs/container_1515354119376_0002_01_000
001/hadoop
```

- Eliminamos la aplicación

```
yarn application -kill application_1515354119376_0002
```

- Comprobamos que se ha eliminado

19. Scheduler

- Vamos a configurar tres colas para nuestro planificador de capacidad

NOMBRE	CAPACIDAD
default	10
rrhh	50
marketing	20
ventas	20

- Accedemos a /opt/hadoop/etc/hadoop
- Abrimos el fichero capacity-scheduler.xml
- Buscamos la siguiente propiedad

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

- La cambiamos por lo siguiente

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

- Ahora creamos una propiedad con la capacidad para cada una de las queues que hemos creado.

```
<property>
  <name>yarn.scheduler.capacity.root.default.capacity</name>
  <value>10</value>
  <description>Default queue target capacity.</description>
</property>

<property>
```



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

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

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

- Guardamos el fichero
- Refrescamos las colas

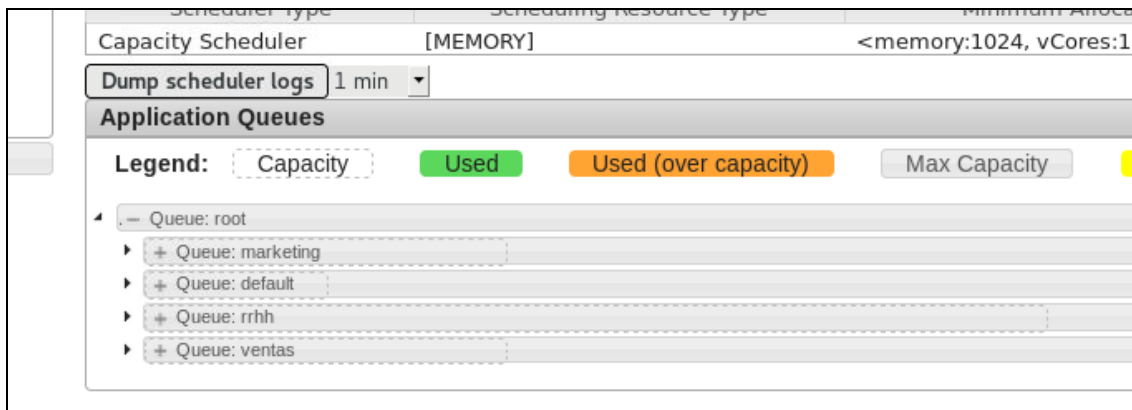
```
yarn radmin -refreshQueues
```

- Comprobamos con el comando mapred que están activas

```
mapred queue -list
18/01/07 20:43:18 INFO client.RMProxy: Connecting to
ResourceManager at nodo1/192.168.56.101:8032
=====
Queue Name : marketing
Queue State : running
Scheduling Info : Capacity: 20.0, MaximumCapacity: 100.0,
CurrentCapacity: 0.0
=====
Queue Name : default
Queue State : running
Scheduling Info : Capacity: 10.0, MaximumCapacity: 100.0,
CurrentCapacity: 0.0
=====
Queue Name : rrhh
```

```
Queue State : running
Scheduling Info : Capacity: 50.0, MaximumCapacity: 100.0,
CurrentCapacity: 0.0
=====
Queue Name : ventas
Queue State : running
Scheduling Info : Capacity: 20.0, MaximumCapacity: 100.0,
CurrentCapacity: 0.0
```

- Comprobamos con la página Web que están activas



20. Instalar, crear conexión, bases de datos y tablas

- Descomprimir el software en la carpeta /opt/hadoop y denominamos al directorio hive. Debe quedar /opt/hadoop/hive
- Poner lo siguiente en el .bashrc o en el fichero de arranque

```
export HIVE_HOME=/opt/hadoop/hive
export PATH=$PATH:$HIVE_HOME/bin:$HIVE_HOME/conf
```

- Accedemos al directorio conf y copiamos los ficheros de templates para convertirlos en xml

```
cp hive-default.xml.template hive-site.xml
cp hive-env.sh.template hive-env.sh
cp hive-exec-log4j.properties.template hive-execlog4j.properties
cp hive-log4j.properties.template hive-log4j.properties
```

- Modificamos el hive.env.sh y configuramos los siguientes parámetros

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

- Creamos dentro de HDFS la estructura de HIVE. Por defecto está apuntando a estos directorios

```
$ hdfs dfs -mkdir /tmp
$ hdfs dfs -mkdir -p /user/hive/warehouse
$ hdfs dfs -chmod g+w /tmp
$ hdfs dfs -chmod g+w /user/hive/warehouse
```

- Accedemos a hive-site.xml y ponemos estos valores de configuración al principio del fichero

```
<property>
  <name>system:java.io.tmpdir</name>
  <value>/tmp/hive/java</value>
</property>
<property>
  <name>system:user.name</name>
  <value>${user.name}</value>
</property>
```

- Creamos un directorio donde guardar la metastore

```
mkdir bbdd
cd bbdd
```

- Creamos la Base de Datos y su configuración

```
schematool -dbType derby -initSchema
```

SLF4J: Class path contains multiple SLF4J bindings.

SLF4J: Found binding in [jar:file:/opt/hadoop/hive/lib/log4j-slf4j-impl-2.6.2.jar!/org/slf4j/impl/StaticLoggerBinder.class]

SLF4J: Found binding in [jar:file:/opt/hadoop/share/hadoop/common/lib/slf4j-log4j12-1.7.25.jar!/org/slf4j/impl/StaticLoggerBinder.class]

SLF4J: See http://www.slf4j.org/codes.html#multiple_bindings for an explanation.

SLF4J: Actual binding is of type [org.apache.logging.slf4j.Log4jLoggerFactory]

Metastore connection URL:

jdbc:derby::databaseName=metastore_db;create=true

Metastore Connection Driver : org.apache.derby.jdbc.EmbeddedDriver

Metastore connection User: APP

Starting metastore schema initialization to 2.1.0

Initialization script hive-schema-2.1.0.derby.sql

Initialization script completed

schemaTool completed

- Arrancamos el cliente hive

hive

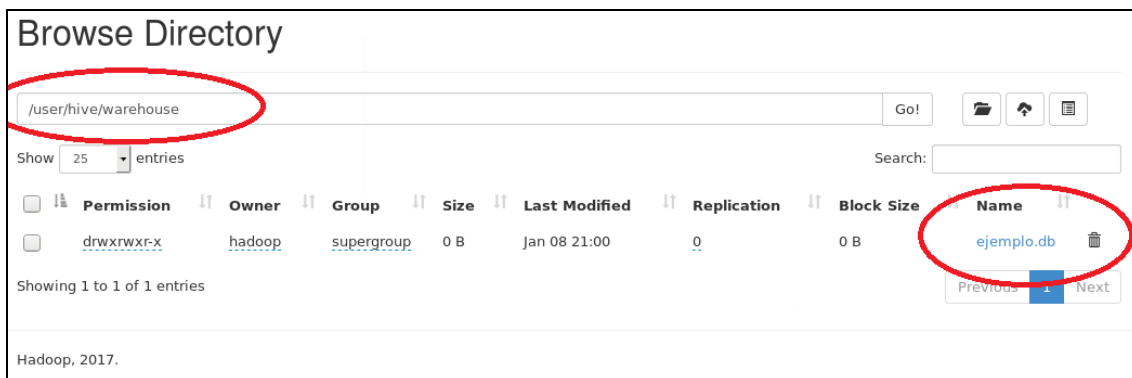
- Comprobar las bases de datos

show databases;

- Crear la Base de datos

create database ejemplo;

- Comprobar en HDFS que existe



Browse Directory

/user/hive/warehouse

Go!

Show 25 entries

Search:

Permission	Owner	Group	Size	Last Modified	Replication	Block Size	Name
drwxrwxr-x	hadoop	supergroup	0 B	Jan 08 21:00	0	0 B	ejemplo.db

Showing 1 to 1 of 1 entries

Previous Next

Hadoop, 2017.

- Acceder a la Base de datos

use ejemplo;

- Crear una pequeña tabla

create table if not exists t1

(
name string

);

- Comprobar en hdfs que existe
-

Browse Directory

/user/hive/warehouse/ejemplo.db

Show 25 entries

Permission	Owner	Group	Size	Last Modified	Replication	Block Size	Name
drwxrwxr-x	hadoop	supergroup	0 B	Jan 08 21:00	0	0 B	t1

Showing 1 to 1 of 1 entries

Hadoop, 2017.

- Insertar alguna fila en la tabla y comprobar el proceso mapreduce

Insert into t1 values ('mi nombre');

- Comprobamos que la ha guardado

select * from t1;

- Como práctica adicional, visualizar, el contenido de la tabla a través del comando siguiente. Debemos ver el fichero que ha creado

h Hdfs dfs -cat /xxxxxxx

21. Tablas internas

- Comprobar si hay bases de datos

```
show databases;
```

```
OK
```

```
+-----+
```

```
| database_name |
```

```
+-----+
```

```
| default      |
```

```
| ejemplo      |
```

```
| prueba      |
```

```
+-----+
```

```
3 rows selected (0,836 seconds)
```

- 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;
```

```
Loading data to table ejemplo.empleados_internal
```

```
INFO      : Loading data to table ejemplo.empleados_internal from
file:/home/curso/Escritorio/employee.txt
```

```
Table ejemplo.empleados_internal stats: [numFiles=1, numRows=0,
totalSize=227, rawDataSize=0]
```

OK

INFO : Table ejemplo.empleados_internal stats: [numFiles=1, numRows=0, totalSize=227, rawDataSize=0]

No rows affected (0,421 seconds)

0: jdbc:hive2://localhost:10000> select * from empleados_internal;

OK

```

+-----+-----+-----+-----+
+-----+-----+-----+-----+
| empleados_internal.name | empleados_internal.work_place |
empleados_internal.sex_age | empleados_internal.skills_score |
empleados_internal.depart_title |
+-----+-----+-----+-----+
+-----+-----+-----+-----+
| Michael | ["Montreal","Toronto"] | {"sex":"Male","age":30} |
{"DB":80} | {"Product":["Developer","Lead"]} |
| Will | ["Montreal"] | {"sex":"Male","age":35} | {"Perl":85}
| {"Product":["Lead"],"Test":["Lead"]} |
| Shelley | ["New York"] | {"sex":"Female","age":27} |
{"Python":80} | {"Test":["Lead"],"COE":["Architect"]} |
| Lucy | ["Vancouver"] | {"sex":"Female","age":57} |
{"Sales":89,"HR":94} | {"Sales":["Lead"]} |
+-----+-----+-----+-----+
+-----+-----+-----+-----+
4 rows selected (0,215 seconds)

```

- 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

SLF4J: Failed to load class "org.slf4j.impl.StaticLoggerBinder".

SLF4J: Defaulting to no-operation (NOP) logger implementation

SLF4J: See <http://www.slf4j.org/codes.html#StaticLoggerBinder> for further details.

Found 2 items

```

drwxrwxr-x - root supergroup 0 2015-06-11 11:15
/user/hive/warehouse/ejemplo.db/empleados_internal
drwxrwxr-x - root supergroup 0 2015-06-11 10:54
/user/hive/warehouse/ejemplo.db/t1

```

- 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

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

Loading data to table ejemplo.empleados_external

Table ejemplo.empleados_external stats: [numFiles=0, numRows=0, totalSize=0, rawDataSize=0]

OK

INFO : Loading data to table ejemplo.empleados_external from file:/home/curso/Escritorio/employee.txt

INFO : Table ejemplo.empleados_external stats: [numFiles=0, numRows=0, totalSize=0, rawDataSize=0]

No rows affected (0,7 seconds)

- Probamos que estén las filas

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

OK

```
+-----+-----+-----+-----+
| empleados_external.name | empleados_external.work_place |
empleados_external.sex_age | empleados_external.skills_score |
empleados_external.depart_title |
+-----+-----+-----+-----+
| Michael | ["Montreal","Toronto"] | {"sex":"Male","age":30} |
{"DB":80} | {"Product":["Developer","Lead"]} |
| Will | ["Montreal"] | {"sex":"Male","age":35} | {"Perl":85}
| {"Product":["Lead"],"Test":["Lead"]} |
| Shelley | ["New York"] | {"sex":"Female","age":27} |
{"Python":80} | {"Test":["Lead"],"COE":["Architect"]} |
```


Lucy	["Vancouver"]	{"sex":"Female","age":57}	
{"Sales":89,"HR":94}	{"Sales":["Lead"]}		
+	+	+	+
+	+	+	+

4 rows selected (0,137 seconds)

- Comprobar que existen el directorio datos
- Hacer alguna SELECT por ejemplo para buscar al empleado "Lucy"
- Borrar las dos tablas
- Comprobar que ha borrado la interna, pero los datos de la externa permanecen.

22. Conexiones remotas

23. Vamos ahora a trabar con el servidor HiveServer 2

24. Primero debemos configurar la siguiente propiedad en el fichero hive-site.xml. La ponemos a FALSE. Hay que buscarla, no crearla de nuevo.

25. De esa forma nos conectamos como usuario hadoop y desactivamos ciertas acciones de seguridad. En un curso más avanzado podríamos ver otras opciones, para esta formación es suficiente.

```
<property>
  <name>hive.server2.enable.doAs</name>
  <value>>false</value>
  <description>
    Setting this property to true will have HiveServer2 execute
    Hive operations as the user making the calls to it.
  </description>
</property>
```

26. Arrancamos el servidor desde el directorio bbdd que hemos creado antes

```
hiveserver2 &
```

27. Nos conectamos con beeline. No hace falta poner usuario

```
beeline
beeline> !connect jdbc:hive2://localhost:10000
Connected to: Apache Hive (version 2.3.2)
Driver: Hive JDBC (version 2.3.2)
Transaction isolation: TRANSACTION_REPEATABLE_READ
0: jdbc:hive2://nodo1:10000>
```

-

- Vamos a probar ahora que podemos conectarnos en remoto desde otro nodo al tener HIVESERVER2
- Copiamos el directorio hive al nodo2, en la misma ruta, es decir en /opt/hadoop
- Copiamos también el .bashrc al /home/hadoop del segundo nodo. Esto nos permite tener la misma configuración
- Entramos en beeline y le ponemos la dirección del nodo1

```
beeline> !connect jdbc:hive2://nodo1:10000
Connecting to jdbc:hive2://nodo1:10000
Enter username for jdbc:hive2://nodo1:10000:
Enter password for jdbc:hive2://nodo1:10000:
Connected to: Apache Hive (version 2.3.2)
Driver: Hive JDBC (version 2.3.2)
Transaction isolation: TRANSACTION_REPEATABLE_READ
```

- Comprobamos que vemos lo mismo

```
show databases;
+-----+
| database_name |
+-----+
| default      |
| ejemplo      |
| prueba       |
+-----+
3 rows selected (0,438 seconds)
```

28. Prácticas adicionales. Meteoritos

- Vamos a realizar unas cuantas SELECT contra una DataSet de la NASA, que contiene las caídas de meteoritos registradas durante las últimas décadas.
- El fichero meteoros.csv tiene los datos
- Primero creamos la siguiente tabla en HIVE

```
Create table meteoros
(nombre      string  ,
id          bigint  ,
tipo        string  ,
clase       string  ,
peso        bigint  ,
recogida    string,
fecha       string  ,
latitud     double  ,
longitud    double  ,
geolocation string )
ROW FORMAT DELIMITED
FIELDS TERMINATED BY ','
;
```

- Luego la cargamos con los datos del fichero de meteoros

```
load data local inpath '/tmp/meteoros.csv' into table meteoros;
No rows affected (7,21 seconds)
```

- Ver las cinco primeras filas

```
select * from meteoros limit 5;
```

meteoros.nombre	meteoros.id	meteoros.tipo	meteoros.clase	meteoros.peso	meteoros.recogida	meteoros.fecha	meteoros.latitud	meteoros.longitud	meteoros.geolocation
Aachen	1	Valid	L5	21	Fell	01/01/1880 12:00:00 AM	50.775	6.08333	"(50.775000
Aarhus	2	Valid	H6	720	Fell	01/01/1951 12:00:00 AM	56.18333	10.23333	"(56.183330

Abee	6	Valid	EH4	107000	Fell
01/01/1952 12:00:00 AM	54.21667	-113.0	"(54.216670		
Acapulco	10	Valid	Acapulcoite	1914	Fell
01/01/1976 12:00:00 AM	16.88333	-99.9	"(16.883330		
Achiras	370	Valid	L6	780	Fell
01/01/1902 12:00:00 AM	-33.16667	-64.95	"(-33.166670		
-----+	-----+	-----+	-----+	-----+	-----+
-----+	-----+	-----+	-----+	-----+	-----+
5 rows selected (0,927 seconds)					

•

29. 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

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

Loading data to table ejemplo.empleados_external
Table ejemplo.empleados_external stats: [numFiles=0, numRows=0,
totalSize=0, rawDataSize=0]
OK
INFO : Loading data to table ejemplo.empleados_external from
file:/home/curso/Escritorio/employee.txt
INFO : Table ejemplo.empleados_external stats: [numFiles=0, numRows=0,
totalSize=0, rawDataSize=0]
No rows affected (0,7 seconds)
```

- Probamos que estén las filas

```
0: jdbc:hive2://localhost:10000> select * from empleados_external;
OK
+-----+-----+-----+-----+
| empleados_external.name | empleados_external.work_place | empleados_external.sex_age | empleados_external.skills_score | empleados_external.depart_title |
+-----+-----+-----+-----+
|                          |                                |                             |                                  |                                  |
|                          |                                |                             |                                  |                                  |
|                          |                                |                             |                                  |                                  |
```

```

+-----+-----+-----+-----+
+-----+-----+-----+-----+
| Michael      | ["Montreal","Toronto"] | {"sex":"Male","age":30} | |
| {"DB":80}    | {"Product":["Developer","Lead"]} |
| Will         | ["Montreal"]           | {"sex":"Male","age":35} | {"Perl":85}
| {"Product":["Lead"],"Test":["Lead"]} |
| Shelley      | ["New York"]           | {"sex":"Female","age":27} | |
| {"Python":80} | {"Test":["Lead"],"COE":["Architect"]} |
| Lucy         | ["Vancouver"]          | {"sex":"Female","age":57} | |
| {"Sales":89,"HR":94} | {"Sales":["Lead"]}      |
+-----+-----+-----+-----+
+-----+-----+-----+-----+
4 rows selected (0,137 seconds)

```

- 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.

30. HiveServer 2 y conexiones remotas

- Vamos ahora a trabajar con el servidor HiveServer 2
- Primero debemos configurar la siguiente propiedad en el fichero hive-site.xml. La ponemos a FALSE. Hay que buscarla, no crearla de nuevo.
- De esa forma nos conectamos como usuario hadoop y desactivamos ciertas acciones de seguridad. En un curso más avanzado podríamos ver otras opciones, para esta formación es suficiente.

```
<property>
  <name>hive.server2.enable.doAs</name>
  <value>>false</value>
  <description>
    Setting this property to true will have HiveServer2 execute
    Hive operations as the user making the calls to it.
  </description>
</property>
```

- Arrancamos el servidor desde el directorio bbdd que hemos creado antes

```
hiveserver2 &
```

- Nos conectamos con beeline. No hace falta poner usuario

```
beeline
beeline> !connect jdbc:hive2://localhost:10000
Connected to: Apache Hive (version 2.3.2)
Driver: Hive JDBC (version 2.3.2)
Transaction isolation: TRANSACTION_REPEATABLE_READ
0: jdbc:hive2://nodo1:10000>
```

-
- Vamos a probar ahora que podemos conectarnos en remoto desde otro nodo al tener HIVESERVER2
- Copiamos el directorio hive al nodo2, en la misma ruta, es decir en /opt/hadoop
- Copiamos también el .bashrc al /home/hadoop del segundo nodo. Esto nos permite tener la misma configuración
- Entramos en beeline y le ponemos la dirección del nodo1

```
beeline> !connect jdbc:hive2://nodo1:10000
Connecting to jdbc:hive2://nodo1:10000
Enter username for jdbc:hive2://nodo1:10000:
```

```
Enter password for jdbc:hive2://nodo1:10000:
Connected to: Apache Hive (version 2.3.2)
Driver: Hive JDBC (version 2.3.2)
Transaction isolation: TRANSACTION_REPEATABLE_READ
```

- Comprobamos que vemos lo mismo

```
show databases;
+-----+
| database_name |
+-----+
| default      |
| ejemplo      |
| prueba       |
+-----+
3 rows selected (0,438 seconds)
```

-
- Vamos a probar ahora que podemos conectarnos en remoto desde otro nodo al tener HIVESERVER2
- Copiamos el directorio hive al nodo2, en la misma ruta, es decir en /opt/hadoop
- Copiamos también el .bashrc al /home/hadoop del segundo nodo. Esto nos permite tener la misma configuración
- Entramos en beeline y le ponemos la dirección del nodo1

```
beeline> !connect jdbc:hive2://nodo1:10000
Connecting to jdbc:hive2://nodo1:10000
Enter username for jdbc:hive2://nodo1:10000:
Enter password for jdbc:hive2://nodo1:10000:
Connected to: Apache Hive (version 2.3.2)
Driver: Hive JDBC (version 2.3.2)
Transaction isolation: TRANSACTION_REPEATABLE_READ
```

- Comprobamos que vemos lo mismo

```
show databases;
+-----+
| database_name |
+-----+
| default      |
| ejemplo      |
| prueba       |
+-----+
```


+-----+
3 rows selected (0,438 seconds)

31. Práctica adicional. Deslizamientos de tierra

- Vamos a realizar unas cuantas SELECT contra una DataSet de la NASA, que contiene información sobre deslizamientos de tierra ocurridos alrededor del mundo
- El fichero deslizamientos.csv tiene los datos
- Primero creamos la siguiente tabla en HIVE

```
create table deslizamientos
(
id                bigint,
fecha             string ,
hora              string ,
country           string      ,
nearest_places    string      ,
hazard_type       string      ,
landslide_type    string      ,
motivo            string      ,
storm_name        string      ,
  fatalities       bigint      ,
  injuries         string      ,
  source_name      string      ,
  source_link      string      ,
  location_description string  ,
  location_accuracy string    ,
  landslide_size   string      ,
  photos_link      string      ,
  cat_src          string      ,
  cat_id           bigint      ,
  countryname      string      ,
  near             string      ,
  distance         double      ,
  adminname1       string      ,
  adminname2       string      ,
  population       bigint      ,
  countrycode      string      ,
  continentcode    string      ,
  key              string      ,
  version          string      ,
  tstamp           string      ,
```

```
changeset_id      string      ,
latitude          double      ,
longitude         double      ,
geolocation       string
)
ROW FORMAT DELIMITED
FIELDS TERMINATED BY ';'

```

- Luego la cargamos con los datos del fichero de los deslizamientos. Previamente, hemos copiado el fichero a /tmp

```
load data local inpath '/tmp/deslizamientos.csv' into table deslizamientos;
No rows affected (7,21 seconds)

```

- Hagamos ahora algunos ejemplos de consultas
- Ver el nombre y fecha de las cinco primeras filas

```
select country,fecha from deslizamientos limit 5;
+-----+-----+
| country | fecha |
+-----+-----+
| United Kingdom | 01/02/2007 |
| Peru          | 01/03/2007 |
| Brazil        | 01/05/2007 |
| Brazil        | 01/05/2007 |
| Brazil        | 01/05/2007 |

```

- Averiguar el país, el tipo de deslizamiento y el motivo de aquellos sitios donde haya habido más de 100 víctimas.

```
select country,fecha, landslide_type,motivo,fatalities from
deslizamientos where fatalities > 100;
+-----+-----+-----+-----+-----+
+
| country | fecha | landslide_type | motivo | fatalities |
+-----+-----+-----+-----+-----+
+
| Afghanistan | 03/28/2007 | Landslide | Flooding | 114 |
| Bangladesh  | 06/11/2007 | Landslide | Monsoon  | 128 |
| China        | 05/17/2008 | Mudslide  | Earthquake | 200 |
| China        | 09/08/2008 | Complex   | Dam_Embankment_Collapse | 277 |
| Brazil       | 11/24/2008 | Landslide | Continuous_rain | 109 |
| Taiwan       | 08/10/2009 | Complex   | Tropical_Cyclone | 491 |

```

Philippines	10/09/2009	Landslide	Tropical_Cyclone	104
Uganda	03/01/2010	Complex	Downpour	388
Brazil	04/07/2010	Mudslide	Downpour	196
India	08/06/2010	Landslide	Downpour	234
India	08/06/2010	Landslide	Downpour	182
China	08/07/2010	Landslide	Downpour	1765
Indonesia	10/04/2010	Landslide	Downpour	145
Brazil	01/12/2011	Mudslide	Downpour	424
Brazil	01/12/2011	Mudslide	Downpour	378
Philippines	12/04/2012	Mudslide	Tropical_Cyclone	430
India	06/16/2013	Debris_Flow	Downpour	5000
Afghanistan	05/02/2014	Landslide	Continuous_rain	2100
India	07/30/2014	Mudslide	Continuous_rain	151
Nepal	08/02/2014	Landslide	Continuous_rain	174
	01/04/2006	Mudslide	Downpour	240
	12/12/2014	Landslide	Monsoon	108
	04/28/2015	Mudslide	Snowfall_snowmelt	250
	10/01/2015	Mudslide	Rain	280
	08/02/2015	Landslide	Downpour	253
	05/18/2016	Mudslide	Monsoon	101
	04/02/2016	Landslide	Unknown	104
+-----+-----+-----+-----+				
+				

- Averiguar los deslizamientos ocurridos por tipos de deslizamiento (landslide_type)

```
select landslide_type, count(*) from deslizamientos group by landslide_type;
```

WARNING: Hive-on-MR is deprecated in Hive 2 and may not be available in the future versions. Consider using a different execution engine (i.e. spark, tez) or using Hive 1.X releases.

```
+-----+-----+
```

```
| landslide_type | _c1 |
```

```
+-----+-----+
|          | 18  |
| Complex  | 232 |
| Creep    | 5   |
| Debris_Flow | 173 |
| Earthflow | 3   |
| Lahar    | 7   |
| Landslide | 6637 |
| Mudslide | 1826 |
| Other    | 66  |
| Riverbank_Collapse | 28  |
| Rockfall | 484 |
| Rockslide | 1   |
| Snow_Avalanche | 7   |
| Translational_Slide | 6   |
| Unknown  | 18  |
| landslide | 4   |
| mudslide | 7   |
+-----+-----+;
```

- Averiguar los que han ocurrido agrupados por motivo

```
select motivo, count(*) from deslizamientos group by motivo;
+-----+-----+
|      motivo      | _c1 |
+-----+-----+
|          | 756 |
| Construction    | 52  |
| Continuous_Rain  | 36  |
| Continuous_rain  | 514 |
| Dam_Embankment_Collapse | 9   |
| Downpour         | 4437 |
| Earthquake       | 76  |
| Flooding         | 49  |
| Freeze_thaw      | 26  |
| Mining_digging   | 74  |
| Monsoon          | 122 |
| No_Apparent_Trigger | 2   |
| No_Apparent_trigger | 18  |
| Other           | 15  |
| Rain            | 1912 |
```

Snowfall_snowmelt	74	
Tropical_Cyclone	538	
Unknown	748	
Volcano	1	
monsoon	2	
unknown	61	
+-----+-----+		

- Indicar los 10 países con más deslizamientos registrados

```
select country,count(*) as total from deslizamientos group by country
order by total desc limit 10;
```

country	total	
+-----+-----+		
	3387	
United States	1439	
India	884	
Philippines	546	
China	347	
Nepal	324	
Indonesia	282	
Brazil	205	
United Kingdom	147	
Malaysia	110	
+-----+-----+		

- Crear la siguiente table de países.

```
create table paises
(
nombre string,
cod string)
ROW FORMAT DELIMITED
FIELDS TERMINATED BY ','
```

- Cargamos la tabla countries.csv que tenemos en los recursos de la práctica

```
load data local inpath '/tmp/countries.csv' into table paises;
```

- Comprobamos que ha cargado los registros

```
select * from paises limit 10;
+-----+-----+
| paises.nombre | paises.cod |
```

```
+-----+-----+
| Name          | Code    |
| Afghanistan   | AF      |
| Åland Islands | AX      |
| Albania       | AL      |
| Algeria       | DZ      |
| American Samoa | AS      |
| Andorra       | AD      |
| Angola        | AO      |
| Anguilla      | AI      |
| Antarctica    | AQ      |
+-----+-----+
10 rows selected (0,665 seconds)
```

- Vamos ahora a visualizar el código del país, el nombre y el número de corrimientos de tierra agrupados por motivo

```
select a.cod,b.country,b.motivo,count(*) from paises a join deslizamientos b
on a.nombre=b.country group by a.cod,b.country,b.motivo;
```

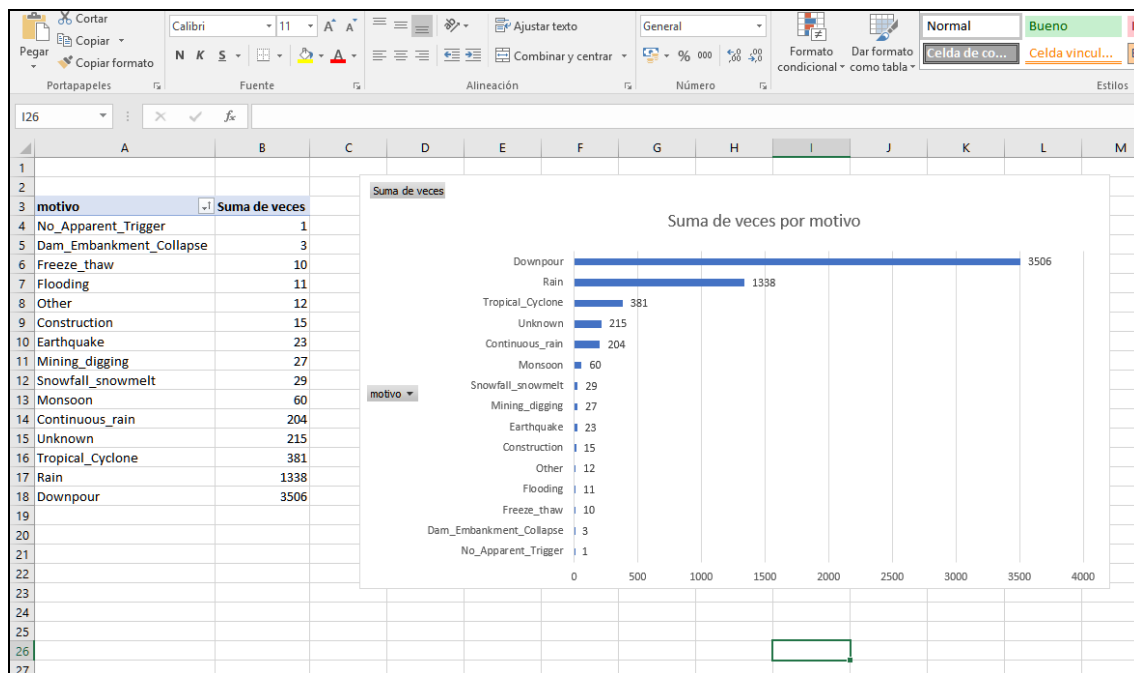
```
+-----+-----+-----+-----+
| a.cod | b.country          | b.motivo          | _c3 |
+-----+-----+-----+-----+
| AE    | United Arab Emirates | Rain              | 1   |
| AF    | Afghanistan         | Continuous_rain   | 1   |
| AF    | Afghanistan         | Downpour          | 4   |
| AF    | Afghanistan         | Flooding          | 1   |
| AF    | Afghanistan         | Rain              | 4   |
| AL    | Albania             | Downpour          | 1   |
| AM    | Armenia             | Downpour          | 3   |
| AO    | Angola              | Downpour          | 3   |
| AR    | Argentina           | Downpour          | 5   |
| AR    | Argentina           | Rain              | 1   |
| AS    | American Samoa      | Downpour          | 4   |
| AS    | American Samoa      | Rain              | 1   |
| AT    | Austria             | Downpour          | 7   |
| AT    | Austria             | Rain              | 2   |
| AT    | Austria             | Snowfall_snowmelt | 1   |
| AU    | Australia           | Continuous_rain   | 2   |
| AU    | Australia           | Downpour          | 56  |
| AU    | Australia           | Mining_digging    | 1   |
| AU    | Australia           | Rain              | 15  |
| AU    | Australia           | Tropical_Cyclone  | 1   |
| AU    | Australia           | Unknown           | 4   |
| AZ    | Azerbaijan          | Downpour          | 16  |
```

AZ	Azerbaijan	Snowfall_snowmelt	1	
AZ	Azerbaijan	Unknown	2	
BA	Bosnia and Herzegovina	Downpour	1	
BA	Bosnia and Herzegovina	Rain	4	
BB	Barbados	Downpour	1	
BD	Bangladesh	Downpour	20	
BD	Bangladesh	Monsoon	3	
BD	Bangladesh	Rain	8	
BD	Bangladesh	Unknown	2	

- Ahora, como práctica final, vamos a exportarlo a un fichero.
- Podemos hacerlo con este comando. Lo dejamos en un directorio denominado datos

```
insert overwrite local directory '/tmp/datos' row format delimited fields
terminated by ',' select a.cod,b.country,b.motivo,count(*) from paises a
join deslizamientos b on a.nombre=b.country group by a.cod,b.country,b.motivo;
```

- Dentro va a generar un fichero denominado 00000_0
- Ahora, por último vamos a importarlo en un Excel para ver el resultado y hacer un gráfico. Sería el punto y final de un trabajo con Big Data



32. Preparación inicial. Descargar y compilar.

- Antes de poder instalar HUE necesitamos realizar una serie de instalaciones en el servidor, en concreto lo haremos en el nodo1.
- Nos conectamos como root y lanzamos los siguientes yum
- NOTA IMPORTANTE: dependiendo del equipo y la versión de Linux con la que lo hagáis, seguramente muchos de los paquetes ya estarán instalados.
- Es conveniente comprobar que todos se han instalado correctamente.
- El proceso puede tardar bastante tiempo porque descarga algunos componentes pesados.
- Asegúrate de tener conexión a internet correcta.

```
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 poder instalar HUE, primero debemos compilarlo.
- Los descargamos de la página de HUE
- Lo descomprimos

```
tar xvf hue-XXXX
```

- Accedemos al directorio que se ha creado
- Lanzamos la compilación

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

33. Configuración de HUE

- Una vez terminado el proceso sin errores, podemos acceder al directorio

```
cd /opt/hadoop/hue
```

- Accedemos al directorio de configuración de HUE
- /opt/hadoop/hue/desktop/conf
- Dentro, debemos tener el fichero hue.ini
- Debemos configurar los valores para:
 - HDFS
 - YARN
 - HIVE
- No hace falta ninguno más porque no tenemos otros productos instalados y no funcionarían.
- Debemos poner correctamente la dirección. En nuestro caso, debería bastar con sustituir localhost por “nodo1” y cambiar el puerto de 8220 a 9000 en la parte de HDFS
- 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.
## 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

34. Arrancar y probar HUE

- Para arrancar el cluster debemos ejecutar el siguiente comando (sería interesante añadir este directorio al PATH de Linux para no tener que buscarlo). Con la opción -d lo lanzamos en background

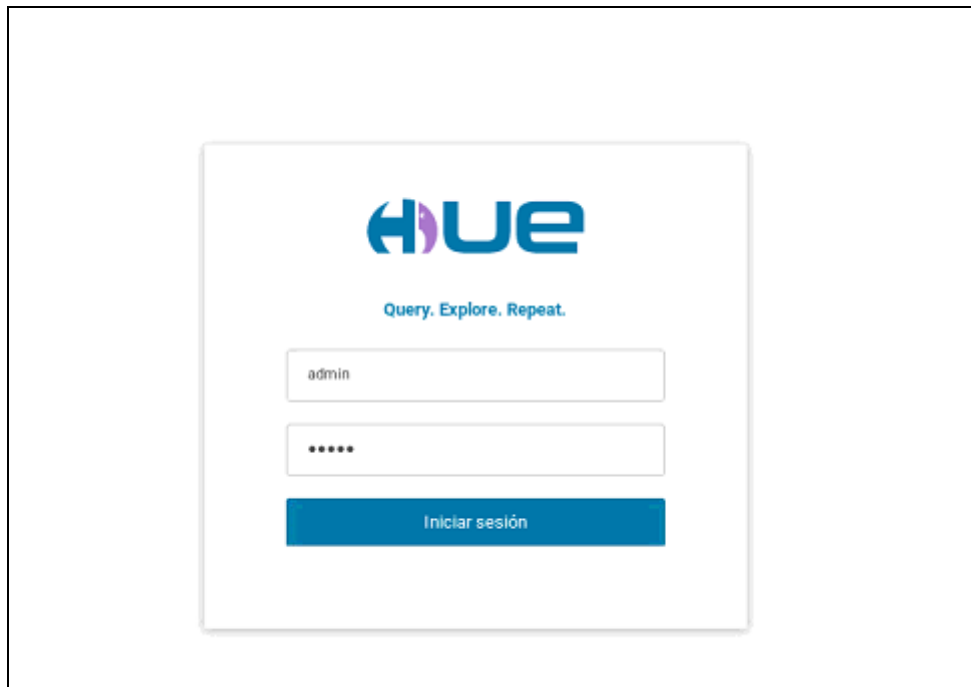
```
/opt/hadoop/hue/build/env/bin/supervisor -d
```

- Si todos va bien debemos tener el proceso funcionando

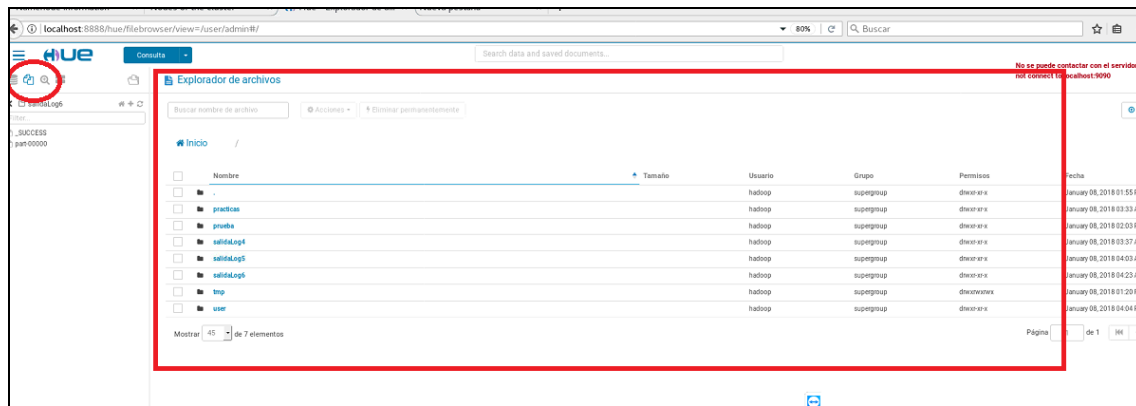
```
ps -ef | grep supervisor
```

```
hadoop      16257      1      1  11:57  ?                00:00:00
/opt/hadoop/hue/build/env/bin/python2.7
/opt/hadoop/hue/build/env/bin/supervisor -d
```

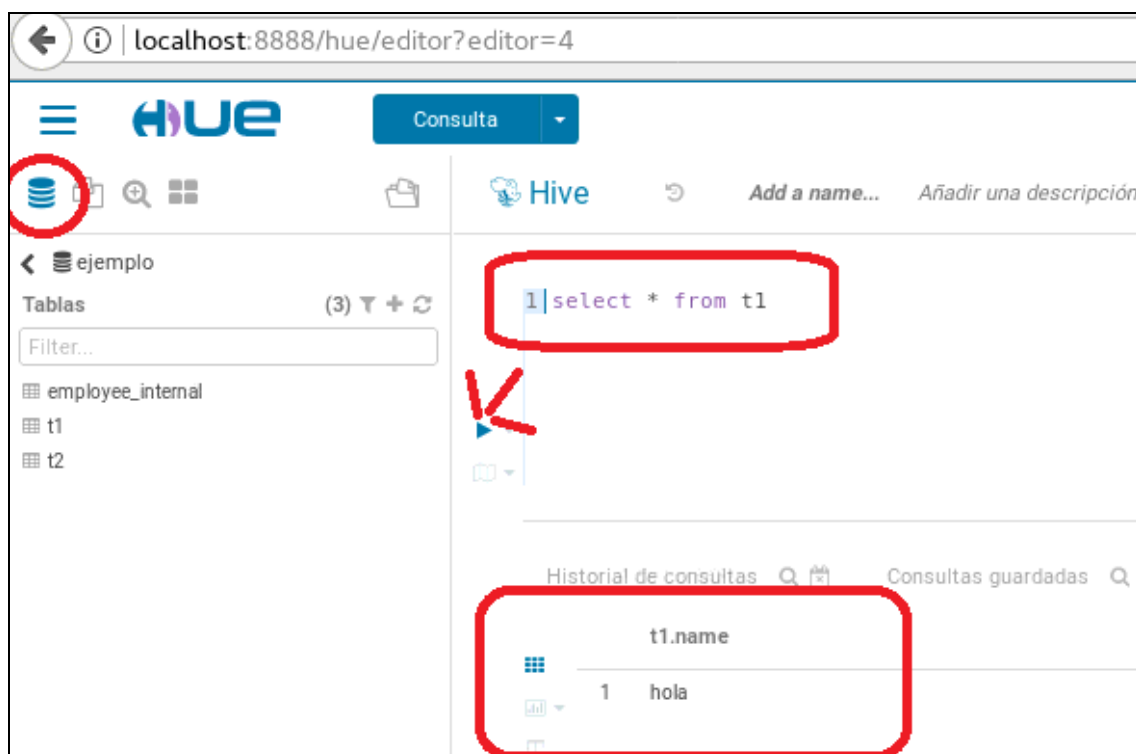
- Ahora abrimos el firefox y nos conectamos por el puerto 8888
- En la primera pantalla se nos pide establecer un usuario para la herramienta, en este caso lo llamo admin, puedes poner el nombre que quieras



- Una vez dentro, tenemos múltiples opciones, vamos a acceder al Browser de HDFS, donde podemos ver lo que tenemos.



- También podemos acceder a HIVE y ver lo que hemos hecho en las prácticas anteriores

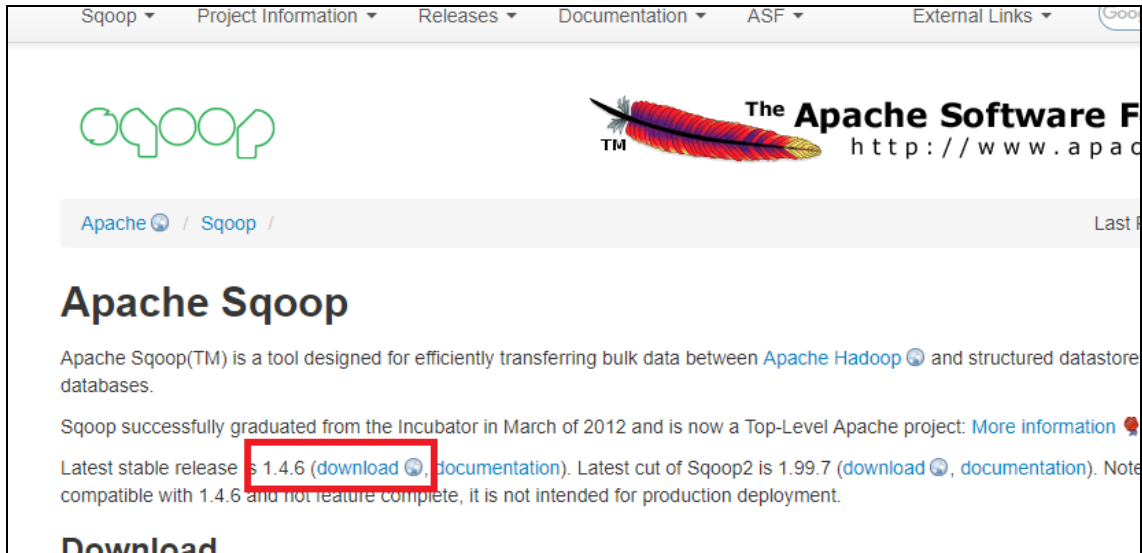


- Desde la opción Consulta → Editor podemos acceder a numerosos editores de distintos productos.
- Como vemos, podemos hacer bastantes cosas con la herramienta

35.

36. SQOOP

- **Descarga e instalación**
- Descargamos el software de la página Web, en concreto la 1.4.6 en el momento de hacer este manual



- Lo descomprimos en /opt/hadoop

```
cd /opt/hadoop/
```

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

- Lo renombramos a sqoop

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

- Ahora editamos el fichero "/home/hadoop/.bashrc" para incluir el HOME y el BIN de sqoop.
- Debe quedar algo parecido a lo siguiente:

```
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/share/hadoop/mapreduce/

#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 zookeeper config dir is
```

- Probamos que funciona

```
sqoop-version
Warning: /opt/hadoop/sqoop/../hbase does not exist! HBase imports will fail.
Please set $HBASE_HOME to the root of your HBase installation.
Warning: /opt/hadoop/sqoop/../hcatalog does not exist! HCatalog jobs will fail.
Please set $HCAT_HOME to the root of your HCatalog installation.
Warning: /opt/hadoop/sqoop/../accumulo does not exist! Accumulo imports will fail.
Please set $ACCUMULO_HOME to the root of your Accumulo installation.
Warning: /opt/hadoop/sqoop/../zookeeper does not exist! Accumulo imports will fail.
Please set $ZOOKEEPER_HOME to the root of your Zookeeper installation.
18/01/14 21:39:37 INFO sqoop.Sqoop: Running Sqoop version: 1.4.6
Sqoop 1.4.6
git commit id c0c5a81723759fa575844a0a1eae8f510fa32c25
Compiled by root on Mon Apr 27 14:38:36 CST 2015
```

- Salen algunos mensajes de los productos que no hemos configurado, pero podemos olvidarlos.
-

37. SQOOP

- **Importación de tablas. Sqoop-import**

- Primero vamos a recordar como podemos ver los datos de Oracle
- Nos vamos al directorio

```
/u01/app/oracle/product/11.2.0/xe/bin
```

- Ejecutamos el siguiente script. Hay que asegurarse de que dejamos los espacios en blanco después del primer punto. Si no, no funciona

```
. ./oracle_env.s
```

- Ahora entramos en Sqlplus

```
sqlplus HR/HR
```

```
SQL*Plus: Release 11.2.0.2.0 Production on Lun Ene 15 20:20:48 2018
```

```
Copyright (c) 1982, 2011, Oracle. All rights reserved.
```

```
Connected to:
```

```
Oracle Database 11g Express Edition Release 11.2.0.2.0 - 64bit Production
```

```
SQL>
```

- Podemos usar el siguiente comando para ver los datos de la tabla DEPARTMENTS

```
select * from departments;
```

```
DEPARTMENT_ID DEPARTMENT_NAME          MANAGER_ID
LOCATION_ID
```

```
-----
      10 Administration                200      1700
      20 Marketing                    201      1800
      30 Purchasing                   114      1700
      40 Human Resources               203      2400
      50 Shipping                     121      1500
      60 IT                           103      1400
      70 Public Relations              204      2700
      80 Sales                        145      2500
      90 Executive                     100      1700
     100 Finance                      108      1700
     110 Accounting                   205      1700
```

```
DEPARTMENT_ID DEPARTMENT_NAME          MANAGER_ID
LOCATION_ID
-----
```

120 Treasury	1700
--------------	------

- Ejecutamos el siguiente comando para cargar la tabla DEPARTMENTS

```
sqoop-import --connect jdbc:oracle:thin:@nodo1:1521:XE --username HR --password HR --table DEPARTMENTS --target-dir /empleados --as-textfile
```

- Podemos comprobar que lo ha cargado...

```
hdfs dfs -ls /ejemplo1
Found 5 items
-rw-r--r--  3 hadoop supergroup      0 2018-01-15 02:54 /ejemplo1/_SUCCESS
-rw-r--r--  3 hadoop supergroup    165 2018-01-15 02:54 /ejemplo1/part-m-00000
-rw-r--r--  3 hadoop supergroup    136 2018-01-15 02:54 /ejemplo1/part-m-00001
-rw-r--r--  3 hadoop supergroup    197 2018-01-15 02:54 /ejemplo1/part-m-00002
-rw-r--r--  3 hadoop supergroup    174 2018-01-15 02:54 /ejemplo1/part-m-00003
```

- Y por ultimo comprobar el resultado de alguno de los ficheros

```
hdfs dfs -cat /empleados/part-m-00000
10,Administration,200,1700
20,Marketing,201,1800
30,Purchasing,114,1700
40,Human Resources,203,2400
50,Shipping,121,1500
60,IT,103,1400
70,Public Relations,204,2700
```

- Podemos ver todos los ficheros para ver el resultado final

```
hdfs dfs -cat /empleados/*
10,Administration,200,1700
20,Marketing,201,1800
30,Purchasing,114,1700
40,Human Resources,203,2400
50,Shipping,121,1500
60,IT,103,1400
70,Public Relations,204,2700
80,Sales,145,2500
90,Executive,100,1700
100,Finance,108,1700
110,Accounting,205,1700
120,Treasury,null,1700
130,Corporate Tax,null,1700
140,Control And Credit,null,1700
150,Shareholder Services,null,1700
```

```
160,Benefits,null,1700
170,Manufacturing,null,1700
180,Construction,null,1700
190,Contracting,null,1700
200,Operations,null,1700
210,IT Support,null,1700
220,NOC,null,1700
230,IT Helpdesk,null,1700
240,Government Sales,null,1700
250,Retail Sales,null,1700
260,Recruiting,null,1700
270,Payroll,null,1700
```

- También podemos cargar solo determinadas columnas. Por ejemplo, si solo quiero department_name y department_id

```
sqoop-import --connect jdbc:oracle:thin:@nodo1:1521:XE --username HR --password HR --table DEPARTMENTS --columns "DEPARTMENT_ID,DEPARTMENT_NAME" --target-dir /practicass/departamentos --as-textfile
```

- El resultado puede ser el siguiente:

```
hdfs dfs -cat /practicass/departamentos/*
```

```
10,Administration
20,Marketing
30,Purchasing
40,Human Resources
50,Shipping
60,IT
70,Public Relations
80,Sales
90,Executive
100,Finance
110,Accounting
120,Treasury
130,Corporate Tax
140,Control And Credit
150,Shareholder Services
160,Benefits
170,Manufacturing
180,Construction
190,Contracting
200,Operations
210,IT Support
```

```
220,NOC
230,IT Helpdesk
240,Government Sales
250,Retail Sales
260,Recruiting
270,Payroll
```

- También puedo poner alguna condición, con la cláusula where

```
sqoop-import --connect jdbc:oracle:thin:@nodo1:1521:XE --username HR --password HR --table DEPARTMENTS --columns "DEPARTMENT_ID,DEPARTMENT_NAME" --where "department_id>200" --target-dir /practicass/dept_200 --as-textfile
```

- El resultado es:

```
dfs dfs -cat /practicass/dept_200/*210,IT Support
220,NOC
230,IT Helpdesk
240,Government Sales
250,Retail Sales
260,Recruiting
270,Payroll
```

- Ahora vamos a realizar una condición más compleja, de tipo free-form. Por ejemplo

```
sqoop import --connect jdbc:oracle:thin:@192.168.43.94:1521:xe --username HR --password HR --query " SELECT FIRST_NAME, LAST_NAME, DEPARTMENT_NAME, SALARY FROM EMPLOYEES A, DEPARTMENTS B WHERE A.DEPARTMENT_ID=B.DEPARTMENT_ID AND SALARY > 4000 AND \${CONDITIONS}" --target-dir /practicass/dept_emple --as-textfile --split-by first_name
```

- El resultado sería

```
hdfs dfs -cat /practicass/dept_emple/*Den,Raphaely,Purchasing,11000
Alexis,Bull,Shipping,4100
Adam,Fripp,Shipping,8200
David,Austin,IT,4800
Bruce,Ernst,IT,6000
Alexander,Hunold,IT,9000
Diana,Lorentz,IT,4200
David,Bernstein,Sales,9500
Eleni,Zlotkey,Sales,10500
Alberto,Errazuriz,Sales,12000
Christopher,Olsen,Sales,8000
Allan,McEwen,Sales,9000
Clara,Vishney,Sales,10500
Danielle,Green,Sales,9500
```

David, Lee, Sales, 6800
 Amit, Banda, Sales, 6200
 Elizabeth, Bates, Sales, 7300
 Ellen, Abel, Sales, 11000
 Alyssa, Hutton, Sales, 8800
 Charles, Johnson, Sales, 6200
 Daniel, Faviet, Finance, 9000
 Jennifer, Whalen, Administration, 4400
 Kevin, Mourgos, Shipping, 5800
 Hermann, Baer, Public Relations, 10000
 Gerald, Cambrault, Sales, 11000
 Jack, Livingston, Sales, 8400
 Jonathon, Taylor, Sales, 8600
 Harrison, Bloom, Sales, 10000
 Janette, King, Sales, 10000
 John, Russell, Sales, 14000
 Karen, Partners, Sales, 13500
 Lex, De Haan, Executive, 17000
 John, Chen, Finance, 8200
 Ismael, Sciarra, Finance, 7700
 Jose Manuel, Urman, Finance, 7800
 Pat, Fay, Marketing, 6000
 Michael, Hartstein, Marketing, 13000
 Nandita, Sarchand, Shipping, 4200

38. ZOOKEEPER

Instalación y configuración

- Descargamos ZooKeeper de la página zookeeper.apache.org
- Lo descomprimos 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/bin:$H
IVE_HOME/bin:$ZOOKEEPER_HOME/bin
```

- Si entramos de nuevo en un terminal tendremos ya cargadas las variables.
- Nos situamos en el directorio de configuración

```
cd /opt/hadoop/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

```
ps
20912 ResourceManager
17315 RunJar
20456 NameNode
16905 RunJar
22377 Jps
20699 SecondaryNameNode
26475 QuorumPeerMain
```

- Podemos preguntar el estado y si es leader o follower

```
zkServer.sh status
ZooKeeper JMX enabled by default
```



```
Using config: /opt/hadoop/zoo/bin/../conf/zoo.cfg
Mode: follower
```

Trabajar con el cliente

- Accedemos al cliente con zkClient.sh

```
zkClient.sh
```

- Escribimos help para ver la ayuda disponible en el cliente

```
[zk: localhost:2181(CONNECTED) 0] help
ZooKeeper -server host:port cmd args
    stat path [watch]
    set path data [version]
    ls path [watch]
    delquota [-n|-b] path
    ls2 path [watch]
    setAcl path acl
    setquota -n|-b val path
    history
    redo cmdno
    printwatches on|off
    delete path [version]
    sync path
    listquota path
    rmr path
    get path [watch]
    create [-s] [-e] path data acl
    addauth scheme auth
    quit
    getAcl path
    close
    connect host:port
[zk: localhost:2181(CONNECTED) 1]
```

- 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
Created /m1
```

- Comprobamos el resultado

```
get /m1
v1
cZxid = 0x100000008
ctime = Thu Feb 01 22:15:06 CET 2018
mZxid = 0x100000008
mtime = Thu Feb 01 22:15:06 CET 2018
pZxid = 0x100000008
cversion = 0
dataVersion = 0
aclVersion = 0
ephemeralOwner = 0x0
dataLength = 2
numChildren = 0
```

- 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 /
[zookeeper]
```

39. Alta Disponibilidad HDFS

Propiedades a configurar.

- Debemos incorporar las siguientes propiedades en nuestros ficheros. NOTA. No son los ficheros completos, solo las propiedades que habría que añadir o modificar.
- 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-cluster</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.ConfiguredFailoverProxyProvider
</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 satisfactoriamente 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
- Preparamos y arrancamos el zkController

```
hdfs zkfc -formatZK
hadoop-daemon.sh start zkfc
```

- Vamos al nodo2
- Preparamos y arrancamos el zkController

```
hdfs zkfc -formatZK
hadoop-daemon.sh start zkfc
```

- Comprobamos que tenemos todos los procesos funcionando, con jps

```
5186 JournalNode
5411 DFSZKFailoverController
6634 NameNode
5213 QuorumPeerMain
7838 Jps
```

- Paramos todo el cluster

```
stop-dfs.sh
```

- Arrancamos de nuevo el cluster para comprobar que todo arranca satisfactoriamente

```
start-dfs.sh
Starting namenodes on [nodo1 nodo2]
nodo1: starting namenode, logging to /opt/hadoop/logs/hadoop-hadoop-namenode-nodo1.out
nodo2: starting namenode, logging to /opt/hadoop/logs/hadoop-hadoop-namenode-nodo2.out
nodo5: starting datanode, logging to /opt/hadoop/logs/hadoop-hadoop-datanode-nodo5.out
nodo3: starting datanode, logging to /opt/hadoop/logs/hadoop-hadoop-datanode-nodo3.out
nodo7: starting datanode, logging to /opt/hadoop/logs/hadoop-hadoop-datanode-nodo7.out
nodo4: starting datanode, logging to /opt/hadoop/logs/hadoop-hadoop-datanode-nodo4.out
nodo6: starting datanode, logging to /opt/hadoop/logs/hadoop-hadoop-datanode-nodo6.out
Starting journal nodes [nodo3 nodo2 nodo1]
nodo1: starting journalnode, logging to /opt/hadoop/logs/hadoop-hadoop-journalnode-nodo1.out
nodo2: starting journalnode, logging to /opt/hadoop/logs/hadoop-hadoop-journalnode-nodo2.out
nodo3: starting journalnode, logging to /opt/hadoop/logs/hadoop-hadoop-journalnode-nodo3.out
Starting ZK Failover Controllers on NN hosts [nodo1 nodo2]
nodo2: starting zkfc, logging to /opt/hadoop/logs/hadoop-hadoop-zkfc-nodo2.out
nodo1: starting zkfc, logging to /opt/hadoop/logs/hadoop-hadoop-zkfc-nodo1.out
```

- Arrancamos la Web Admin del nodo1

- Arrancamos el Web Admin del nodo2. Vemos que está en standby

www.apasoft-training.com

- También lo podemos ver desde línea de comandos:

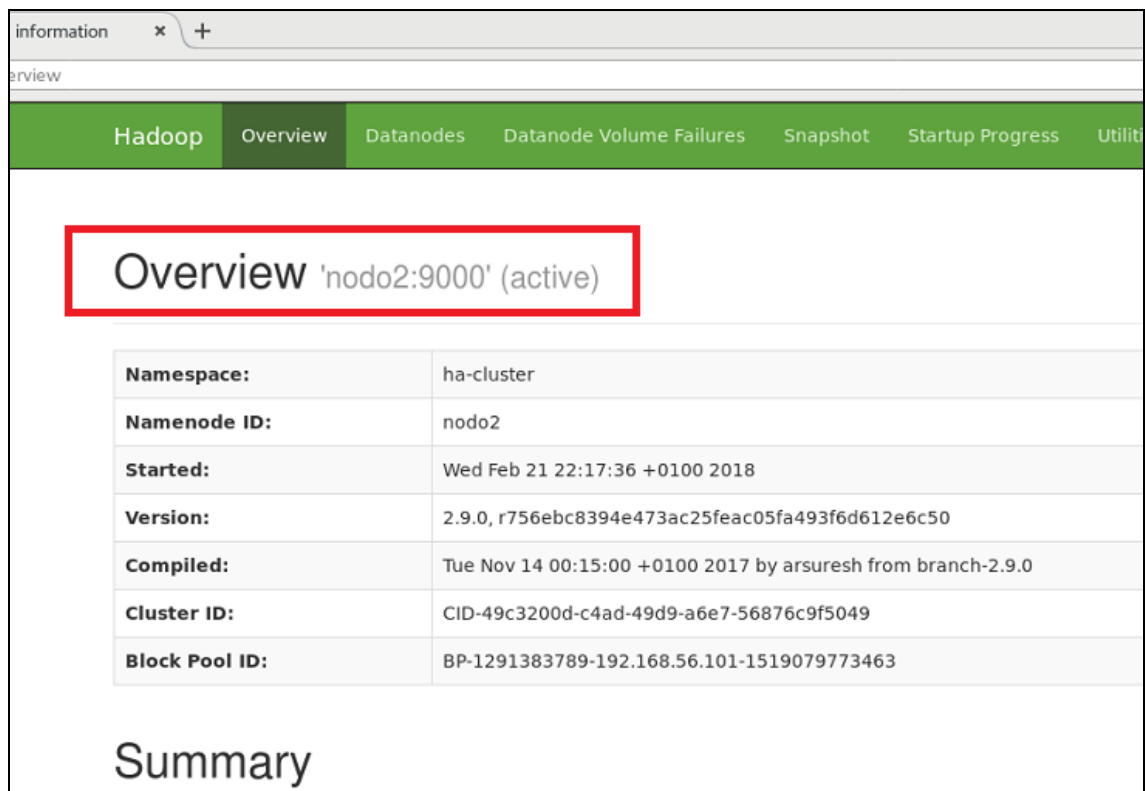
```
$hdfs haadmin -getServiceState nodo1
active
$ hdfs haadmin -getServiceState nodo2
standby
$ hdfs haadmin -getAllServiceState
nodo1:9000    active
nodo2:9000    standby
```

- 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



information x +

erview

Hadoop Overview Datanodes Datanode Volume Failures Snapshot Startup Progress Utilities

Overview 'nodo2:9000' (active)

Namespace:	ha-cluster
Namenode ID:	nodo2
Started:	Wed Feb 21 22:17:36 +0100 2018
Version:	2.9.0, r756ebc8394e473ac25feac05fa493f6d612e6c50
Compiled:	Tue Nov 14 00:15:00 +0100 2017 by arsuresh from branch-2.9.0
Cluster ID:	CID-49c3200d-c4ad-49d9-a6e7-56876c9f5049
Block Pool ID:	BP-1291383789-192.168.56.101-1519079773463

Summary

40. Instalar Spark

- Acceder a la página Web de Apache
- Acceder a Downloads
- Descargarnos una versión sin haddop
-