

TECNOLOGÍAS DE COMPUTACIÓN DE DATOS MASIVOS

```
hdmaster@NameNode:~$ hdfs dfs.FileSystemCat salidaWordCount.txt
a 45639
aa 1038
aachen 965
aaliyah 2063
aardvark 2056
aardvarks 992
aaron 1005
ab 1019
aba 8
abaci 1032
aback 1049
abacus 2031
abacuses 989
abad 70
abade 2
abadejo 4
abades 11
abadesa 33
abadesas 1
abadia 7
abaft 990
abaja 1
abajada 1
abajan 1
abajar 7
abajarse 3
abajase 1
abaje 1
abajen 1
abajes 1
abajito 1
abajo 409
abajó 2
abalanzaba 1
```

Consola Consola nº 2
(hdmaster) NameNode - KDE Terminal Emulator

PRÁCTICA 2 – SISTEMA HDFS

MOISÉS FRUTOS PLAZA 48488132-S
moises.frutos@um.es
MÁSTER EN BIG DATA

ENTREGADO EL 20 DE NOVIEMBRE DE 2016

ÍNDICE

Índice de contenido

1. INTRODUCCIÓN.....	3
2. DESARROLLAR Y COMPILAR EN JAVA LOS PROGRAMAS FILESYSTEMCAT Y FILESYSTEMCOPYSECONDHALF.....	3
3. SUBIR LOS PROGRAMAS COMPILADOS Y EL FICHERO SALIDAWORDCOUNT.TXT DESDE LA MÁQUINA LOCAL AL NAMENODE.....	5
4. PREPARAR EL ENTORNO ANTES DE LA EJECUCIÓN DE LOS PROGRAMAS EN EL NAMENODE.....	5
5. EJECUCIÓN Y RESULTADOS DE LOS PROGRAMAS.....	6
6. CREACIÓN DE UN DIRECTORIO CON CUOTA LIMITADA A DOS FICHEROS.....	7
7. COMPROBACIÓN DE QUE NO DEJA COPIAR TRES O MÁS FICHEROS AL DIRECTORIO	8
8. EJECUCION Y SALIDA DEL CHECK HDFS EN TODO EL SISTEMA HDFS DEL CLÚSTER	9
9. CREACIÓN DE UN DIRECTORIO MEDIANTE WEBHDFS.....	10
10. ENVÍO DE FICHERO LOCAL AL SISTEMA HDFS DEL CLÚSTER MEDIANTE WEBHDFS.....	11
11. COMPROBACIÓN DE QUE EL FICHERO SE ENVIÓ CORRECTAMENTE ABRIÉNDOLO MEDIANTE WEBHDFS.....	13

1. INTRODUCCIÓN

Esta práctica tiene como propósito probar el sistema de ficheros HDFS de un clúster Hadoop, para ello se siguen una serie de pasos:

- Desarrollar y compilar en Java los programas a ejecutar FileSystemCat y FileSystemCopySecondHalf mediante Eclipse y línea de comandos
- Subir los programas compilados y el fichero salidaWordCount.txt mediante scp desde la máquina local al NameNode
- Preparar el entorno antes de la ejecución de los programas en el NameNode
- Ejecución y resultados de los programas
- Creación de un directorio con cuota limitada a dos ficheros
- Comprobación de que no deja copiar tres o más ficheros al directorio
- Ejecución y salida del check hdfs en todo el sistema HDFS del clúster
- Creación de un directorio mediante webhdfs
- Envío de fichero local al sistema HDFS del clúster mediante webhdfs
- Comprobación de que el fichero se envió correctamente abriéndolo mediante webhdfs

A continuación se irá viendo, apartado por apartado, el desarrollo de toda la práctica dos en su totalidad incluyendo la parte opcional (***aparte de este documento de seguimiento se incluyen todos los ficheros fuentes, compilados, readme, capturas de pantalla, ficheros de texto de prueba, salidas, etc. dentro del propio fichero comprimido enviado***).

2. DESARROLLAR Y COMPILAR EN JAVA LOS PROGRAMAS FILESYSTEMCAT Y FILESYSTEMCOPYSECONDDHALF

Para desarrollar los programas requeridos he utilizado tanto el IDE Eclipse para programar el código como la línea de comandos para su posterior compilación. En definitiva se han seguido los siguientes pasos:

1. Abrimos Eclipse y creamos un nuevo proyecto Java que contendrá los ficheros fuente .java que vamos a programar, en mi caso he llamado al proyecto *filesystemcat*.
2. Creamos un paquete dentro de *src/main/java* llamado *hdfs*, ya que en él incluiremos las dos clases .java que vamos a desarrollar.
3. Creamos la primera clase llamándola *FileSystemCat.java* y copiamos el código de las transparencias del tema HDFS. Nos saldrán algunos errores marcados en rojo porque tenemos que importar las librerías *hadoop common* para *hdfs*.
4. Para importar las librerías yo he optado por incluir en el Buildpath la librería externa *hadoop-common-2.7.3.jar* que, en mi caso, se encontraba dentro del directorio

`/home/moises/hadoop/share/hadoop/common.`

5. Tras esto se realizan cuantas importaciones sean necesarias hasta que desaparezcan los errores marcados en rojo.
6. Aprovechando que el código de la primera clase ya estaba listo, he desarrollado el código de la segunda clase que he denominado *FileSystemCopySecondHalf.java*.
7. Tras tener los dos .java creados sin errores de marcas en rojo era el momento de compilarlos a .class mediante la línea de comandos.
8. Para poder compilarlos necesitaba el classpath de Hadoop, el cual lo podemos obtener poniendo el comando de aquí

```
$hadoop classpath
```

9. Después he aprovechado para copiar el resultado de la salida y crearme una variable de entorno llamada HADOOP_CLASSPATH incluyéndole la cadena “:.” al final, que en mi máquina quedaba tal que así:

```
$export
HADOOP_CLASSPATH=/home/moises/hadoop/etc/hadoop:/home/moises/hadoop/share/hadoop/common/lib/*:/home/moises/hadoop/share/hadoop/common/*:/home/moises/hadoop/share/hadoop/hdfs:/home/moises/hadoop/share/hadoop/hdfs/lib/*:/home/moises/hadoop/share/hadoop/hdfs/*:/home/moises/hadoop/share/hadoop/yarn/lib/*:/home/moises/hadoop/share/hadoop/yarn/*:/home/moises/hadoop/share/hadoop/mapreduce/lib/*:/home/moises/hadoop/share/hadoop/mapreduce/*:/contrib/capacity-scheduler/*.jar:.
```

10. De paso he aprovechado para meterla dentro de mi .bashrc para no tener que volver a exportarla cuando vuelva a iniciar sesión.
11. Antes de compilar hay que situarse en el directorio donde se encuentra el paquete hdfs creado previamente en nuestro proyecto Java. En mi caso era así:

```
$cd /home/moises/workspaceMARS/filesystemcat/src/main/java/hdfs
```

12. Tras todo esto, ya era posible poder compilar los dos .java para convertirlos en .class mediante las siguientes instrucciones:

```
$javac -classpath $HADOOP_CLASSPATH FileSystemCat.java
$javac -cp $HADOOP_CLASSPATH FileSystemCopySecondHalf.java
```

Tras esto ya tenemos la plataforma de Azure lista para preparar nuestra primera máquina virtual que nos va a servir de base para montar el clúster Hadoop.

3. SUBIR LOS PROGRAMAS COMPILADOS Y EL FICHERO SALIDAWORDCOUNT.TXT DESDE LA MÁQUINA LOCAL AL NAMENODE

Antes de nada iniciamos todo el clúster como es de costumbre mediante *azure login* y el script *encenderMaquinas.sh* creado en la práctica anterior. Tras esperar a que se inicie al menos el NameNode entramos en el mismo mediante *ssh* como usuario *hdmaster* e iniciamos los demonios ejecutando el script *iniciarDemonios.sh* también creado en la práctica anterior. Tras esto ya podemos continuar con la subida de los programas compilados y el fichero *salidaWordCount.txt* que lo necesitaremos para probar los programas.

13. Ya que tenemos el paquete *hdfs* con los *.java* y los *.class* compilados, podemos subirlo entero al NameNode mediante *scp*:

```
$scp -r ./hdfs hdmaster@NameNode:~
```

14. Subimos también el fichero *salidaWordCount.txt* al NameNode (primero tenemos que situarnos donde tengamos el fichero):

```
$scp -r salidaWordCount.txt hdmaster@Namenode:~
```

15. Una vez dentro del NameNode y con todos los demonios ya iniciados

En principio, ya tenemos todo lo que necesitamos en el NameNode. Ahora lo que precisamos hacer es preparar el entorno para poder ejecutar los programas.

4. PREPARAR EL ENTORNO ANTES DE LA EJECUCIÓN DE LOS PROGRAMAS EN EL NAMENODE

Antes de poder ejecutar los programas tenemos que preparar el entorno del NameNode, para ello seguimos los siguientes pasos:

16. Una vez dentro del NameNode y con todos los demonios ya iniciados, tenemos que salir del modo *hdfs safemode* con:

```
$hdfs dfsadmin -safemode leave
```

17. Incluir el fichero *salidaWordCount.txt* en el sistema HDFS con:

```
$hdfs dfs -put salidaWordCount.txt
```

18. Ver el classpath actual de hadoop mediante la orden:

```
$hadoop classpath
```

19. Añadir la variable de entorno HADOOP_CLASSPATH concatenándole al classpath de hadoop anterior la cadena “:.” (tal y como habíamos hecho previamente en la máquina local, pero con el classpath hadoop del NameNode):

```
$export
HADOOP_CLASSPATH=/opt/yarn/hadoop/etc/hadoop:/opt/yarn/hadoop/share/hadoop/common/lib/*:/opt/yarn/hadoop/share/hadoop/common/*:/opt/yarn/hadoop/share/hadoop/hdfs:/opt/yarn/hadoop/share/hadoop/hdfs/lib/*:/opt/yarn/hadoop/share/hadoop/yarn/lib/*:/opt/yarn/hadoop/share/hadoop/yarn/*:/opt/yarn/hadoop/share/hadoop/mapreduce/lib/*:/opt/yarn/hadoop/share/hadoop/mapreduce/*:/contrib/capacity-scheduler/*.jar:.
```

20. Añadir los ficheros .class al sistema hdfs usando:

```
$hdfs dfs -put ./hdfs/FileSystemCat.class
$hdfs dfs -put ./hdfs/FileSystemCopySecondHalf.class
```

Con esto ya tenemos el entorno del NameNode listo para ejecutar los programas.

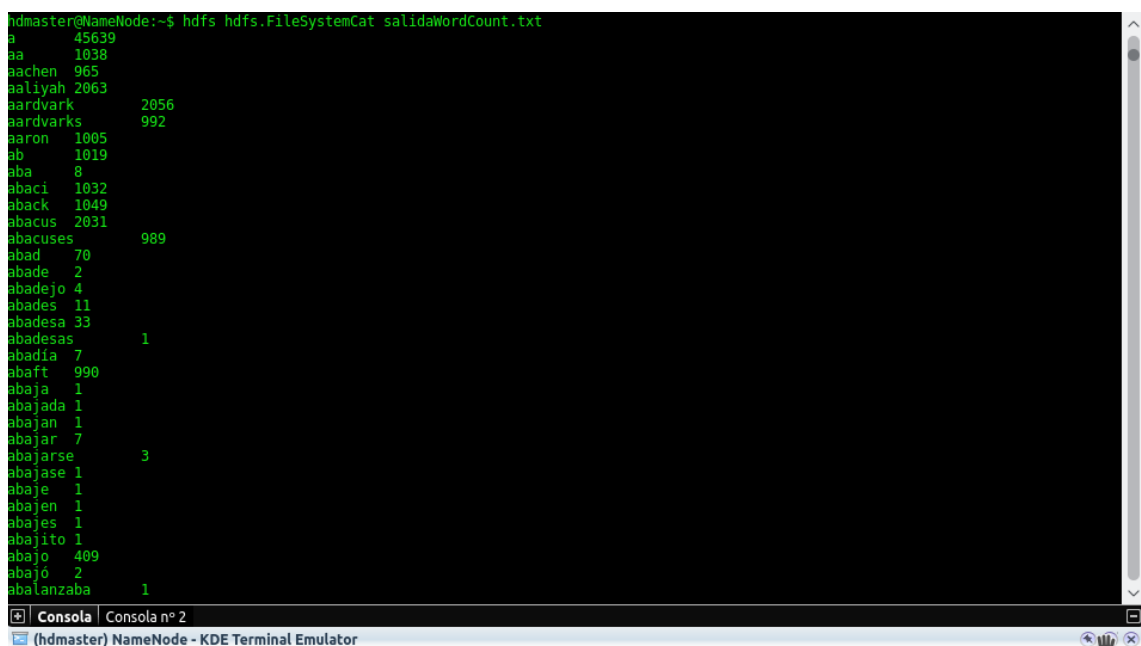
5. EJECUCIÓN Y RESULTADOS DE LOS PROGRAMAS

Ahora es cuando vamos a ejecutar los programas y ver los resultados obtenidos.

21. Ejecutamos primero el FileSystemCat mediante el comando:

```
$hdfs hdfs.FileSystemCat salidaWordCount.txt
```

En la salida podemos comprobar efectivamente que muestra el contenido del fichero *salidaWordCount.txt*



```
hdmaster@NameNode:~$ hdfs hdfs.FileSystemCat salidaWordCount.txt
aa 45639
a 1038
aachen 965
aaliyah 2063
aardvark 2056
aardvarks 992
aaron 1005
ab 1019
aba 8
abaci 1032
aback 1049
abacus 2031
abacuses 989
abad 70
abade 2
abadejo 4
abades 11
abadesa 33
abadesas 1
abadia 7
abaft 990
abaja 1
abajada 1
abajan 1
abajar 7
abajarse 3
abajase 1
abajaje 1
abajen 1
abajes 1
abajito 1
abajo 409
abajó 2
abalanza 1
```

22. Ejecutamos después el `FileSystemCopySecondHalf` con el comando:

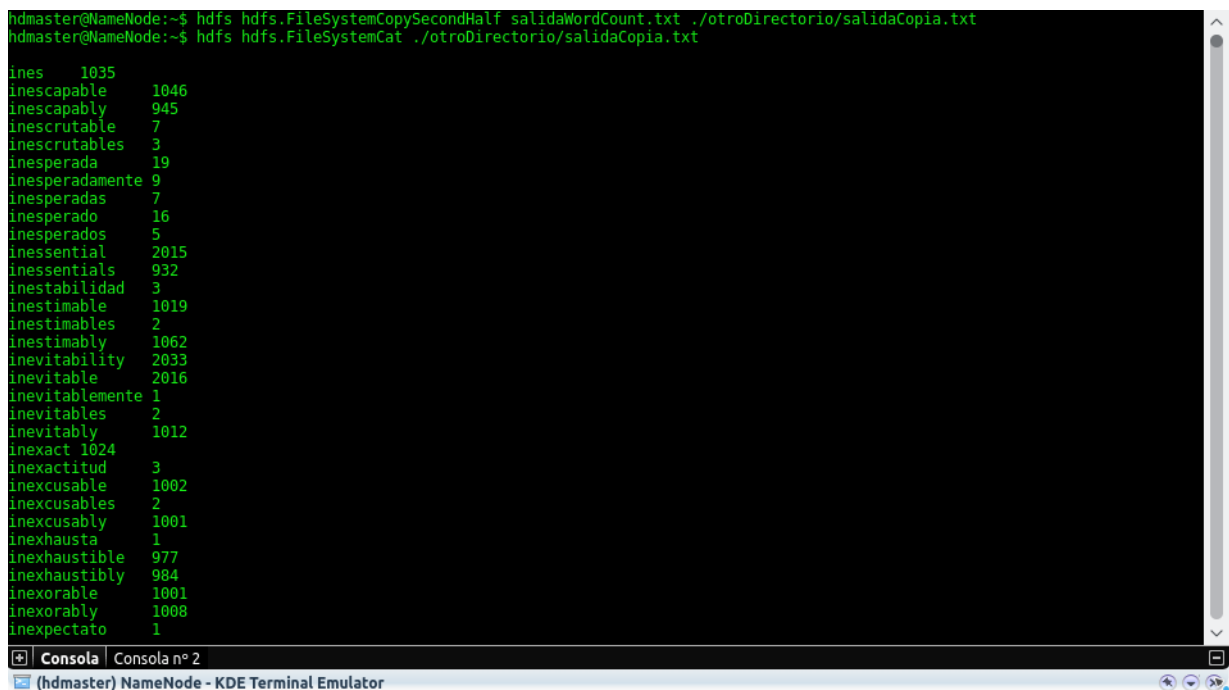
```
$hdfs hdfs.FileSystemCopySecondHalf salidaWordCount.txt
./otroDirectorio/salidaCopia.txt
```

La salida en este caso no nos tiene que devolver nada ya que lo que en realidad hace es copiar la mitad inferior del fichero *salidaWordCount.txt* en *salidaCopia.txt* creando además del propio fichero de salida el directorio llamado '*otroDirectorio*'

23. Podemos ejecutar de nuevo `FileSystemCat` con el nuevo fichero creado para comprobar que se ha copiado correctamente:

```
$hdfs hdfs.FileSystemCat ./otroDirectorio/salidaCopia.txt
```

La salida esta vez sí que nos muestra el contenido del nuevo fichero, el cual sólo contiene la mitad inferior del de *salidaWordCount.txt*



```
hdmaster@NameNode:~$ hdfs hdfs.FileSystemCopySecondHalf salidaWordCount.txt ./otroDirectorio/salidaCopia.txt
hdmaster@NameNode:~$ hdfs hdfs.FileSystemCat ./otroDirectorio/salidaCopia.txt
ines 1035
inescapable 1046
inescapably 945
inescrutable 7
inescrutables 3
inesperada 19
inesperadamente 9
inesperadas 7
inesperado 16
inesperados 5
inessential 2015
inessentials 932
inestabilidad 3
inestimable 1019
inestimables 2
inestimably 1062
inevitability 2033
inevitable 2016
inevitablemente 1
inevitables 2
inevitably 1012
inexact 1024
inexactitud 3
inexcusable 1002
inexcusables 2
inexcusably 1001
inexhausta 1
inexhaustible 977
inexhaustibly 984
inexorable 1001
inexorably 1008
inexpectato 1
```

Una vez hecho todo esto ya podemos decir que hemos completado los puntos 1 y 2 de la práctica. A continuación, en el siguiente apartado comenzaremos con el punto 3.

6. CREACIÓN DE UN DIRECTORIO CON CUOTA LIMITADA A DOS FICHEROS

En este apartado comienza el punto 3 de la práctica, que pide, por una parte, crear un directorio y establecerle 2 ficheros de cuota, y comprobar que no se pueden copiar en él 3 ficheros o más, y por otra, hacer un check a todo el sistema de ficheros hdfs y mostrar el resultado. Dentro de este apartado veremos como crear dicho directorio y como establecerle la cuota de dos ficheros.

24. Creamos primero el directorio en hdfs:

```
$hdfs dfs -mkdir directorioSolo2Copias
```

25. Después le definimos la cuota a sólo dos ficheros:

```
$hdfs dfsadmin -setQuota 2 directorioSolo2Copias
```

7. COMPROBACIÓN DE QUE NO DEJA COPIAR TRES O MÁS FICHEROS AL DIRECTORIO

26. Copiamos varias veces, por ejemplo, el fichero salidaWordCount.txt en el mismo directorio para tener tres copias del mismo además del original:

```
$cp salidaWordCount.txt 1.txt  
$cp salidaWordCount.txt 2.txt  
$cp salidaWordCount.txt 3.txt
```

27. Comprobamos primero que de hecho tenemos 3 copias o más de ficheros .txt con el comando:

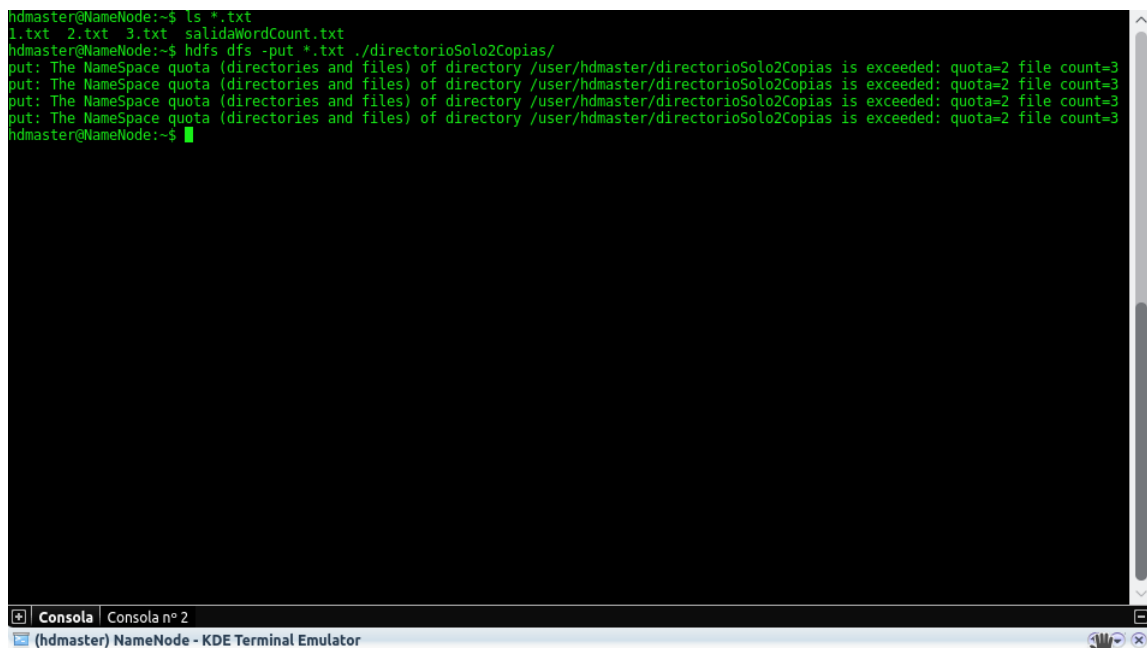
```
$ls *.txt
```

En nuestro caso tenemos 4 ficheros .txt

28. Ahora podemos comprobar si nos deja copiar o no los cuatro ficheros .txt al directorio que hemos creado con sólo una cuota de dos ficheros:

```
$hdfs dfs -put *.txt ./directorioSolo2Copias/
```

La salida nos dice por cada uno de los ficheros que intenta copiar dentro del directorio que estamos superando la cuota de ficheros del mismo, y por tanto no nos deja copiarlos.



```
hdmaster@NameNode:~$ ls *.txt  
1.txt 2.txt 3.txt salidaWordCount.txt  
hdmaster@NameNode:~$ hdfs dfs -put *.txt ./directorioSolo2Copias/  
put: The NameSpace quota (directories and files) of directory /user/hdmaster/directorioSolo2Copias is exceeded: quota=2 file count=3  
put: The NameSpace quota (directories and files) of directory /user/hdmaster/directorioSolo2Copias is exceeded: quota=2 file count=3  
put: The NameSpace quota (directories and files) of directory /user/hdmaster/directorioSolo2Copias is exceeded: quota=2 file count=3  
put: The NameSpace quota (directories and files) of directory /user/hdmaster/directorioSolo2Copias is exceeded: quota=2 file count=3  
hdmaster@NameNode:~$
```

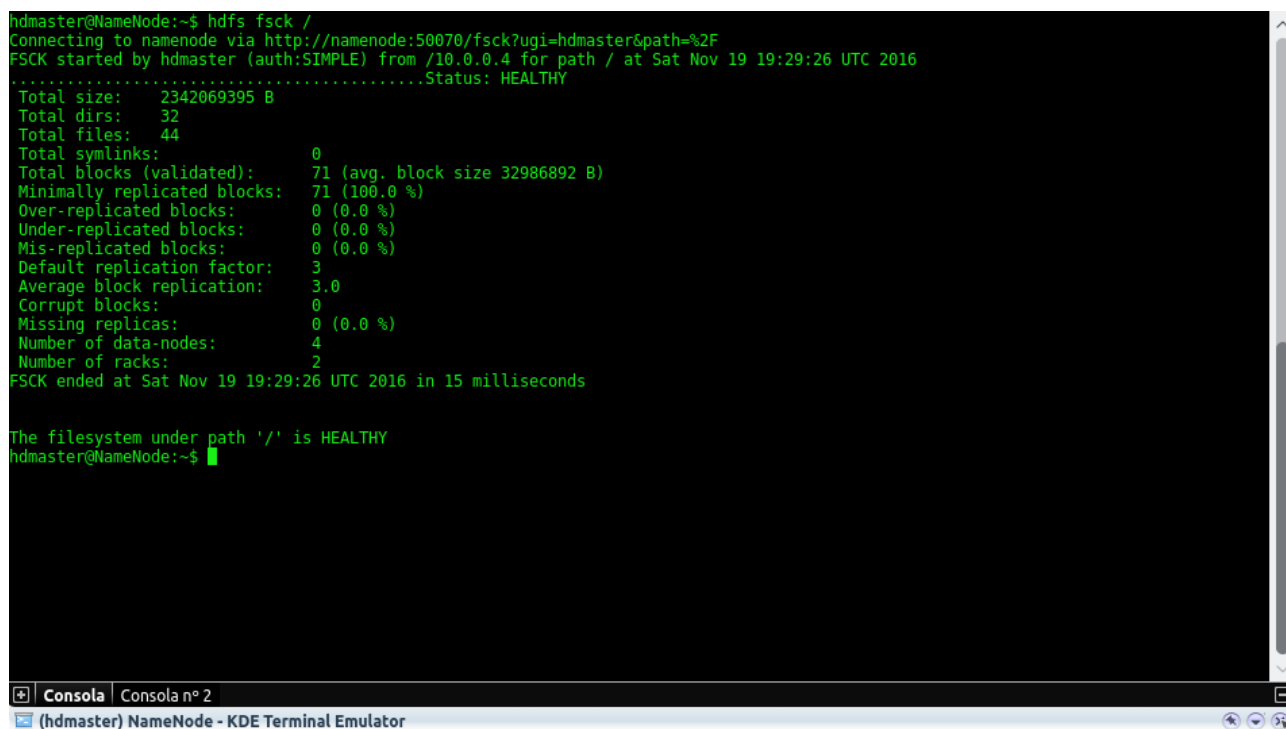

8. EJECUCION Y SALIDA DEL CHECK HDFS EN TODO EL SISTEMA HDFS DEL CLÚSTER

En este apartado veremos la ejecución del check para el sistema de ficheros hdfs y el resultado que nos muestra.

29. Para hacer el check a todo el sistema de ficheros hdfs del clúster tenemos que ejecutar el siguiente comando:

```
$hdfs fsck /
```

La salida nos muestra que el estado del sistema de ficheros es saludable, el tamaño total en Bytes, los directorios totales, los ficheros totales, enlaces simbólicos, total de bloques validados, el factor de replicación por defecto (en nuestro caso 3), bloques corruptos, replicas perdidas, números de datanodes, y número de racks. Por último nos indica el tiempo invertido en hacer el chequeo, en este caso sólo 15 milisegundos.



```
hdmaster@NameNode:~$ hdfs fsck /
Connecting to namenode via http://namenode:50070/fsck?ugi=hdmaster&path=%2F
FSCK started by hdmaster (auth:SIMPLE) from /10.0.0.4 for path / at Sat Nov 19 19:29:26 UTC 2016
.....Status: HEALTHY
Total size:      2342069395 B
Total dirs:      32
Total files:     44
Total symlinks:   0
Total blocks (validated): 71 (avg. block size 32986892 B)
Minimally replicated blocks: 71 (100.0 %)
Over-replicated blocks: 0 (0.0 %)
Under-replicated blocks: 0 (0.0 %)
Mis-replicated blocks: 0 (0.0 %)
Default replication factor: 3
Average block replication: 3.0
Corrupt blocks: 0
Missing replicas: 0 (0.0 %)
Number of data-nodes: 4
Number of racks: 2
FSCK ended at Sat Nov 19 19:29:26 UTC 2016 in 15 milliseconds

The filesystem under path '/' is HEALTHY
hdmaster@NameNode:~$
```

Aquí terminaría el punto 3 de la práctica y, por tanto, la parte obligatoria de la misma. A partir del siguiente apartado se ve la parte opcional de la práctica.

9. CREACIÓN DE UN DIRECTORIO MEDIANTE WEBHDFS

En este apartado veremos como se puede crear un directorio para el usuario *moises* (usuario *no hdmaster*) en el sistema de ficheros HDFS del Namenode del clúster, ejecutando una llamada webhdfs desde la máquina local.

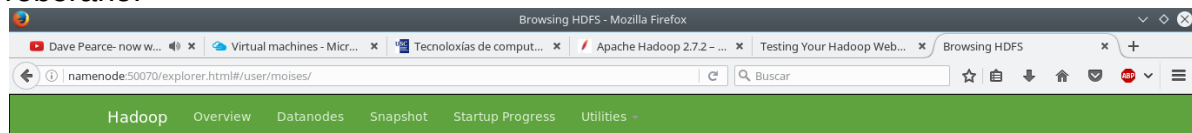
30. Creamos un directorio en el sistema de ficheros HDFS del Namenode usando *webhdfs* desde la máquina local para el usuario *moises* con:

```
$ curl -i -X PUT
"http://NameNode:50070/webhdfs/v1/user/moises/directorioWebhdfs?
op=MKDIRS&user.name=moises"
```

La salida que nos devuelve es:

```
HTTP/1.1 200 OK
Cache-Control: no-cache
Expires: Sat, 19 Nov 2016 23:04:21 GMT
Date: Sat, 19 Nov 2016 23:04:21 GMT
Pragma: no-cache
Expires: Sat, 19 Nov 2016 23:04:21 GMT
Date: Sat, 19 Nov 2016 23:04:21 GMT
Pragma: no-cache
Content-Type: application/json
Set-Cookie:
hadoop.auth="u=moises&p=moises&t=simple&e=1479632661757&s=QQWNf4QQnX1ZvRa0W
CawBTgkBZA="; Path=/; Expires=dom, 20-nov-2016 09:04:21 GMT; HttpOnly
Transfer-Encoding: chunked
Server: Jetty(6.1.26)
{"boolean":true}
```

Esto nos indica que la llamada se efectuó con éxito y creo el directorio en el lugar donde le hemos indicado. Si echamos un vistazo al explorador de la interfaz web podemos corroborarlo.



Browse Directory

Permission	Owner	Group	Size	Last Modified	Replication	Block Size	Name
drwxr-xr-x	moises	supergroup	0 B	20/11/2016 0:04:21	0	0 B	directorioWebhdfs
drwxr-xr-x	moises	supergroup	0 B	1/11/2016 21:40:14	0	0 B	libros
drwxr-xr-x	moises	supergroup	0 B	12/11/2016 18:51:10	0	0 B	salidawc

Hadoop, 2016.

10. ENVÍO DE FICHERO LOCAL AL SISTEMA HDFS DEL CLÚSTER MEDIANTE WEBHDFS

En este apartado veremos como se puede enviar un fichero al sistema de ficheros HDFS del Namenode del clúster, ejecutando una llamada webhdfs desde la máquina local.

31. Primero hacemos una llamada para indicar al Namenode que queremos crear un fichero en el sistema hdfs y que nos diga donde lo tenemos que copiar:

```
$curl -i -X PUT
"http://NameNode:50070/webhdfs/v1/user/moises/directorioWebhdfs/?
op=CREATE&user.name=moises"
```

La salida que nos devuelve es:

```
HTTP/1.1 307 TEMPORARY_REDIRECT
Cache-Control: no-cache
Expires: Sat, 19 Nov 2016 23:11:28 GMT
Date: Sat, 19 Nov 2016 23:11:28 GMT
Pragma: no-cache
Expires: Sat, 19 Nov 2016 23:11:28 GMT
Date: Sat, 19 Nov 2016 23:11:28 GMT
Pragma: no-cache
Content-Type: application/octet-stream
Set-Cookie:
hadoop.auth="u=moises&p=moises&t=simple&e=1479633088927&s=oGs47gqOChhhNGEgx
CnOkvhEnZ8="; Path=/; Expires=dom, 20-nov-2016 09:11:28 GMT; HttpOnly
Location: http://datanode4:50075/webhdfs/v1/user/moises/directorioWebhdfs/?
op=CREATE&user.name=moises&namenoderpcaddress=namenode:9000&overwrite=false
Content-Length: 0
Server: Jetty(6.1.26)
```

Esto nos indica de que ha creado una redirección temporal a una localización que está dentro del datanode4 y que usa el puerto 50075. La localización completa la tendremos que usar en la próxima llamada para enviar el fichero y escribirlo allí.

32. A continuación le enviamos el fichero local *prueba.txt* a la localización enviada por el Namenode como respuesta de la llamada anterior con:

```
$curl -i -X PUT -T prueba.txt
"http://datanode4:50075/webhdfs/v1/user/moises/directorioWebhdfs/prueba.txt?
op=CREATE&user.name=moises&namenoderpcaddress=namenode:9000&overwrite=false"
```

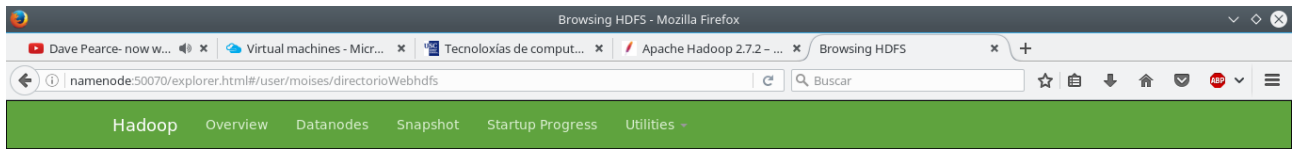
La salida que nos devuelve ahora es:

```
HTTP/1.1 100 Continue

HTTP/1.1 201 Created
Location: hdfs://namenode:9000/user/moises/directorioWebhdfs/prueba.txt
Content-Length: 0
```

Connection: close

Esto significa que, en primer lugar continua por donde se quedó, y en segundo lugar nos devuelve una notificación como que el fichero se ha creado, aunque ahora no nos devuelve la localización interna del fichero (datanode4), sino la externa (la del Namenode). Una vez más nos vamos al explorador del Namenode para corroborar que esto es así.



Browse Directory

/user/moises/directorioWebhdfs Go!

Permission	Owner	Group	Size	Last Modified	Replication	Block Size	Name
-rwxr-xr-x	moises	supergroup	294 B	20/11/2016 0:17:28	3	64 MB	prueba.txt

Hadoop, 2016.

File information - prueba.txt ×

Download

Block information -- Block 0

Block ID: 1073742058

Block Pool ID: BP-1332237322-10.0.0.4-1478024217080

Generation Stamp: 1234

Size: 294

Availability:

- datanode2
- datanode1
- datanode4

Close

11. COMPROBACIÓN DE QUE EL FICHERO SE ENVIÓ CORRECTAMENTE ABRIÉNDOLO MEDIANTE WEBHDFS

Por último, en este apartado comprobaremos desde el propio *webhdfs* que podemos abrir un fichero dentro del sistema de ficheros HDFS del Namenode desde nuestra propia máquina local. Y en nuestro caso usaremos el fichero *prueba.txt* recién enviado para intentar ver su contenido.

33. Para abrir el fichero que acabamos de copiar simplemente tenemos que poner lo siguiente:

```
$curl -i -L
"http://NameNode:50070/webhdfs/v1/user/moises/directorioWebhdfs/pr
ueba.txt?op=OPEN&user.name=moises"
```

La salida que nos devuelve es:

```
HTTP/1.1 307 TEMPORARY_REDIRECT
Cache-Control: no-cache
Expires: Sat, 19 Nov 2016 23:25:12 GMT
Date: Sat, 19 Nov 2016 23:25:12 GMT
Pragma: no-cache
Expires: Sat, 19 Nov 2016 23:25:12 GMT
Date: Sat, 19 Nov 2016 23:25:12 GMT
Pragma: no-cache
Set-Cookie:
hadoop.auth="u=moises&p=moises&t=simple&e=1479633912864&s=mqlhR9P8tJzBlL8re
4DNMLf4ZGs="; Path=/; Expires=dom, 20-nov-2016 09:25:12 GMT; HttpOnly
Location:
http://datanode1:50075/webhdfs/v1/user/moises/directorioWebhdfs/prueba.txt?
op=OPEN&user.name=moises&namenoderpcaddress=namenode:9000&offset=0
Content-Type: application/octet-stream
Content-Length: 0
Server: Jetty(6.1.26)
```

```
HTTP/1.1 200 OK
Access-Control-Allow-Methods: GET
Access-Control-Allow-Origin: *
Content-Type: application/octet-stream
Connection: close
Content-Length: 294
```

Éste es un fichero de prueba para comprobar que efectivamente se envía este fichero desde mi equipo al sistema HDFS de mi clúster HADOOP utilizando webhdfs.

Después de enviarlo probaremos a abrirlo. Si se abre tiene que mostrar el contenido del mismo por la salida estándar.

Moisés F.

Esto nos indica dos cosas, primero que hace una redirección interna hacia donde está una de las réplicas del fichero *prueba.txt*, en este caso la réplica que está en el datanode1; y, segundo, que una vez la encuentra, la obtiene, la abre y muestra su contenido por la salida estándar como se puede apreciar.

Y con este apartado concluye la parte opcional de la práctica, y por ende toda la práctica también.