

## LAB6

### Hadoop, HDFS y MapReduce

En este laboratorio vamos a evaluar el uso de Hadoop en un entorno sencillo como la máquina virtual de Cloudera.

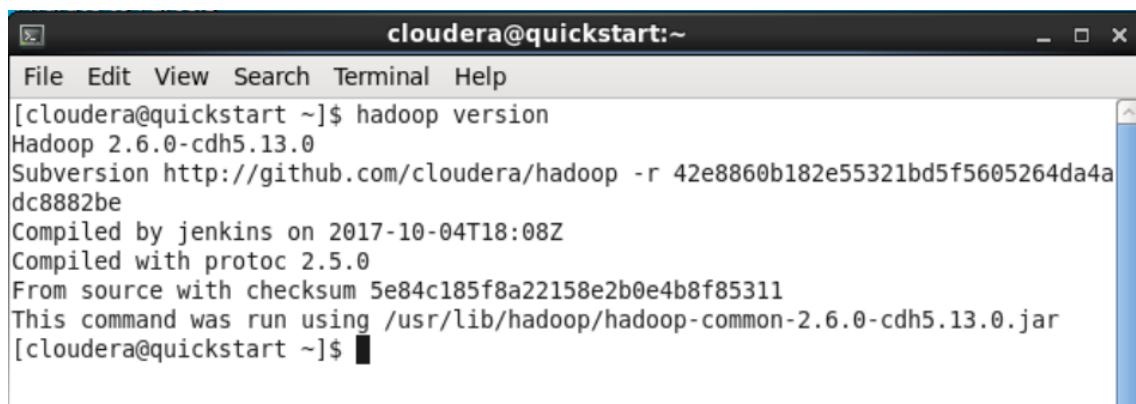
#### PARTE 1

Revisaremos las herramientas básicas y el uso de este sistema.



1. Lo primero que hemos hecho es ir a la Terminal. Aquí comprobar cuál es la versión que se está utilizando de Hadoop, en este caso es la 2.6.0:

```
$ hadoop version
```

A screenshot of a terminal window titled "cloudera@quickstart:~". The window shows the output of the "hadoop version" command. The output is as follows:

```
File Edit View Search Terminal Help
[cloudera@quickstart ~]$ hadoop version
Hadoop 2.6.0-cdh5.13.0
Subversion http://github.com/cloudera/hadoop -r 42e8860b182e55321bd5f5605264da4a
dc8882be
Compiled by jenkins on 2017-10-04T18:08Z
Compiled with protoc 2.5.0
From source with checksum 5e84c185f8a22158e2b0e4b8f85311
This command was run using /usr/lib/hadoop/hadoop-common-2.6.0-cdh5.13.0.jar
[cloudera@quickstart ~]$
```

2. Y ahora vamos a comprobar si están los servicios activados y cuál es su calidad de servicio. Utilizamos **dfsadmin** le pedimos que haga un **report** y veremos como resultado una serie de datos y dentro de esos datos nos interesa ver cuántos nodos activos tenemos.

```
$ hdfs dfsadmin -report
```

En este caso vemos que hay 1 (Live datanodes).

```
cloudera@quickstart:~$ hdfs dfsadmin -report
[cloudera@quickstart ~]$ hdfs dfsadmin -report
Configured Capacity: 58531520512 (54.51 GB)
Present Capacity: 46699715358 (43.49 GB)
DFS Remaining: 45827073268 (42.68 GB)
DFS Used: 872642090 (832.22 MB)
DFS Used%: 1.87%
Under replicated blocks: 0
Blocks with corrupt replicas: 0
Missing blocks: 0
Missing blocks (with replication factor 1): 0

-----
Live datanodes (1):

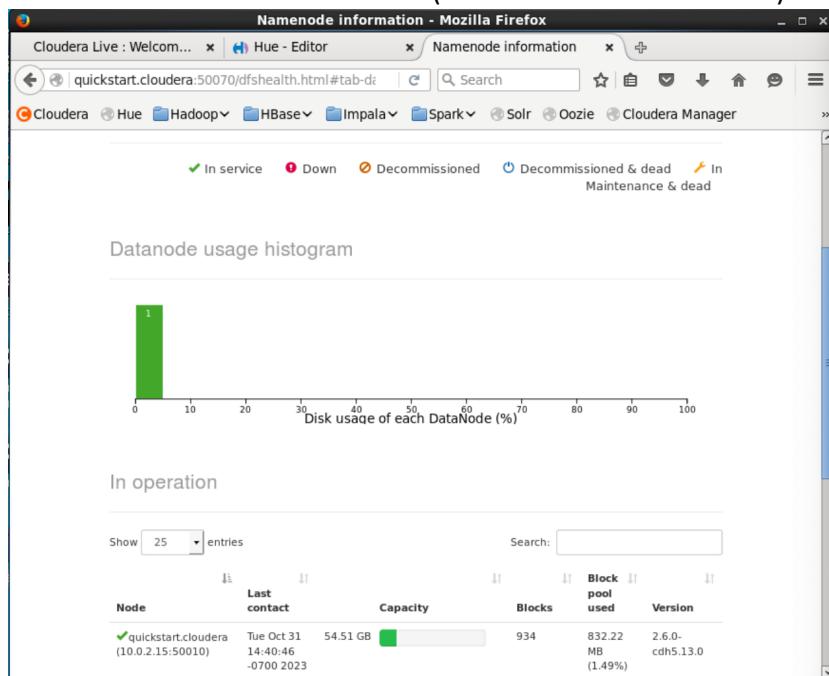
Name: 10.0.2.15:50010 (quickstart.cloudera)
Hostname: quickstart.cloudera
Decommission Status : Normal
Configured Capacity: 58531520512 (54.51 GB)
DFS Used: 872642090 (832.22 MB)
Non DFS Used: 8583317974 (7.99 GB)
DFS Remaining: 45827073268 (42.68 GB)
DFS Used%: 1.49%
DFS Remaining%: 78.29%
Configured Cache Capacity: 0 (0 B)
Cache Used: 0 (0 B)
Cache Remaining: 0 (0 B)
Cache Used%: 100.00%
Cache Remaining%: 0.00%
Xceivers: 6
Last contact: Tue Oct 31 14:37:33 PDT 2023
```

3. Existe otra manera de comprobar si están los servicios activados y es abrir el navegador Firefox y acercarnos a la página web de datos de calidad del servidor en Hadoop – HDFS NameNode:

Vemos que existe un nodo activo (Live Nodes):

Configured Capacity:	54.51 GB
DFS Used:	829.15 MB (1.49%)
Non DFS Used:	9.94 GB
DFS Remaining:	40.73 GB (74.72%)
Block Pool Used:	829.15 MB (1.49%)
DataNodes usages% (Min/Median/Max/stdDev):	1.49% / 1.49% / 1.49% / 0.00%
Live Nodes	1 (Decommissioned: 0, In Maintenance: 0)
Dead Nodes	0 (Decommissioned: 0, In Maintenance: 0)
Decommissioning Nodes	0
Entering Maintenance Nodes	0

Y tenemos más información sobre ese nodo (dando click en Live Nodes)::



Una vez que hemos comprobado que el nodo está activo se puede asegurar que tenemos los servicios listos para utilizar.

Ahora **vamos a utilizar HDFS** y crear una carpeta, copiar ficheros, ver cómo funciona y cuál es su uso básico.

4. Primero miramos qué carpetas están contenidas en el directorio /user  
 \$ hdfs dfs -ls /user

```
cloudera@quickstart:~$ hdfs dfs -ls /user
File Edit View Search Terminal Help
Non DFS Used: 10677392654 (9.94 GB)
DFS Remaining: 43736213347 (40.73 GB)
DFS Used%: 1.49%
DFS Remaining%: 74.72%
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: 6
Last contact: Tue Jun 23 14:27:45 PDT 2020

[cloudera@quickstart ~]$ hdfs dfs -ls /user
Found 8 items
drwxr-xr-x  - cloudera cloudera      0 2017-07-19 05:33 /user/cloudera
drwxr-xr-x  - mapred   hadoop        0 2017-07-19 05:34 /user/history
drwxrwxrwx  - hive     supergroup    0 2017-07-19 05:36 /user/hive
drwxrwxrwx  - hue      supergroup    0 2017-07-19 05:35 /user/hue
drwxrwxrwx  - jenkins  supergroup    0 2017-07-19 05:35 /user/jenkins
drwxrwxrwx  - oozie    supergroup    0 2017-07-19 05:35 /user/oozie
drwxrwxrwx  - root     supergroup    0 2017-07-19 05:35 /user/root
drwxr-xr-x  - hdfs    supergroup    0 2017-07-19 05:36 /user/spark
[cloudera@quickstart ~]$
```

5. Creamos una carpeta hadoop en el directorio /user, donde vamos a guardar nuestros datos. Utilizamos dfs -mkdir (make dir) para crear la carpeta.

\$ hdfs dfs -mkdir -p /user/hadoop

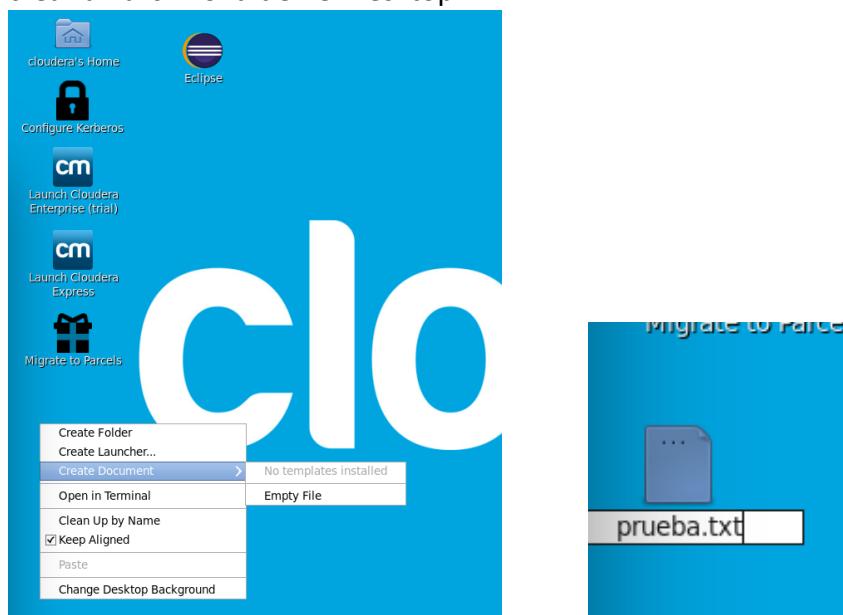
```
[cloudera@quickstart ~]$ hdfs dfs -mkdir -p /user/hadoop
[cloudera@quickstart ~]$
```

Ahora lo que vamos a hacer es comprobar que esa carpeta existe utilizando nuevamente el comando ls.

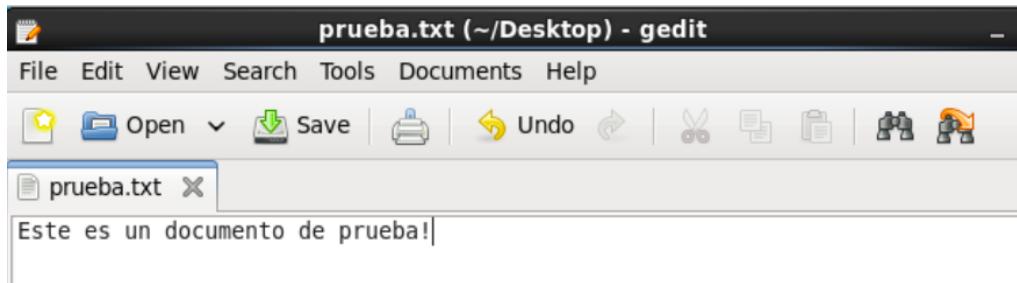
```
$ hdfs dfs -ls /user
```

```
[cloudera@quickstart ~]$ hdfs dfs -ls /user
Found 8 items
drwxr-xr-x  - cloudera  cloudera          0 2017-10-23 09:14 /user/cloudera
drwxr-xr-x  - mapred    hadoop           0 2017-10-23 09:15 /user/history
drwxrwxrwx  - hive      supergroup        0 2017-10-23 09:17 /user/hive
drwxrwxrwx  - hue       supergroup        0 2017-10-23 09:16 /user/hue
drwxrwxrwx  - jenkins   supergroup        0 2017-10-23 09:15 /user/jenkins
drwxrwxrwx  - oozie     supergroup        0 2017-10-23 09:16 /user/oozie
drwxrwxrwx  - root      supergroup        0 2017-10-23 09:16 /user/root
drwxr-xr-x  - hdfs     supergroup        0 2017-10-23 09:17 /user/spark
[cloudera@quickstart ~]$ hdfs dfs -mkdir -p /user/hadoop
[cloudera@quickstart ~]$ hdfs dfs -ls /user
Found 9 items
drwxr-xr-x  - cloudera  cloudera          0 2017-10-23 09:14 /user/cloudera
drwxr-xr-x  - cloudera  supergroup         0 2023-10-31 14:44 /user/hadoop
drwxr-xr-x  - mapred    hadoop           0 2017-10-23 09:15 /user/history
drwxrwxrwx  - hive      supergroup        0 2017-10-23 09:17 /user/hive
drwxrwxrwx  - hue       supergroup        0 2017-10-23 09:16 /user/hue
drwxrwxrwx  - jenkins   supergroup        0 2017-10-23 09:15 /user/jenkins
drwxrwxrwx  - oozie     supergroup        0 2017-10-23 09:16 /user/oozie
drwxr-xr-x  - root      supergroup        0 2017-10-23 09:16 /user/root
drwxr-xr-x  - hdfs     supergroup        0 2017-10-23 09:17 /user/spark
[cloudera@quickstart ~]$
```

6. A partir de aquí lo que vamos a hacer es copiar archivos de prueba y comprobar cuál es el típico uso de los comandos de acceso a los archivos. Lo primero que hacemos es crear un archivo .txt en el Desktop:



7. Dar doble click en archivo creado para poder editar, luego guardar:



- Con ls vemos que el archivo está en el Desktop

```
[cloudera@quickstart ~]$ ls Desktop
compartidaMV Enterprise.desktop Kerberos.desktop prueba.txt
Eclipse.desktop Express.desktop Parcels.desktop prueba.txt~
[cloudera@quickstart ~]$
```

8. Copiar el fichero guardado en el Desktop (**que de hecho se encuentra en el file system de Centos, no de hadoop OJO**) en la carpeta que hemos creado en HDFS /user/hadoop.
- \$ hdfs dfs -put Desktop/prueba.txt /user/hadoop

**NOTA:** El comando put lleva un archivo **local** a un archivo **remoto** en nuestro servidor de datos HDFS.

9. Una vez que está copiado y no tenemos ningún mensaje de error comprobamos que ese archivo existe.

\$ hdfs dfs -ls /user/hadoop

```
[cloudera@quickstart ~]$ hdfs dfs -ls /user/hadoop
Found 1 items
-rw-r--r-- 1 cloudera supergroup          32 2023-10-31 14:50 /user/hadoop/prueba.txt
[cloudera@quickstart ~]$
```

**NOTA:** también puedes navegar desde el browser en Firefox y verificar tus archivos en el hadoop file system: Hadoop – HDFS NameNode - Utilities – Browse the file system - user



## Browse Directory

/user/hadoop

Permission	Owner	Group	Size	Last Modified	Replication	Block Size	Name
-rw-r--r--	cloudera	supergroup	32 B	Tue Oct 31 14:50:33 -0700 2023	1	128 MB	prueba.txt

10. Otra cosa que podemos hacer para comprobar si el contenido está bien, es visualizando su contenido con el comando cat.

```
$ hdfs dfs -cat /user/hadoop/prueba.txt
```

NOTA: HDFS nos va a devolver el contenido de ese archivo.

```
[cloudera@quickstart ~]$ hdfs dfs -cat /user/hadoop/prueba.txt
Este es un documento de prueba!
[cloudera@quickstart ~]$
```

11. Cuando no necesitamos más el archivo, lo que hacemos es eliminarlo. Usamos el comando -rm. Tenemos que especificar el fichero que queremos eliminar y ese archivo dejará de existir. Después del comando nos avisa que lo ha eliminado.

```
$ hdfs dfs -rm /user/hadoop/prueba.txt
```

```
[cloudera@quickstart ~]$ hdfs dfs -rm /user/hadoop/prueba.txt
Deleted /user/hadoop/prueba.txt
[cloudera@quickstart ~]$
```

12. Para comprobación final vemos que nuestra carpeta está efectivamente vacía y que no hay ningún archivo de datos en esa carpeta.

```
$ hdfs dfs -ls /user/Hadoop
```

```
[cloudera@quickstart ~]$ hdfs dfs -ls /user/hadoop
[cloudera@quickstart ~]$
```

## PARTE 2

Vamos a lanzar un típico trabajo de Hadoop (job) para ver cómo funciona.

En este caso vamos a utilizar uno de los ejemplos de librería de Hadoop que es **terasort**. (Para conocer más puede investigar sobre el uso de terasort <https://hadoop.apache.org/docs/r3.2.0/api/org/apache/hadoop/examples/terasort/package-summary.html>).

1. Llamamos al ejemplo de entre los ejemplos de Hadoop que vienen con la instalación de esta manera:

```
$ hadoop jar /usr/lib/hadoop-mapreduce/hadoop-mapreduce-examples.jar teragen
1000 /user/hadoop/terainput
```

Llamamos al programa *teragen* (y se ejecuta un trabajo mapreduce) y establecemos el parámetro 1000 y el resultado tiene que estar en una **carpeta nueva** que se va a llamar *terainput* (al ejecutar el comando anterior se crea automáticamente la carpeta *terainput* en el directorio */user/hadoop/*). Entonces se generan 1000 registros en esa carpeta que corresponden a 1000 números aleatorios.

```
cloudera@quickstart:~$ hadoop jar /usr/lib/hadoop-mapreduce/hadoop-mapreduce-examples.jar teragen 1000 /user/hadoop/terainput
23/10/31 15:02:05 INFO client.RMProxy: Connecting to ResourceManager at /0.0.0.0:8032
23/10/31 15:02:05 INFO terasort.TeraGen: Generating 1000 using 2
23/10/31 15:02:05 INFO mapreduce.JobSubmitter: number of splits:2
23/10/31 15:02:06 INFO mapreduce.JobSubmitter: Submitting tokens for job: job_1698787909287_0001
23/10/31 15:02:06 INFO impl.YarnClientImpl: Submitted application application_1698787909287_0001
23/10/31 15:02:06 INFO mapreduce.Job: The url to track the job: http://quickstart.cloudera:8088/proxy/application_1698787909287_0001/
23/10/31 15:02:06 INFO mapreduce.Job: Running job: job_1698787909287_0001
23/10/31 15:02:13 INFO mapreduce.Job: Job job_1698787909287_0001 running in uber mode : false
23/10/31 15:02:13 INFO mapreduce.Job: map 0% reduce 0%
23/10/31 15:02:19 INFO mapreduce.Job: map 50% reduce 0%
23/10/31 15:02:22 INFO mapreduce.Job: map 100% reduce 0%
23/10/31 15:02:22 INFO mapreduce.Job: Job job_1698787909287_0001 completed successfully
23/10/31 15:02:22 INFO mapreduce.Job: Counters:
  File System Counters
    FILE: Number of bytes read=0
    FILE: Number of bytes written=286474
    FILE: Number of read operations=0
    FILE: Number of large read operations=0
    FILE: Number of write operations=0
    HDFS: Number of bytes read=164
    HDFS: Number of bytes written=100000
    HDFS: Number of read operations=8
    HDFS: Number of large read operations=0
    HDFS: Number of write operations=4
  Job Counters
    Launched map tasks=2
```

```
cloudera@quickstart:~$ hadoop jar /usr/lib/hadoop-mapreduce/hadoop-mapreduce-examples.jar teragen 1000 /user/hadoop/terainput
File Edit View Search Terminal Help
File System Counters
  FILE: Number of bytes read=0
  FILE: Number of bytes written=286474
  FILE: Number of read operations=0
  FILE: Number of large read operations=0
  FILE: Number of write operations=0
  HDFS: Number of bytes read=164
  HDFS: Number of bytes written=100000
  HDFS: Number of read operations=8
  HDFS: Number of large read operations=0
  HDFS: Number of write operations=4
Job Counters
  Launched map tasks=2
  Other local map tasks=2
  Total time spent by all maps in occupied slots (ms)=8891
  Total time spent by all reduces in occupied slots (ms)=0
  Total time spent by all map tasks (ms)=8891
  Total vcore-milliseconds taken by all map tasks=8891
  Total megabyte-milliseconds taken by all map tasks=9104384
Map-Reduce Framework
  Map input records=1000
  Map output records=1000
  Input split bytes=164
  Spilled Records=0
  Failed Shuffles=0
  Merged Map outputs=0
  GC time elapsed (ms)=115
  CPU time spent (ms)=1520
  Physical memory (bytes) snapshot=404377600
  Virtual memory (bytes) snapshot=3125948416
  Total committed heap usage (bytes)=442499072
org.apache.hadoop.examples.terasort.TeraGen$Counters
  CHECKSUM=2173251765740
  File Input Format Counters
    Bytes Read=0
  File Output Format Counters
    Bytes Written=100000
```

Este es un ejemplo pequeño que ha generado 1000 registros que están en la carpeta terainput con un total de 100000 bytes escritos.

Veamos que /user/hadoop ahora contiene la carpeta terainput:

```
[cloudera@quickstart ~]$ hdfs dfs -ls /user/hadoop
Found 1 items
drwxr-xr-x  - cloudera supergroup          0 2023-10-31 15:02 /user/hadoop/terainput
[cloudera@quickstart ~]$
```

El contenido en terainput suma esos 100000 bytes:

`$ hdfs dfs -ls /user/hadoop/terainput`

```
[cloudera@quickstart ~]$ hdfs dfs -ls /user/hadoop/terainput
Found 3 items
-rw-r--r--  1 cloudera supergroup          0 2023-10-31 15:02 /user/hadoop/terainput/_SUCCESS
-rw-r--r--  1 cloudera supergroup      50000 2023-10-31 15:02 /user/hadoop/terainput/part-r00000
-rw-r--r--  1 cloudera supergroup      50000 2023-10-31 15:02 /user/hadoop/terainput/part-r00001
[cloudera@quickstart ~]$
```

2. Ahora lo que hacemos es utilizar la segunda parte del ejemplo de *terasort* que es ordenar los 1000 registros (1000 números aleatorios que se generaron).

Para ordenar usamos la misma librería de ejemplos que está en `usr/lib/hadoop-mapreduce` y llamamos al programa *terasort*.

```
$ hadoop jar /usr/lib/hadoop-mapreduce/hadoop-mapreduce-examples.jar terasort
/user/hadoop/terainput /user/hadoop/teraooutput
```

NOTA: el nombre del programa viene después de la clase donde está el código (.jar). La carpeta terainput tiene los datos de entrada y teraooutput será donde queremos que se genere la salida que es la ordenación de todos estos números de entrada. Esta carpeta de salida no debe de existir, se crea automáticamente como parte de la ejecución del programa:

```
cloudera@quickstart:~$ hadoop jar /usr/lib/hadoop-mapreduce/hadoop-mapreduce-examples.jar terasort /user/hadoop/terainput /user/hadoop/teraooutput
23/10/31 15:08:36 INFO terasort.TeraSort: starting
23/10/31 15:08:37 INFO input.FileInputFormat: Total input paths to process : 2
Spent 125ms computing base-splits.
Spent 2ms computing TeraScheduler splits.
Computing input splits took 127ms
Sampling 2 splits of 2
Making 1 from 1000 sampled records
Computing partitions took 174ms
Spent 303ms computing partitions.
23/10/31 15:08:37 INFO client.RMProxy: Connecting to ResourceManager at /0.0.0.0:8032
23/10/31 15:08:37 WARN hdfs.DFSClient: Caught exception
java.lang.InterruptedException
    at java.lang.Object.wait(Native Method)
    at java.lang.Thread.join(Thread.java:1281)
    at java.lang.Thread.join(Thread.java:1355)
    at org.apache.hadoop.hdfs.DFSOutputStream$DataStreamer.closeResponder(DFSOutputStream.java:967)
    at org.apache.hadoop.hdfs.DFSOutputStream$DataStreamer.endBlock(DFSOutputStream.java:705)
    at org.apache.hadoop.hdfs.DFSOutputStream$DataStreamer.run(DFSOutputStream.java:894)
23/10/31 15:08:38 INFO mapreduce.JobSubmitter: number of splits:2
23/10/31 15:08:38 INFO mapreduce.JobSubmitter: Submitting tokens for job: job_1698787909287_0002
23/10/31 15:08:38 INFO impl.YarnClientImpl: Submitted application application_1698787909287_0002
23/10/31 15:08:38 INFO mapreduce.Job: The url to track the job: http://quickstart.cloudera:8088/proxy/application_1698787909287_0002/
23/10/31 15:08:38 INFO mapreduce.Job: Running job: job_1698787909287_0002
23/10/31 15:08:44 INFO mapreduce.Job: Job job_1698787909287_0002 running in uber mode : false
23/10/31 15:08:44 INFO mapreduce.Job: map 0% reduce 0%
23/10/31 15:08:50 INFO mapreduce.Job: map 50% reduce 0%
23/10/31 15:08:51 INFO mapreduce.Job: map 100% reduce 0%
23/10/31 15:08:55 INFO mapreduce.Job: map 100% reduce 100%
```

Finalmente se mira que el resultado se ha escrito en dos ficheros.

```
cloudera@quickstart:~
```

File Edit View Search Terminal Help

File System Counters

- FILE: Number of bytes read=104006
- FILE: Number of bytes written=642483
- FILE: Number of read operations=0
- FILE: Number of large read operations=0
- FILE: Number of write operations=0
- HDFS: Number of bytes read=100262
- HDFS: Number of bytes written=100000
- HDFS: Number of read operations=9
- HDFS: Number of large read operations=0
- HDFS: Number of write operations=2

Job Counters

- Launched map tasks=2
- Launched reduce tasks=1
- Data-local map tasks=2
- Total time spent by all maps in occupied slots (ms)=7437
- Total time spent by all reduces in occupied slots (ms)=2954
- Total time spent by all map tasks (ms)=7437
- Total time spent by all reduce tasks (ms)=2954
- Total vcore-milliseconds taken by all map tasks=7437
- Total vcore-milliseconds taken by all reduce tasks=2954
- Total megabyte-milliseconds taken by all map tasks=7615488
- Total megabyte-milliseconds taken by all reduce tasks=3024896

Map-Reduce Framework

- Map input records=1000
- Map output records=1000
- Map output bytes=102000
- Map output materialized bytes=104012
- Input split bytes=262
- Combine input records=0
- Combine output records=0
- Reduce input groups=1000
- Reduce shuffle bytes=104012
- Reduce input records=1000
- Reduce output records=1000
- Spilled Records=2000
- Shuffled Maps =2
- Failed Shuffles=0
- Merged Map outputs=2
- GC time elapsed (ms)=105
- CPU time spent (ms)=1630
- Physical memory (bytes) snapshot=804818944
- Virtual memory (bytes) snapshot=4705198080
- Total committed heap usage (bytes)=874512384

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=100000

File Output Format Counters

- Bytes Written=100000

23/10/31 15:08:55 INFO terasort.TeraSort: done

[cloudera@quickstart ~]\$

Hay que recordar que se han ordenado los 100000 bytes que estaban como input, lo cual indica que el resultado en la carpeta de salida también debe contener 100000 bytes.

\$ hdfs dfs -ls /user/hadoop  
\$ hdfs dfs -ls /user/hadoop/teraooutput

```
[cloudera@quickstart ~]$ hdfs dfs -ls /user/hadoop
Found 2 items
drwxr-xr-x - cloudera supergroup 0 2023-10-31 15:02 /user/hadoop/terainput
drwxr-xr-x - cloudera supergroup 0 2023-10-31 15:08 /user/hadoop/teraooutput
[cloudera@quickstart ~]$ hdfs dfs -ls /user/hadoop/teraooutput
Found 3 items
-rw-r--r-- 1 cloudera supergroup 0 2023-10-31 15:08 /user/hadoop/teraooutput/_SU
CCES
-rw-r--r-- 10 cloudera supergroup 0 2023-10-31 15:08 /user/hadoop/teraooutput/_pa
rtition.lst
-rw-r--r-- 1 cloudera supergroup 100000 2023-10-31 15:08 /user/hadoop/teraooutput/part
-r-00000
[cloudera@quickstart ~]$
```

3. Finalmente, cuando hemos realizado un procesamiento con hadoop lo que debemos hacer es eliminar los datos de salida cuando ya no los vamos a usar más.

Utilizamos el comando que hemos visto antes que es rm. En este caso utilizamos rm -r para eliminar la carpeta entera.

Entonces vamos a borrar todas las carpetas que hemos creado, tanto la de entrada como la de salida.

```
$ hdfs dfs -rm -r -skipTrash /user/hadoop/tera*
```

```
[cloudera@quickstart ~]$ hdfs dfs -rm -r -skipTrash /user/hadoop/tera*
Deleted /user/hadoop/terainput
Deleted /user/hadoop/teraooutput
[cloudera@quickstart ~]$
```

### **PARTE 3: TRABAJO GRUPAL**

En este ejercicio vas a realizar una pequeña comprobación del buen funcionamiento de varios servicios relacionados con Hadoop:

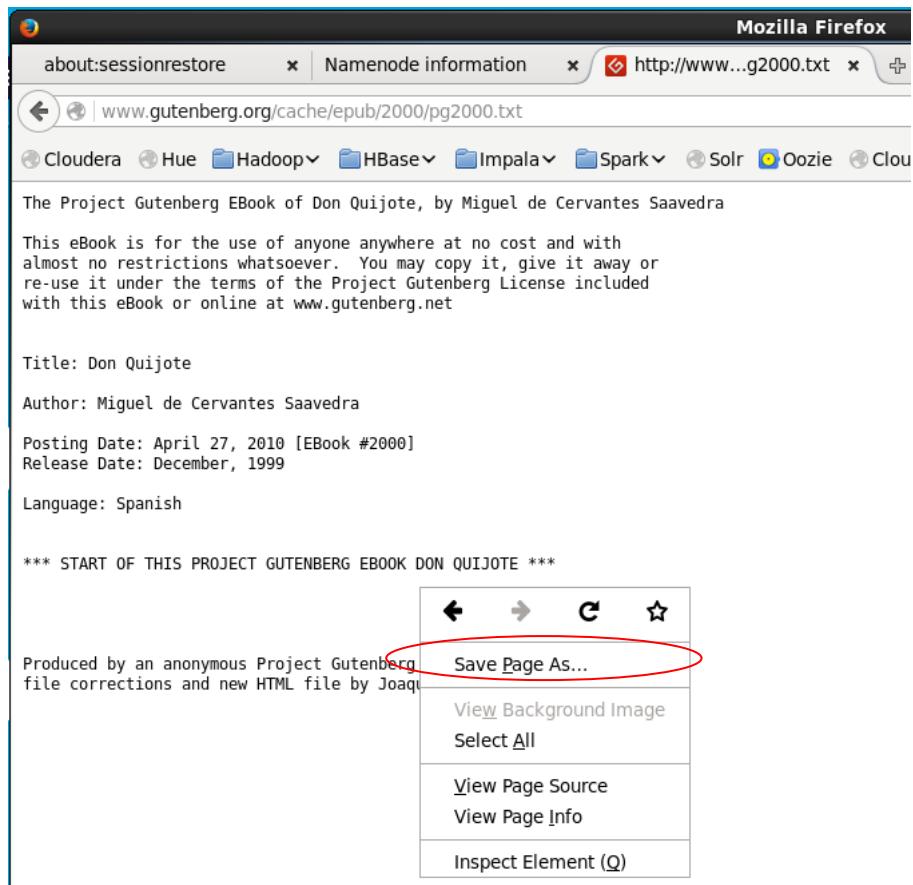
- Ciclo de creación, lectura y eliminación de un archivo en HDFS.
- Lanzar un trabajo sencillo de prueba (word count job)
- Comprobar los resultados obtenidos en HDFS

Primero, debes descargar un archivo de datos de prueba "texto.txt" para resolver una serie de pasos de la práctica. Para ello:

1. Entra en la máquina virtual MV\_Cloudera,
2. Abre el navegador web, y,
3. Descarga los datos desde esta dirección:  
<http://www.gutenberg.org/cache/epub/2000/pg2000.txt>
4. Cambia el nombre a texto.txt del archivo descargado

A continuación, realiza los siguientes pasos:

1. Descarga el fichero de datos "texto.txt" a una carpeta de usuario en la máquina virtual (puede ser en el Desktop)  
Save Page As -> texto.txt



The Project Gutenberg EBook of Don Quijote, by Miguel de Cervantes Saavedra

This eBook is for the use of anyone anywhere at no cost and with almost no restrictions whatsoever. You may copy it, give it away or re-use it under the terms of the Project Gutenberg License included with this eBook or online at [www.gutenberg.net](http://www.gutenberg.net)

Title: Don Quijote  
 Author: Miguel de Cervantes Saavedra  
 Posting Date: April 27, 2010 [EBook #2000]  
 Release Date: December, 1999  
 Language: Spanish

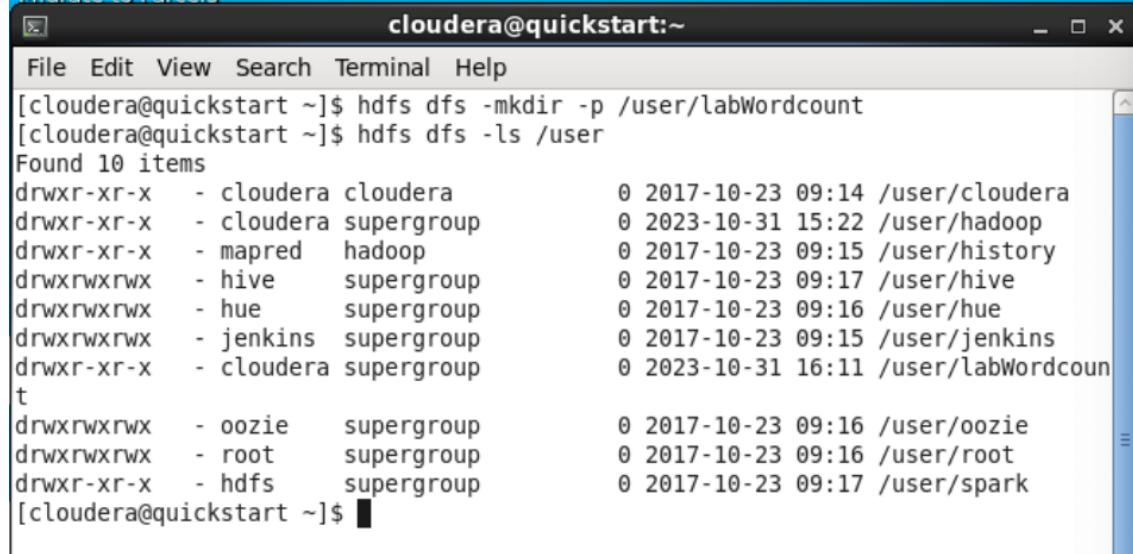
\*\*\* START OF THIS PROJECT GUTENBERG EBOOK DON QUIJOTE \*\*\*

Produced by an anonymous Project Gutenberg file corrections and new HTML file by Joaq

Save Page As...

## 2. Crea una carpeta de datos de entrada en HDFS

```
$ hdfs dfs -mkdir -p /user/labWordcount
```



```
cloudera@quickstart:~
```

```
File Edit View Search Terminal Help
```

```
[cloudera@quickstart ~]$ hdfs dfs -mkdir -p /user/labWordcount
[cloudera@quickstart ~]$ hdfs dfs -ls /user
Found 10 items
drwxr-xr-x  - cloudera cloudera      0 2017-10-23 09:14 /user/cloudera
drwxr-xr-x  - cloudera supergroup    0 2023-10-31 15:22 /user/hadoop
drwxr-xr-x  - mapred   hadoop       0 2017-10-23 09:15 /user/history
drwxrwxrwx  - hive     supergroup    0 2017-10-23 09:17 /user/hive
drwxrwxrwx  - hue     supergroup    0 2017-10-23 09:16 /user/hue
drwxrwxrwx  - jenkins  supergroup    0 2017-10-23 09:15 /user/jenkins
drwxr-xr-x  - cloudera supergroup    0 2023-10-31 16:11 /user/labWordcount
drwxrwxrwx  - oozie   supergroup    0 2017-10-23 09:16 /user/oozie
drwxrwxrwx  - root    supergroup    0 2017-10-23 09:16 /user/root
drwxr-xr-x  - hdfs   supergroup    0 2017-10-23 09:17 /user/spark
[cloudera@quickstart ~]$
```

## 3. Copia el archivo "texto.txt" a la carpeta de datos de entrada en HDFS

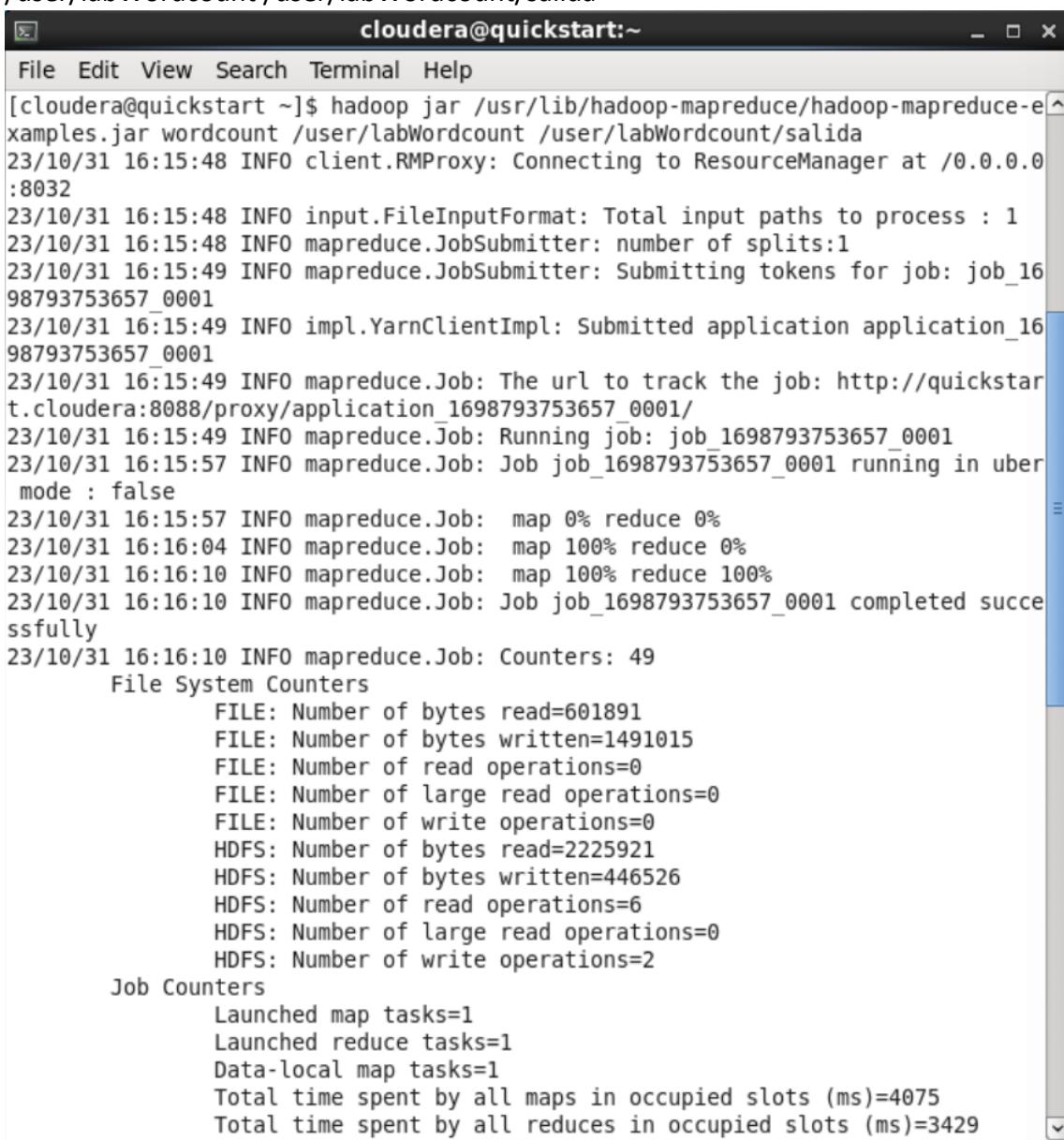
```
$ hdfs dfs -put Desktop/texto.txt /user/labWordcount
```

```
$ hdfs dfs -ls /user/labWordcount
```

```
[cloudera@quickstart ~]$ hdfs dfs -put Desktop/texto.txt /user/labWordcount
[cloudera@quickstart ~]$ hdfs dfs -ls /user/labWordcount
Found 1 items
-rw-r--r-- 1 cloudera supergroup 2225797 2023-10-31 16:13 /user/labWordcount/texto.txt
[cloudera@quickstart ~]$
```

4. Utiliza la biblioteca de ejemplos de Hadoop para contar las palabras en el archivo texto.txt usando: hadoop jar /usr/lib/hadoop-mapreduce/hadoop-mapreduce-examples.jar wordcount <carpeta HDFS de entrada> <nueva carpeta HDFS de salida>

```
$ hadoop jar /usr/lib/hadoop-mapreduce/hadoop-mapreduce-examples.jar wordcount
/usr/labWordcount /user/labWordcount/salida
```



```
cloudera@quickstart:~ - □ ×
File Edit View Search Terminal Help
[cloudera@quickstart ~]$ hadoop jar /usr/lib/hadoop-mapreduce/hadoop-mapreduce-examples.jar wordcount /user/labWordcount /user/labWordcount/salida
23/10/31 16:15:48 INFO client.RMProxy: Connecting to ResourceManager at /0.0.0.0:8032
23/10/31 16:15:48 INFO input.FileInputFormat: Total input paths to process : 1
23/10/31 16:15:48 INFO mapreduce.JobSubmitter: number of splits:1
23/10/31 16:15:49 INFO mapreduce.JobSubmitter: Submitting tokens for job: job_1698793753657_0001
23/10/31 16:15:49 INFO impl.YarnClientImpl: Submitted application application_1698793753657_0001
23/10/31 16:15:49 INFO mapreduce.Job: The url to track the job: http://quickstart.cloudera:8088/proxy/application_1698793753657_0001/
23/10/31 16:15:49 INFO mapreduce.Job: Running job: job_1698793753657_0001
23/10/31 16:15:57 INFO mapreduce.Job: Job job_1698793753657_0001 running in uber mode : false
23/10/31 16:15:57 INFO mapreduce.Job: map 0% reduce 0%
23/10/31 16:16:04 INFO mapreduce.Job: map 100% reduce 0%
23/10/31 16:16:10 INFO mapreduce.Job: map 100% reduce 100%
23/10/31 16:16:10 INFO mapreduce.Job: Job job_1698793753657_0001 completed successfully
23/10/31 16:16:10 INFO mapreduce.Job: Counters: 49
  File System Counters
    FILE: Number of bytes read=601891
    FILE: Number of bytes written=1491015
    FILE: Number of read operations=0
    FILE: Number of large read operations=0
    FILE: Number of write operations=0
    HDFS: Number of bytes read=2225921
    HDFS: Number of bytes written=446526
    HDFS: Number of read operations=6
    HDFS: Number of large read operations=0
    HDFS: Number of write operations=2
  Job Counters
    Launched map tasks=1
    Launched reduce tasks=1
    Data-local map tasks=1
    Total time spent by all maps in occupied slots (ms)=4075
    Total time spent by all reduces in occupied slots (ms)=3429
```

```
cloudera@quickstart:~
```

Total time spent by all reduce tasks (ms)=3429  
Total vcore-milliseconds taken by all map tasks=4075  
Total vcore-milliseconds taken by all reduce tasks=3429  
Total megabyte-milliseconds taken by all map tasks=4172800  
Total megabyte-milliseconds taken by all reduce tasks=3511296  
Map-Reduce Framework  
 Map input records=38054  
 Map output records=389641  
 Map output bytes=3740184  
 Map output materialized bytes=601891  
 Input split bytes=124  
 Combine input records=389641  
 Combine output records=39749  
 Reduce input groups=39749  
 Reduce shuffle bytes=601891  
 Reduce input records=39749  
 Reduce output records=39749  
 Spilled Records=79498  
 Shuffled Maps =1  
 Failed Shuffles=0  
 Merged Map outputs=1  
 GC time elapsed (ms)=84  
 CPU time spent (ms)=3460  
 Physical memory (bytes) snapshot=562921472  
 Virtual memory (bytes) snapshot=3138342912  
 Total committed heap usage (bytes)=549978112  
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=2225797  
File Output Format Counters  
 Bytes Written=446526

```
[cloudera@quickstart ~]$
```

NOTA: Es importante asegurarse de que la carpeta de entrada contiene nuestro archivo de texto y que no existe la carpeta de salida en HDFS antes de lanzar el job.

5. Comprueba el resultado en la carpeta de salida en HDFS. ¿Cuántas veces aparece la palabra "vaca" en el archivo de texto part-r-00000?

Más info en <https://geekland.eu/uso-del-comando-grep-en-linux-y-unix-con-ejemplos/#:~:text=El%20comando%20grep%20nos%20permite,o%20palabra%20que%20estamos%20buscando>

```
$ hdfs dfs -ls /user/labWordcount/salida
```

```
$ hdfs dfs -cat /user/labWordcount/salida/part-r-00000
```

```
$ hdfs dfs -cat /user/ labWordcount/salida/part-r-00000| more
```

```
$ hdfs dfs -cat /user/labWordcount/salida/part-r-00000 | grep vaca
```

```
[cloudera@quickstart ~]$ hdfs dfs -cat /user/hadoop/salida/part-r-00000 |grep vaca
vaca      5
vaca,     2
vaca;     1
vacada,   1
vacase,   1
[cloudera@quickstart ~]$
```

## 6. Se puede verificar que existe el archivo de salida desde el Browser

• To see [the output](#) through browser

Click on "Hadoop->HDFS Namenode->Utilities > Browse [the](#) file system" → / → user → cloudera → Practice → outwc → part-r-00000

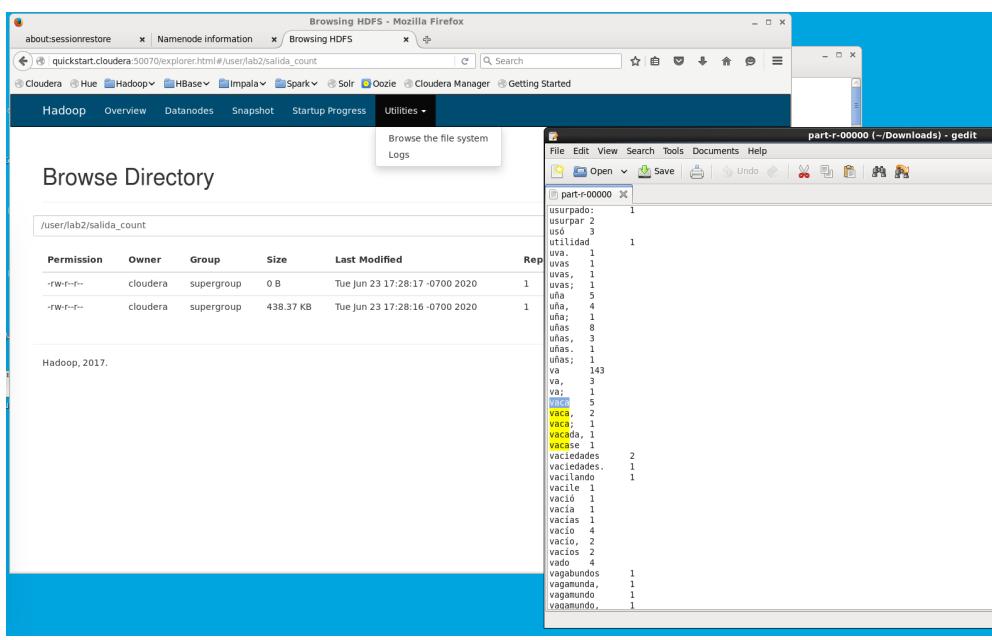




**Get this book in print ▾**

**★★★★★**  
0 Reviews  
Write review

**Hadoop Practice Guide: SQOOP, PIG, HIVE, HBASE for Beginners**  
By Jisha Mariam Jose



## O copia el archivo de hdfs a local

- Extraer un archivo de una carpeta HDFS.
6. Después de ejecutar nuestro procesamiento con Hadoop, haremos la operación inversa. Es decir, extraeremos los archivos de datos de HDFS y los llevaremos a una carpeta local de la máquina virtual Cloudera. La sintaxis de la operación get es: hdfs dfs -get <origen hdfs> <destino local>. En nuestro ejemplo, primero comprobaremos que los datos se han generado bien:

```
hdfs dfs -ls /user/labWordcount/salida
```

Debemos comprobar que podemos ver los archivos \_SUCCESS y part-r-00000 como resultado del procesamiento. Ahora extraeremos los resultados de HDFS:

```
hdfs dfs -get /user/labWordcount/salida/part-r-00000 /home/cloudera/Desktop
[cloudera@quickstart ~]$ hdfs dfs -get /user/labWordcount/salida/part-r-00000 /home/cloudera/Desktop
[cloudera@quickstart ~]$ ls Desktop
compartidaMV  Express.desktop  part-r-00000  texto.txt
Eclipse.desktop  Kerberos.desktop  prueba.txt
Enterprise.desktop  Parcels.desktop  prueba.txt~
[cloudera@quickstart ~]$
```

De nuevo usamos un editor de texto para comprobar que los resultados están bien:

```
gedit /home/cloudera/Downloads/part-r-00000
```

#### HAZLO TÚ MISMO

Busca o genera de 3 a 5 archivos .txt (y puedes usar tu carpeta compartida para ingresarlos en Centos), pasa esos archivos a una carpeta en HDFS, aplica un job wordcount y mira los resultados, cuántas 'map tasks' y 'reduce tasks' se levantaron? Cuántas particiones se crearon? Haz una captura de tus resultados de frecuencia de palabras...

#### NOTA FINAL

If you expect 10TB of input data and have a blocksize of 128MB, you'll end up with 82,000 maps, unless Configuration.set(MRJobConfig.NUM\_MAPS, int) (which only provides a hint to the framework) is used to set it even higher.