

# Prácticas BigData

## 1. Prácticas con HDFS

### 1.1. 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

```

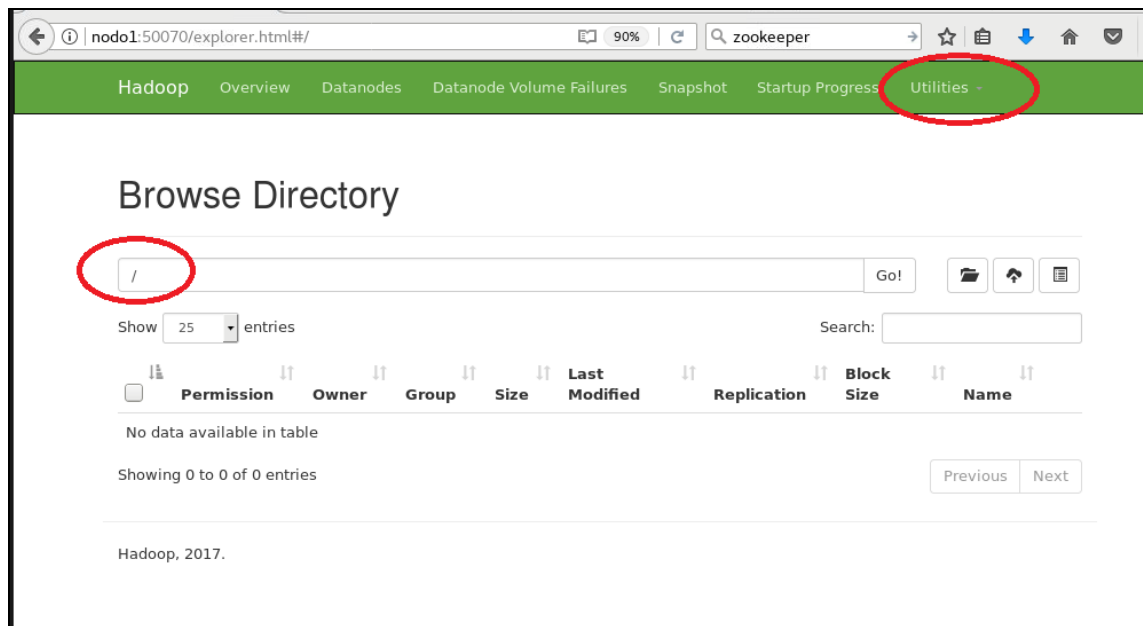
The general command line syntax is:

```
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
```

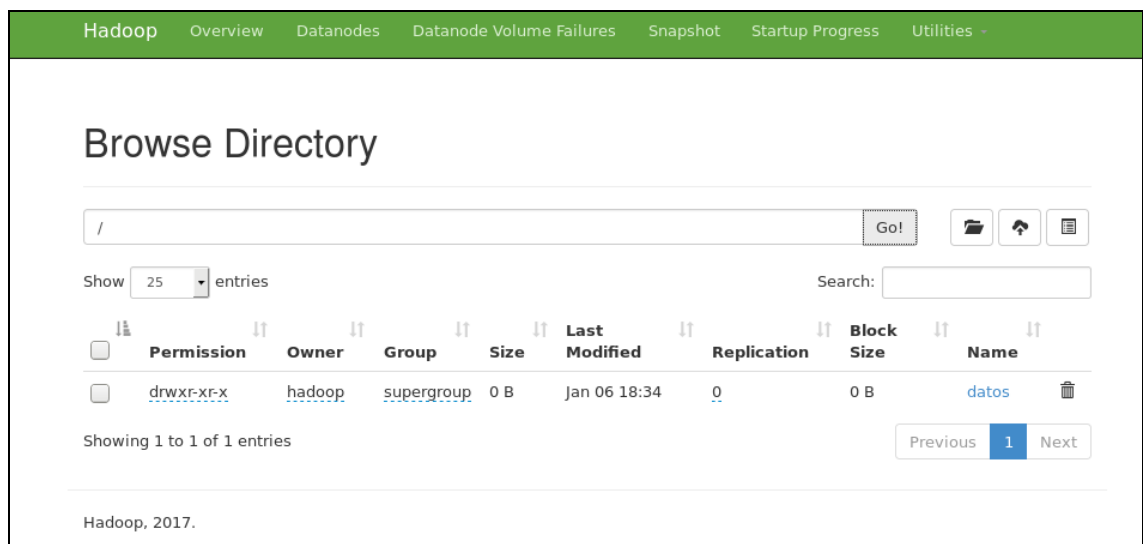
- Comprobar que existe

```
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
```

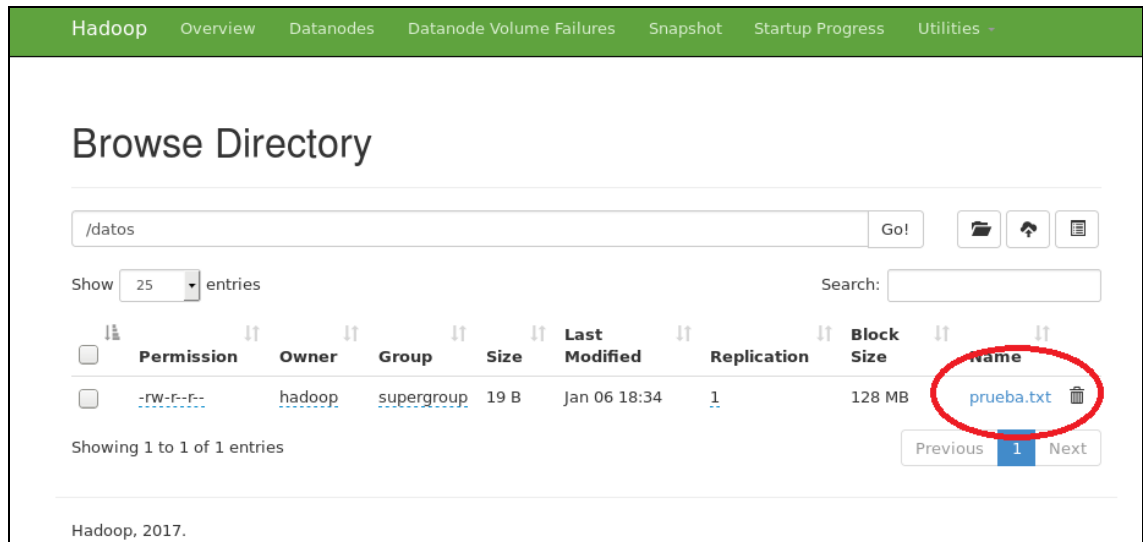
- Comprobar su existencia

```
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:

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

Showing 1 to 1 of 1 entries Previous 1 Next

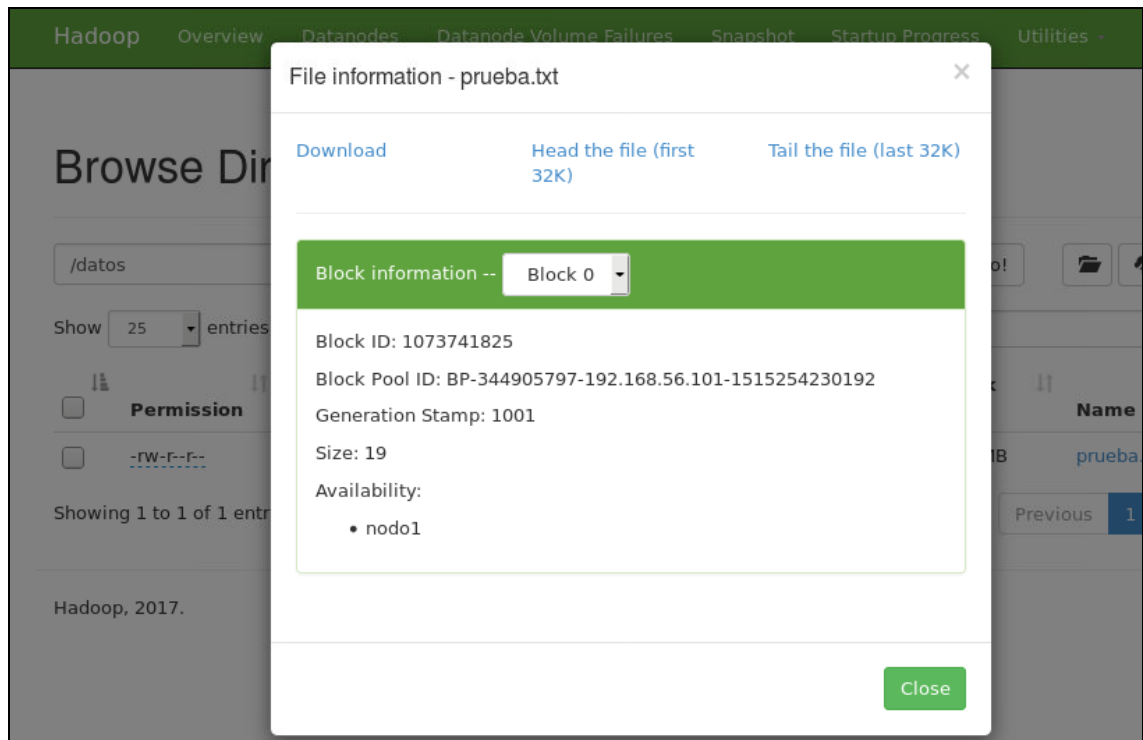
Hadoop, 2017.

- 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 the following information:

- Block information -- Block 0 (selected)
- Block ID: 107374182
- Block Pool ID: BP-34
- Generation Stamp: 1
- Size: 134217728
- Availability:
  - nodo1

A red arrow points to the file name "fic\_grande.dat" in the file list on the right side of the interface.

- 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 “practicas”

```
hdfs dfs -mkdir /practicas
```

- Copiamos prueba.txt desde datos a prácticas

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

- Comprobamos el contenido

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

- Borramos el fichero

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

- Vemos que los comandos son muy parecidos a Linux

## 1.2. 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
    
```