Prática sesión 5 - OPERACIONES BÁSICAS hdfs y yarn

El objetivo de la siguiente práctica es que el alumno se familiarice con las operaciones básicas sobre HDFS y YARN. Para ello se propone una serie de ejercicios que deben ir resolviéndose gradualmente para ir adquiriendo el conocimiento y destrezas necesarias para operar en el entorno distribuido de Hadoop.

Resumen

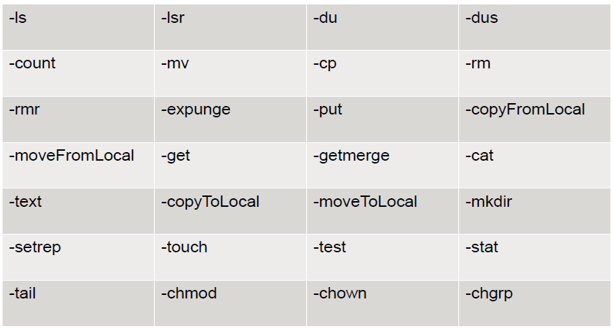
## Ejercicios HDFS

### Operaciones básicas

* Arrancamos HDFS:

|  |
| --- |
| start-dfs.sh |

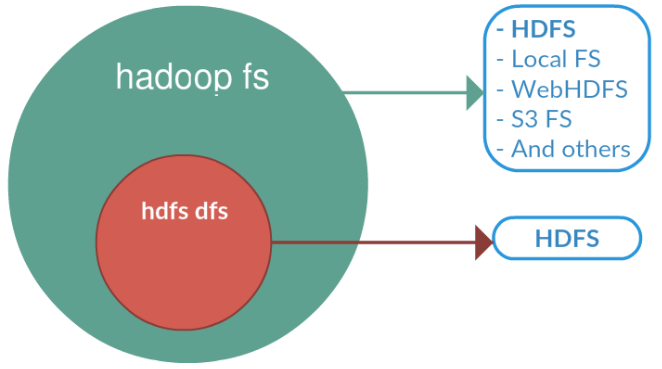
* Con el comando jps comprobamos que esté levantado el servicio.
* Accedemos al UI de HDFS en localhost:9870
* Recordamos algunos de los comandos que se pueden utilizar:



* Lanzamos los siguientes comandos:

|  |
| --- |
| hadoop fs –ls /user/  hadoop dfs –ls /user/  hdfs dfs –ls /user/ |

* *hadoop fs* se relaciona con un sistema de archivos genérico que puede apuntar a cualquier sistema de archivos como local, HDFS, FTP, S3, etc.
* *hadoop dfs* es específico de HDFS pero ha quedado ya obsoleto por lo que debemos usar hdfs dfs en su lugar.



* Creamos un directorio

|  |
| --- |
| hdfs dfs –mkdir /user/prueba |

* Subimos un archivo cualquiera, por ejemplo, alguno de hadoop, al directorio que acabamos de crear.

|  |
| --- |
| hdfs dfs –put /opt/hadoop/logs/hadoop.log /user/prueba/ |

* Con el comando –count podemos ver: el número de directorios, ficheros, tamaño y path

|  |
| --- |
| hdfs dfs –count /user/prueba/ |

* Descargamos el fichero que hemos subido anteriormente al directorio /tmp

|  |
| --- |
| hdfs dfs –copyToLocal /user/prueba/hadoop.log /tmp/ |

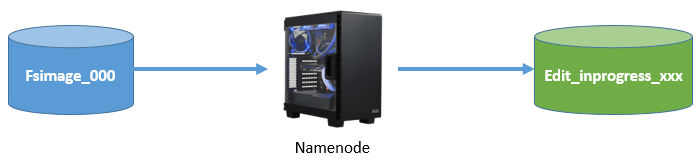
* Borramos el directorio que hemos creado

|  |
| --- |
| hdfs dfs –rm -r /user/prueba |

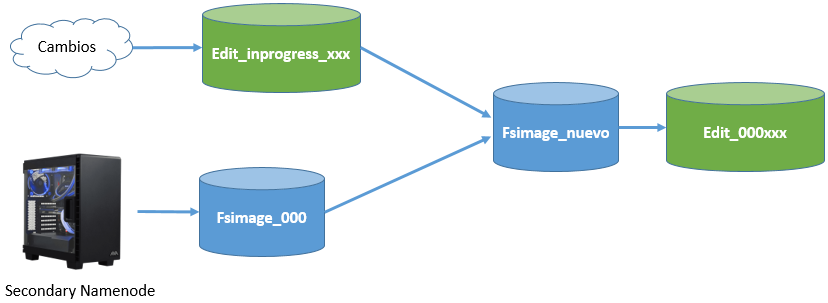
### 1.2 Fsimage y Editlog

Para gestionar los cambios que se producen en el clúster, HDFS utiliza los siguientes ficheros:

* + Edits\_000XXX
    - Contiene todos los cambios que se han producido en los metadatos de HDFS.
  + Edits\_inprogress\_000XXX
    - Donde se están escribiendo los cambios en los metadatos en la sesión activa.
  + Fsimage\_000XXX
    - Guarda una instantánea del estado de HDFS en un tiempo determinado.
    - Cuando se arranca HDFS se carga en memoria el último fichero fsimage que se encuentre disponible.
    - Fsimage no se modifica
    - Edit\_inprogress es quien va recogiendo los cambios que se van produciendo.



* + - Cada cierto tiempo o cada vez que se llena un bloque Secondary namenode, coge el fsimage anterior junto con todos los cambios que están en Edit\_inprogress y crea un edit\_000XX, para a continuación volver a crear un Edit\_inprogress.



* Antes de comenzar con los siguientes ejercicios, debemos movernos hasta el directorio de configuración de hadoop y consultar dónde se guarda la información del namenode: */opt/hadoop/etc/hadoop/conf/hdfs-site.xml*

|  |
| --- |
|  |

* Una vez localizada la ruta donde se almacena los metadatos del namenode, listamos los ficheros que contiene:

|  |
| --- |
| ls -l /datos/namenode/current |

* Comprobamos el id en el fichero VERSION

|  |
| --- |
| **hadoop@hadoop-VirtualBox:~$ cat /datos/namenode/current/VERSION**  #Fri Jun 04 11:26:50 CEST 2021  namespaceID=1402219917  clusterID=CID-83ed185a-1907-476b-bfda-7ce072ed045c  cTime=1618596221610  storageType=NAME\_NODE  blockpoolID=BP-1410034788-192.168.0.101-1618596221610  layoutVersion=-65 |

* Vamos a realizar un checkpoint manual para sincronizar el sistema de ficheros.
* Para ello entramos en modo SAFE para impedir que se trabaje con el sistema de ficheros mientras lanzamos el checkpoint.

|  |
| --- |
| hdfs dfsadmin -safemode enter |

* Comprobamos en el UI de HDFS que realmente esta en modo ON

|  |
| --- |
|  |

* Realizamos el checkpoint

|  |
| --- |
| hdfs dfsadmin -saveNamespace |

* Volvemos a entrar en modo normal

|  |
| --- |
| hdfs dfsadmin -safemode leave |

* Comprobamos que los fimage del namenode son iguales.

|  |
| --- |
| [Adjuntar captura] |

### 1.3. Administración de HDFS

* Realizamos un report del sistema HDFS

|  |
| --- |
| hdfs dfsadmin -report |

* Comprobamos con hdfs fsck el estado del sistema de ficheros

|  |
| --- |
| hdfs fsck /user/ |

* Comprobamos el estado de un determinado directorio, por ejemplo, temporal1

|  |
| --- |
| hdfs fsck /user/temporal1 |

* Comprobar la topología que tenemos en este momento

|  |
| --- |
| hdfs dfsadmin -printTopology |

* Comprobamos si hay algún fichero abierto

|  |
| --- |
| hdfs dfsadmin -listOpenFiles |

### 1.4. Bloques

* Lo primero que vamos a hacer es crear un directorio dentro del hdfs llamado temporal

|  |
| --- |
| hdfs dfs -mkdir /user/temporal/ |

* Una vez creado subimos el archivo de la carpeta Recurso al directorio de dfs creado en el paso anterior

|  |
| --- |
| hdfs dfs -put Descargas/el\_quijote.txt /user/temporal/ |

* Con el fichero subido nos vamos al hdfs UI: localhost:9870 y comprobamos que el Block Pool ID del block information, coincide con el del directorio de datos del datanode, dentro del directorio current:

|  |
| --- |
|  |

* Dentro de este subdirectorio existe otro current/finalized, donde Hadoop irá creando una estructura de subdirectorios subdir()… donde albergará los bloques de datos. En uno aparecen los datos y en el otro los metadatos

|  |
| --- |
| ls -l /datos/datanode/current/BP-1410034788-192.168.0.101-1618596221610/current/finalized/subdir0/subdir0/ |

* Creamos un nuevo directorio llamado temporal1 y copiamos el fichero prueba.txt del directorio temporal a temporal1.

|  |
| --- |
| hdfs dfs -mkdir /user/temporal1/ |

* Borramos el directorio temporal

|  |
| --- |
| hdfs dfs -rm -r /user/temporal/ |

* Ahora vamos a crear un fichero grande. Para ello lanzamos este comando que nos va a generar un fichero de 1G en /tmp, llamado giga\_test.dat que estará lleno de ceros.

|  |
| --- |
| dd if=/dev/zero of=/tmp/giga\_test.dat bs=1024 count=1000000 |

* Subimos el fichero a un directorio que creamos conveniente

|  |
| --- |
| hdfs dfs -put /tmp/giga\_test.dat /user/temporal1/ |

* Una vez subido nos vamos a hdfs UI file browser para ver los bloques que ha creado

|  |
| --- |
|  |

* Ahora nos vamos al directorio subdir() de datanode y podremos comprobar todos los bloques

|  |
| --- |
| ls -l /datos/datanode/current/BP-1410034788-192.168.0.101-1618596221610/current/finalized/subdir0/subdir0/ |

### 1.5. Snapshots

* Creamos un pequeño documento

|  |
| --- |
| echo "Hola" > hola.txt |

* Creamos un directorio HDFS para probar

|  |
| --- |
| hdfs dfs -mkdir /user/temporal1 (el que ja teníem) |

* Subimos el fichero que hemos creado

|  |
| --- |
| hdfs dfs -put hola.txt /user/temporal1/ |

* Ejecutamos un fsck sobre el fichero. Queremos obtener información sobre: Ficheros, bloques, nodos…

|  |
| --- |
| hdfs fsck /user/temporal1/hola.txt |

* En base a lo que hemos obtenido con la ejecución del comando anterior:

1. *BP-344905797-192.168.56.101-1515254230192:blk\_1073741837\_1013 len=20 Live\_repl=1 [DatanodeInfoWithStorage[127.0.0.1:50010,DS-173cc83b694a-425e-ad0f-c4c86352e2f6,DISK]]*

Buscamos el fichero en el sistema de ficheros de Linux a partir de su número de bloque

|  |
| --- |
| cat subdir0/subdir0/blk\_1073742032 |

* Habilitamos los snapshot sobre el directorio /datos2

|  |
| --- |
| hdfs dfsadmin -allowSnapshot /user/temporal1 |

* Creamos un snapshot llamado “s1” en el directorio

|  |
| --- |
| hdfs dfs -createSnapshot /user/temporal1 s1 |

* Comprobamos que se ha creado satisfactoriamente

|  |
| --- |
| hdfs dfs -ls /user/temporal1/.snapshot |

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

|  |
| --- |
| hdfs dfs -ls /user/temporal1/.snapshot/s1 |

* Borramos el fichero f1.txt

|  |
| --- |
| hdfs dfs -rm /user/temporal1/hola.txt |

* Podemos comprobar que ya no existe

|  |
| --- |
| hdfs dfs -ls /user/temporal1  Found 1 items  -rw-r--r-- 1 hadoop supergroup 1060259 2021-06-04 12:46 /user/temporal1/el\_quijote.txt |

* Sin embargo, con snapshot es muy fácil recuperarlo, simplemente lo copiamos de nuevo a su sitio original.

|  |
| --- |
| hdfs dfs -cp /user/temporal1/.snapshot/s1/hola.txt /user/temporal1 |

## Ejercicios YARN

### 1.1. Gestión de procesos

* Arrancamos yarn y comprobamos que esté arriba con jps

|  |
| --- |
| **hadoop@hadoop-VirtualBox:~$ start-yarn.sh**  Starting resourcemanager  Starting nodemanagers  **hadoop@hadoop-VirtualBox:~$ jps**  10896 ResourceManager  6736 DataNode  11376 Jps  3601 org.eclipse.equinox.launcher\_1.6.0.v20200915-1508.jar  6588 NameNode  11054 NodeManager  6910 SecondaryNameNode |

* Subimos el fichero el\_quijote.txt que puede descargarse de aquí: <https://gist.github.com/jsdario/6d6c69398cb0c73111e49f1218960f79>  a hdsf /user/temporal1/ o a cualquier otro directorio que creamos conveniente.

|  |
| --- |
| hdfs dfs -put Descargas/el\_quijote.txt /user/temporal1 |

* Ahora vamos a utilizar algunas de las rutinas mapreduce que vienen en el directorio share de hadoop: /opt/hadoop/share/hadoop/mapreduce/

|  |
| --- |
| hadoop jar hadoop-mapreduce-examples-3.2.2.jar wordcount /user/temporal1 /user/temporal1/resultat |

* Comprobamos el log que genera mapreduce

|  |
| --- |
| [Adjuntar captura] |

* Accedemos al directorio y comprobamos el resultado

|  |
| --- |
| **hdfs dfs -ls /user/temporal1**  Found 3 items  -rw-r--r-- 1 hadoop supergroup 1060259 2021-06-04 12:46 /user/temporal1/el\_quijote.txt  -rw-r--r-- 1 hadoop supergroup 5 2021-06-04 13:12 /user/temporal1/hola.txt  drwxr-xr-x - hadoop supergroup 0 2021-06-04 13:28 /user/temporal1/resultat |

* Descargamos el resultado al directorio /tmp con otro nombre para visualizarlo

|  |
| --- |
| **hdfs dfs -get /user/temporal1/resultat/part-r-00000 /tmp/comptar\_el\_quijote.txt**  **cat /tmp/comptar\_el\_quijote.txt** |

* Ahora accedemos a la Web de Administración de YARN y en “Applications” podemos ver la aplicación que acabamos de lanzar

|  |
| --- |
| Respuesta |

* Si en la misma fila, pulsamos sobre <<history>> obtendremos más información (\*) Nota al final del documento para que funcione **history**.

|  |
| --- |
| **mapred --daemon start historyserver** |

Para el seguimiento de la práctica enviar a [clopez@teralco.com](mailto:clopez@teralco.com)

Nota:

Para ver <<history>> Hay que ejecutar mapred --daemon start historyserver antes de ejecutar cualquier proceso. Si no mostrará el siguiente error:

