# HADOOP

## ACERCA DE HADOOP

El núcleo de Hadoop (**Hadoop core**) se compone de:

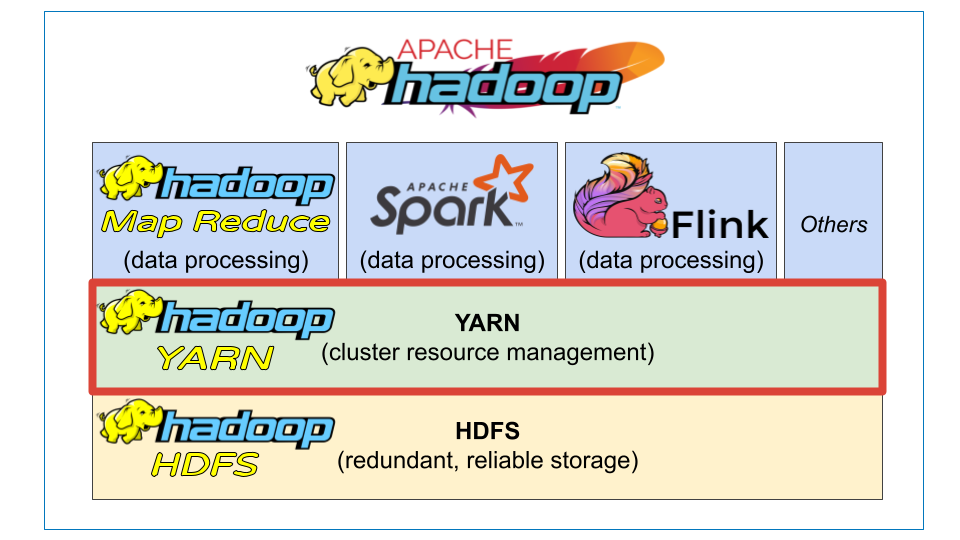
* un conjunto de utilidades comunes (**Hadoop common**)
* un sistema de ficheros distribuidos (Hadoop Distributed File System ↔ **HDFS**).
* un gestor de recursos para el manejo del clúster y la planificación de procesos (**YARN**)
* un sistema para procesamiento paralelo de grandes conjuntos de datos (**MapReduce**)

Ya hemos visto que es y cómo funciona HDFS así que antes de instalar Hadoop vamos a ver un poco cómo funciona YARN y MapReduce para poder montar un clúster Hadoop y hacer algunas pruebas.

### YARN

YARN (Yet Another Resource Negotiator) es un componente fundamental en el ecosistema Hadoop que se utiliza para administrar los recursos y programar trabajos en un clúster de Hadoop. Su función principal es asignar recursos de manera eficiente a las aplicaciones que se ejecutan en un clúster de Hadoop, lo que permite la ejecución simultánea de múltiples aplicaciones de manera coordinada.

Forma parte de Hadoop desde la versión 2. YARN soporta varios frameworks de procesamiento distribuido, como MapReduce, Tez, Impala, Spark, etc.



Componentes clave de YARN:

* + **ResourceManager (RM)**: el ResourceManager es el maestro del clúster y se encarga de administrar los recursos globales del clúster, asignando los recursos disponibles a las aplicaciones que lo solicitan.
  + **NodeManager (NM)**: cada nodo en el clúster Hadoop tiene un NodeManager que se encarga de administrar los recursos locales en ese nodo y proporciona información sobre la utilización de los recursos locales al ResourceManager.
  + **ApplicationMaster (AM)**: cada aplicación que se ejecuta en el clúster tiene su propio ApplicationMaster. El AM se comunica con el ResourceManager para solicitar recursos y gestionar la ejecución de la aplicación.

Flujo de trabajo de una aplicación YARN:

* + Cuando un usuario inicia una aplicación en el clúster, se crea una solicitud de recursos que se envía al ResourceManager.
  + El ResourceManager verifica la disponibilidad de recursos y asigna un contenedor en un nodo para el ApplicationMaster de la aplicación.
  + El ApplicationMaster se inicia en el contenedor asignado y se comunica con el ResourceManager para solicitar recursos adicionales según las necesidades de la aplicación.
  + El ResourceManager asigna contenedores adicionales en los nodos disponibles según las solicitudes del ApplicationMaster.
  + La aplicación se ejecuta en los contenedores asignados hasta que se complete.
  + El ResourceManager gestiona la asignación de recursos y la monitorización de la aplicación en ejecución.

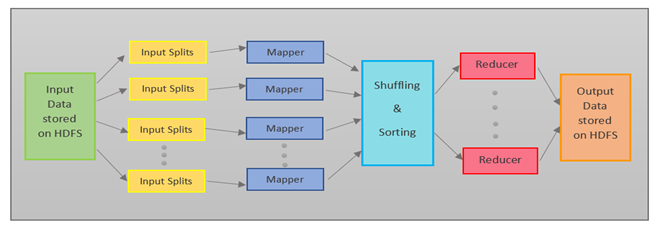
YARN gestiona recursos en términos de CPU y memoria en unidades llamadas "contenedores". Un contenedor representa una cantidad fija de recursos disponibles en un nodo del clúster.

Diagrama

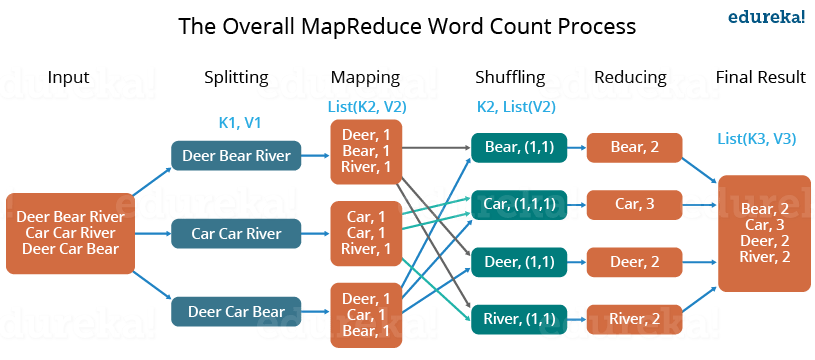
Descripción generada automáticamente

### MAPREDUCE

MapReduce se trata de un paradigma de programación funcional en dos fases, la de mapeo y la de reducción, y define el algoritmo que utiliza Hadoop para paralelizar las tareas. Un algoritmo MapReduce divide los datos, los procesa en paralelo, los reordena, combina y agrega de vuelta los resultados mediante un formato clave/valor.



El siguiente gráfico muestra un ejemplo de MapReduce que cuenta las apariciones de cada palabra en un fichero:



Como sugiere el nombre, el proceso se divide principalmente en dos fases:

* **Fase de mapeo (map):** los documentos se parten en pares de clave/valor. Hasta que no se reduzca, podemos tener muchos duplicados.
* **Fase de reducción (reduce):** es en cierta medida similar a un "group by" de SQL. Las ocurrencias similares se agrupan, y dependiendo de la función de reducción, se puede crear un resultado diferente.

MapReduce ya no se usa casi en favor de Spark a la hora de procesar los datos.

# INSTALACIÓN Y CONFIGURACIÓNDE UN CLÚSTER HADOOP MANUALMENTE

## INSTALACIÓN

### CREAR MÁQUINA VIRTUAL

Vamos a crear una máquina virtual en virtual box con las configuraciones e instalaciones básicas y después la clonaremos:

* Nombre: master (las otras dos máquinas se llamarán nodo1, nodo2 y nodo3)
* RAM: 8GB (si es posible)
* Procesadores: 2
* Disco duro: 100GB
* Interfaz de red: Red NAT ("BigData"): 192.0.2.5 (la ip de las otras dos máquinas será 192.0.2.10, 192.0.2.20 y 192.0.2.30)
* Sistema operativo: Ubuntu 22
* Usuario / password: iabd / Password0

#### FICHERO /etc/hosts

Modificamos el fichero /etc/hosts para que cuando clonemos la máquina reconozca los tres sistemas por el nombre:

***sudo nano /etc/hosts***

***/etc/hosts***

192.0.2.5 master

192.0.2.10 nodo1

192.0.2.20 nodo2

192.0.2.30 nodo3

#### SSH

Instalamos ssh para poder conectarnos al equipo vía ssh

***sudo apt-get update***

***sudo apt-get install openssh-server***

#### JAVA

Instalamos java con los siguientes comandos:

***sudo apt-get install default-jdk***

Con el siguiente comando comprobamos cual es la versión que tenemos:

***java -version***

Ahora necesitamos saber cuál es la ruta donde se ha instalado Java

***dirname $(dirname $(readlink -f $(which java)))***

*usr*lib/jvm/java-21-openjdf-amd6

#### HADOOP

Instalamos Hadoop:

Descargamos el fichero binario de Hadoop desde la página oficial: <https://hadoop.apache.org/releases.html> y comprobamos en la carpeta Downloads que se ha descargado y el nombre del archivo

***cd /home/iabd/Downloads***

***ls***

Vamos a hacer la instalación de Hadoop en el directorio /opt

***cd /opt***

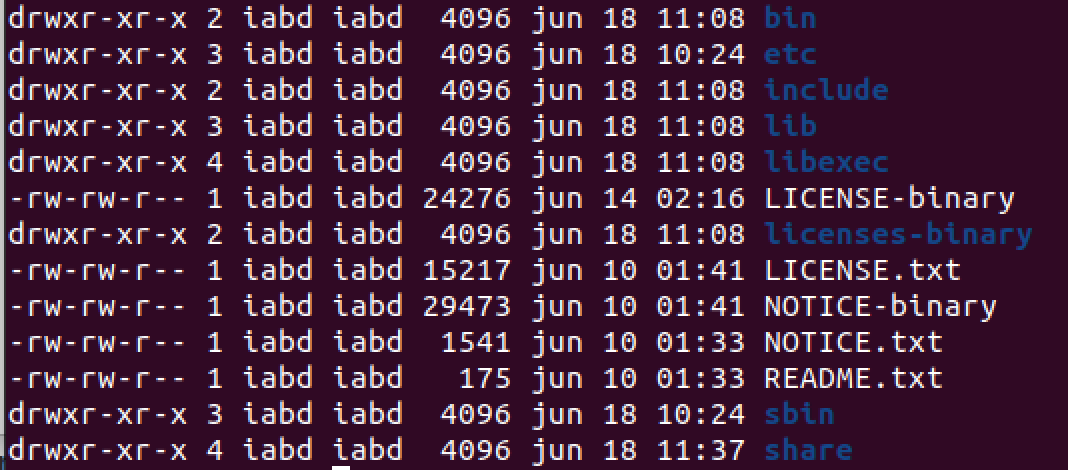
***sudo tar -zxvf /home/iabd/Downloads/hadoop-X.Y.Z.tar.gz***

Esto debe haber creado un directorio denominado hadoop-X.Y.Z y para hacer más sencillo el trabajo lo cambiamos de nombre a hadoop

***sudo mv hadoop-X.Y.Z hadoop***

Comprobamos que existen los ficheros desempaquetados en el directorio

***ls -l /opt/hadoop***



En el directorio bin van a estar los ejecutables, en el directorio etc los ficheros de configuración y en el directorio sbin algunos scripts para realizar ciertas tareas.

En caso de que el usuario iabd no sea el propietario deberíamos cambiarlo para que pertenezcan al usuario iabd, que es con el que vamos a trabajar (también se puede crear un usuario específico para Hadoop).

***sudo chown -R iabd:iabd hadoop***

#### VARIABLES DE ENTORNO

Creamos algunas variables de entorno necesarias en el fichero /home/iabd/.bashrc. Al modificar el fichero .bashrc estamos añadiendo las variables de entorno solo para el usuario iabd, hay que añadir las siguientes líneas al final del fichero

***cd /home/iabd***

***nano /home/iabd/.bashrc***

***/home/iabd/.bashrc***

export JAVA\_HOME=/usr/lib/jvm/java-11-openjdk-amd64

export HADOOP\_HOME=/opt/hadoop

export HADOOP\_INSTALL=$HADOOP\_HOME

export HADOOP\_MAPRED\_HOME=$HADOOP\_HOME

export HADOOP\_COMMON\_HOME=$HADOOP\_HOME

export HADOOP\_HDFS\_HOME=$HADOOP\_HOME

export HADOOP\_YARN\_HOME=$HADOOP\_HOME

export HADOOP\_COMMON\_LIB\_NATIVE\_DIR=$HADOOP\_HOME/lib/native

export PATH=$PATH:$HADOOP\_HOME/bin:$HADOOP\_HOME/sbin

export HADOOP\_OPTS="-Djava.library.path=$HADOOP\_HOME/lib/native"

Para probarlo, salimos de la sesión y volvemos a entrar, o bien ejecutamos el siguiente comando desde /home/iabd:

***source ./.bashrc***

Para comprobar la instalación de Hadoop ejecutamos el siguiente comando:

***hadoop version***

Ahora tenemos que asegurarnos que en el fichero /opt/hadoop/etc/hadoop-env.sh el valor que se asigna a JAVA\_HOME es el correcto:

***nano /opt/hadoop/etc/hadoop/hadoop-env.sh***

***/opt/hadoop/etc/hadoop/hadoop-env.sh***

export JAVA\_HOME=/usr/lib/jvm/java-11-openjdk-amd64

#### DIRECTORIO DATOS

Creamos un directorio denominado /datos, donde crearemos nuestros directorios de datos

***sudo mkdir -p /datos/{namenode,datanode}***

Damos permisos sobre el directorio al usuario iadb

***sudo chown -R iabd:iabd /datos***

### CLONAR MÁQUINA

Vamos a clonar la máquina creada tres veces (recordar cambiar la MAC) y después hay que hacer algunos cambios en las máquinas clonadas:

* Cambiar el hostname: nodo1, nodo2 y nodo3
* Cambiar la IP:
  + nodo1  192.0.2.10
  + nodo2  192.0.2.20
  + nodo3  192.0.2.30

### DISTRIBUIR PARES DE CLAVES DE AUTENTICACIÓN PARA EL USUARIO DE HADOOP

El nodo master utilizará una conexión SSH para conectarse a los otros nodos con autenticación de par de claves.

En el equipo master:

* Generamos la clave ssh (para el usuario iabd que es con el que hemos iniciado sesión) con el siguiente comando:

***cd /home/iabd***

***ssh-keygen -t rsa***

Al generar la clave dejamos el campo de la contraseña en blanco.

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

***ls /home/iabd/.ssh***

* Creamos el fichero/home/iabd/.ssh/authorized\_keys que necesitamos para luego pasar la clave pública al resto de nodos.

***cat /home/iab ubuntu d/.ssh/id\_rsa.pub >> /home/ubuntu/.ssh/authorized\_keys***

* Copiamos la clave del master a los dos nodos para tener conectividad ssh.

***scp /home/ubuntu/.ssh/authorized\_keys ubuntu@nodo1:/home/ubuntu/.ssh/authorized\_keys***

***scp /home/ubuntu/.ssh/authorized\_keys ubuntu@nodo2:/home/ubuntu/.ssh/authorized\_keys***

***scp /home/ubuntu/.ssh/authorized\_keys***

***ubuntu@node3:/home/ubuntu/.ssh/authorized\_keys***

Si queremos poder trabajar desde todos los nodos debemos hacer los mismo en el resto de los equipos y pasar la clave al resto.

nodo1

***cd /home/iabd***

***ssh-keygen -t rsa***

***cat /home/iabd/.ssh/id\_rsa.pub >> /home/ubuntu/.ssh/authorized\_keys***

***scp /home/ubuntu/.ssh/authorized\_keys ubuntu@master:/home/ubuntu/.ssh/authorized\_keys***

***scp /home/ubuntu/.ssh/authorized\_keys ubuntu@node2:/home/ubuntu/.ssh/authorized\_keys***

***scp /home/ubuntu/.ssh/authorized\_keys ubuntu@node3:/home/ubuntu/.ssh/authorized\_keys***

nodo2

***cd /home/iabd***

***ssh-keygen -t rsa***

***cat /home/ubuntu/.ssh/id\_rsa.pub >> /home/ubuntu/.ssh/authorized\_keys***

***scp /home/ubuntu/.ssh/authorized\_keys ubuntu@master:/home/ubuntu/.ssh/authorized\_keys***

***scp /home/ubuntu/.ssh/authorized\_keys ubuntu@node1:/home/ubuntu/.ssh/authorized\_keys***

***scp /home/ubuntu/.ssh/authorized\_keys ubuntu@node3:/home/ubuntu/.ssh/authorized\_keys***

nodo3

***cd /home/iabd***

***ssh-keygen -t rsa***

***cat /home/ubuntu/.ssh/id\_rsa.pub >> /home/ubuntu/.ssh/authorized\_keys***

***scp /home/ubuntu/.ssh/authorized\_keys ubuntu@master:/home/ubuntu/.ssh/authorized\_keys***

***scp /home/ubuntu/.ssh/authorized\_keys ubuntu@node1:/home/ubuntu/.ssh/authorized\_keys***

***scp /home/ubuntu/.ssh/authorized\_keys ubuntu@node2:/home/ubuntu/.ssh/authorized\_keys***

## CONFIGURACIÓN DEL MASTER

La configuración se realizará en el master y se replicará en los otros nodos.

### NAMENODE

Actualizamos /opt/hadoop/etc/hadoop/core-site.xml para establecer la ubicación del Namenode en el master en el puerto 9000, hay que añadir o modificar las siguientes property en configuration:

***nano /opt/hadoop/etc/hadoop/core-site.xml***

**/opt/hadoop/etc/hadoop/core-site.xml**

*<configuration>*

*<property>*

*<name>fs.default.name</name>*

*<value>hdfs://master:9000</value>*

*</property>*

*</configuration>*

### HDFS

Edita /opt/hadoop/etc/hadoop/hdfs-site.conf para indicar donde se van a guardar los datos en el Namenode y en los Datanode. También se configura el acceso web (para poder acceder desde otras aplicaciones). Hay que añadir o modificar las siguientes property en configuration:

***nano /opt/hadoop/etc/hadoop/hdfs-site.xml***

**/opt/hadoop/etc/hadoop/hdfs-site.xml**

*<property>*

*<name>dfs.namenode.name.dir</name>*

*<value>/datos/namenode</value>*

*</property>*

*<property>*

*<name>dfs.datanode.data.dir</name>*

*<value>/datos/datanode</value>*

*</property>*

*<property>*

*<name>dfs.replication</name>*

*<value>2</value>*

*</property>*

*<property>*

*<name>dfs.webhdfs.enabled</name>*

*<value>true</value>*

*</property>*

*<property>*

*<name>dfs.namenode.http-address</name>*

*<value>master:9870</value>*

*</property>*

La última propiedad, dfs.replication, indica cuántas veces se replican los datos en el clúster y el valor no puede ser un valor más alto que el número real de nodos trabajadores.

El sistema de archivos HDFS necesita ser formateado como cualquier otro sistema de archivo clásico. En el master ejecutamos el siguiente comando:

***hdfs namenode -format***

Nuestra instalación de Hadoop está ahora configurada y lista para funcionar.

### YARN

Edita /opt/hadoop/etc/hadoop/mapred-site.xml para añadir o modificar las siguientes property en configuration:

***nano /opt/hadoop/etc/hadoop/mapred-site.xml***

**/opt/hadoop/etc/hadoop/mapred-site.xml**

*<property>*

*<name>mapreduce.framework.name</name>*

*<value>yarn</value>*

*</property>*

*<property>*

*<name>yarn.app.mapreduce.am.env</name>*

*<value>HADOOP\_MAPRED\_HOME=$HADOOP\_HOME</value>*

*</property>*

*<property>*

*<name>mapreduce.map.env</name>*

*<value>HADOOP\_MAPRED\_HOME=$HADOOP\_HOME</value>*

*</property>*

*<property>*

*<name>mapreduce.reduce.env</name>*

*<value>HADOOP\_MAPRED\_HOME=$HADOOP\_HOME</value>*

*</property>*

Edita /opt/hadoop/etc/hadoop/yarn-site.xml para añadir o modificar las siguientes property en configuratio. En el campo value para el campo yarn.resourcemanager.hostname, hay que poner el nombre del master:

***nano /opt/hadoop/etc/hadoop/yarn-site.xml***

**/opt/hadoop/etc/hadoop/yarn-site.xml**

*<property>*

*<name>yarn.resourcemanager.hostname</name>*

*<value>master</value>*

*</property>*

*<property>*

*<name>yarn.nodemanager.aux-services</name>*

*<value>mapreduce\_shuffle</value>*

*</property>*

### WORKERS

Edita /opt/hadoop/etc/hadoop/workers para incluir todos los nodos trabajadores:

***nano /opt/hadoop/etc/hadoop/workers***

**/opt/hadoop/etc/hadoop/workers**

*nodo1*

*nodo2*

*nodo3*

### REPLICAR CONFIGURACIÓN EN LOS WORKERS

Hay que replicar algunos de los archivos modificados en los nodos trabajadores

***cd /opt/hadoop/etc/hadoop***

***sudo scp core-site.xml ubuntu@192.0.2.10:/opt/hadoop/etc/hadoop/***

***sudo scp hdfs-site.xml ubuntu@192.0.2.10:/opt/hadoop/etc/hadoop/***

***sudo scp mapred-site.xml ubuntu@192.0.2.10:/opt/hadoop/etc/hadoop/***

***sudo scp yarn-site.xml ubuntu@192.0.2.10:/opt/hadoop/etc/hadoop/***

***sudo scp workers ubuntu@192.0.2.10:/opt/hadoop/etc/hadoop/***

***sudo scp core-site.xml ubuntu@192.0.2.20:/opt/hadoop/etc/hadoop/***

***sudo scp hdfs-site.xml ubuntu@192.0.2.20:/opt/hadoop/etc/hadoop/***

***sudo scp mapred-site.xml ubunt@192.0.2.20:/opt/hadoop/etc/hadoop/***

***sudo scp yarn-site.xml ubuntu@192.0.2.20:/opt/hadoop/etc/hadoop/***

***sudo scp workers ubuntu@192.0.2.20:/opt/hadoop/etc/hadoop/***

***sudo scp core-site.xml ubuntu@192.0.2.30:/opt/hadoop/etc/hadoop/***

***sudo scp hdfs-site.xml ubuntu@192.0.2.30:/opt/hadoop/etc/hadoop/***

***sudo scp mapred-site.xml ubuntu@192.0.2.30:/opt/hadoop/etc/hadoop/***

***sudo scp yarn-site.xml ubuntu@192.0.2.30:/opt/hadoop/etc/hadoop/***

***sudo scp workers ubuntu@192.0.2.30:/opt/hadoop/etc/hadoop/***

## ARRANCAR, MONITORIZAR Y DETENER LOS SERVICIOS

#### ARRANCAR Y DETENER

Iniciamos los servicios ejecutando el siguiente script desde el master:

***start-all.sh***

Podemos comprobar con el comando jps los procesos activos en cada equipo.

***jps***

had

En el master deberíamos ver los siguientes procesos:

* Jps
* NameNode
* SecondaryNameNode
* ResourceManager

En nodo1, nodo2 y nodo3 deberíamos ver los siguientes procesos:

* DataNode
* NodeManager
* Jps

Para detener los servicios tenemos los siguientes comandos:

***stop-all.sh***

#### MONITORIZAR

**MODO COMANDO**

Podemos obtener información útil sobre el clúster HDFS con el siguiente comando:

***hdfs dfsadmin -report***

El siguiente comando muestra la topología, identificando los nodos que tenemos y al rack al que pertenece cada nodo.

***hdfs dfsadmin -printTopology***

Esto imprimirá información (por ejemplo, capacidad y uso) para todos los Datanodes en ejecución. Para obtener la descripción de todos los comandos disponibles:

***hdfs dfsadmin -help***

Con el siguiente comando podemos imprimir un informe de los nodos en ejecución:

***yarn node -list***

Con el siguiente comando podemos obtener una lista de aplicaciones en ejecución:

***yarn application -list***

**INTERFAZ WEB**

Tenemos una interfaz web para monitorizar el clúster HDFS:

[***http://master:9870***](http://master:9870/)

Interfaz de usuario gráfica, Aplicación

Descripción generada automáticamente

También tenemos una interfaz web para monitorizar el clúster HDFS:

***http://master:8088***

Interfaz de usuario gráfica, Aplicación

Descripción generada automáticamente

# TRABAJANDO CON HADOOP

Para interactuar con el almacenamiento desde un terminal, se utiliza el comando hdfs.

Diagrama

Descripción generada automáticamente

**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\_HOME/sbin/start-dfs.sh**

**hdfs dfs**se utiliza para acceder a HDFS, el cual requiere de otro argumento (empezando con un guion) el cual será uno de los comandos Linux. Podéis consultar la lista de comandos en la [documentación oficial](https://hadoop.apache.org/docs/current/hadoop-project-dist/hadoop-common/FileSystemShell.html),

***hdfs dfs -comandos***

Por ejemplo, para mostrar todos los archivos que tenemos en el directorio raíz de HDFS haríamos:

***hdfs dfs -ls /***

Los comandos más utilizados son:

* ***put***: coloca un archivo dentro de HDFS
* ***get***: recupera un archivo de HDFS y lo lleva a nuestro sistema local.
* ***cat / head / tail***: visualiza el contenido de un archivo.
* ***mkdir / rmdir***: crea / borra una carpeta.
* ***count***: cuenta el número de elementos (número de carpetas, ficheros, tamaño y ruta).
* ***cp / mv / rm***: copia / mueve-renombra / elimina un archivo.

¿Sabes qué realiza cada uno de los siguientes comandos?

* ***hdfs dfs -mkdir -p /user/iabd/datos***
* ***hdfs dfs -put ejemplo.txt /user/iabd/datos/***
* ***hdfs dfs -put ejemplo.txt /user/iabd/datos/ejemploRenombrado.txt***
* ***hdfs dfs -ls /user/iabd/datos***
* ***hdfs dfs -count /user/iabd/datos***
* ***hdfs dfs -mv datos/ejemploRenombrado.txt /user/iabd/datos/otroNombre.json***
* ***hdfs dfs -get /user/iabd/datos/otroNombre.json /tmp***
* ***hdfs dfs -rm -r /user***

A continuación vamos a ver cómo trabaja internamente HDFS con los bloques. Para ello, vamos a trabajar con un archivo que ocupe más de un bloque, como pueden ser los datos de 25 millones de películas:

***wget https://files.grouplens.org/datasets/movielens/ml-25m.zip***

Descomprimimos el fichero y colocamos el archivo ratings.csv dentro de HDFS.

***unzip ml-25m.zip***

***hdfs dfs -mkdir /prueba***

***hdfs dfs -put ml-25m/ratings.csv /prueba***

Si queremos saber cuántos bloques ha creado y cuanto ocupa cada uno de ellos, podemos utilizar la utilidad fsck que permite comprobar la salud del sistema de archivos. Si le pasamos las opciones -files y -blocks nos mostrará información tanto de los archivos como de los bloques contenidos:

***hdfs fsck /prueba -files -blocks***

Desde la interfaz gráfica también podemos navegar por las carpetas de nuestro sistema de ficheros. Abrimos la interfaz gráfica ([http://master:9870](http://master:9870/)) y en el menú tenemos que elegir ***Utilities  Browse the file system***.

Interfaz de usuario gráfica, Texto, Aplicación, Correo electrónico

Descripción generada automáticamente

Si intentamos crear una carpeta o eliminar algún archivo recibimos un mensaje del tipo un mensaje de error de permiso denegado. Por defecto, los recursos vía web los crea el usuario dr.who.

Interfaz de usuario gráfica, Texto, Aplicación, Correo electrónico

Descripción generada automáticamente

Si queremos habilitar los permisos para que desde este la interfaz gráfica podamos crear/modificar/eliminar recursos, podemos cambiar permisos a la carpeta:

***hdfs dfs -mkdir /pruebas***

***hdfs dfs -chmod 777 /pruebas***

Otra posibilidad es modificar el archivo de configuración core-site.xml y añadir una propiedad para modificar el usuario estático:

***nano /opt/hadoop/etc/hadoop/core-site.xml***

**/opt/hadoop/etc/hadoop/core-site.xml**

*<property>*

*<name>hadoop.http.staticuser.user</name>*

*<value>iabd</value>*

*</property>*

Tras reiniciar Hadoop, ya podremos crear los recursos como el usuario iabd.

***stop-all.sh***

***start-all.sh***

Desde la interfaz gráfica también podemos obtener la información sobre los bloques del archivo haciendo click sobre el archivo:

Interfaz de usuario gráfica, Texto, Aplicación, Chat o mensaje de texto

Descripción generada automáticamente

Si desplegamos el combo Block information, podremos ver cómo se ha partido el archivo CSV en 6 bloques (650 MB aproximadamente que ocupa el fichero CSV / 128 MB del tamaño del bloque).

Interfaz de usuario gráfica, Texto, Aplicación, Chat o mensaje de texto

Descripción generada automáticamente

### SNAPSHOTS

Mediante las snapshots podemos crear una instantánea que almacena cómo está en un determinado momento nuestro sistema de ficheros, a modo de copia de seguridad de los datos, para en un futuro poder realizar una recuperación.

El primer paso es activar el uso de snapshots, mediante el comando de administración indicando sobre qué carpeta vamos a habilitar su uso:

***hdfs dfsadmin -allowSnapshot /prueba***

El siguiente paso es crear un snapshot, para ello se indica tanto la carpeta como un nombre para la captura:

***hdfs dfs -createSnapshot /prueba snapshot1***

Esta captura se creará dentro de una carpeta oculta dentro de la ruta indicada (en nuestro caso creará la carpeta /prueba/.snapshot/snapshot1/ la cual contendrá la información de la instantánea).

A continuación, vamos a borrar uno de los archivos creados anteriormente en la carpeta prueba y comprobar que ya no existe:

***hdfs dfs -rm /prueba/ratings.csv***

***hdfs dfs -ls /prueba***

Para comprobar el funcionamiento de los snapshots, vamos a recuperar el archivo desde la captura creada anteriormente y comprobamos que volvemos a tener el fichero.

***hdfs dfs -cp /prueba/.snapshot/snapshot1/ratings.csv /prueba***

***hdfs dfs -ls /prueba***

Si queremos saber que carpetas soportan las instantáneas podemos usar el comando:

***hdfs lsSnapshottableDir***

Finalmente, si queremos deshabilitar las snapshots de una determinada carpeta, primero hemos de eliminarlas y luego deshabilitarlas:

***hdfs dfs -deleteSnapshot /prueba snapshot1***

***hdfs dfsadmin -disallowSnapshot /prueba***

### HDFS Y PYTHON

Para acceder desde Python a HDFS podemos utilizar la librería [HdfsCLI](https://hdfscli.readthedocs.io/en/latest/).

Primero hemos de instalarla mediante pip:

***pip install hdfs***

Vamos a ver un sencillo ejemplo de lectura y escritura en HDFS:

***nano ejemplo.py***

**ejemplo.py**

*#!/usr/bin/python3*

*from hdfs import InsecureClient*

*#LEER FICHERO*

*url="http://nombre\_servidor:50070"*

*usuario="nombre\_usuario"*

*client = InsecureClient(url, usuario)*

*fichero = "ruta\_y\_nombre\_fichero"*

*with client.read(fichero) as hdfs\_file:*

*texto = hdfs\_file.read()*

*print(texto)*

*#ESCRIBIR FICHERO*

*datos="xxxxxxxxxxxxxxx"*

*client.write("ruta\_y\_nombre\_fichero", datos)*

# EJEMPLOS

### EJEMPLO 1

Creamos un directorio en el directorio raíz de Hadoop

***hdfs dfs -mkdir -p /books***

Utilicemos como ejemplo algunos libros de texto del [proyecto Gutenberg](https://www.gutenberg.org/).

***cd /home/iabd/Downloads***

***wget -O alice.txt https://www.gutenberg.org/files/11/11-0.txt***

***wget -O holmes.txt https://www.gutenberg.org/files/1661/1661-0.txt***

***wget -O frankenstein.txt https://www.gutenberg.org/files/84/84-0.txt***

Subimos los tres libros al directorio /books de Hadoop:

***hdfs dfs -put alice.txt holmes.txt frankenstein.txt /books***

Veamos el contenido del fichero /books de Hadoop:

***hdfs dfs -ls /books***

Veamos como descargar uno de los libros de Hadoop al sistema de archivos local y comprobamos que se haya descargado correctamente:

***hdfs dfs -get /books/alice.txt /home/iabd***

***ls /home/iabd***

También se puede visualizar el contenido del libro directamente desde el HDFS:

***hdfs dfs -cat /books/alice.txt***

### EJEMPLO 2

Vamos ahora a ver cómo realizar un trabajo distribuido gracias a YARN y MapReduce

Los trabajos empaquetados en jar se envían a YARN para su ejecución con el comando ***yarn jar***.

El paquete de instalación Hadoop proporciona aplicaciones de muestra que pueden ejecutarse para probar el clúster. Vamos a usar la aplicación de muestra que cuenta las palabras para contar las palabras de los tres libros que previamente hemos subido a HDFS.

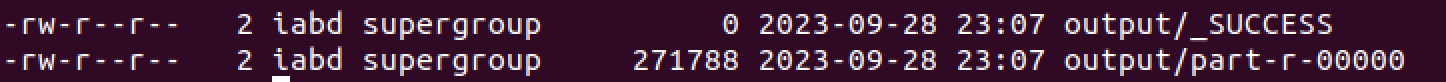
En el master tenemos que ejecutar:

***yarn jar /opt/hadoop/share/hadoop/mapreduce/hadoop-mapreduce-examples-3.3.6.jar wordcount "/books/\*" output***

El último argumento es donde se guardará la salida del trabajo en HDFS como hemos dado una ruta relativa lo va a crear en /user/iabd (crea una carpeta para nuestro usuario).

Una vez finalizado el trabajo, podemos obtener el resultado con los siguientes comandos:

***hdfs dfs -ls output***



***hdfs dfs -cat output/part-r-00000 | less***

Podemos mostrar por pantalla las 10 palabras que más veces aparecen en el texto, junto con el número de apariciones, pero para hacerlo primero vamos a descargar el documento con la solución.

***hdfs dfs -get output/part-r-00000***

***sort part-r-00000 -k2 -n -r | head***

Podemos monitorizar la ejecución de los procesos a través de la interfaz web del ResourceManager. Por ejemplo, en el apartado Cluster -> Applications -> Finished podemos ver la información del trabajo lanzado.

¿Cuánto tiempo tardó en realizarse?

### EJEMPLO 3

Ejecuta los siguientes comandos y piensa que es lo que hacen:

***hdfs dfs -mkdir /input***

***hdfs dfs -copyFromLocal /opt/hadoop/etc/hadoop/\*.xml /input***

***yarn jar /opt/hadoop/share/hadoop/mapreduce/hadoop-mapreduce-examples-3.3.6.jar grep /input output3 'dfs[a-z.]'***

***hdfs dfs -cat output3/\****

En este ejemplo puedes ver la estructura y funcionamiento básico de una ejecución de Hadoop:

1. Copiamos los ficheros de configuración de Hadoop en HDFS en la carpeta /input
2. Se lanza un trabajo, en concreto realizar un "grep" con una expresión regular ('dfs[a-z.] +') sobre los ficheros de entrada situados en /input.
3. El resultado se escribe en un directorio de salida /user/iabd/output3
4. Consulta los ficheros de salida

Además de la función grep, Hadoop trae otros ejemplos de trabajos MapReduce preconfigurados. Puedes consultarlos ejecutando el siguiente código:

***yarn jar /opt/hadoop/share/hadoop/mapreduce/hadoop-mapreduce-examples-3.3.6.jar***

### EJEMPLO 4

Veamos el ejemplo de MapReduce que permite calcular la media de la longitud de las palabras en una serie de ficheros de texto. Realizamos el cálculo sobre los ficheros de texto de los libros presentados en anteriores ejemplos.

***yarn jar /opt/hadoop/share/hadoop/mapreduce/hadoop-mapreduce-examples-3.3.6.jar wordmean /books/\* media\_libros***

Visualizamos el resultado:

***hdfs dfs -cat media\_libros/part-r-00000***

Imagen de la pantalla de un video juego

Descripción generada automáticamente con confianza baja

### EJEMPLO 5

Podemos monitorizar la ejecución de los procesos a través de la interfaz web del ResourceManager (***http://master:8088***). Por ejemplo, en el apartado ***Cluster  Applications  Finished*** podemos ver la información del trabajo lanzado.

¿Cuánto tiempo tardó en realizarse?