IES SAN ANDRÉS

ESPECIALIZACIÓN EN INTELIGENCIA ARTIFICIAL Y BIG DATA

BIG DATA APLICADO

UT02: ALMACENAMIENTO DISTRIBUIDO. HDFS

ÍNDICE

1 ARQUITECTURA Y CONCEPTOS DE HADOOP

CARACTERÍSTICAS

Víctor J. González

Sistema de ficheros **distribuido** de Hadoop

Tolerancia a fallos | Acceso a datos en flujo | Escalabilidad | Simplicidad | Alta disponibilidad

HDFS está diseñado para tener una alta tolerancia a fallos en el sistema con acciones tanto **correctivas** como **preventivas**.

Los ficheros que almacena se dividen en fragmentos denominados **bloques**, cada uno de 64 MB o 128 MB

Los bloques se **replican** a lo largo del clúster con un **factor de replicación** que suele ser 3, por lo que, aunque falle una máquina siempre habrá disponibles otras dos copias del bloque

CARACTERÍSTICAS

Sistema de ficheros **distribuido** de Hadoop

HDFS se basa en el principio **escribe una vez, lee muchas** (*write once, read many*).

Los datos son servidor por el cliente HDFS en forma de **flujos**.

Esto significa que **no hay que esperar a que se haya leído todo el fichero para obtener los datos**, sino que en el momento en que se empieza a leer se comienzan a enviar los datos al cliente para que pueda empezar a procesarlos.

Sistema de ficheros **distribuido** de Hadoop

Tolerancia a fallos | Acceso a datos en flujo | **Escalabilidad** | Simplicidad | Alta disponibilidad

HDFS es un sistema de ficheros **altamente escalable** y se le pueden añadir cualquier número de máquinas para incrementar su capacidad de almacenamiento.

Hay que tener en cuenta que es habitual clústeres con cientos de nodos o incluso miles de nodos.

Yahoo (origen de Hadoop) ha llegado a tener un clúster con 42000 nodos y 600 petabytes (600000 TB) de datos en HDFS (https://www.globalbigdataconference.com/news/129559/how-yahoo-using-hadoop-in-real-time.html)

CARACTERÍSTICAS

Sistema de ficheros **distribuido** de Hadoop

Tolerancia a fallos	Acceso a datos en flujo	Escalabilidad	**Simplicidad**	Alta disponibilidad

HDFS es sencillo de configurar y administrar. Está escrito en Java y proporciona una interfaz en línea de comandos muy similar a los comandos de Linux, lo que hace muy sencillo su aprendizaje.

CARACTERÍSTICAS

Sistema de ficheros **distribuido** de Hadoop

Tolerancia a fallos	Acceso a datos en flujo	Escalabilidad	Simplicidad	**Alta disponibilidad**

HDFS está diseñado para ser un sistema de alta disponibilidad.

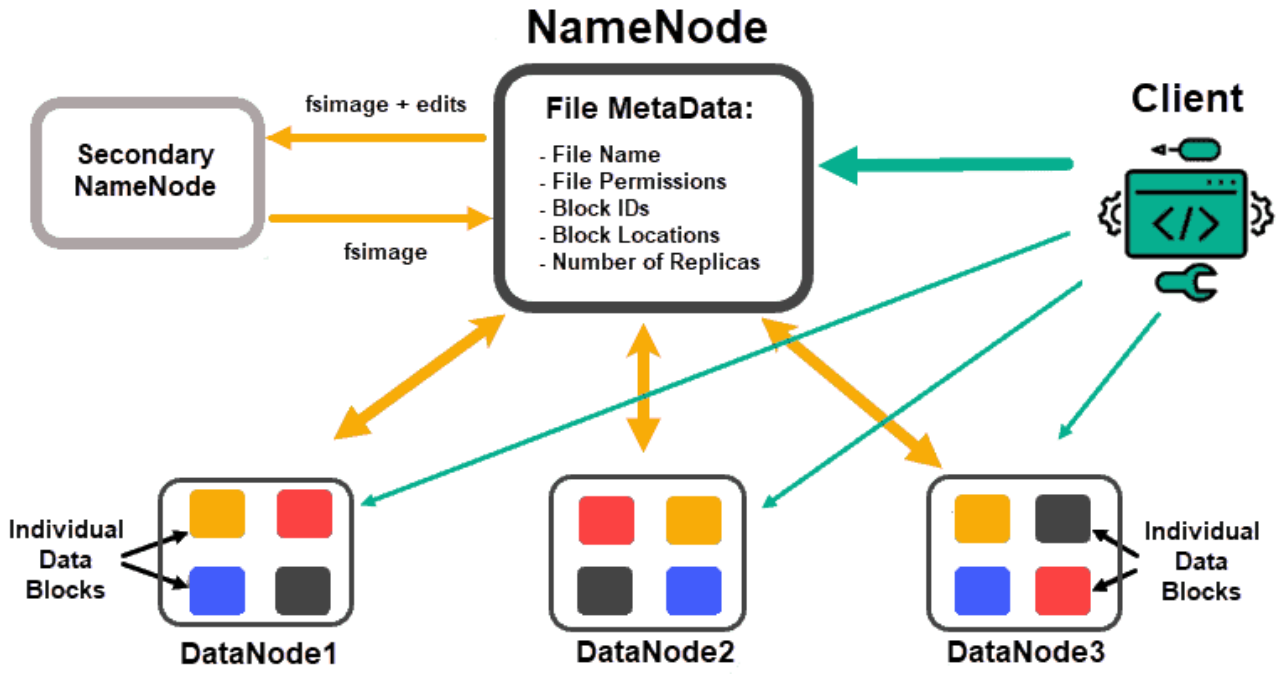
Cada solicitud de lectura/escritura se dirige a un **nodo maestro** (NameNode), dado que esto crea un **punto único de fallo** (*SPOF – Single Point of Failure*) HDFS dispone de **otro NameNode** en espera que listo para asumir el control si el NameNode principal falla.

ARQUITECTURA DE HDFS

Víctor J. González

La arquitectura de HDFS se basa en un **modelo maestro-esclavo** cuyos componentes son:

- **NameNode** (maestro)
- **DataNodes** (esclavos)
- **Secondary NameNode**
- **Cliente HDFS**



ARQUITECTURA DE HDFS

Víctor J. González

NameNode

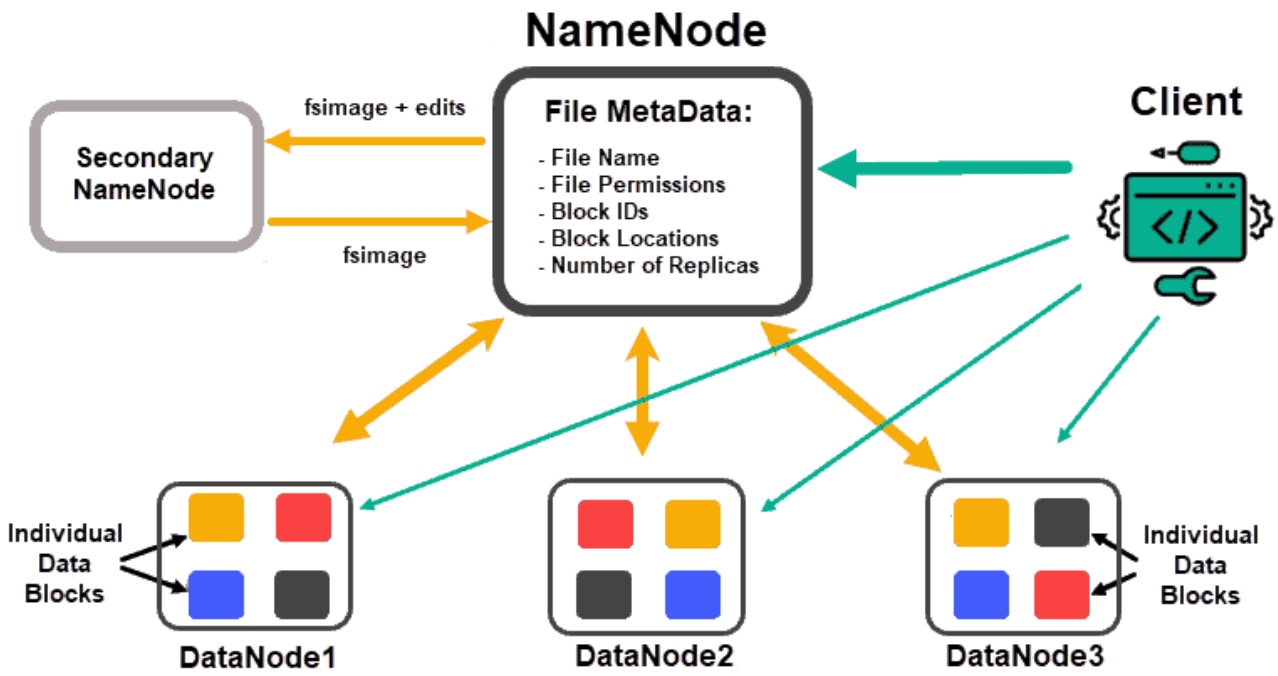
DataNodes

Secondary NameNode

Cliente HDFS

Es el **nodo maestro** del sistema.

Gestiona el **espacio de nombres del sistema de archivos**, manteniendo la jerarquía de directorios y los metadatos de todos los archivos y directorios (*permisos, propietario, ubicación de los bloques de datos*)



ARQUITECTURA DE HDFS

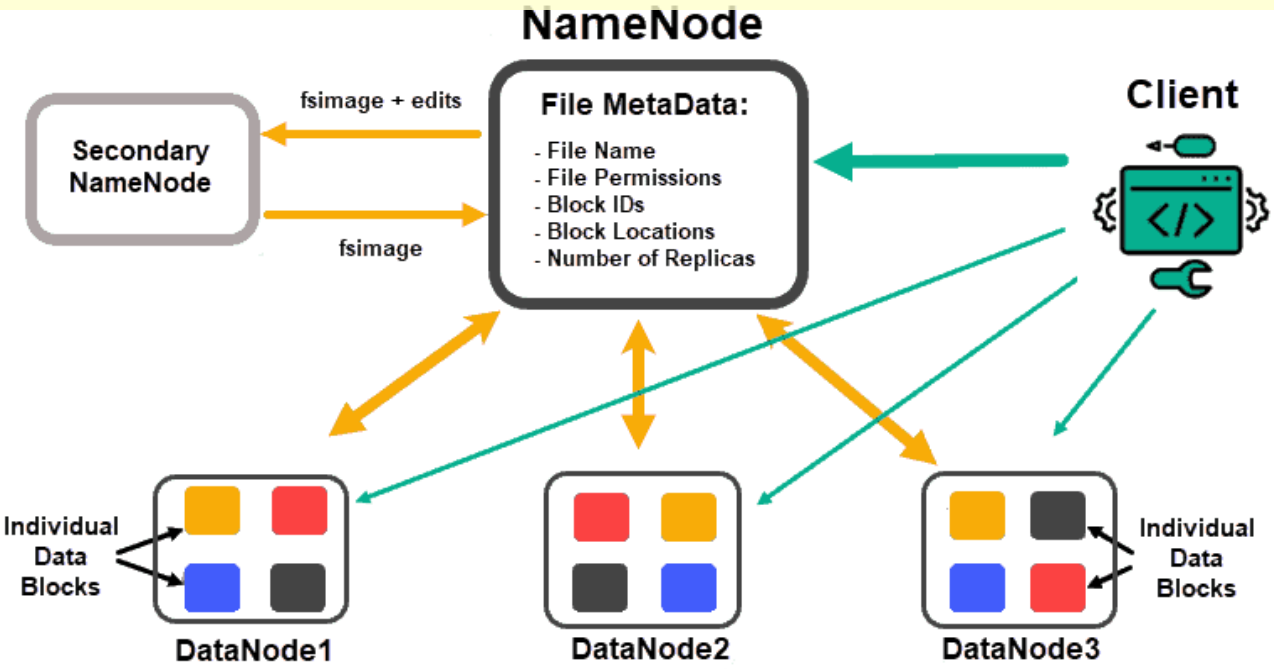
Víctor J. González

No almacena los datos, solo **almacena los metadatos** en RAM

Conoce en qué DataNodes se encuentran los bloques de cada archivo

Recibe peticiones de los clientes y les proporciona la ubicación de los bloques

Es un **punto único de fallo (SPOF)** en las versiones tradicionales de HDFS

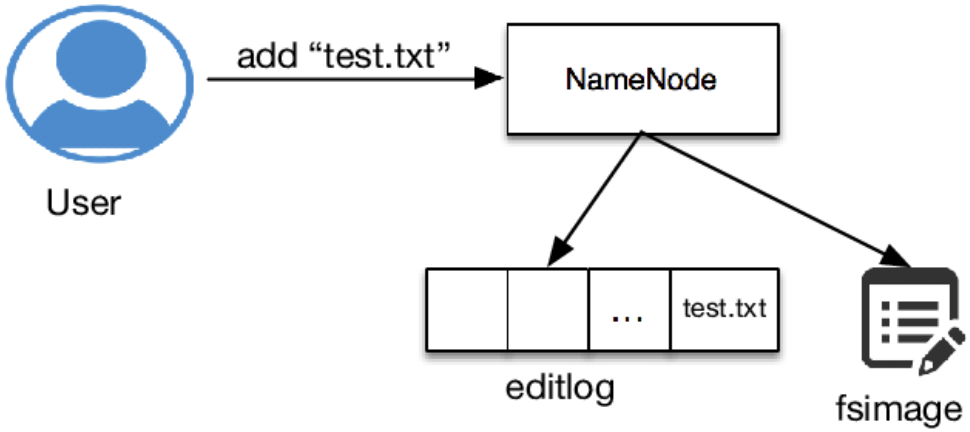


ARQUITECTURA DE HDFS

Víctor J. González

Dos elementos importantes son **fsimage** y **editlogs**:

- **fsimage**
  - Es una **instantánea** completa del sistema de archivos en un momento determinado
  - Contiene la **estructura de directorios y archivos**, así como las **ubicaciones** de los bloques en los DataNodes
  - Se almacena en el disco del NameNode y se carga en memoria al arrancar Hadoop



NameNode

DataNodes

Secondary NameNode

Cliente HDFS

- **editlogs**
  - Es un **registro de cambios** en el sistema de archivos HDFS desde la última actualización de *fsimage*
  - Registra todas las operaciones (creación, eliminación, …)
  - Si el NameNode falla y se reinicia, aplica estos logs sobre *fsimage* para recuperar el estado más reciente.

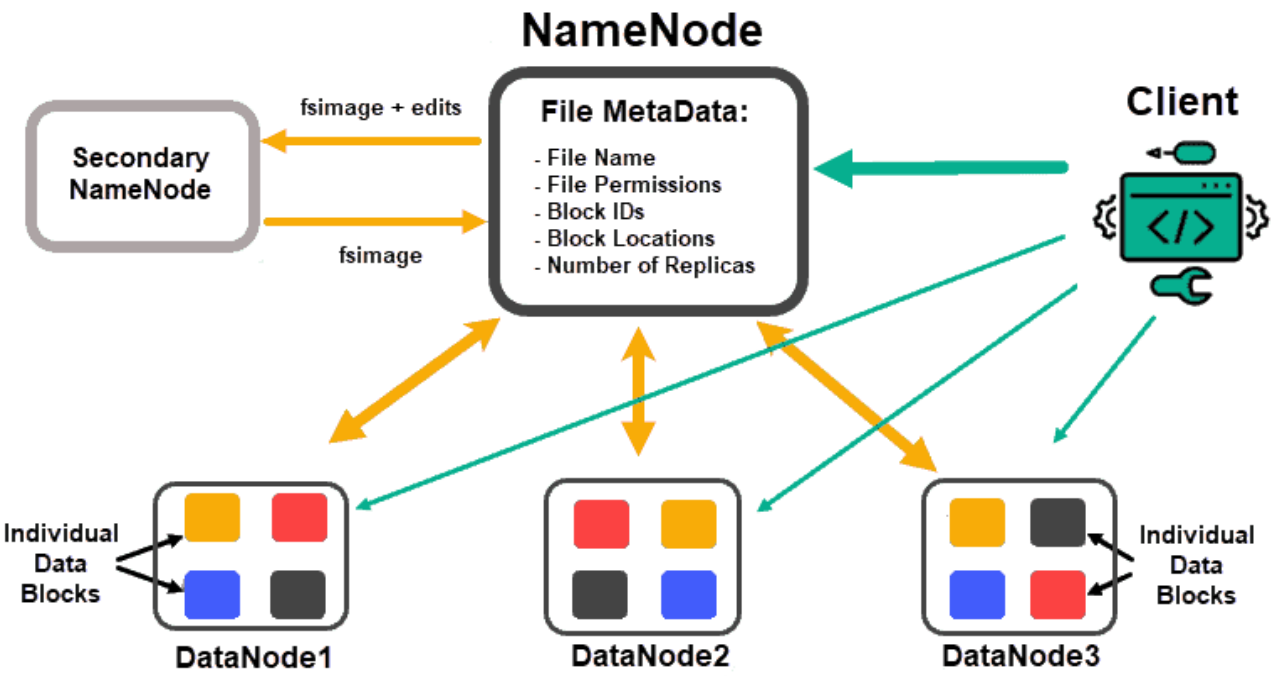
ARQUITECTURA DE HDFS

Víctor J. González

Son los **nodos esclavos**.

Cada *DataNode* gestiona un conjunto de bloques en su sistema de archivos local.

Reciben instrucciones del *NameNode* para **crear, eliminar y replicar bloques.**

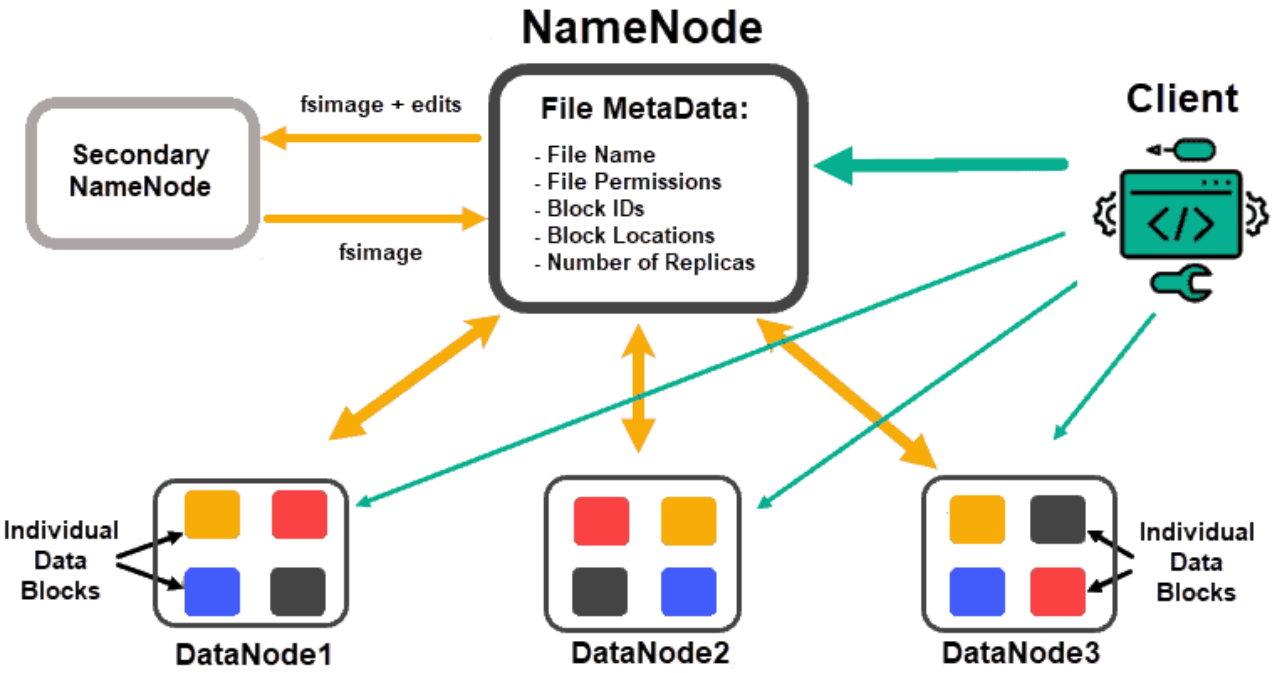


ARQUITECTURA DE HDFS

Víctor J. González

Envían periódicamente **informes de bloques y señales de vida (*heartbeats*)** al NameNode para informarle de su estado.

Hay dos conceptos importantes: **bloques** y **replicación**



ARQUITECTURA DE HDFS

Víctor J. González

Bloques:

- Define la cantidad mínima de datos que HDFS puede leer/escribir de una vez
- El valor por defecto es de 64 MB o 128 MB

Tamaño	Ventajas	Inconvenientes
Pequeño (<64 MB)	Ahorra espacio en archivos pequeños. Mayor paralelismo en archivos pequeños	Más metadatos Fragmentación en archivos grandes
Mediano	Optimizado para cargas de trabajo estándar	
Grande (>512 MB)	Menos metadatos Mejor rendimiento en archivos grandes	Menos paralelismo en el procesamiento Se pierde almacenamiento

NameNode

DataNodes

Secondary NameNode

Cliente HDFS

Replicación:

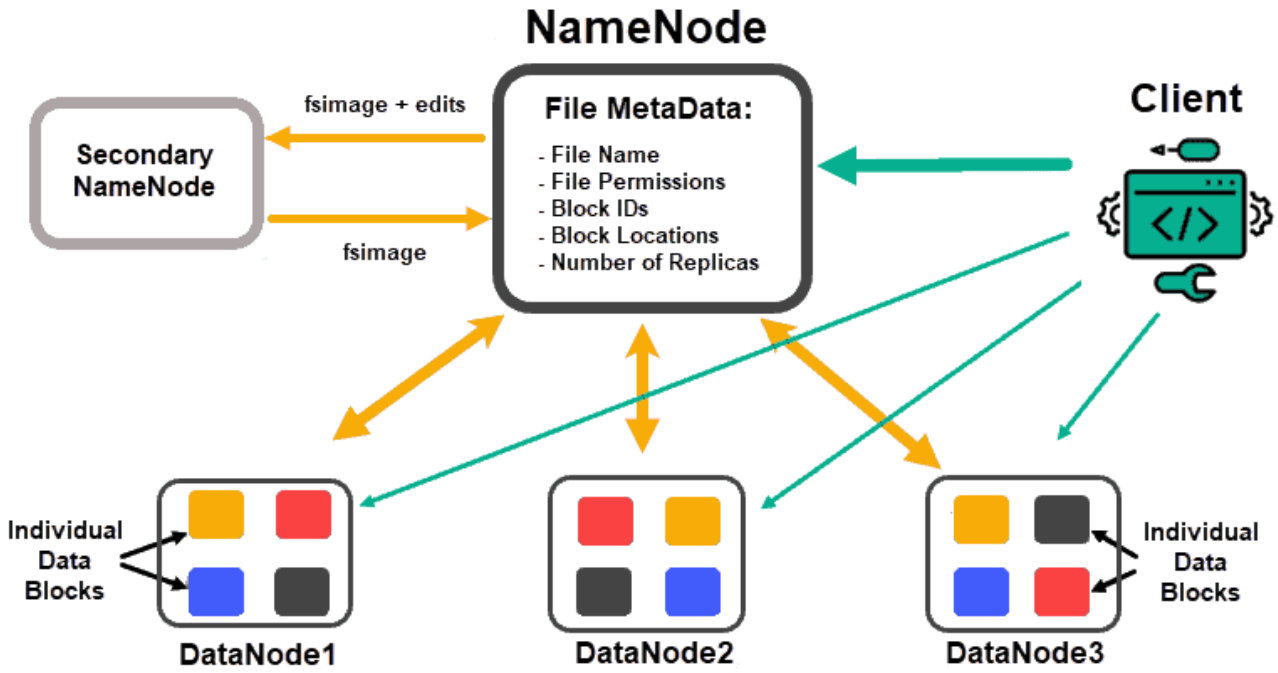
- Por defecto, HDFS tiene un factor de replicación de 3
- La política de replicación es ***rack aware***, de modo que tiene en cuenta si los nodos están en el mismo rack o no.
  - Intenta distribuir los bloques en varios racks para el caso de que falle un rack completo
  - Las transmisiones de datos son más rápidas entre nodos de un mismo rack que entre diferentes racks.

ARQUITECTURA DE HDFS

Víctor J. González

Su función principal es realizar **puntos de control (*checkpoints*)** del estado del *NameNode* para prevenir pérdida de datos.

Ayuda a reducir el tiempo de recuperación del NameNode en caso de fallo ya que puede cargar el último *checkpoint* en lugar de tener que procesar todos los EditLogs desde el inicio.



ARQUITECTURA DE HDFS

Víctor J. González

NameNode

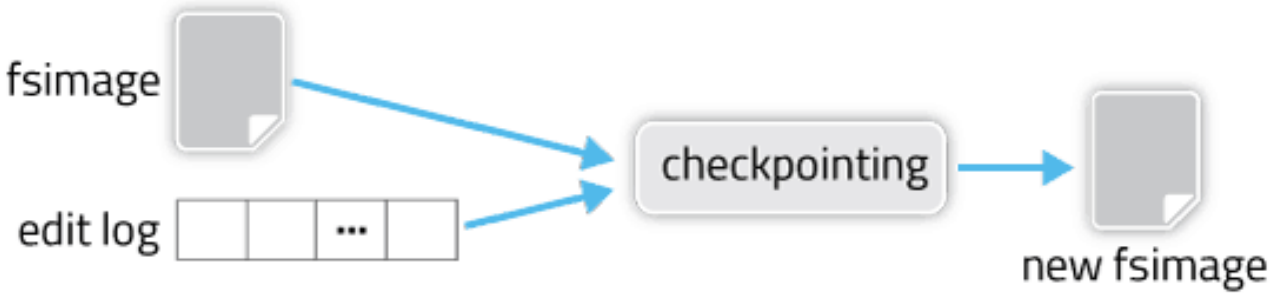
DataNodes

Secondary NameNode

Cliente HDFS

Cuando los *editlogs* crecen demasiado, el *Secondary NameNode* realiza un **checkpoint**

- Fusiona *fsimage* con *editLogs*
- Genera una nueva *fsimage* limpia y vacía los *editLogs*
- Transfiere la nueva *fsimage* al NameNode



ARQUITECTURA DE HDFS

Víctor J. González

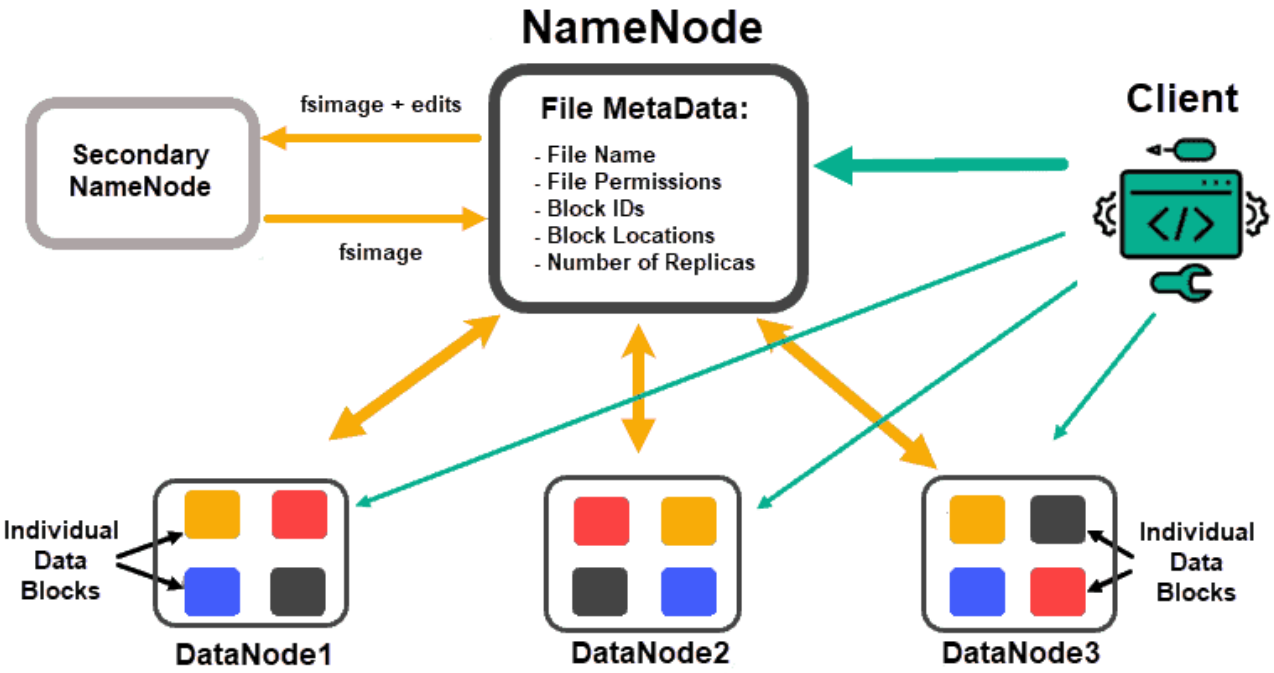
NameNode

DataNodes

Secondary NameNode

Cliente HDFS

En las arquitecturas de Alta Disponibilidad (HA), esta función se reemplaza por un ***Standy NameNode Activo***



ARQUITECTURA DE HDFS

Víctor J. González

NameNode

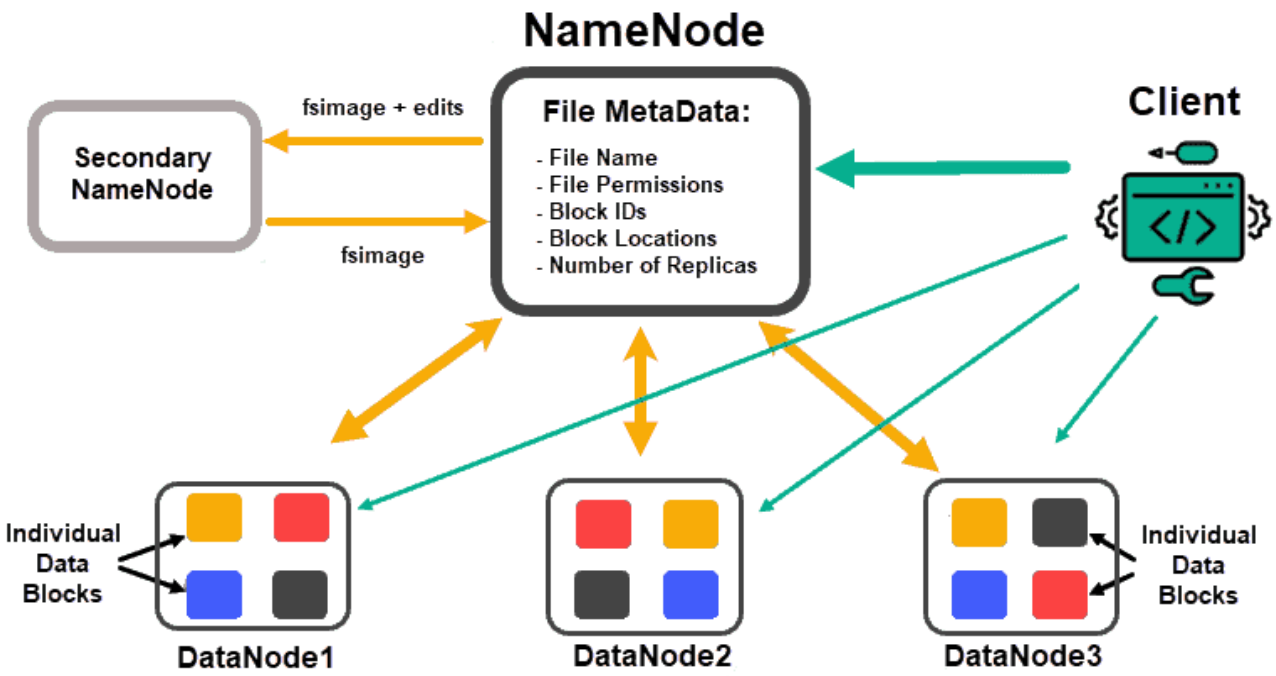
DataNodes

Secondary NameNode

Cliente HDFS

Es la **interfaz** que utilizan las aplicaciones para interactuar con el sistema de archivos HDFS.

Puede ser una **API** (como la de Java o Python) o herramientas de **línea de comandos**.



ARQUITECTURA DE HDFS

Víctor J. González

NameNode

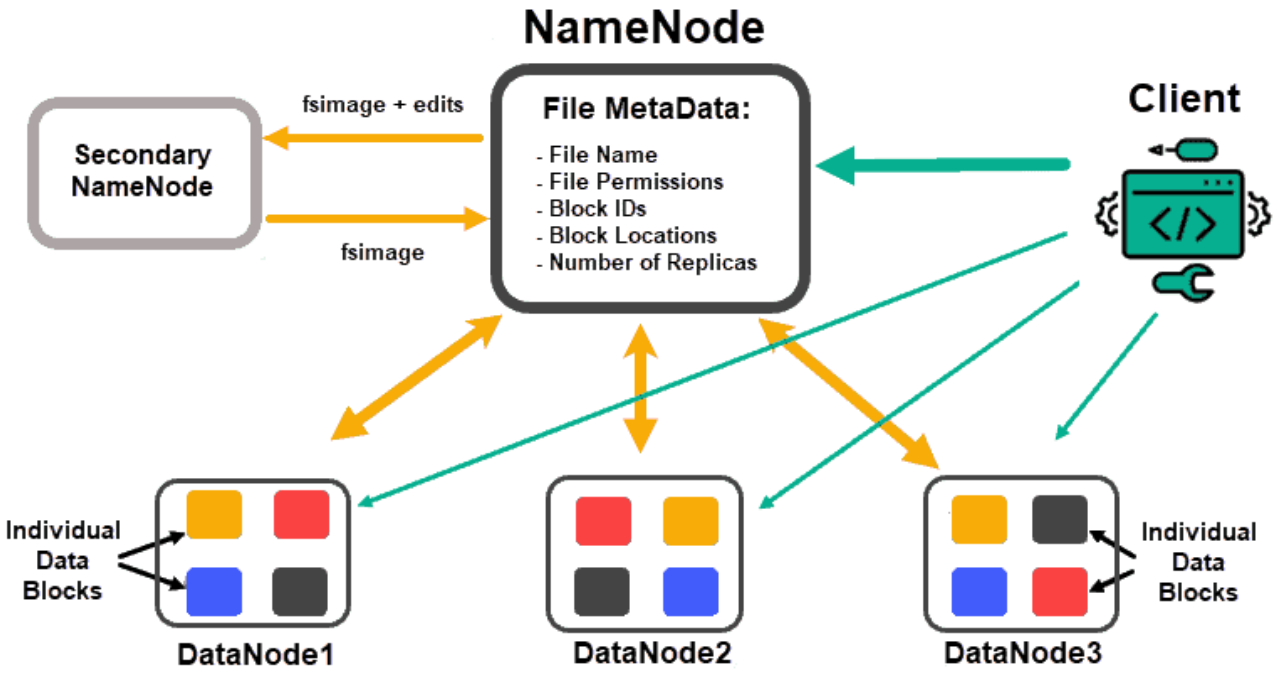
DataNodes

Secondary NameNode

Cliente HDFS

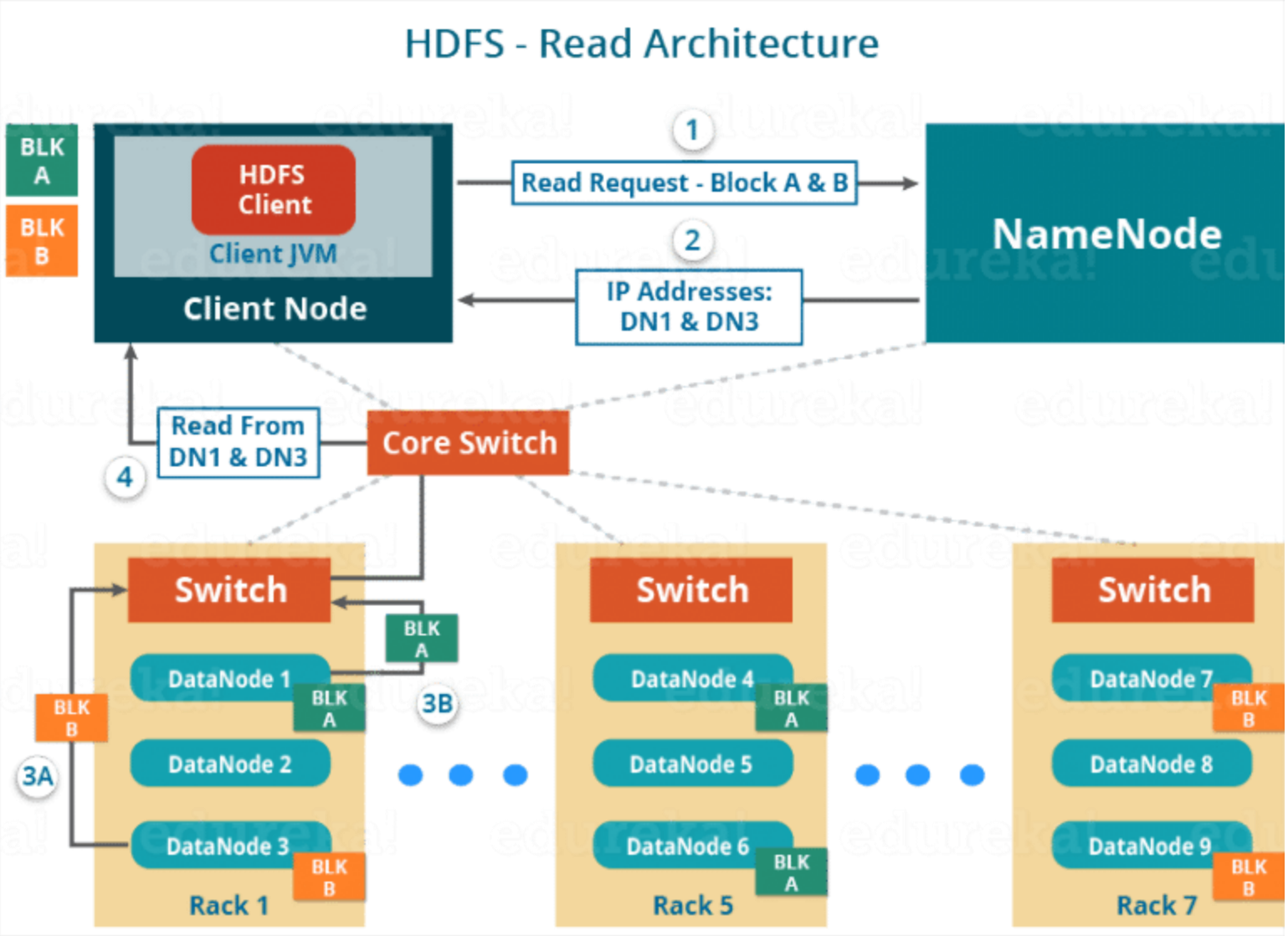
Cuando un cliente quiere **leer un archivo**, contacta al NameNode para obtener la ubicación de los bloques y luego se comunica directamente con los *DataNodes*.

Cuando un cliente quiere **escribir un archivo**, el NameNode le indica en qué DataNodes debe crear los bloques y el cliente envía los datos a esos DataNodes, que luego se encargan de la replicación.



ARQUITECTURA DE HDFS

Víctor J. González



2

INSTALACIÓN DE HADOOP Y HDFS

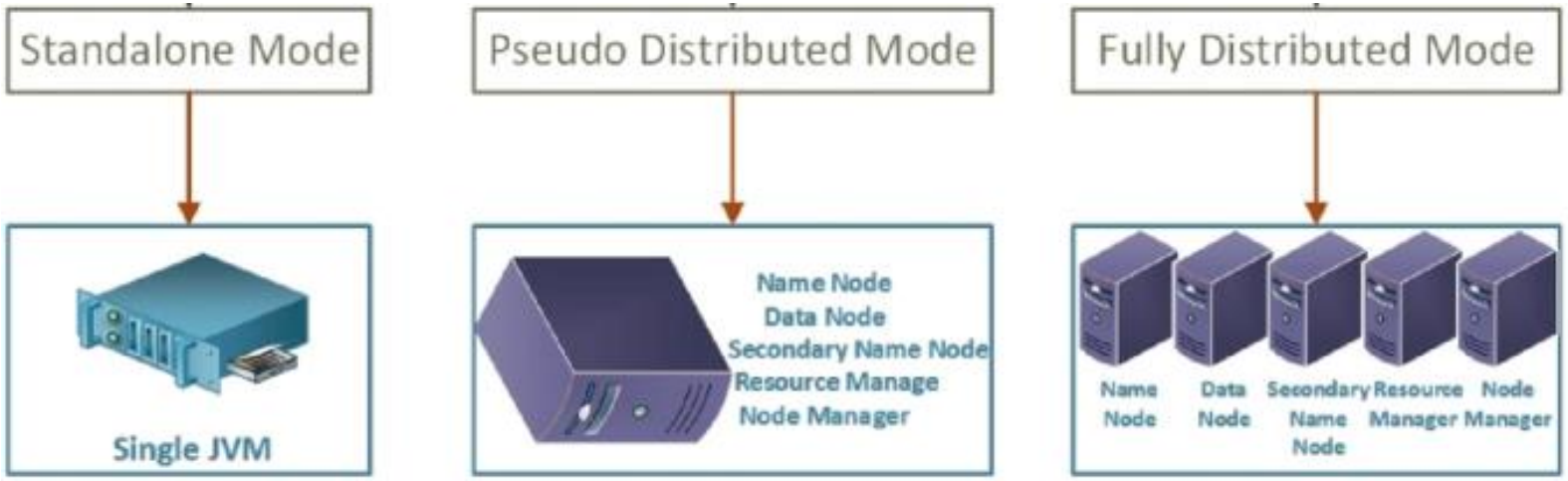
3 INSTALACIÓN DE HADOOP Y HDFS

3.1 INSTALACIÓN DE HADOOP EN MODO PSEUDO-DISTRIBUIDO

Hadoop se puede instalar en tres modos diferentes:

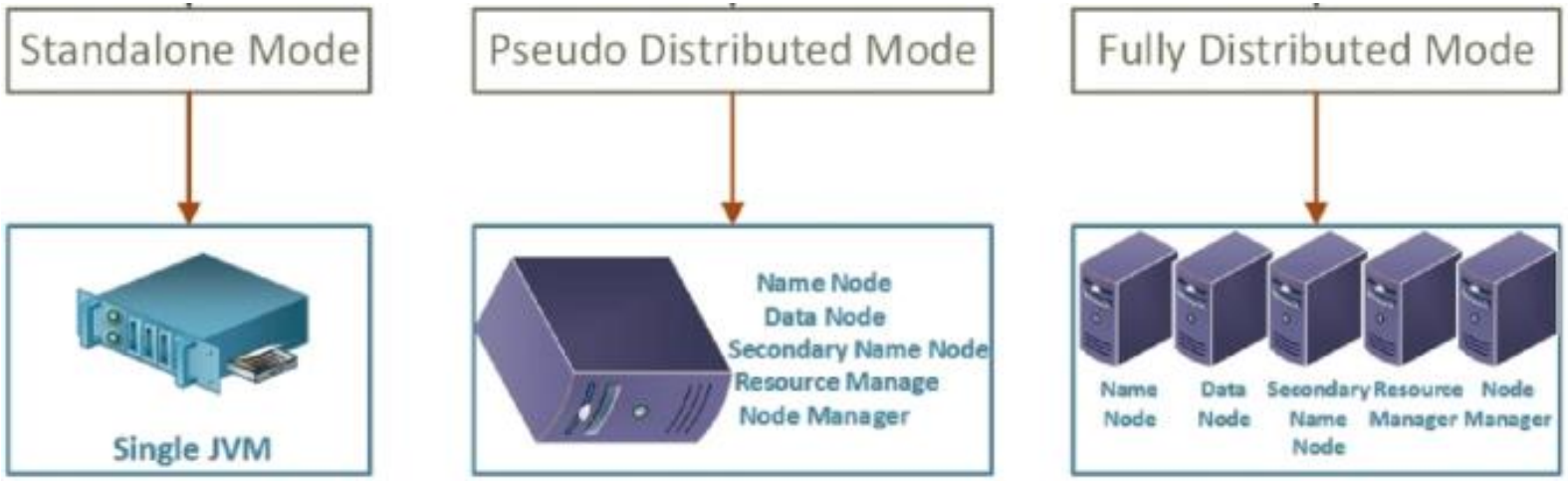
Standalone

- Hadoop se ejecuta como **un solo proceso Java** en una máquina, sin usar HDFS (usa el sistema de archivos local del SO) ni YARN.
- No hay demonios Hadoop (*NameNode*, *DataNode*, …)
- Se usa para desarrollo local y realizar pruebas de MapReduce sin necesidad de distribución.



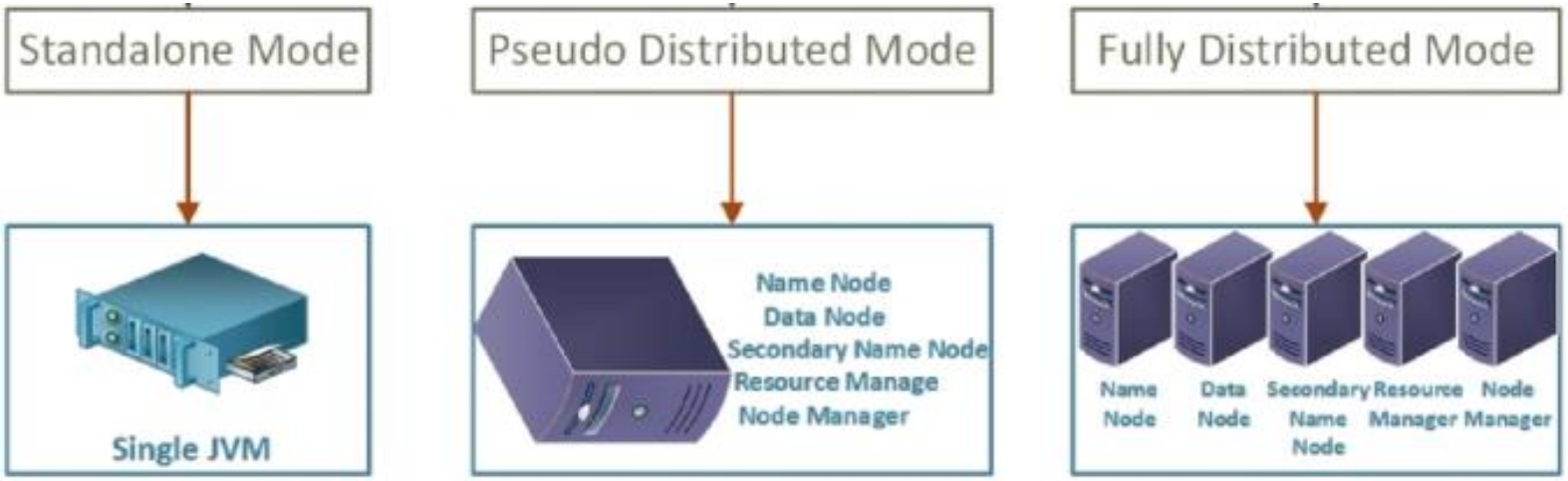
Pseudo-distribuido

- Hadoop se ejecuta en una única máquina, pero **simula un clúster real**.
- Todos los demonios (*NameNode*, *DataNode*, *ResourceManager* y *NodeManager*) se ejecutan en la misma máquina, pero como procesos separados.
- Útil para desarrollo y aprendizaje
- Se configura mediante **archivos XML** (core-site.xml, hdfs-site.xml, …)
- Se puede acceder a **interfaces web**



Distribuido

- Hadoop se ejecuta en múltiples máquinas conectadas en red, formando un **clúster real**.
- Hay nodos **maestros** (*Master*) y nodos **trabajadores** (*Workers*)
- Se usa en entornos de producción o pruebas a gran escala



PREPARACIÓN DE LAS MÁQUINAS VIRTUALES

Víctor J. González

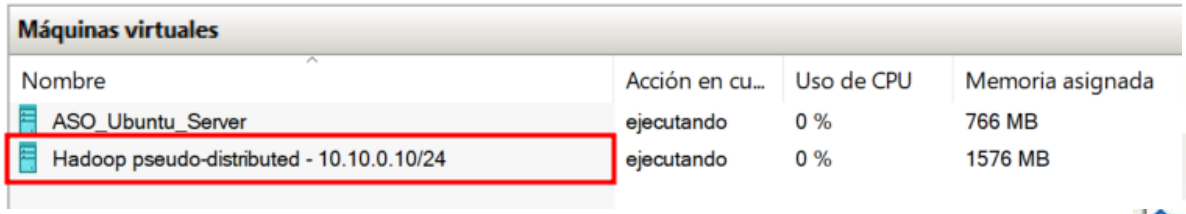
CARACTERÍSTICA	STANDALONE	PSEUDO-DISTRIBUTED	DISTRIBUTED
Número de máquinas	1	1	Múltiples (2 o más)
HDFS	No	Sí	Sí
YARN	No	Sí	Sí
Procesos separados	No	Sí (todos en una máquina)	Sí (distribuidos en nodos)
Uso de red	No	Sí (localhost)	Sí (entre nodos reales)
Propósito	Pruebas simples	Desarrollo / aprendizaje	Producción / alto rendimiento
Escalabilidad	Nula	Limitada	Alta
Tolerancia a fallos	No	No (un solo punto de fallo)	Sí (con replicación)
Configuración requerida	Mínima	Moderada	Compleja

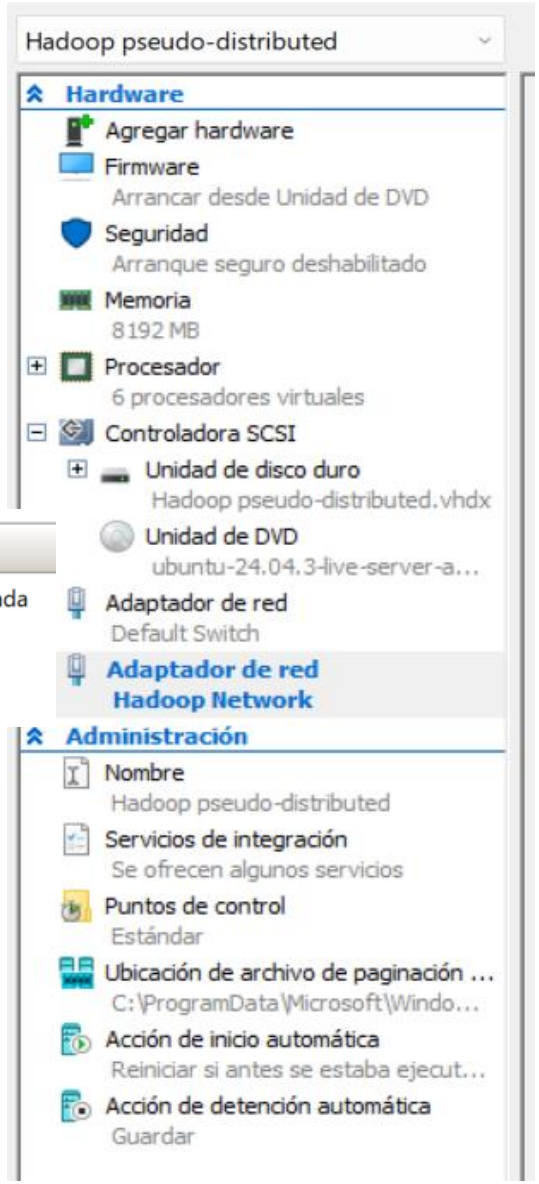
PREPARACIÓN DE LAS MÁQUINAS VIRTUALES

Víctor J. González

Creamos una máquina virtual y nos aseguramos de que tenga **dos adaptadores de red**, uno con salida a Internet (*Default switch*) y otro conectado a una red interna que interconectará todas nuestras máquinas virtuales.

En mi caso he instalado un Ubuntu 22.04





PREPARACIÓN DE LAS MÁQUINAS VIRTUALES

Víctor J. González

Arrancamos la máquina en modo *headless* y nos conectamos por SSH

```
PS C:\> ssh vgonzalez@10.0.0.1
vgonzalez@10.0.0.1's password:
Welcome to Ubuntu 22.04.2 LTS (GNU/Linux 5.15.0-134-generic x86_64)

 * Documentation:  https://help.ubuntu.com
 * Management:     https://landscape.canonical.com
 * Support:        https://ubuntu.com/advantage
```

PREPARACIÓN DE LAS MÁQUINAS VIRTUALES

Víctor J. González

La instalación de Hadoop requiere una serie de pasos:

- Crear un usuario y directorio
- Descargar Hadoop
- Instalar Hadoop
- Instalar las JDK de Java
- Comprobación de Hadoop
- Configurar SSH

CREACIÓN DEL USUARIO

Comenzamos creando un usuario llamado **hadoop**

```
vgonzalez@hadoop:/opt$ sudo adduser hadoop
Adding user `hadoop' ...
Adding new group `hadoop' (1001) ...
Adding new user `hadoop' (1001) with group `hadoop' ...
Creating home directory `/home/hadoop' ...
Copying files from `/etc/skel' ...
New password:
Retype new password:
passwd: password updated successfully
```

Agregamos al nuevo usuario al grupo de *sudoers*

```
victor@hdfs-server:/opt$ sudo usermod -a -G sudo hadoop
[sudo] password for victor:
victor@hdfs-server:/opt$
```

CREACIÓN DEL USUARIO

Debemos escoger un directorio para alojar todos los ficheros de Hadoop. Dado que el directorio **/opt** de Linux se suele utilizar para instalar aplicaciones opcionales (de terceros), crearemos el directorio **/opt/hadoop** y hacemos propietario al usuario **hadoop**

```
vgonzalez@hadoop:~$ cd /opt
vgonzalez@hadoop:/opt$ sudo mkdir hadoop
vgonzalez@hadoop:/opt$ sudo chown hadoop /opt/hadoop
```

Por último, comprobamos que el usuario tenga permisos sobre dicho fichero

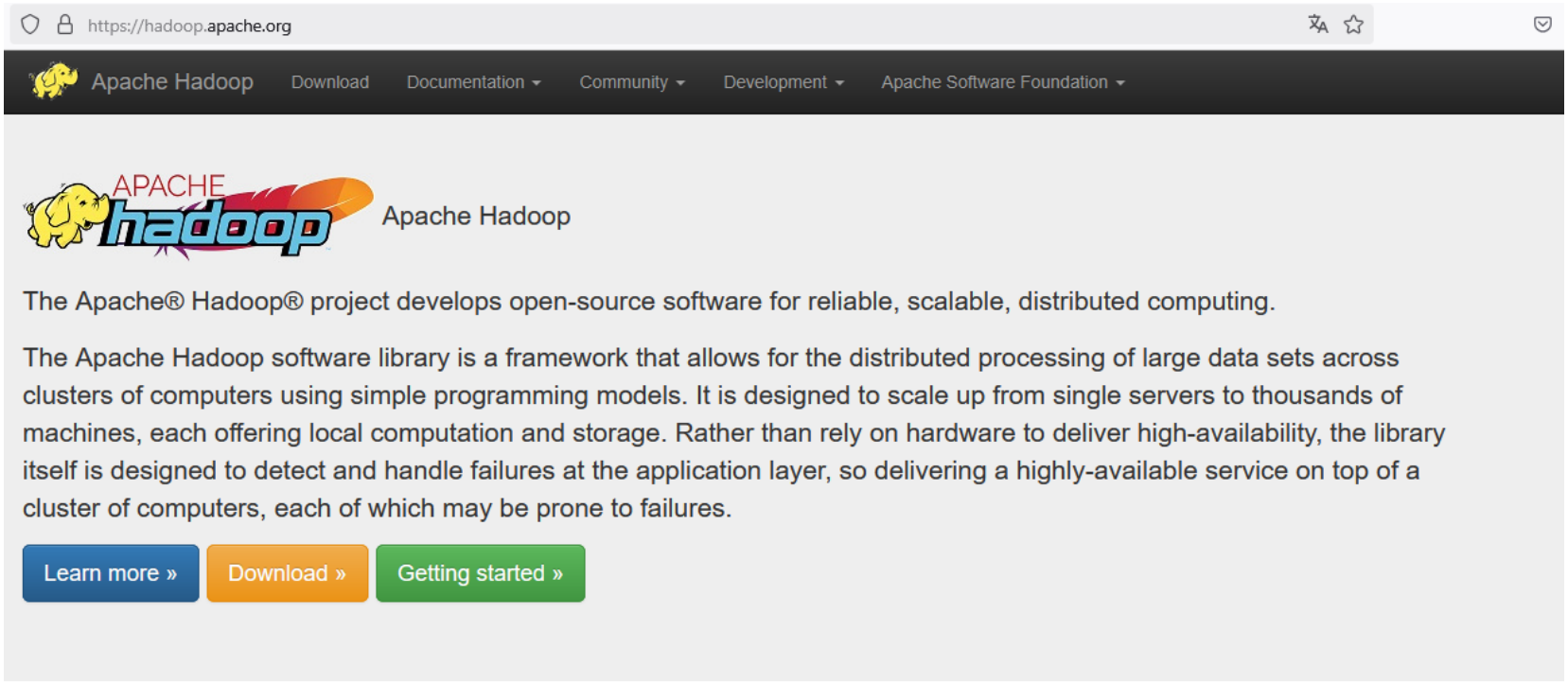
```
vgonzalez@hadoop:/opt$ su hadoop
Password:
hadoop@hadoop:/opt$ cd hadoop
hadoop@hadoop:/opt/hadoop$ touch file
hadoop@hadoop:/opt/hadoop$ ls
file
```

A partir de aquí, ya trabajaremos con el usuario **hadoop**

DESCARGA DE APACHE HADOOP

Víctor J. González

Para descargar Hadoop debemos ir a https://hadoop.apache.org/



DESCARGA DE APACHE HADOOP

Víctor J. González

En el apartado *Download* vemos que hay disponibles diversas versiones (en mi caso escojo la 3.4.1)

Download

Hadoop is released as source code tarballs with corresponding binary tarballs for convenience. The downloads are distributed via mirror sites and should be checked for tampering using GPG or SHA-512.

Version	Release date	Source download	Binary download	Release notes
3.4.1	2024 Oct 18	source (checksum signature)	binary (checksum signature) binary-aarch64 (checksum signature) binary-lean (checksum signature)	Announcement
3.4.0	2024 Mar 17	source (checksum signature)	binary (binary-(	
3.3.6	2023 Jun 23	source (checksum signature)	binary (binary-(	
2.10.2	2022 May 31	source (checksum signature)	binary (	

Comienzo descargando el fichero con el SHA (**checksum**) para comprobar posteriormente la integridad del fichero descargado.

Copio el enlace del mismo.

DESCARGA DE APACHE HADOOP

Víctor J. González

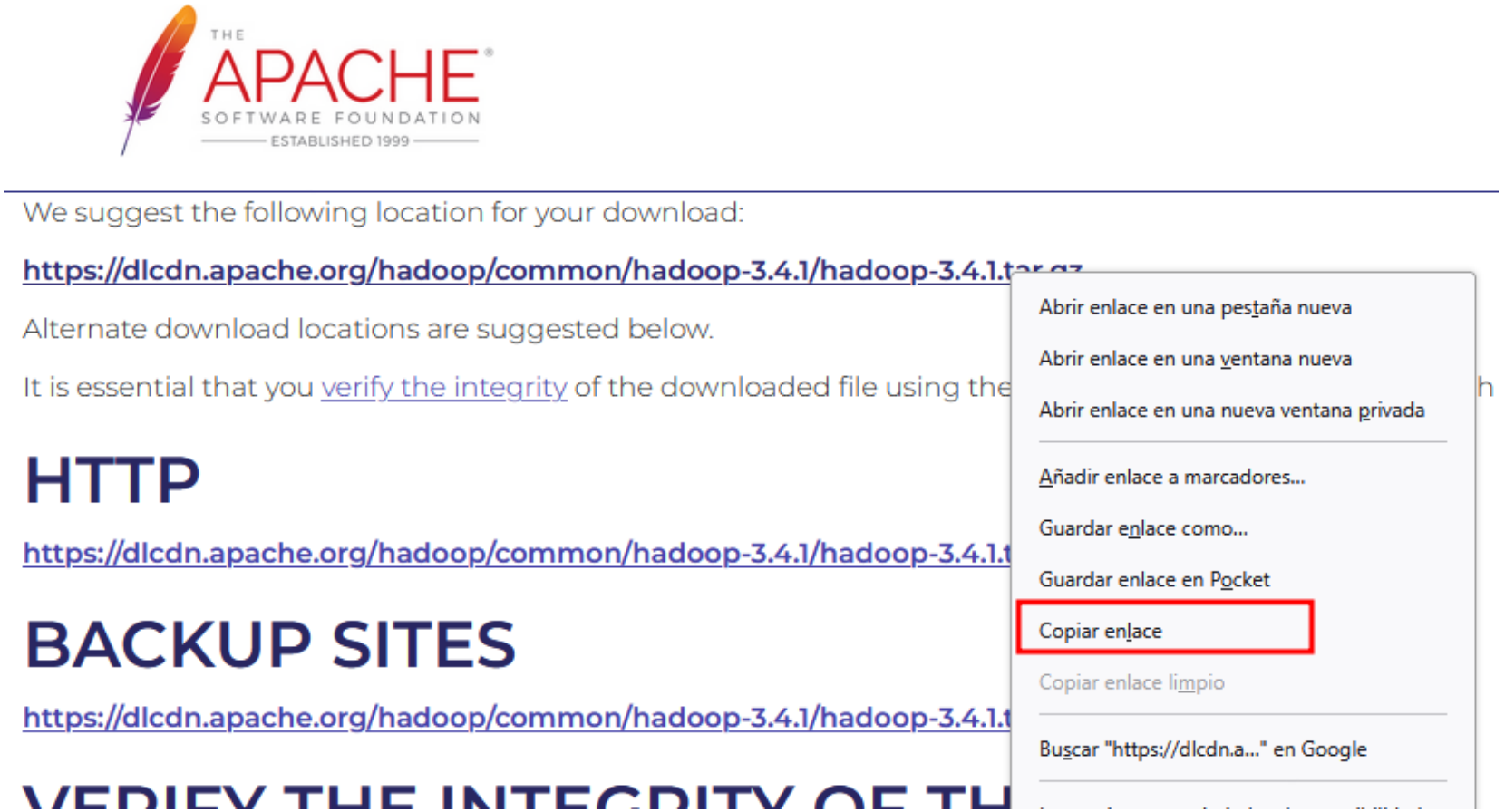
Y lo descargo usando el comando **wget**

```
hadoop@hadoop:~$ wget https://downloads.apache.org/hadoop/common/hadoop-3.4.1/hadoop-3.4.1.tar.gz.sha512
--2025-03-27 17:49:57--  https://downloads.apache.org/hadoop/common/hadoop-3.4.1/hadoop-3.4.1.tar.gz.sha512
Resolving downloads.apache.org (downloads.apache.org)... 88.99.208.237, 135.181.214.104, 2a01:4f8:10a:39da::2, ...
Connecting to downloads.apache.org (downloads.apache.org)|88.99.208.237|:443... connected.
HTTP request sent, awaiting response... 200 OK
Length: 160 [text/plain]
Saving to: ‘hadoop-3.4.1.tar.gz.sha512’

hadoop-3.4.1.tar.gz.sha512                    100%[=============================================================

2025-03-27 17:49:57 (95,2 MB/s) - ‘hadoop-3.4.1.tar.gz.sha512’ saved [160/160]

hadoop@hadoop:~$ ls -l
total 4
-rw-rw-r-- 1 hadoop hadoop 160 oct 10 10:53 hadoop-3.4.1.tar.gz.sha512
```

DESCARGA DE APACHE HADOOP

Víctor J. González

Y ahora descargo los binarios de la opción elegida. Repito el mismo proceso copiando primero la URL del fichero



DESCARGA DE APACHE HADOOP

Víctor J. González

Y lo descargo con **wget**.

```
hadoop@hadoop:~$ wget https://dlcdn.apache.org/hadoop/common/hadoop-3.4.1/hadoop-3.4.1.tar.gz
--2025-03-27 17:51:22--  https://dlcdn.apache.org/hadoop/common/hadoop-3.4.1/hadoop-3.4.1.tar.gz
Resolving dlcdn.apache.org (dlcdn.apache.org)... 151.101.2.132, 2a04:4e42::644
Connecting to dlcdn.apache.org (dlcdn.apache.org)|151.101.2.132|:443... connected.
HTTP request sent, awaiting response... 200 OK
Length: 974002355 (929M) [application/x-gzip]
Saving to: ‘hadoop-3.4.1.tar.gz’

hadoop-3.4.1.tar.gz                                 100%[=======================================================

2025-03-27 17:51:52 (58,3 MB/s) - ‘hadoop-3.4.1.tar.gz’ saved [974002355/974002355]
```

De esta forma, tendré en el mismo directorio el fichero descargado y el fichero con su SHA 512

```
hadoop@hadoop:~$ ls -l
total 951184
-rw-rw-r-- 1 hadoop hadoop 974002355 oct 10 10:53 hadoop-3.4.1.tar.gz
-rw-rw-r-- 1 hadoop hadoop       160 oct 10 10:53 hadoop-3.4.1.tar.gz.sha512
```

DESCARGA DE APACHE HADOOP

Vamos a comprobar ahora la **integridad** del fichero.

Para ello usamos el comando **shasum** con los siguientes modificadores:

- **-a**: algoritmo utilizado (en este caso es SHA 512)
- **-c:** para verificar la suma

```
hadoop@hadoop:~$ shasum -a 512 -c hadoop-3.4.1.tar.gz.sha512
hadoop-3.4.1.tar.gz: OK
```

Si nos devuelve OK quiere decir que el fichero que he hemos descargado no ha sufrido ningún tipo de modificación (datos corruptos, malware, …)

Esta comprobación es recomendable al instalar un servidor de pruebas e imprescindible si estamos preparando un servidor para producción,

Una vez descargado nos queda descomprimir el fichero en el directorio /opt/hadoop

```
hadoop@hadoop:~$ tar xvf /home/hadoop/hadoop-3.4.1.tar.gz \
> --strip-components=1 `
> -C /opt/hadoop
```

El fichero que descargamos

Por defecto, al descomprimir creará un directorio y, dentro de él, todos los directorios de Hadoop. Con **strip-components=1** le indicamos que no cree este directorio

Con **–C** le indicamos el directorio donde queremos descomprimir los datos. En nuestro caso en /opt/hadoop

Si vamos a /opt/hadoop vemos que ha descomprimido un montón de ficheros y directorios.

```
hadoop@hadoop:/opt/hadoop$ ls -l
total 108
drwxr-xr-x 2 hadoop hadoop  4096 oct  9 16:36 bin
drwxr-xr-x 3 hadoop hadoop  4096 oct  9 14:59 etc
-rw-rw-r-- 1 hadoop hadoop     0 mar 27 17:43 file
drwxr-xr-x 2 hadoop hadoop  4096 oct  9 16:36 include
drwxr-xr-x 3 hadoop hadoop  4096 oct  9 16:36 lib
drwxr-xr-x 4 hadoop hadoop  4096 oct  9 16:36 libexec
-rw-rw-r-- 1 hadoop hadoop 23759 sep 16  2024 LICENSE-binary
drwxr-xr-x 2 hadoop hadoop  4096 oct  9 16:36 licenses-binary
-rw-rw-r-- 1 hadoop hadoop 15696 jul 15  2024 LICENSE.txt
-rw-rw-r-- 1 hadoop hadoop 27165 jul 15  2024 NOTICE-binary
-rw-rw-r-- 1 hadoop hadoop  1541 jul 15  2024 NOTICE.txt
-rw-rw-r-- 1 hadoop hadoop   175 jul 15  2024 README.txt
drwxr-xr-x 3 hadoop hadoop  4096 oct  9 14:59 sbin
drwxr-xr-x 4 hadoop hadoop  4096 oct  9 17:09 share
```

INSTALACIÓN DE APACHE HADOOP

Víctor J. González

Veamos que contienen todos los directorios que se han generado.

El directorio **bin** contiene los ejecutables que necesita Hadoop para funcionar

```
hadoop@hadoop:/opt/hadoop$ ls bin -l
total 2040
-rwxr-xr-x 1 hadoop hadoop 957016 oct  9 16:00 container-executor
-rwxr-xr-x 1 hadoop hadoop   9201 oct  9 14:57 hadoop
-rwxr-xr-x 1 hadoop hadoop  11114 oct  9 14:57 hadoop.cmd
-rwxr-xr-x 1 hadoop hadoop  11730 oct  9 15:03 hdfs
-rwxr-xr-x 1 hadoop hadoop   8566 oct  9 15:03 hdfs.cmd
-rwxr-xr-x 1 hadoop hadoop   6586 oct  9 16:03 mapred
-rwxr-xr-x 1 hadoop hadoop   6311 oct  9 16:03 mapred.cmd
-rwxr-xr-x 1 hadoop hadoop  35008 oct  9 16:00 oom-listener
-rwxr-xr-x 1 hadoop hadoop 993152 oct  9 16:00 test-container-executor
-rwxr-xr-x 1 hadoop hadoop  13208 oct  9 16:00 yarn
-rwxr-xr-x 1 hadoop hadoop  13321 oct  9 16:00 yarn.cmd
```

El comando principal de Hadoop, permite interactuar con todos sus componentes

El comando para interactuar con el sistema de archivos de Hadoop (HDFS)

Comando para interactuar con MapReduce, el motor de procesamiento por lotes clásico de Hadoop

Con el comando yarn podremos gestionar YARN (Yet Another Resource Manager), el gestor de recursos y ejecución de aplicaciones de Hadoop

En el directorio **etc** está la configuración

core.site.xml contiene la configuración global de Hadoop

mapred-site.xml para la configuración de MapReduce (framework de ejecución, historial de Jobs, …)

```
hadoop@hadoop:/opt/hadoop$ ls etc/hadoop/
capacity-scheduler.xml            httpfs-env.sh              mapred-site.xml
configuration.xsl                 httpfs-log4j.properties    shellprofile.d
container-executor.cfg            httpfs-site.xml            ssl-client.xml.example
core-site.xml                     kms-acls.xml               ssl-server.xml.example
hadoop-env.cmd                    kms-env.sh                 user_ec_policies.xml.template
hadoop-env.sh                     kms-log4j.properties       workers
hadoop-metrics2.properties        kms-site.xml               yarn-env.cmd
hadoop-policy.xml                 log4j.properties           yarn-env.sh
hadoop-user-functions.sh.example  mapred-env.cmd             yarnservice-log4j.properties
hdfs-rbf-site.xml                 mapred-env.sh              yarn-site.xml
hdfs-site.xml                     mapred-queues.xml.template
```

yarn-site.xml almacena la configuración de YARN

hdfs.site.xml lo utilizaremos para la configuración especificad e HDFS (NameNode, DataNode, réplicas, …)

workers contiene la lista de nodos esclavos

INSTALACIÓN DE APACHE HADOOP

Víctor J. González

El directorio **sbin** contiene **scripts**, por ejemplo, para arrancar y parar diversos servicios

```
hadoop@hadoop:/opt/hadoop$ ls sbin
distribute-exclude.sh    start-all.sh          stop-balancer.sh
FederationStateStore     start-balancer.sh     stop-dfs.cmd
hadoop-daemon.sh         start-dfs.cmd         stop-dfs.sh
hadoop-daemons.sh        start-dfs.sh          stop-secure-dns.sh
httpfs.sh                start-secure-dns.sh   stop-yarn.cmd
kms.sh                   start-yarn.cmd        stop-yarn.sh
mr-jobhistory-daemon.sh  start-yarn.sh         workers.sh
refresh-namenodes.sh     stop-all.cmd          yarn-daemon.sh
start-all.cmd            stop-all.sh           yarn-daemons.sh
```

Los directorios **include**, **lib** y **libexec** se utilizarían si queremos compilar el código optimizado para la máquina.

El directorio **share** contiene las **librerías** de Hadoop

INSTALACIÓN DE JAVA JDK

Víctor J. González

Hay dos opciones:

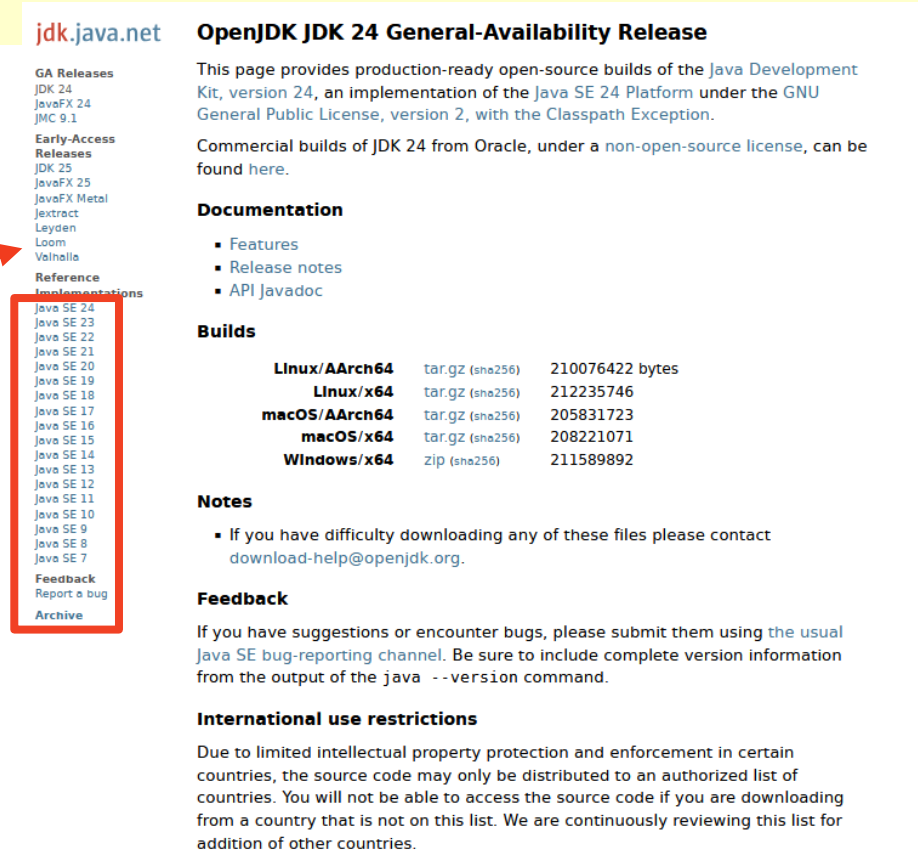
- OpenJDK (https://openjdk.org/)
- Oracle JDK (https://www.oracle.com/es/java/technologies/downloads/)

Problema: a partir de la versión 8 de Oracle Java ha cambiado el licenciamiento de forma que solo es gratuita para uso personal.

Así que optaremos por utilizar **OpenJDK**

Si vamos a la documentación de Hadoop veremos que Hadoop solo soporta Java 8 y Java 11.

En nuestro caso vamos a descargar la versión **Java 8**. Para descargarlo vamos a https://jdk.java.net/



INSTALACIÓN DE JAVA JDK

Víctor J. González

jdk.java.net

GA Releases
JDK 24
JavaFX 24
JMC 9.1

Early-Access Releases
JDK 25
JavaFX 25
JavaFX Metal
Jextract
Leyden
Loom
Valhalla

Reference Implementations
Java SE 24
Java SE 23
Java SE 22
Java SE 21
Java SE 20
Java SE 19
Java SE 18
Java SE 17
Java SE 16
Java SE 15
Java SE 14
Java SE 13
Java SE 12
Java SE 11
Java SE 10
Java SE 9
Java SE 8
Java SE 7

Feedback
Report a bug

Archive

Java Platform, Standard Edition 8 Reference Implementations

The official Reference Implementations for Java SE 8 (JSR 337) are based solely upon open-source code available from the JDK 8 Project in the OpenJDK Community. This Reference Implementation applies to JSR 337 Maintenance Release 6 (July 2024). Reference Implementation for Maintenance Release 1 (Mar 2015), Maintenance Release 2 (Mar 2019), Maintenance Release 3 (Feb 2020), Maintenance Release 4 (Jul 2022), and Maintenance Release 5 (Jul 2023) contains RIs for these releases.

These binaries are for reference use only!

These binaries are provided for use by implementers of the Java SE 8 Platform Specification and are for reference purposes only. These Reference Implementations have been approved through the Java Community Process. Binaries for development and production will be available from Oracle and in most popular Linux distributions.

RI Binaries (build 1.8.0_44-b02) under the GNU General Public License version 2

- Oracle Linux 8.6 x64 Java Development Kit (sha256) 167 MB
- ~~Windows 11 i586 Java Development Kit (sha256) 92 MB~~
- Oracle Linux 8.6 i586 Java Runtime Environment for Compact Profiles
  - Compact Profile 1 (sha256) 15 MB
  - Compact Profile 2 (sha256) 18 MB
  - Compact Profile 3 (sha256) 20 MB
  - Full JRE (sha256) 40 MB

RI Source Code

The source code of the RI binaries is available under the GPLv2 in a single zip file (sha256) 123 MB.

INSTALACIÓN DE JAVA JDK

Víctor J. González

Al igual que hicimos con Hadoop, copiamos el enlace al JDK y lo descargamos en la máquina virtual

RI Binaries (build 1.8.0_44-b02) under the GNU General Public License version 2

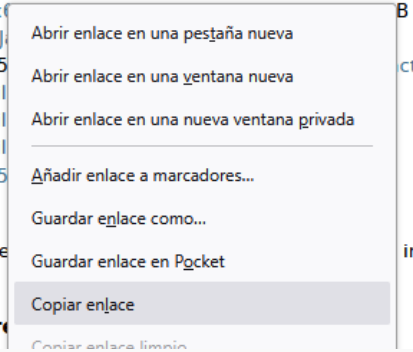
- Oracle Linux 8.6 x
- Windows 11 i586 J
- Oracle Linux 8.6 i5
  - Compact Profil
  - Compact Profil
  - Compact Profil
  - Full JRE (sha25

RI Source Code

The source code of the in a single zip file (sha256) 123 MB.

International use r

Due to limi




```
hadoop@hadoop:~$ wget https://download.java.net/openjdk/jdk8u44/ri/openjdk-8u44-linux-x64.tar.gz
--2025-03-27 18:38:05--  https://download.java.net/openjdk/jdk8u44/ri/openjdk-8u44-linux-x64.tar.gz
Resolving download.java.net (download.java.net)... 2.20.100.121
Connecting to download.java.net (download.java.net)|2.20.100.121|:443... connected.
HTTP request sent, awaiting response... 200 OK
Length: 216015848 (206M) [application/x-gzip]
Saving to: ‘openjdk-8u44-linux-x64.tar.gz’

openjdk-8u44-linux-x64.tar.gz    53%[===========================>                    ] 109,98M  46,6MB/s
```

```
hadoop@hadoop:~$ ls -l
total 210956
-rw-rw-r-- 1 hadoop hadoop 216015848 may 22  2024 openjdk-8u44-linux-x64.tar.gz
```

INSTALACIÓN DE JAVA JDK

Java se puede instalar a **nivel global** para todos los usuarios y a **nivel local** para un único usuario.

Dado que es una versión muy antigua de Java lo propio será instalarlo únicamente para el usuario hadoop para no interferir con otros usuarios.

Lo descomprimimos en el directorio **/opt/hadoop**

```
hadoop@hadoop:~$ tar xvf openjdk-8u44-linux-x64.tar.gz -C /opt/hadoop/
```

```
hadoop@hadoop:~$ ls /opt/hadoop/
bin   include          libexec          LICENSE.txt    README.txt
etc   java-se-8u44-ri  LICENSE-binary   NOTICE-binary  sbin
file  lib              licenses-binary  NOTICE.txt     share
```

INSTALACIÓN DE JAVA JDK

Como tiene un nombre muy largo lo **renombramos** para que sea más cómodo trabajar con ese directorio

```
hadoop@hadoop:~$ cd /opt/hadoop/
hadoop@hadoop:/opt/hadoop$ mv ./java-se-8u44-ri/ jdk
hadoop@hadoop:/opt/hadoop$ ls -l
total 112
drwxr-xr-x 2 hadoop hadoop  4096 oct  9 16:36 bin
drwxr-xr-x 3 hadoop hadoop  4096 oct  9 14:59 etc
-rw-rw-r-- 1 hadoop hadoop     0 mar 27 17:43 file
drwxr-xr-x 2 hadoop hadoop  4096 oct  9 16:36 include
drwxr-xr-x 9 hadoop hadoop  4096 may 22  2024 jdk
drwxr-xr-x 3 hadoop hadoop  4096 oct  9 16:36 lib
drwxr-xr-x 4 hadoop hadoop  4096 oct  9 16:36 libexec
-rw-rw-r-- 1 hadoop hadoop 23759 sep 16  2024 LICENSE-binary
```

INSTALACIÓN DE JAVA JDK

Comprobamos que Java funciona correctamente y que es la versión que deseamos.

```
hadoop@hadoop:/opt/hadoop$ cd jdk/bin/
hadoop@hadoop:/opt/hadoop/jdk/bin$ ./java -version
openjdk version "1.8.0_44"
OpenJDK Runtime Environment (build 1.8.0_44-b02)
OpenJDK 64-Bit Server VM (build 25.40-b25, mixed mode)
```

INSTALACIÓN DE JAVA JDK

Víctor J. González

Ahora necesitamos configurar algunas **variables de entorno** para poder ejecutar directamente tanto Hadoop como Java sin necesidad de indicar la ruta completa del ejecutable.

Para ello, vamos a añadir las siguientes líneas al fichero **.bashrc**

```
## HADOOP
export HADOOP_HOME=/opt/hadoop
export JAVA_HOME=/opt/hadoop/jdk
export HADOOP_MAPRED_HOME=${HADOOP_HOME}
export HADOOP_COMMON_HOME=${HADOOP_HOME}
export HADOOP_HDFS_HOME=${HADOOP_HOME}
export YARN_HOME=${HADOOP_HOME}
export PATH=$PATH:$HADOOP_HOME/bin:$HADOOP_HOME/sbin:$JAVA_HOME/bin
```

INSTALACIÓN DE JAVA JDK

Y recargamos el fichero para que se apliquen los cambios

```
hadoop@hadoop:~$ . ./.bashrc
hadoop@hadoop:~$ hadoop version
Hadoop 3.4.1
Source code repository https://github.com/apache/hadoop.git -r 4d7825309348956336b8
Compiled by mthakur on 2024-10-09T14:57Z
Compiled on platform linux-x86_64
Compiled with protoc 3.23.4
From source with checksum 7292fe9dba5e2e44e3a9f763fce3e680
This command was run using /opt/hadoop/share/hadoop/common/hadoop-common-3.4.1.jar
hadoop@hadoop:~$ java -version
openjdk version "1.8.0_44"
OpenJDK Runtime Environment (build 1.8.0_44-b02)
OpenJDK 64-Bit Server VM (build 25.40-b25, mixed mode)
```

INSTALACIÓN DE JAVA JDK

También tenemos que añadir la siguiente línea en el fichero `/opt/hadoop/etc/hadoop/`**`hadoop-env.sh`**

```
  GNU nano 6.2                                  hadoop-env.sh *

# The java implementation to use. By default, this environment
# variable is REQUIRED on ALL platforms except OS X!
# export JAVA_HOME=
export JAVA_HOME=/opt/hadoop/jdk

# The language environment in which Hadoop runs. Use the English
# environment to ensure that logs are printed as expected.
export LANG=en_US.UTF-8

# Location of Hadoop.  By default, Hadoop will attempt to determine
# this location based upon its execution path.
# export HADOOP_HOME=

# Location of Hadoop's configuration information.  i.e., where this
# file is living. If this is not defined, Hadoop will attempt to
# locate it based upon its execution path.
```

COMPROBACIÓN DE FUNCIONAMIENTO

Víctor J. González

Vamos a **verificar ahora que todo es correcto** y que Hadoop funciona.

En el directorio **share** de Hadoop están las librerías de Hadoop, pero si buscamos en **share/hadoop/mapreduce** podemos ver que hay un fichero con ejemplos de MapReduce

```
hadoop@hadoop:~$ cd /opt/hadoop/
hadoop@hadoop:/opt/hadoop$ cd share/hadoop
hadoop@hadoop:/opt/hadoop/share/hadoop$ ls
client  common  hdfs  mapreduce  tools  yarn
hadoop@hadoop:/opt/hadoop/share/hadoop$ cd mapreduce
hadoop@hadoop:/opt/hadoop/share/hadoop/mapreduce$ ls
hadoop-mapreduce-client-app-3.4.1.jar              hadoop-mapreduce-client-nativetask-3.4.1.jar
hadoop-mapreduce-client-common-3.4.1.jar           hadoop-mapreduce-client-shuffle-3.4.1.jar
hadoop-mapreduce-client-core-3.4.1.jar             hadoop-mapreduce-client-uploader-3.4.1.jar
hadoop-mapreduce-client-hs-3.4.1.jar               hadoop-mapreduce-examples-3.4.1.jar
hadoop-mapreduce-client-hs-plugins-3.4.1.jar       jdiff
hadoop-mapreduce-client-jobclient-3.4.1.jar        sources
hadoop-mapreduce-client-jobclient-3.4.1-tests.jar
```

COMPROBACIÓN DE FUNCIONAMIENTO

Vamos a ver el contenido de este fichero que tiene extensión .jar.

Para ello, utilizaremos el **comando `jar`**, que muestra todo el contenido de un archivo con esta extensión.

```
hadoop@hadoop:/opt/hadoop/share/hadoop/mapreduce$ jar tvf hadoop-mapreduce-examples-3.4.1.jar
     0 Wed Oct 09 16:02:38 UTC 2024 META-INF/
   187 Wed Oct 09 16:02:36 UTC 2024 META-INF/MANIFEST.MF
     0 Wed Oct 09 16:02:38 UTC 2024 org/
     0 Wed Oct 09 16:02:38 UTC 2024 org/apache/
     0 Wed Oct 09 16:02:38 UTC 2024 org/apache/hadoop/
     0 Wed Oct 09 16:02:38 UTC 2024 org/apache/hadoop/examples/
     0 Wed Oct 09 16:02:38 UTC 2024 org/apache/hadoop/examples/pi/
     0 Wed Oct 09 16:02:38 UTC 2024 org/apache/hadoop/examples/pi/math/
     0 Wed Oct 09 16:02:38 UTC 2024 org/apache/hadoop/examples/dancing/
     0 Wed Oct 09 16:02:38 UTC 2024 org/apache/hadoop/examples/terasort/
 29535 Wed Oct 09 16:02:38 UTC 2024 org/apache/hadoop/examples/RandomTextWriter.class
  4894 Wed Oct 09 16:02:38 UTC 2024 org/apache/hadoop/examples/RandomWriter$RandomMapper.class
  2679 Wed Oct 09 16:02:38 UTC 2024 org/apache/hadoop/examples/BaileyBorweinPlouffe$BbpInputFormat$1.class
  1252 Wed Oct 09 16:02:38 UTC 2024 org/apache/hadoop/examples/AggregateWordCount.class
  2453 Wed Oct 09 16:02:38 UTC 2024 org/apache/hadoop/examples/MultiFileWordCount$WordOffset.class
  2864 Wed Oct 09 16:02:38 UTC 2024 org/apache/hadoop/examples/SecondarySort$Reduce.class
```

COMPROBACIÓN DE FUNCIONAMIENTO

Víctor J. González

Para probar el funcionamiento de Hadoop vamos a utilizar **Grep**

```
2800 Wed Oct 09 16:02:38 UTC 2024 org/apache/hadoop/examples/DBCountPageView$PageviewReducer.class
7011 Wed Oct 09 16:02:38 UTC 2024 org/apache/hadoop/examples/Join.class
3950 Wed Oct 09 16:02:38 UTC 2024 org/apache/hadoop/examples/Grep.class
2418 Wed Oct 09 16:02:38 UTC 2024 org/apache/hadoop/examples/WordMean$WordMeanReducer.class
2961 Wed Oct 09 16:02:38 UTC 2024 org/apache/hadoop/examples/QuasiMonteCarlo$QmcMapper.class
2416 Wed Oct 09 16:02:38 UTC 2024 org/apache/hadoop/examples/WordMedian$WordMedianReducer.class
```

Este ejemplo lanza un **proceso MapReduce** que recoge un conjunto de ficheros, se le pasa una expresión regular con el texto a buscar y devuelve el número de ocurrencias de la expresión regular entre todos los datos procesados

COMPROBACIÓN DE FUNCIONAMIENTO

Víctor J. González

Como necesitamos datos vamos a descargar, por ejemplo, el libro del Quijote (https://babel.upm.es/~angel/teaching/pps/quijote.txt)

```
hadoop@hadoop:/tmp$ mkdir /tmp/entrada
hadoop@hadoop:/tmp$ wget https://babel.upm.es/~angel/teaching/pps/quijote.txt
--2025-03-27 18:54:37--  https://babel.upm.es/~angel/teaching/pps/quijote.txt
Resolving babel.upm.es (babel.upm.es)... 138.100.12.136
Connecting to babel.upm.es (babel.upm.es)|138.100.12.136|:443... connected.
HTTP request sent, awaiting response... 200 OK
Length: 2141519 (2,0M) [text/plain]
Saving to: ‘quijote.txt’

quijote.txt                         100%[=====================================
```

Creamos un directorio **/tmp/entrada** y descargamos ahí los datos con los que vamos a trabajar.

Ahora me sitúo en el directorio que contiene el fichero .jar con los ejemplos y ejecuto la siguiente orden

El comando **hadoop jar** sirve para ejecutar aplicaciones empaquetadas en archivos Hadoop

```
hadoop@hadoop:/opt/hadoop/share/hadoop/mapreduce$ hadoop jar \
> hadoop-mapreduce-examples-3.4.1.jar \
> grep \
> /tmp/quijote.txt \
> /tmp/salida \
> '[Ss]ancho'
```

El fichero jar que contiene la aplicación que quiero ejecutar

La aplicación que quiero ejecutar

Los datos de entrada, puede ser un fichero o un directorio, en cuyo caso procesaría todo su contenido

La expresión regular que quiero buscar

Donde quiero que guarde el resultado

COMPROBACIÓN DE FUNCIONAMIENTO

Si vamos ahora al directorio `/tmp/salida` veremos que se nos han generado dos ficheros.

```
hadoop@hadoop:/tmp$ cd /tmp/salida
hadoop@hadoop:/tmp/salida$ ls
part-r-00000  _SUCCESS
```

Este fichero está vacío y sirve para indicar que el proceso ha finaliza con éxito

Este fichero contiene el resultado. Si mostramos su contenido veremos que hay 2147 cadenas que coinciden con la expresión regular

```
hadoop@hadoop:/tmp/salida$ cat part-r-00000
2147    Sancho
```

CONFIGURACIÓN DE SSH

Víctor J. González

Hadoop esta conformado por un nodo maestro y varios nodos esclavo, todas las comunicaciones entre estos nodos las realiza utilizando el **protocolo SSH**, por lo que necesitamos configurar las relaciones de confianza entre estos equipos mediante pares de clave pública-privada para que estas conexiones puedan realizarse.

1.- Comenzamos **generando el par de claves** en el equipo, esto se hace mediante el comando **ssh-keygen**

```
hadoop@hadoop:~$ ssh-keygen
Generating public/private rsa key pair.
Enter file in which to save the key (/home/hadoop/.ssh/id_rsa):
Created directory '/home/hadoop/.ssh'.
Enter passphrase (empty for no passphrase):
```

CONFIGURACIÓN DE SSH

Víctor J. González

2.- Las claves generadas se guardan en el directorio **~/.ssh** con los nombres:

- **id_rsa:** esta es la clave privada y debe permanecer secreta
- **id_rsa.pub:** esta es la clave pública y es la que se debe enviar a otros equipos. Podré conectarme de forma transparente a cualquier equipo que tenga mi clave pública

```
hadoop@hadoop:~$ ls /home/hadoop/.ssh/
id_rsa  id_rsa.pub
hadoop@hadoop:~$ cat /home/hadoop/.ssh/id_rsa.pub
ssh-rsa AAAAB3NzaC1yc2EAAAADAQABAAABgQCsKCwgHNO6AnNRUTYhuPFvIT70SbQAnWN90mY/w
3Ip4KhCRR1pchHf0YOYcqlzNuRIMWqIkGyH8yS6i8asixTtD22bhqcoCWpVy8ugDRa1niCM+hQfyO
gS3/4aHfsmsJ6Cf2bUURb4orG71uy1O3KDd3Dd7wTOlU2pADJw418MZ3TbojM8E10GvOl+eVXldDP
Zj3Hw5OV63abVGRj3qmwbYhQdbdnFf5eKg+8WFKb4r5Ao5d+HCVdoZ1EecS7VhZWqUUBp+X6RZNJh
```

3.- Cuando se envía la clave pública a otro equipo se debe almacenar en el directorio **~/.ssh/authorized_keys**

CONFIGURACIÓN DE SSH

Víctor J. González

3.- Aunque solo haya un nodo, las comunicaciones son por SSH, por lo que hay que incluir la propia clave pública en el fichero **authorized_keys**

```
hadoop@hdfs-server:~/.ssh$ cp ~/.ssh/id_rsa.pub ~/.ssh/authorized_keys
hadoop@hdfs-server:~/.ssh$ cat authorized_keys
ssh-rsa AAAAB3NzaC1yc2EAAAADAQABAAABgQCb4tyZGlXrKIDxTGXqdcHItHxsKR9O7woQqjaJ7xAU/UTs
c3MZ1JFkQ3ntICXbtrrYxtLVwH8S/uS4DVSSSX2TY+xH1kbl1mZNRcWnKYs7ox0imyJYfyf2qcWZ16HXR0T5
DOrLRqUep/KnsXEPxpoE84UH8jKkXV3jjNU0pcEUCcif9vbPM59ibhu0Y9Zzz1PcZPI+GSuj7l/bj4LkB07/
```

4.- Asignamos permisos

```
hadoop@hdfs-server:~$ sudo chmod go-w $HOME $HOME/.ssh
hadoop@hdfs-server:~$ sudo chmod 600 $HOME/.ssh/authorized_keys
hadoop@hdfs-server:~$ sudo chown `whoami` $HOME/.ssh/authorized_keys
```

CONFIGURACIÓN DE SSH

5.- Comprobamos que funciona correctamente

```
hadoop@hdfs-server:~/.ssh$ ssh localhost
Welcome to Ubuntu 22.04.5 LTS (GNU/Linux 5.15.0-151-generic x86_64)

 * Documentation:  https://help.ubuntu.com
 * Management:     https://landscape.canonical.com
 * Support:        https://ubuntu.com/pro

 System information as of lun 04 ago 2025 07:26:45 UTC
```

CONFIGURACIÓN DE SSH

Víctor J. González

Según la documentación oficial, Hadoop no admite IPv6 para gestionar correctamente el clúster, por lo que debemos deshabilitarlo.

Esto lo hacemos en el fichero `/etc/sysctl.conf`

```
hadoop@hdfs-server:~/.ssh$ sudo nano /etc/sysctl.conf
```

Añadimos las siguientes líneas

```
net.ipv6.conf.all.disable_ipv6 = 1
net.ipv6.conf.default.disable_ipv6 = 1
net.ipv6.conf.lo.disable_ipv6 = 1
```

Y reiniciamos la máquina para que se apliquen los cambios

```
hadoop@hdfs-server:~/.ssh$ sudo reboot now
Connection to 10.0.0.101 closed by remote host.
Connection to 10.0.0.101 closed.
```

3 INSTALACIÓN DE HADOOP Y HDFS

3.2 CONFIGURACIÓN DE HDFS

CONFIGURACIÓN DE HDFS

Víctor J. González

Con la instalación que hemos hecho tenemos un nodo de Hadoop instalado en la máquina.

Si queremos configurar Hadoop en modo **pseudo-distribuido** debemos modificar algunos ficheros de configuración.

Empezamos por el fichero `/opt/hadoop/etc/hadoop/core-site.xml` añadiendo lo siguiente:

```
<configuration>
        <property>
                <name>fs.default.name</name>
                <value>hdfs://localhost:8020</value>
                <description>Nombre del filesystem por defecto</description>
        </property>
</configuration>
```

Esto configura el directorio HDFS por defecto en *localhost*

CONFIGURACIÓN DE HDFS

Lo siguiente que hacemos es configurar algunas propiedades del sistema de ficheros mediante el fichero `/opt/hadoop/etc/hadoop/hdfs-site.xml`

Establecemos los parámetros:

- `dfs.namenode.name.dir`: ruta del sistema de ficheros donde el NameNode almacenará los metadatos.
- `dfs.datanode.data.dir`: ruta del sistema de ficheros donde el DataNode almacenará los bloques.
- `dfs.replication`: factor de replicación. Como solo tenemos una máquina establecemos el valor en 1.

CONFIGURACIÓN DE HDFS

Víctor J. González

```
<configuration>
        <property>
                <name>dfs.namenode.name.dir</name>
                <value>file:/opt/hadoop/workspace/dfs/name</value>
                <description>Ruta almacenamiento metadatos</description>
        </property>
        <property>
                <name>dfs.datanode.data.dir</name>
                <value>file:/opt/hadoop/workspace/dfs/data</value>
                <description>Ruta almacenamiento de los bloques </description>
        </property>
        <property>
                <name>dfs.replication</name>
                <value>1</value>
                <description>Factor de replicación.</description>
        </property>
</configuration>
```

CONFIGURACIÓN DE HDFS

Debemos crear los directorios que hemos indicado en el fichero de configuración

```
hadoop@hdfs-server:/opt/hadoop$ sudo mkdir -p /opt/hadoop/workspace/dfs/name
hadoop@hdfs-server:/opt/hadoop$ sudo mkdir -p /opt/hadoop/workspace/dfs/data
```

CONFIGURACIÓN DE HDFS

Víctor J. González

Finalmente, formateamos el sistema de ficheros con la sentencia `hdfs namenode -format`

```
hadoop@hdfs-server:/opt/hadoop/workspace/dfs$ hdfs namenode -format
2025-08-04 19:39:55,763 INFO namenode.NameNode: STARTUP_MSG:
/************************************************************
STARTUP_MSG: Starting NameNode
STARTUP_MSG:   host = hdfs-server/127.0.1.1
STARTUP_MSG:   args = [-format]
STARTUP_MSG:   version = 3.4.1
STARTUP_MSG:   classpath = /opt/hadoop/etc/hadoop:/opt/hadoop/share/hadoop/common/l
ler-ssl-ocsp-4.1.100.Final.jar:/opt/hadoop/share/hadoop/common/lib/jackson-databind
opt/hadoop/share/hadoop/common/lib/commons-compress-1.26.1.jar:/opt/hadoop/share/ha
op/common/lib/failureaccess-1.0.jar:/opt/hadoop/share/hadoop/common/lib/jaxb-api-2.
ar:/opt/hadoop/share/hadoop/common/lib/netty-transport-classes-epoll-4.1.100.Final.
al.jar:/opt/hadoop/share/hadoop/common/lib/jsr305-3.0.2.jar:/opt/hadoop/share/hadoo
-2.0.3.jar:/opt/hadoop/share/hadoop/common/lib/jersey-server-1.19.4.jar:/opt/hadoop
mmon/lib/zookeeper-3.8.4.jar:/opt/hadoop/share/hadoop/common/lib/metrics-core-3.2.4
oop/share/hadoop/common/lib/netty-codec-memcache-4.1.100.Final.jar:/opt/hadoop/shar
```

Si miramos ahora en el directorio que creamos para los metadatos veremos que se han creado una serie de ficheros.

```
hadoop@hdfs-server:/opt/hadoop/workspace/dfs$ ls name
current
hadoop@hdfs-server:/opt/hadoop/workspace/dfs$ ls name/current/
fsimage_0000000000000000000  fsimage_0000000000000000000.md5  seen_txid  VERSION
```

CONFIGURACIÓN DE HDFS

Víctor J. González

Ahora vamos a **arrancar el sistema HDFS**. Para ello utilizamos el script que se encuentra en el directorio **sbin**.

```
hadoop@hdfs-server:/opt/hadoop/sbin$ ls
distribute-exclude.sh    refresh-namenodes.sh  start-yarn.cmd    stop-secure-dns.sh
FederationStateStore     start-all.cmd         start-yarn.sh     stop-yarn.cmd
hadoop-daemon.sh         start-all.sh          stop-all.cmd      stop-yarn.sh
hadoop-daemons.sh        start-balancer.sh     stop-all.sh       workers.sh
httpfs.sh                start-dfs.cmd         stop-balancer.sh  yarn-daemon.sh
kms.sh                   start-dfs.sh          stop-dfs.cmd      yarn-daemons.sh
mr-jobhistory-daemon.sh  start-secure-dns.sh   stop-dfs.sh
```

CONFIGURACIÓN DE HDFS

Víctor J. González

Ahora vamos a **arrancar el sistema HDFS**. Para ello utilizamos el script que se encuentra en el directorio **sbin**.

```
hadoop@hdfs-server:/opt/hadoop/etc/hadoop$ start-dfs.sh
Starting namenodes on [localhost]
Starting datanodes
Starting secondary namenodes [hdfs-server]
hadoop@hdfs-server:/opt/hadoop/etc/hadoop$
```

Podemos comprobar que estos procesos se han puesto en marcha usando el comando `jps`, que muestra los procesos Java activos en la máquina.

```
hadoop@hdfs-server:/opt/hadoop/etc/hadoop$ jps
3811 NameNode
3942 DataNode
4166 SecondaryNameNode
4270 Jps
```

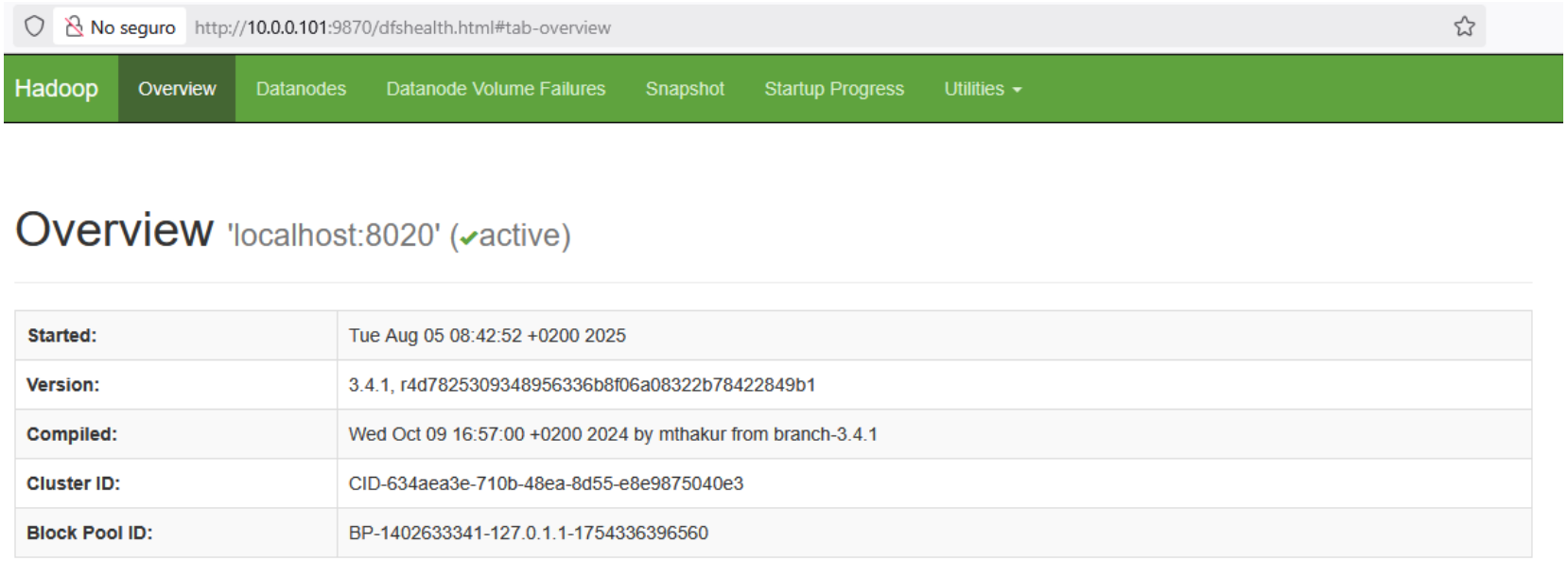
También podríamos ver estos procesos con `ps –ef | grep java`

CONFIGURACIÓN DE HDFS

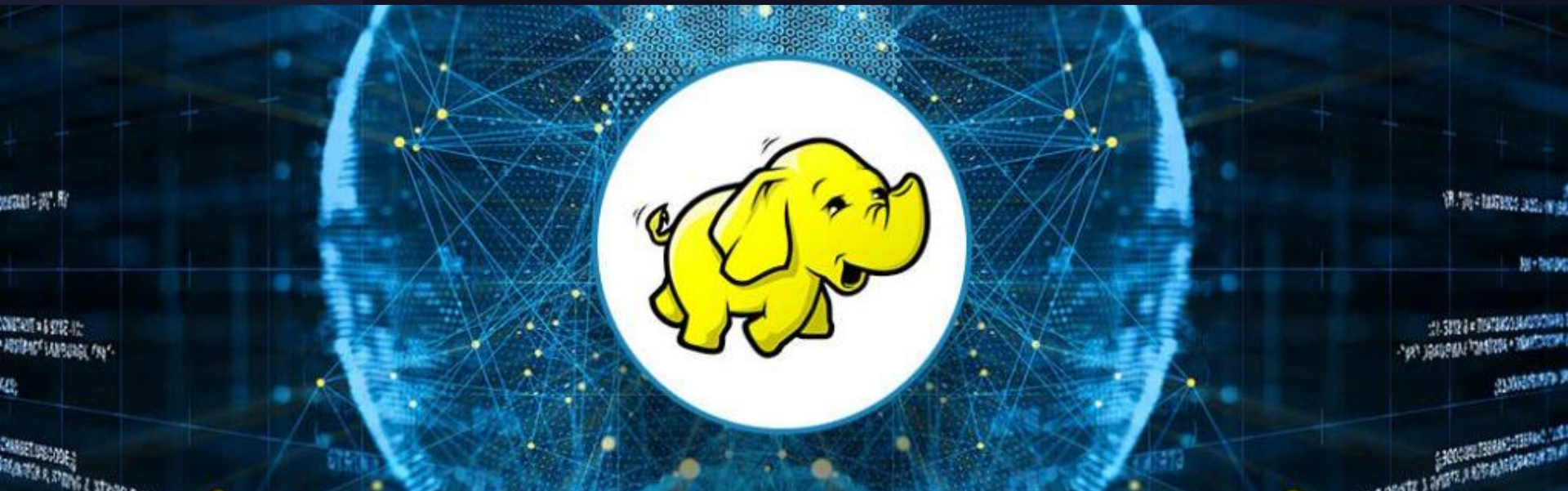
Ahora que ya tenemos arrancado DFS ya podremos ver que hay información en el directorio data

```
hadoop@hdfs-server:~$ ls /opt/hadoop/workspace/dfs/data
current  in_use.lock
hadoop@hdfs-server:~$ ls /opt/hadoop/workspace/dfs/data/current/
BP-1402633341-127.0.1.1-1754336396560  VERSION
```

También podemos comprobar que se ha iniciado correctamente accediendo al puerto **9870** de la máquina

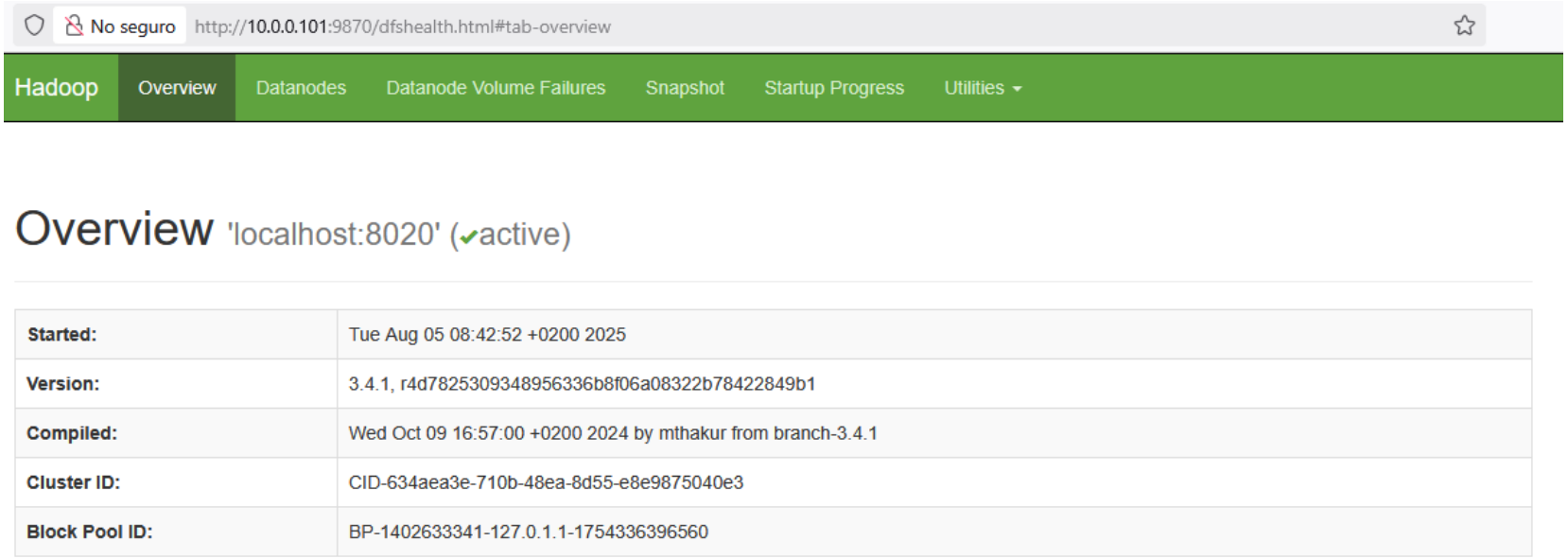


4 ADMINISTRACIÓN Y PUESTA EN MARCHA



Veamos las opciones disponibles en la interfaz web

Overview: muestra una visión general del estado del sistema



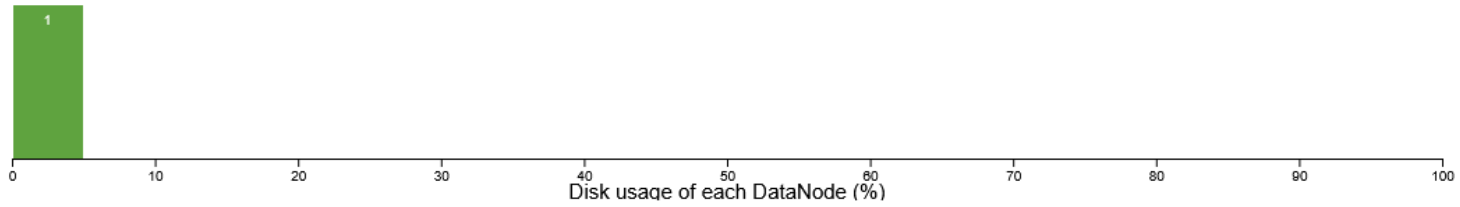
Configured Capacity:	123.41 GB
Configured Remote Capacity:	0 B
DFS Used:	28 KB (0%)
Non DFS Used:	7.6 GB
DFS Remaining:	109.5 GB (88.73%)
Block Pool Used:	28 KB (0%)
DataNodes usages% (Min/Median/Max/stdDev):	0.00% / 0.00% / 0.00% / 0.00%
Live Nodes	1 (Decommissioned: 0, In Maintenance: 0)
Dead Nodes	0 (Decommissioned: 0, In Maintenance: 0)
Decommissioning Nodes	0
Entering Maintenance Nodes	0
Total Datanode Volume Failures	0 (0 B)
Number of Under-Replicated Blocks	0
Number of Blocks Pending Deletion (including replicas)	0
Block Deletion Start Time	Tue Aug 05 08:42:52 +0200 2025
Last Checkpoint Time	Tue Aug 05 08:42:53 +0200 2025
Last HA Transition Time	Never
Enabled Erasure Coding Policies	RS-6-3-1024k

Memoria disponible y utilizada por el sistema de ficheros HDFS

Nodos disponibles y estado

Datanodes: información detallada de cada uno de los *datanodes* que tenemos en el sistema

Datanode usage histogram



In operation

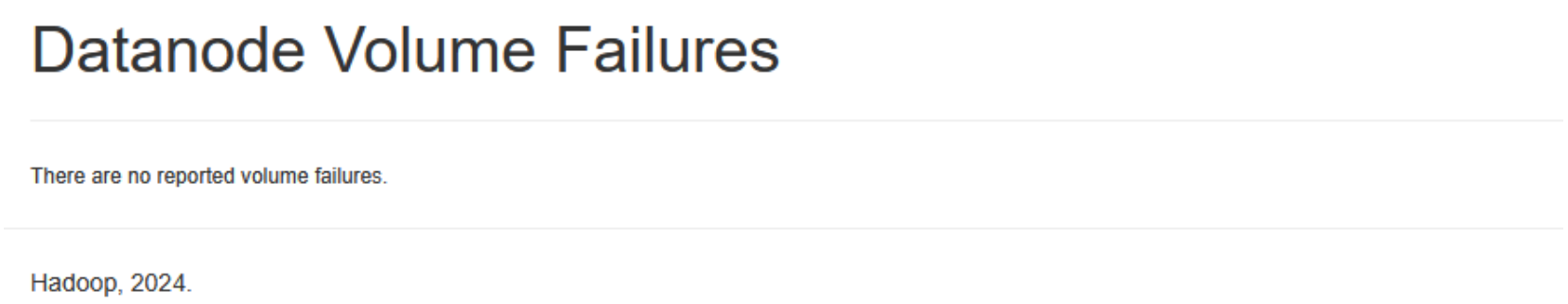
DataNode State All Show 25 entries Search:

Node	Http Address	Last contact	Last Block Report	Used	Non DFS Used	Capacity	Blocks	Block pool used	Block pool usage StdDev	Version
✔/default-rack/hdfs-server:9866 (127.0.0.1:9866)	http://hdfs-server:9864	0s	24m	28 KB	7.6 GB	123.41 GB	0	28 KB (0%)	0%	3.4.1

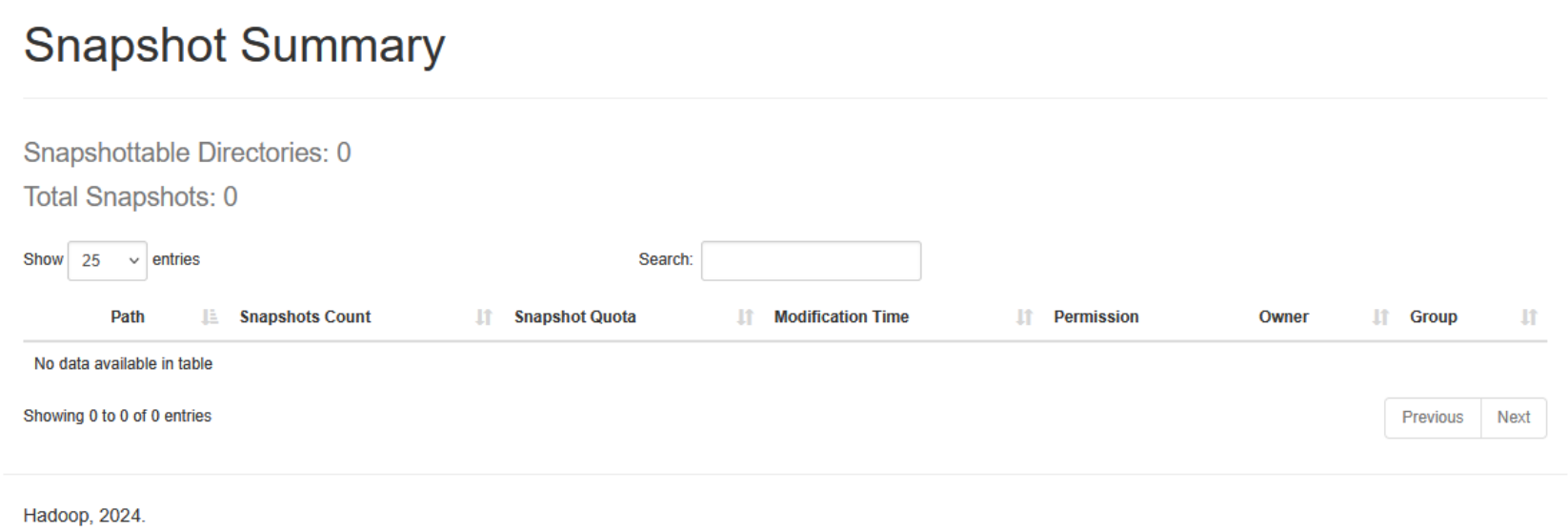
Showing 1 to 1 of 1 entries Previous 1 Next

Entering Maintenance

Datanode volumen failures: informe de errores que haya en alguno de los datanodes



Snapshot: ya veremos esta funcionalidad más adelante, pero aquí se mostrarán los snapshots que tengamos en HDFS



Startup progress: estado del arranque del sistema

Startup Progress

Elapsed Time: 0 sec, Percent Complete: 100%

Phase	Completion	Elapsed Time
Loading fsimage /opt/hadoop/workspace/dfs/name/current/fsimage_0000000000000000000 401 B	**100%**	**0 sec**
erasure coding policies (0/0)	100%	
inodes (1/1)	100%	
delegation tokens (0/0)	100%	
cache pools (0/0)	100%	
Loading edits	**100%**	**0 sec**
Saving checkpoint	**100%**	**0 sec**
erasure coding policies /opt/hadoop/workspace/dfs/name/current/fsimage.ckpt_0000000000000000000 (0/0)	100%	
inodes /opt/hadoop/workspace/dfs/name/current/fsimage.ckpt_0000000000000000000 (0/0)	100%	
delegation tokens /opt/hadoop/workspace/dfs/name/current/fsimage.ckpt_0000000000000000000 (0/0)	100%	
cache pools /opt/hadoop/workspace/dfs/name/current/fsimage.ckpt_0000000000000000000 (0/0)	100%	
Safe mode	**100%**	**0 sec**
awaiting reported blocks (0/0)	100%	

Hadoop, 2024.

Browse the file system: nos mostrará qué archivos tenemos en el sistema de ficheros.

Logs: aquí podremos acceder a los diferentes ficheros de log de Hadoop

Directory: /logs/

Name ⇑	Last Modified	Size
hadoop-hadoop-datanode-hdfs-server.log	Aug 5, 2025 6:42:56 AM	35,230 bytes
hadoop-hadoop-datanode-hdfs-server.out	Aug 5, 2025 6:42:55 AM	818 bytes
hadoop-hadoop-namenode-hdfs-server.log	Aug 5, 2025 7:12:04 AM	62,331 bytes
hadoop-hadoop-namenode-hdfs-server.out	Aug 5, 2025 7:11:18 AM	6,673 bytes
hadoop-hadoop-secondarynamenode-hdfs-server.log	Aug 5, 2025 6:43:59 AM	37,250 bytes
hadoop-hadoop-secondarynamenode-hdfs-server.out	Aug 5, 2025 6:42:58 AM	818 bytes
SecurityAuth-hadoop.audit	Aug 4, 2025 7:34:25 PM	0 bytes

Metrics: información completa de Hadoop en formato JSON


```
beans:
  0:
    name:                    "Hadoop:service=NameNode,name=DelegationTokenSecretManagerMetrics"
    modelerType:             "DelegationTokenSecretManagerMetrics"
    tag.Context:             "token"
    tag.Hostname:            "hdfs-server"
    RemoveTokenNumOps:       0
    RemoveTokenAvgTime:      0.0 JS:0
    StoreTokenNumOps:        0
    StoreTokenAvgTime:       0.0 JS:0
    TokenFailure:            0
    UpdateTokenNumOps:       0
    UpdateTokenAvgTime:      0.0 JS:0
  1:
    name:                    "Hadoop:service=NameNode,name=JvmMetrics"
    modelerType:             "JvmMetrics"
    tag.Context:             "jvm"
    tag.ProcessName:         "NameNode"
    tag.SessionId:           null
    tag.Hostname:            "hdfs-server"
    MemNonHeapUsedM:         64.50994
    MemNonHeapCommittedM:    66.0 JS:66
    MemNonHeapMaxM:          -1.0 JS:-1
    MemHeapUsedM:            27.426682
    MemHeapCommittedM:       35.558594
```

Configuration: fichero XML con la configuración que tengamos establecida

This XML file does not appear to have any style information associated with it. The document tree is shown below.


```
<configuration>
  <property>
    <name>mapreduce.jobhistory.jhist.format</name>
    <value>binary</value>
    <final>false</final>
    <source>mapred-default.xml</source>
  </property>
  <property>
    <name>dfs.namenode.available-space-rack-fault-tolerant-block-placement-policy.balanced-space-tolerance</name>
    <value>5</value>
    <final>false</final>
    <source>hdfs-default.xml</source>
  </property>
  <property>
    <name>fs.s3a.retry.interval</name>
    <value>500ms</value>
    <final>false</final>
    <source>core-default.xml</source>
  </property>
  <property>
    <name>dfs.block.access.token.lifetime</name>
    <value>600</value>
    <final>false</final>
    <source>hdfs-default.xml</source>
  </property>
  <property>
    <name>mapreduce.job.heap.memory-mb.ratio</name>
    <value>0.8</value>
    <final>false</final>
    <source>mapred-default.xml</source>
  </property>
```

Network topology: topología de red de Hadoop, en nuestro caso solo tenemos una única máquina, pero recuerda que Hadoop puede distribuirse sobre cientos o miles de máquinas

```
Rack: /default-rack
   127.0.0.1:9866 (localhost)
```

5 COMANDOS BÁSICOS Y CLI DE HDFS

COMANDOS BÁSICOS Y CLI DE HDFS

Víctor J. González

Para interactuar con el sistema HDFS debemos utilizar el comando **hdfs**. Si invocamos dicho comando veremos su sintaxis, donde se puede ver que funciona junto con un **subcomando**, que puede ser de diversas categorías.

```
hadoop@hdfs-server:~$ hdfs
Usage: hdfs [OPTIONS] SUBCOMMAND [SUBCOMMAND OPTIONS]

  OPTIONS is none or any of:

--buildpaths                       attempt to add class files from build tree
--config dir                       Hadoop config directory
--daemon (start|status|stop)       operate on a daemon
--debug                            turn on shell script debug mode
--help                             usage information
--hostnames list[,of,host,names]   hosts to use in worker mode
--hosts filename                   list of hosts to use in worker mode
--loglevel level                   set the log4j level for this command
--workers                          turn on worker mode
```

COMANDOS BÁSICOS Y CLI DE HDFS

Víctor J. González

Los comandos de **administración** permiten realizar tareas de administración, como habilitar el cifrado o lanzar el cliente de administración

```
    Admin Commands:

cacheadmin          configure the HDFS cache
crypto              configure HDFS encryption zones
debug               run a Debug Admin to execute HDFS debug commands
dfsadmin            run a DFS admin client
dfsrouteradmin      manage Router-based federation
ec                  run a HDFS ErasureCoding CLI
fsImageValidation   run FsImageValidation to check an fsimage
fsck                run a DFS filesystem checking utility
haadmin             run a DFS HA admin client
jmxget              get JMX exported values from NameNode or DataNode.
oev                 apply the offline edits viewer to an edits file
oiv                 apply the offline fsimage viewer to an fsimage
oiv_legacy          apply the offline fsimage viewer to a legacy fsimage
storagepolicies     list/get/set/satisfyStoragePolicy block storage policies
```

Los comandos de **cliente** son los que utilizaremos habitualmente para trabajar con el sistema de ficheros, especialmente el subcomando **dfs**

```
    Client Commands:

classpath            prints the class path needed to get the hadoop jar and the required
                     libraries
dfs                  run a filesystem command on the file system
envvars              display computed Hadoop environment variables
fetchdt              fetch a delegation token from the NameNode
getconf              get config values from configuration
groups               get the groups which users belong to
lsSnapshot           list all snapshots for a snapshottable directory
lsSnapshottableDir   list all snapshottable dirs owned by the current user
snapshotDiff         diff two snapshots of a directory or diff the current directory conte
                     with a snapshot
version              print the version
```

COMANDOS BÁSICOS Y CLI DE HDFS

Víctor J. González

Por último, los comandos del **demonio** sirven para iniciar manualmente los procesos individuales que componen el sistema de archivos distribuido de Hadoop

```
    Daemon Commands:

balancer             run a cluster balancing utility
datanode             run a DFS datanode
dfsrouter            run the DFS router
diskbalancer         Distributes data evenly among disks on a given node
httpfs               run HttpFS server, the HDFS HTTP Gateway
journalnode          run the DFS journalnode
mover                run a utility to move block replicas across storage types
namenode             run the DFS namenode
nfs3                 run an NFS version 3 gateway
portmap              run a portmap service
secondarynamenode    run the DFS secondary namenode
sps                  run external storagepolicysatisfier
zkfc                 run the ZK Failover Controller daemon

SUBCOMMAND may print help when invoked w/o parameters or with -h.
```

COMANDOS BÁSICOS Y CLI DE HDFS

Víctor J. González

El comando que utilizaremos para trabajar con ficheros es **hdfs dfs**. Si miramos su ayuda podemos ver que los parámetros tienen gran similitud con los comandos de Linux Bash

```
hadoop@hdfs-server:~$ hdfs dfs
Usage: hadoop fs [generic options]
        [-appendToFile [-n] <localsrc> ... <dst>]
        [-cat [-ignoreCrc] <src> ...]
        [-checksum [-v] <src> ...]
        [-chgrp [-R] GROUP PATH...]
        [-chmod [-R] <MODE[,MODE]... | OCTALMODE> PATH...]
        [-chown [-R] [OWNER][:[GROUP]] PATH...]
        [-concat <target path> <src path> <src path> ...]
        [-copyFromLocal [-f] [-p] [-l] [-d] [-t <thread count>] [-q <thread pool qu
calsrc> ... <dst>]
        [-copyToLocal [-f] [-p] [-crc] [-ignoreCrc] [-t <thread count>] [-q <thread
ze>] <src> ... <localdst>]
        [-count [-q] [-h] [-v] [-t [<storage type>]] [-u] [-x] [-e] [-s] <path> ...
        [-cp [-f] [-p | -p[topax]] [-d] [-t <thread count>] [-q <thread pool queue
.. <dst>]
        [-createSnapshot <snapshotDir> [<snapshotName>]]
        [-deleteSnapshot <snapshotDir> <snapshotName>]
        [-df [-h] [<path> ...]]
```

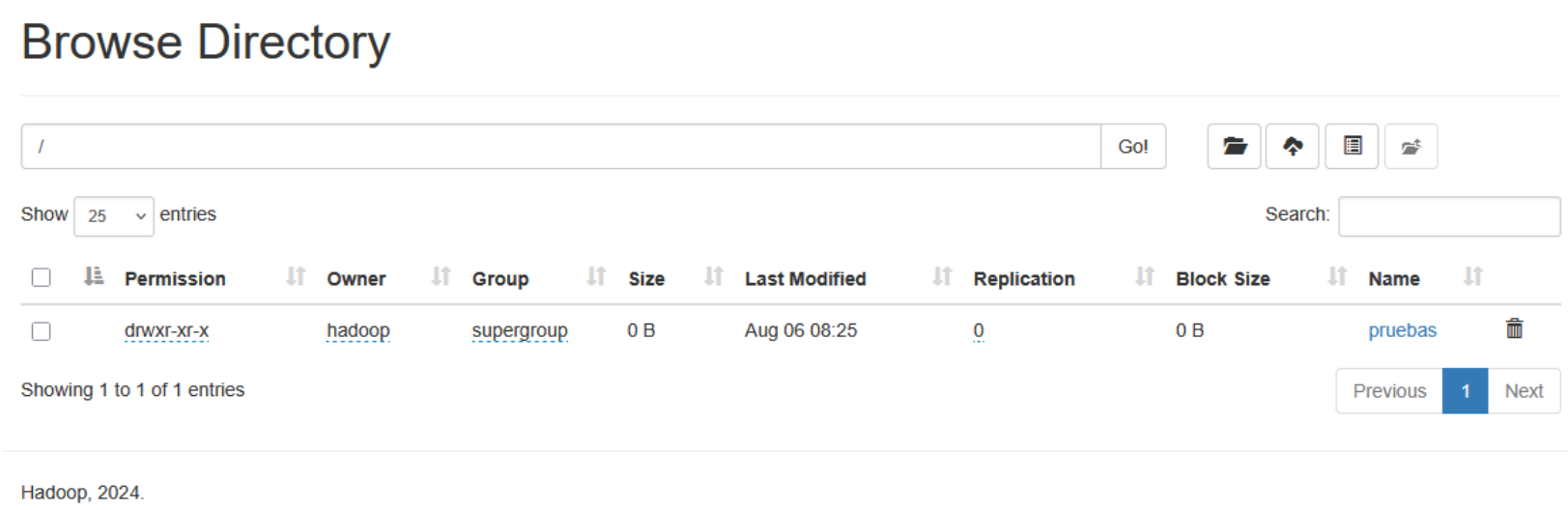
COMANDOS BÁSICOS Y CLI DE HDFS

Víctor J. González

Comando	Función
`hdfs dfs –ls <ruta>`	Lista los archivos y carpetas de un directorio
`hdfs dfs –mkdir <ruta>`	Crea un directorio en la ruta indicada
`hdfs dfs –put <local> <dst>`	Copia un fichero desde el sistema de archivos local a la ruta destino en el sistema HDFS
`hdfs dfs –get <src> <local>`	Extrae un fichero de HDFS al sistema de ficheros local
`hdfs dfs –cat <file>`	Muestra el contenido de un archivo de texto
`hdfs dfs –rm <file>`	Elimina un fichero
`hdfs dfs –mv <src> <dst>`	Mueve y renombra archivos y carpetas
`hdfs dfs –du <path>`	Muestra el espacio usado
`hdfs dfs –cp <src> <dst>`	Copia archivos dentro de HDFS
`hdfs dfs –chmod <mode> <path>`	Cambia los permisos de un archivo
`hdfs dfs –chown <ownr:grp> <path>`	Cambia el propietario de un fichero
`hdfs dfs –text <path>`	Comprueba si un archivo o directorio existe

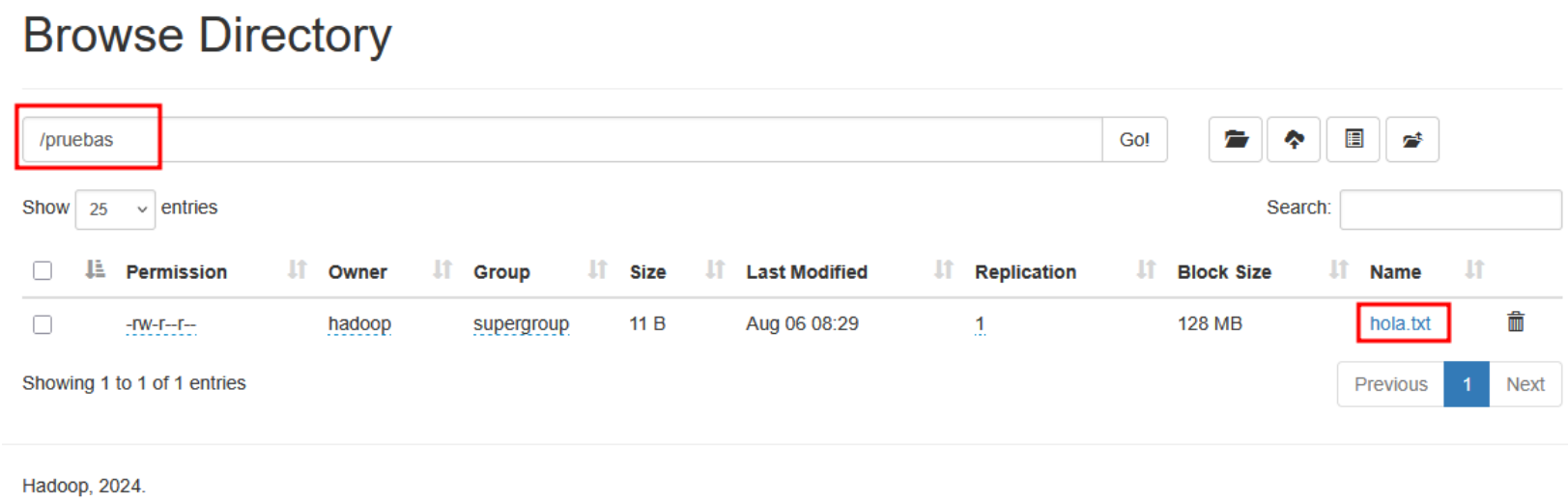
Creación de un directorio

```
hadoop@hdfs-server:~$ hdfs dfs -ls /
hadoop@hdfs-server:~$ hdfs dfs -mkdir /pruebas
hadoop@hdfs-server:~$ hdfs dfs -ls /
Found 1 items
drwxr-xr-x   - hadoop supergroup          0 2025-08-06 06:25 /pruebas
```



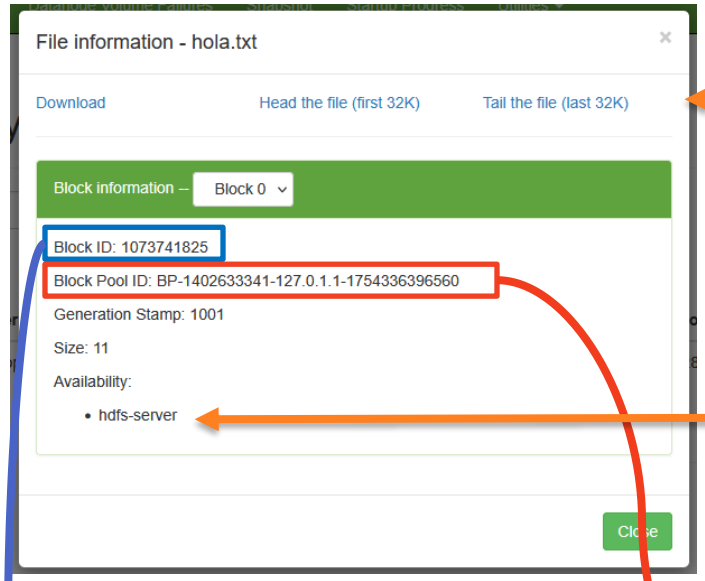
Copia de un archivo a HDFS

```
hadoop@hdfs-server:~$ echo "Hola mundo" > hola.txt
hadoop@hdfs-server:~$ hdfs dfs -put ./hola.txt /pruebas
hadoop@hdfs-server:~$ hdfs dfs -ls /pruebas
Found 1 items
-rw-r--r--   1 hadoop supergroup         11 2025-08-06 06:29 /pruebas/hola.txt
```



COMANDOS BÁSICOS Y CLI DE HDFS

Víctor J. González



Si hacemos click en el fichero vemos información sobre él

Como el factor de replicación es 1, el bloque solo está disponible en un nodo

```
hadoop@hdfs-server:~$ ls -l /opt/hadoop/workspace/dfs/data/current
total 8
drwx------ 4 hadoop hadoop 4096 ago  5 06:42 BP-1402633341-127.0.1.1-1754336396560
-rw-rw-r-- 1 hadoop hadoop  229 ago  5 06:42 VERSION
```

```
hadoop@hdfs-server:~$ ls -l /opt/hadoop/workspace/dfs/data/current/BP-1402633341-127.0.1.1-175433
6396560/current/finalized/subdir0/subdir0/
total 8
-rw-rw-r-- 1 hadoop hadoop 11 ago  6 06:29 blk_1073741825
-rw-rw-r-- 1 hadoop hadoop 11 ago  6 06:29 blk_1073741825_1001.meta
```

Observa cómo se relaciona el Block ID y el Block Pool ID con directorios en el sistema de ficheros local

COMANDOS BÁSICOS Y CLI DE HDFS

Víctor J. González

Incluso podríamos ver el contenido del bloque desde el sistema de ficheros **local**.

```
hadoop@hdfs-server:~$ cat /opt/hadoop/workspace/dfs/data/current/BP-1402633341-127.0.1.1-17543363
96560/current/finalized/subdir0/subdir0/blk_1073741825
Hola mundo
```

Mostrar el contenido de un archivo dentro de HDFS

```
hadoop@hdfs-server:~$ hdfs dfs -ls /pruebas
Found 1 items
-rw-r--r--   1 hadoop supergroup         11 2025-08-06 06:29 /pruebas/hola.txt
hadoop@hdfs-server:~$ hdfs dfs -cat /pruebas/hola.txt
Hola mundo
```

Aquí estamos viendo el contenido del fichero desde **dentro** de HDFS

Eliminar un fichero dentro de HDFS

```
hadoop@hdfs-server:~$ hdfs dfs -rm /pruebas/hola.txt
Deleted /pruebas/hola.txt
hadoop@hdfs-server:~$
```

Copia de fichero (gran tamaño)

```
hadoop@hdfs-server:~$ dd \
> if=/dev/zero \
> of=/tmp/big_file.dat \
> bs=1024 \
> count=1000000
1000000+0 records in
1000000+0 records out
1024000000 bytes (1,0 GB, 977 MiB) copied, 3,62294 s, 283 MB/s
```

Vamos a usar el comando **dd** de Linux, que sirve para copiar y convertir datos a bajo nivel

Usamos como entrada /dev/zero, que genera ceros infinitamente

Definimos un tamaño de bloque de 1024 bytes (1 KB)

Y lo repetimos 1000000 de veces (1 GB)

COMANDOS BÁSICOS Y CLI DE HDFS

Víctor J. González

Aquí está el fichero en el sistema de ficheros local

```
hadoop@hdfs-server:~$ ls -l /tmp/big*
-rw-rw-r-- 1 hadoop hadoop 1024000000 ago  7 06:23 /tmp/big_file.dat
hadoop@hdfs-server:~$ hdfs dfs -put /tmp/big_file.dat /
```

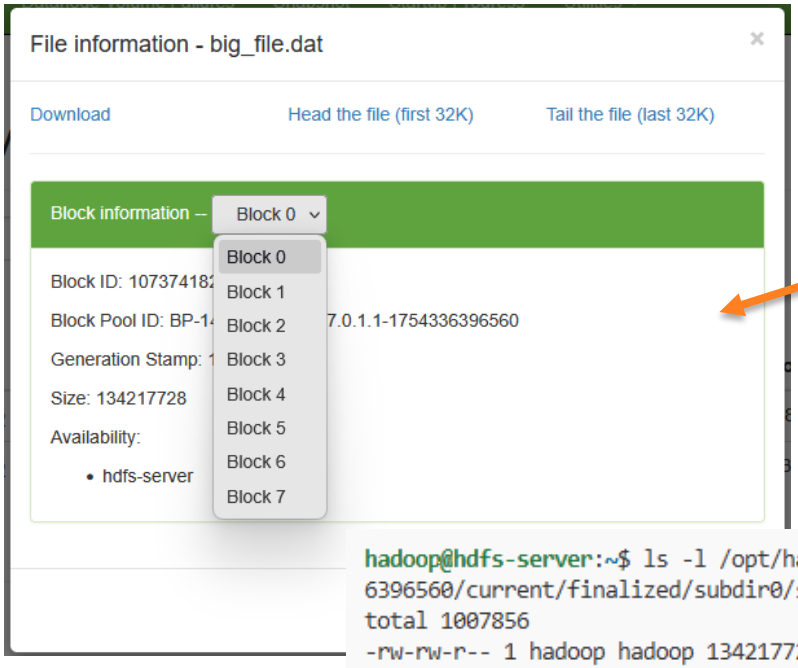
Lo copiamos a HDFS

```
hadoop@hdfs-server:~$ hdfs dfs -ls /
Found 2 items
-rw-r--r--   1 hadoop supergroup 1024000000 2025-08-07 06:30 /big_file.dat
```

Y lo podemos ver en el sistema de ficheros HDFS

COMANDOS BÁSICOS Y CLI DE HDFS

Víctor J. González



Si vamos a la interfaz web ahora veremos que ha dividido el fichero en 8 bloques (cada uno de 128 MB)

Y si comprobamos el directorio data del sistema de ficheros local veremos la correspondencia de esos 8 bloques

```
hadoop@hdfs-server:~$ ls -l /opt/hadoop/workspace/dfs/data/current/BP-1402633341-127.0.1.1-175433
6396560/current/finalized/subdir0/subdir0/
total 1007856
-rw-rw-r-- 1 hadoop hadoop 134217728 ago  7 06:30 blk_1073741827
-rw-rw-r-- 1 hadoop hadoop   1048583 ago  7 06:30 blk_1073741827_1003.meta
-rw-rw-r-- 1 hadoop hadoop 134217728 ago  7 06:30 blk_1073741828
-rw-rw-r-- 1 hadoop hadoop   1048583 ago  7 06:30 blk_1073741828_1004.meta
-rw-rw-r-- 1 hadoop hadoop 134217728 ago  7 06:30 blk_1073741829
-rw-rw-r-- 1 hadoop hadoop   1048583 ago  7 06:30 blk_1073741829_1005.meta
-rw-rw-r-- 1 hadoop hadoop 134217728 ago  7 06:30 blk_1073741830
-rw-rw-r-- 1 hadoop hadoop   1048583 ago  7 06:30 blk_1073741830_1006.meta
-rw-rw-r-- 1 hadoop hadoop 134217728 ago  7 06:30 blk_1073741831
-rw-rw-r-- 1 hadoop hadoop   1048583 ago  7 06:30 blk_1073741831_1007.meta
-rw-rw-r-- 1 hadoop hadoop 134217728 ago  7 06:30 blk_1073741832
-rw-rw-r-- 1 hadoop hadoop   1048583 ago  7 06:30 blk_1073741832_1008.meta
-rw-rw-r-- 1 hadoop hadoop 134217728 ago  7 06:30 blk_1073741833
-rw-rw-r-- 1 hadoop hadoop   1048583 ago  7 06:30 blk_1073741833_1009.meta
-rw-rw-r-- 1 hadoop hadoop  84475904 ago  7 06:30 blk_1073741834
-rw-rw-r-- 1 hadoop hadoop    659975 ago  7 06:30 blk_1073741834_1010.meta
```

Ver espacio ocupado

```
hadoop@hdfs-server:~$ hdfs dfs -du /
1024000000  1024000000  /big_file.dat
0           0           /pruebas
```

Copia de archivo dentro desde HDFS a HDFS

```
hadoop@hdfs-server:~$ hdfs dfs -put ./hola.txt /
hadoop@hdfs-server:~$ hdfs dfs -ls /
Found 3 items
-rw-r--r--   1 hadoop supergroup 1024000000 2025-08-07 06:30 /big_file.dat
-rw-r--r--   1 hadoop supergroup         11 2025-08-07 08:57 /hola.txt
drwxr-xr-x   - hadoop supergroup          0 2025-08-07 06:21 /pruebas
hadoop@hdfs-server:~$ hdfs dfs -cp /hola.txt /pruebas
hadoop@hdfs-server:~$ hdfs dfs -ls /pruebas
Found 1 items
-rw-r--r--   1 hadoop supergroup         11 2025-08-07 08:57 /pruebas/hola.txt
```

Aquí copiamos un fichero del sistema de archivos local a HDFS

Y aquí copiamos un fichero que está en HDFS a otra ubicación también dentro de HDFS

COMANDOS BÁSICOS Y CLI DE HDFS

Víctor J. González

Para realizar tareas de administración tenemos el subcomando **dfsadmin**

```
hadoop@hdfs-server:~$ hdfs dfsadmin
Usage: hdfs dfsadmin
Note: Administrative commands can only be run as the HDFS superuser.
        [-report [-live] [-dead] [-decommissioning] [-enteringmaintenance] [-inmaintenance] [-slownodes]]
        [-safemode <enter | leave | get | wait | forceExit>]
        [-saveNamespace [-beforeShutdown]]
        [-rollEdits]
        [-restoreFailedStorage true|false|check]
        [-refreshNodes]
        [-setQuota <quota> <dirname>...<dirname>]
        [-clrQuota <dirname>...<dirname>]
        [-setSpaceQuota <quota> [-storageType <storagetype>] <dirname>...<dirname>]
        [-clrSpaceQuota [-storageType <storagetype>] <dirname>...<dirname>]
        [-finalizeUpgrade]
        [-rollingUpgrade [<query|prepare|finalize>]]
        [-upgrade <query | finalize>]
        [-refreshServiceAcl]
        [-refreshUserToGroupsMappings]
        [-refreshSuperUserGroupsConfiguration]
        [-refreshCallQueue]
        [-refresh <host:ipc_port> <key> [arg1..argn]
        [-reconfig <namenode|datanode> <host:ipc_port|livenodes> <start|status|properties>]
        [-printTopology]
        [-refreshNamenodes datanode_host:ipc_port]
        [-getVolumeReport datanode_host:ipc_port]
        [-deleteBlockPool datanode_host:ipc_port blockpoolId [force]]
        [-setBalancerBandwidth <bandwidth in bytes per second>]
```

COMANDOS BÁSICOS Y CLI DE HDFS

Víctor J. González

Comando	Función
`hdfs dfsadmin -report`	Muestra un informe sobre el estado del sistema
`hdfs dfsadmin –metasave <file>`	Guarda el estado actual del NameNode en un archivo de texto
`hdfs dfsadmin –listCorruptFileBlocks <ruta>`	Lista bloques corruptos en la ruta indicada
`hdfs dfs –setrep <file> -w <level>`	Cambia el factor de replicación de un archivo (ojo con el subcomando)
`hdfs dfsadmin -refreshNodes`	Vuelve a leer el archivo `dfs.hosts` o `dfs.hosts.exclude` para añadir o excluir nodos sin reiniciar el clúster
`hdfs dfsadmin –safemode get|enter|leave|wait`	Gestiona el modo seguro del NameNode
`hdfs dfsadmin -balancer`	Inicia el balancer, que distribuye los nodos para que uso del disco esté equilibrado
`hdfs dfsadmin -printTopology`	Muestra el número de nodos y a qué máquina pertenecen

Informe de HDFS

```
hadoop@hdfs-server:~$ hdfs dfsadmin -report
Configured Capacity: 132512878592 (123.41 GB)
Present Capacity: 116331134976 (108.34 GB)
DFS Remaining: 115299037184 (107.38 GB)
DFS Used: 1032097792 (984.29 MB)
DFS Used%: 0.89%
Replicated Blocks:
        Under replicated blocks: 0
        Blocks with corrupt replicas: 0
        Missing blocks: 0
        Missing blocks (with replication factor 1): 0
        Low redundancy blocks with highest priority to recover: 0
        Pending deletion blocks: 0
Erasure Coded Block Groups:
        Low redundancy block groups: 0
        Block groups with corrupt internal blocks: 0
        Missing block groups: 0
        Low redundancy blocks with highest priority to recover: 0
        Pending deletion blocks: 0
```

Número de bloques replicados menos veces que las indicadas por el factor de replicación

COMANDOS BÁSICOS Y CLI DE HDFS

Víctor J. González

```
-------------------------------------------------
Live datanodes (1):

Name: 127.0.0.1:9866 (localhost)
Hostname: hdfs-server
Decommission Status : Normal
Configured Capacity: 132512878592 (123.41 GB)
DFS Used: 1032097792 (984.29 MB)
Non DFS Used: 9403174912 (8.76 GB)
DFS Remaining: 115299037184 (107.38 GB)
DFS Used%: 0.78%
DFS Remaining%: 87.01%
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: 0
Last contact: Fri Aug 08 05:58:40 UTC 2025
Last Block Report: Thu Aug 07 09:21:00 UTC 2025
Num of Blocks: 10
```

Modo seguro de HDFS

El modo seguro sirve para impedir escrituras en el sistema para realizar tareas de mantenimiento.

Situaciones en las que se debería usar este modo:

- **Mantenimiento del sistema**: antes de realizar tareas críticas como actualización del sistema de archivos, cambio de configuración de nodos o migraciones de datos.
- **Diagnóstico de problemas**: durante la revisión de bloques corruptos, estados inconsistentes de nodos o pérdida de datos en el clúster
- **Comprobación de integridad al arrancar**: cuando el NameNode arranca, entra automáticamente en modo seguro para verificar cuántos bloques tienen réplicas suficientes y esperar a que los DataNodes se registren correctamente,
- **Balanceo o reconfiguración de nodos**: antes de añadir o eliminar nodos del clúster.
- **Recuperación después de fallo**: si se detecta una situación anómala tras reiniciar HDS, dejar el sistema en modo seguro puede ser útil para recuperar metadatos o copiar datos críticos a otros sistemas.

COMANDOS BÁSICOS Y CLI DE HDFS

Víctor J. González

Comprobamos el estado del modo seguro

Accedemos al modo seguro. Podemos comprobarlo en la interfaz web


```
hadoop@hdfs-server:~$ hdfs dfsadmin -safemode get
Safe mode is OFF
hadoop@hdfs-server:~$ hdfs dfsadmin -safemode enter
Safe mode is ON
hadoop@hdfs-server:~$ hdfs fsck -list-corruptfileblocks /
Connecting to namenode via http://localhost:9870/fsck?ugi=hadoop&listcorruptfileblo
The filesystem under path '/' has 0 CORRUPT blocks
hadoop@hdfs-server:~$ hdfs dfsadmin -safemode leave
Safe mode is OFF
hadoop@hdfs-server:~$
```

Como ejemplo, aquí estamos comprobando si hay bloques corruptos

Cuando hemos acabado, salimos del modo seguro

Topología de HDFS

El nodo está en el **rack lógico** llamado `/default-rack`

```
hadoop@hdfs-server:~$ hdfs dfsadmin -printTopology
Rack: /default-rack
   127.0.0.1:9866 (localhost) In Service
```

Dirección IP y nombre del DataNode

El nodo está activo y disponible para operaciones

HDFS puede estar desplegado en clústeres grandes distribuidos físicamente con los nodos organizados en **racks** (un rack es un conjunto de servidores conectados entre sí a través de un switch de red común.)

HDFS es **rack aware (consciente de rack)**, de forma que intentará situar las réplicas de cada bloque en diferentes racks para aumentar la tolerancia a fallos y la eficiencia de red.

Snapshots

Los **snapshots** permiten guardar el estado de un directorio en un momento dado y restaurarlo más adelante.

Características:

- **Instantáneo:** la creación de un *snapshot* es casi instantánea, ya que no se copia la información real de los bloques de datos.
- **Cero sobrecarga de almacenamiento**: no ocupan espacio adicional de almacenamiento. Solo almacenan los **cambios** (deltas) que ocurren después de que se tomó la instantánea.
- **Lectura única**: los datos dentro de un *snapshot* no pueden ser modificados.

SNAPSHOTS

Solo se pueden tomar snapshots de directorios que han sido **explícitamente habilitados** para ello por un administrador (***Snapshottable Directory***)

Se debe hacer con la orden:

```
hadoop@hadoop:~$ hdfs dfsadmin -allowSnapshot /pruebas
Allowing snapshot on /pruebas succeeded
hadoop@hadoop:~$
```

Una vez habilitado el directorio, los *snapshots* se almacenan en un subdirectorio oculto llamado `.snapshot` dentro de ese directorio

```
hadoop@hdfs-server:~$ hdfs lsSnapshottableDir
drwxr-xr-x 0 hadoop supergroup 0 2025-08-07 08:57 0 65536 /pruebas
```

También podemos deshabilitar un directorio como apto para realizar snapshots

```
hadoop@hdfs-server:~$ hdfs dfsadmin -disallowSnapshot /pruebas
Disallowing snapshot on /pruebas succeeded
hadoop@hdfs-server:~$
```

Para crear un *snapshot* debemos usar el subcomando **createSnapshot**, indicando un nombre opcional

```
hadoop@hadoop:~$ hdfs dfs -createSnapshot /pruebas mi_snapshot
Created snapshot /pruebas/.snapshot/mi_snapshot
hadoop@hadoop:~$
```

Podemos ver los snapshots que he creado en un directorio mostrando el contenido del directorio oculto .snapshot

```
hadoop@hadoop:~$ hdfs dfs -ls /pruebas/.snapshot
Found 2 items
drwxr-xr-x   - hadoop supergroup          0 2025-10-14 07:30 /pruebas/.snapshot/backup_14-10-2025
drwxr-xr-x   - hadoop supergroup          0 2025-10-15 09:27 /pruebas/.snapshot/mi_snapshot
hadoop@hadoop:~$
```

SNAPSHOTS

Víctor J. González

Para renombrar y eliminar snapshots usaremos los subcomandos **renameSnapshot** y **deleteSnapshot** respectivamente

```
hadoop@hadoop:~$ hdfs dfs -ls /pruebas/.snapshot
Found 2 items
drwxr-xr-x   - hadoop supergroup          0 2025-10-14 07:30 /pruebas/.snapshot/backup_14-10-2025
drwxr-xr-x   - hadoop supergroup          0 2025-10-15 09:27 /pruebas/.snapshot/mi_snapshot
hadoop@hadoop:~$ hdfs dfs -renameSnapshot /pruebas mi_snapshot backup_15-10-2025
Renamed snapshot mi_snapshot to backup_15-10-2025 under hdfs://localhost:8020/pruebas
hadoop@hadoop:~$ hdfs dfs -ls /pruebas/.snapshot
Found 2 items
drwxr-xr-x   - hadoop supergroup          0 2025-10-14 07:30 /pruebas/.snapshot/backup_14-10-2025
drwxr-xr-x   - hadoop supergroup          0 2025-10-15 09:32 /pruebas/.snapshot/backup_15-10-2025
hadoop@hadoop:~$ hdfs dfs -deleteSnapshot /pruebas backup_14-10-2025
Deleted snapshot backup_14-10-2025 under hdfs://localhost:8020/pruebas
hadoop@hadoop:~$ hdfs dfs -ls /pruebas/.snapshot
Found 1 items
drwxr-xr-x   - hadoop supergroup          0 2025-10-15 09:32 /pruebas/.snapshot/backup_15-10-2025
```

SNAPSHOTS

Dado que los *snapshots* son de solo lectura, la **restauración** se realiza copiando los datos del *snapshot* de vuelta al directorio activo o a una nueva ubicación con el subcomando **cp**

```
hadoop@hadoop:~$ hdfs dfs -cp /pruebas/.snapshot/backup_15-10-2025/fichero /pruebas/fichero_recuperado
hadoop@hadoop:~$ hdfs dfs -ls /pruebas
Found 2 items
-rw-r--r--   1 hadoop supergroup         41 2025-10-14 07:27 /pruebas/fichero
-rw-r--r--   1 hadoop supergroup         41 2025-10-15 09:38 /pruebas/fichero_recuperado
```

Cuotas

Una **cuota** es una restricción que limita el uso de recursos de almacenamiento en un directorio y sus subdirectorios.

Existen dos tipos de cuotas:

- **Cuota de espacio**, que limita el número total de bytes
- **Cuota de nombres**, que limita el número total de archivos y directorios

CUOTAS

Víctor J. González

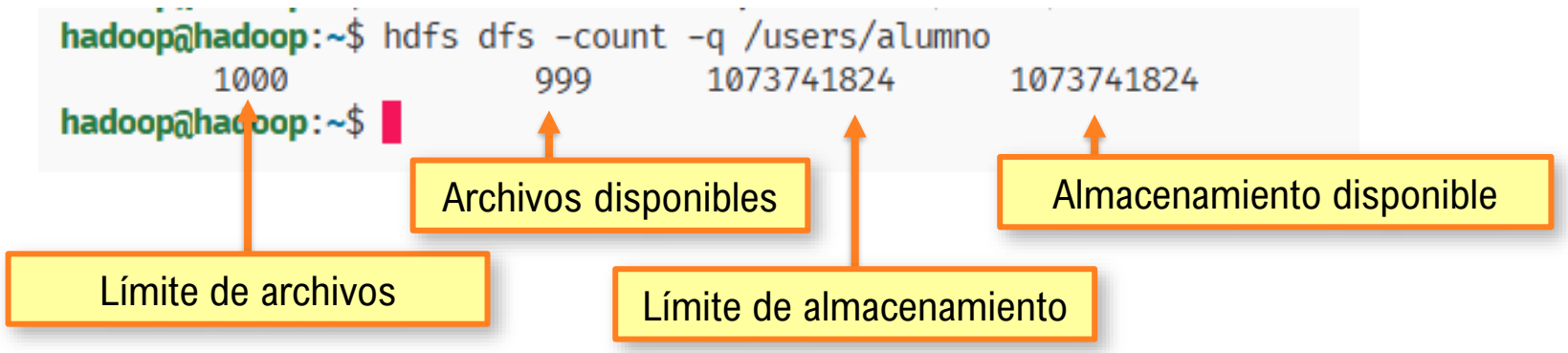
Para crear una cuota usaremos el subcomando **setSpaceQuota** o **setQuota** de **dfsadmin** para cuotas de espacio o de nombres respectivamente.

Podremos ver las cuotas asignadas con el subcomando **–count -q**

```
hadoop@hadoop:~$ hdfs dfsadmin -setSpaceQuota 1g /users/alumno
hadoop@hadoop:~$
```

```
hadoop@hadoop:~$ hdfs dfsadmin -setQuota 1000 /users/alumno
```

```
hadoop@hadoop:~$ hdfs dfs -count -q /users/alumno
        1000             999      1073741824      1073741824            1            0                0 /users/alumno
hadoop@hadoop:~$
```



Permisos

HDFS implementa un modelo de permisos similar al de los sistemas Unix/Linux, basado en usuarios, grupos y permisos de acceso.

Los subcomandos para trabajar con permisos son análogos a los de cualquier sistema Linux: **chmod**, **chown** y **chgrp**