

---

**BIG DATA**

**SISTEMA DE FICHEROS  
HDFS DE HADOOP**

**EDUARD LARA**

# 1. INTRODUCCION HDFS

---

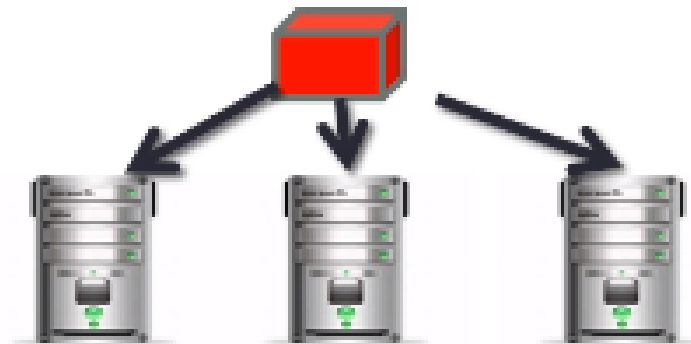
- ❖ Es un sistema de ficheros y almacenamiento
- ❖ Es muy tolerante a fallos
- ❖ Permite almacenar gran cantidad de datos, distribuidos y replicados entre todos los nodos del cluster. Si se cae una maquina voy a tener la seguridad que el dato lo voy a tener en otro
- ❖ Se puede escalar de forma incremental y sobrevivir a fallos de hardware sin perder datos
- ❖ Es la parte de almacenamiento de Datos de Hadop



# 1. INTRODUCCION HDFS

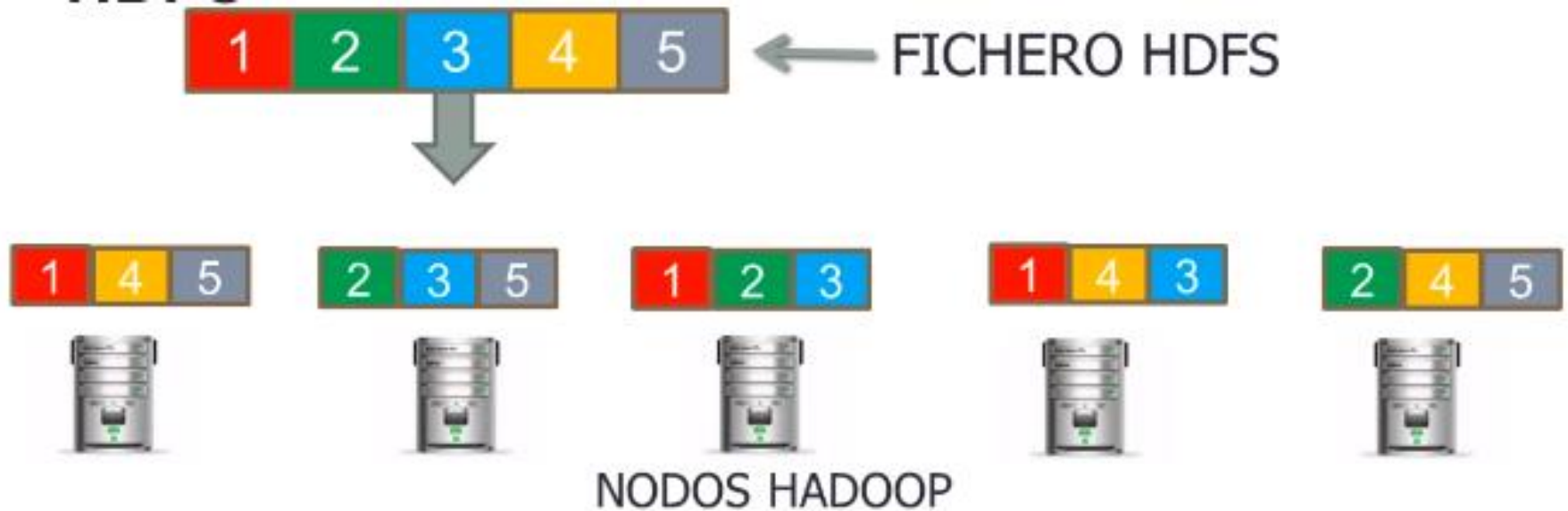
---

- ❖ HDFS gestiona el almacenamiento en el cluster, dividiendo los ficheros en bloques (128 Megas) y almacenando copias duplicadas a través de los nodos
- ❖ Por defecto se replican en 3 nodos distintos
- ❖ Puedo tener mas de tres copias, pero no va a mejorar el rendimiento y va a menguar la capacidad del sistema



# 1. INTRODUCCION HDFS

---



# 1. INTRODUCCION HDFS

---

- ❖ Al crear un cluster hadoop hay:
  - ❖ Un nodo que actúa como maestro de datos. Solo tienen metadatos
  - ❖ El resto de nodos son esclavos . Contiene los datos propiamente dichos



## 2. CREACION CLUSTER PSEUDODISTRIBUIDO

---

### Cluster pseudo-distribuido

- ❖ Instalaremos la parte maestro y la parte esclava en el mismo nodo. Un mismo nodo será Maestro y esclavo a la vez.
- ❖ No habrá replica, ni alta disponibilidad, ni proceso ni nada. En la vida real esto nunca va a pasar. En producción siempre vamos a tener un maestro y múltiples esclavos.

## 2. CREACION CLUSTER PSEUDODISTRIBUIDO

---

**Paso 1.** En el directorio /opt/hadoop/etc/hadoop están los ficheros de configuración, que indicaran a hadoop cómo debe arrancar

```
hadoop@nodol:~$ pwd
/home/hadoop
hadoop@nodol:~$ cd /opt/hadoop/etc/hadoop/
hadoop@nodol:/opt/hadoop/etc/hadoop$ ls
capacity-scheduler.xml      httpfs-env.sh              mapred-env.sh
configuration.xsl           httpfs-log4j.properties   mapred-queues.xml.template
container-executor.cfg      httpfs-signature.secret   mapred-site.xml.template
core-site.xml               httpfs-site.xml           slaves
hadoop-env.cmd              kms-acls.xml               ssl-client.xml.example
hadoop-env.sh               kms-env.sh                 ssl-server.xml.example
hadoop-metrics2.properties kms-log4j.properties      yarn-env.cmd
hadoop-metrics.properties  kms-site.xml               yarn-env.sh
hadoop-policy.xml           log4j.properties          yarn-site.xml
hdfs-site.xml               mapred-env.cmd
```

hadoop@nodol:/opt/hadoop/etc/hadoop\$

## 2. CREACION CLUSTER PSEUDODISTRIBUIDO

---

### Ficheros importantes de hadoop

- ❖ El fichero **core-site.xml** contiene las propiedades, la configuración general del cluster.
- ❖ Luego vamos a ver **hdfs-site.xml** que contiene la configuración para los datos, para el sistema de ficheros HDFS.
- ❖ Luego vamos a ver otro que se llama **mapred-site.xml.template** que es el que contendrá la configuración para Map Reduce.
- ❖ Y luego tenemos el **yarn-site.xml** que nos permitirá configurar el modo de trabajo de proceso yarn.
- ❖ Por ahora nos vale con los dos primeros:
  - ❖ **core-site.xml**
  - ❖ **hdfs-site.xml**



## 2. CREACION CLUSTER PSEUDODISTRIBUIDO

**Paso 2.** Empezaremos con **core-site.xml**. Le vamos a indicar la configuración más básica que queremos utilizar para nuestro cluster. Abrimos core-site.xml y vemos que está vacío. Lo pone hadoop para que lo rellenemos con la información necesaria

```
File Edit View Search Terminal Help
GNU nano 6.2 core-site.xml
<?xml version="1.0" encoding="UTF-8"?>
<?xml-stylesheet type="text/xsl" href="configuration.xsl"?>
<!--
Licensed under the Apache License, Version 2.0 (the "License");
you may not use this file except in compliance with the License.
You may obtain a copy of the License at

http://www.apache.org/licenses/LICENSE-2.0

Unless required by applicable law or agreed to in writing, software
distributed under the License is distributed on an "AS IS" BASIS,
WITHOUT WARRANTIES OR CONDITIONS OF ANY KIND, either express or implied.
See the License for the specific language governing permissions and
limitations under the License. See accompanying LICENSE file.
-->
<!-- Put site-specific property overrides in this file. -->

<configuration>
</configuration>
```

[ Read 20 lines ]

^G Help	^O Write Out	^W Where Is	^K Cut	^T Execute	^C Location
^X Exit	^R Read File	^_ Replace	^U Paste	^J Justify	^_ Go To Line

```
<configuration>
  <property>
    <name>fs.defaultFS</name>
    <value>hdfs://nodo1:9000</value>
  </property>
</configuration>
```

## 2. CREACION CLUSTER PSEUDODISTRIBUIDO

---

- ❖ El fichero tiene que empezar y acabar con el tag configuration.
  - ❖ La única propiedad necesaria para montar nuestro clúster pseudo distribuído es `fs.defaultFS`, que nos indica que sistema de ficheros vamos a utilizar para hadoop.
  - ❖ Utilizaremos el estándar que viene por defecto con hadoop: **hdfs**. En el mundo Hadoop Big Data tenemos algún sistema de ficheros adicional que no vienen incluidos en la distribución
  - ❖ Tenemos que indicar a nuestro sistema hadoop donde localizar al maestro HDFS, es decir el servidor maestro que va a contener los datos, que se suele llamar **NAMENODE**: el maestro de datos.
  - ❖ Estará en el `nodo1` es decir en la máquina donde me encuentro y por el puerto 9000 (no es el por defecto, pero es configurable).
  - ❖ En resumen básicamente se pone el **puerto** y el **tipo de sistema de ficheros**
-

## 2. CREACION CLUSTER PSEUDODISTRIBUIDO

**Paso 3.** El siguiente fichero a tocar es el **hdfs-site.xml**. Indica donde vamos a configurar los datos.

```
File Edit View Search Terminal Help
GNU nano 6.2 hdfs-site.xml
<?xml version="1.0" encoding="UTF-8"?>
<?xml-stylesheet type="text/xsl" href="configuration.xsl"?>
<!--
Licensed under the Apache License, Version 2.0 (the "License");
you may not use this file except in compliance with the License.
You may obtain a copy of the License at

http://www.apache.org/licenses/LICENSE-2.0

Unless required by applicable law or agreed to in writing, software
distributed under the License is distributed on an "AS IS" BASIS,
WITHOUT WARRANTIES OR CONDITIONS OF ANY KIND, either express or implied.
See the License for the specific language governing permissions and
limitations under the License. See accompanying LICENSE file.
-->
<!-- Put site-specific property overrides in this file. -->
<configuration>
</configuration>
[ Read 21 lines ]
^G Help      ^O Write Out ^W Where Is  ^K Cut       ^T Execute   ^C U
^X Exit      ^R Read File ^\ Replace   ^U Paste     ^J Justify   ^/_ G
u 22.04 0:-* 56! 4h14m 0.75 2x2.5GHz 3.8613% 24G54% 2023-01-
```

```
<configuration>
  <property>
    <name>dfs.replication</name>
    <value>1</value>
  </property>
  <property>
    <name>dfs.namenode.name.dir</name>
    <value>/datos/namenode</value>
  </property>
  <property>
    <name>dfs.datanode.data.dir</name>
    <value>/datos/datanode</value>
  </property>
</configuration>
```

## 2. CREACION CLUSTER PSEUDODISTRIBUIDO

---

- ❖ También contiene un tag configuration donde indicaremos la configuración con la que trabajar a nivel de datos, a nivel de hdfs.
  - ❖ El primer parámetro dfs.replication a 1, indica que sólo va a haber un nodo de datos, sin réplica. Normalmente será 3, donde cada bloque de datos (ficheros) se replicara en tres nodos distintos.
  - ❖ El segundo parámetro indica dónde (directorio) se encuentra la información (metadatos) del maestro (en /datos/namenode). Un maestro sólo tiene metadatos, que es la info general del cluster. Nuestro cluster está formado por un maestro llamado namenode.
  - ❖ El tercer parámetro dfs.datanode indica donde (directorio) se guardan los bloques de datos reales en cada esclavo.
  - ❖ Cuando tengamos ya un cluster con varios nodos:
    - ❖ en el maestro sólo estará /datos/namenode/
    - ❖ en los esclavos estará /datos/datanode/
-

## 2. CREACION CLUSTER PSEUDODISTRIBUIDO

---

**Paso 4.** Una vez configurados los dos ficheros básicos, prepararemos el sistema de ficheros HDFS para poder arrancar nuestro cluster. Al tratarse de un cluster pseudodistribuido, siendo el mismo nodo maestro y esclavo, crearemos los directorios de `hdfs-site.xml`:

- ❖ `/datos/namenode/` → para los metadatos del maestro
- ❖ `/datos/datanode/` → para los datos del esclavo

```
File Edit View Search Terminal Help
hadoop@nodol:/opt/hadoop/etc/hadoop$ mkdir -p /datos/namenode
mkdir: cannot create directory '/datos': Permission denied
1 hadoop@nodol:/opt/hadoop/etc/hadoop$ sudo mkdir -p /datos/namenode
hadoop@nodol:/opt/hadoop/etc/hadoop$ sudo mkdir -p /datos/datanode
hadoop@nodol:/opt/hadoop/etc/hadoop$ chown -R hadoop:hadoop datos
chown: cannot access 'datos': No such file or directory
1 hadoop@nodol:/opt/hadoop/etc/hadoop$ sudo chown -R hadoop:hadoop /datos
hadoop@nodol:/opt/hadoop/etc/hadoop$
```

```
hadoop@nodol:/opt/hadoop/etc/hadoop$ ls -l /
total 4019280
lrwxrwxrwx    1 root    root          7 ago   9 11:53 bin -> usr/bin
drwxr-xr-x    4 root    root        4096 ene 11 08:27 boot
drwxr-xr-x    4 hadoop  hadoop      4096 ene 11 10:28 datos
drwxr-xr-x   20 root    root        4180 ene  8 07:48 dev
drwxr-xr-x  133 root    root       12288 ene 11 08:30 etc
drwxr-xr-x    3 root    root        4096 ene  1 19:38 home
lrwxrwxrwx    1 root    root          7 ago   9 11:53 lib -> usr/lib
lrwxrwxrwx    1 root    root          9 ago   9 11:53 lib32 -> usr/lib32
lrwxrwxrwx    1 root    root          9 ago   9 11:53 lib64 -> usr/lib64
lrwxrwxrwx    1 root    root         10 ago   9 11:53 libx32 -> usr/libx32
drwx-----   2 root    root       16384 ene  1 18:16 lost+found
```

### 3. ARRANCAR HDFS

---

**Paso 1.** Antes de poder hacer cualquier operación con hadoop tenemos que crear el sistema de ficheros HDFS

- ❖ En versiones inferiores a 2.7:
  - ❖ **hadoop namenode -format**
- ❖ Desde la versión 2.7 se recomienda usar:
  - ❖ **hdfs namenode -format**

```
hadoop@nodol1:/datos/namenode$ hdfs namenode -format
```

```
23/01/11 10:49:08 INFO namenode.FSImage: FSImageSaver clean checkpoint: txid
when meet shutdown.
23/01/11 10:49:08 INFO namenode.NameNode: SHUTDOWN_MSG:
/*****
SHUTDOWN_MSG: Shutting down NameNode at nodol1/127.0.1.1
*****/
hadoop@nodol1:/opt/hadoop/etc/hadoop$
```

### 3. ARRANCAR HDFS

---

**Paso 2.** En el directorio /datos/namenode se crea el sistema de ficheros hdfs de los metadatos del maestro.

- Dentro se ha creado un directorio llamado current. Aquí encontramos el sistema de ficheros hdfs de metadatos
- En el directorio /datos/datanode no hay nada, está vacío, porque no se ha cargado ningún tipo de fichero

```
hadoop@nodo1:/opt/hadoop/etc/hadoop$ cd /datos/namenode
hadoop@nodo1:/datos/namenode$ ls -l
total 4
drwxrwxr-x 2 hadoop hadoop 4096 ene 11 10:49 current
hadoop@nodo1:/datos/namenode$ ls -l current/
total 16
-rw-rw-r-- 1 hadoop hadoop 325 ene 11 10:49 fsimage_00000000000000000000
-rw-rw-r-- 1 hadoop hadoop 62 ene 11 10:49 fsimage_00000000000000000000.md5
-rw-rw-r-- 1 hadoop hadoop 2 ene 11 10:49 seen_txid
-rw-rw-r-- 1 hadoop hadoop 213 ene 11 10:49 VERSION
hadoop@nodo1:/datos/namenode$ ls -l /datos/datanode
total 0
hadoop@nodo1:/datos/namenode$
```



# 3. ARRANCAR HDFS

---

**Paso 3.** Una vez creados los directorios y realizado el formateo del sistema de ficheros, ahora arrancaremos la parte de datos HDFS.

Recordar que Hadoop tiene dos partes: datos y procesos.

En /opt/hadoop/sbin tenemos una serie de scripts que nos permiten arrancar, parar y hacer operaciones automáticas.

```
hadoop@nodol1:~$ cd /opt/hadoop/sbin
hadoop@nodol1:/opt/hadoop/sbin$ ls
distribute-exclude.sh  slaves.sh  stop-all.sh
FederationStateStore  start-all.cmd  stop-balancer.sh
hadoop-daemon.sh       start-all.sh  stop-dfs.cmd
hadoop-daemons.sh     start-balancer.sh  stop-dfs.sh
hdfs-config.cmd        start-dfs.cmd  stop-secure-dns.sh
hdfs-config.sh         start-dfs.sh  stop-yarn.cmd
httpfs.sh              start-secure-dns.sh  stop-yarn.sh
kms.sh                 start-yarn.cmd  yarn-daemon.sh
mr-jobhistory-daemon.sh  start-yarn.sh  yarn-daemons.sh
refresh-namenodes.sh    stop-all.cmd
hadoop@nodol1:/opt/hadoop/sbin$ start-dfs.sh
Starting namenodes on [nodol1]
nodol1: Error: JAVA_HOME is not set and could not be found.
localhost: Error: JAVA_HOME is not set and could not be found.
Starting secondary namenodes [0.0.0.0]
0.0.0.0: Error: JAVA_HOME is not set and could not be found.
hadoop@nodol1:/opt/hadoop/sbin$
```

Para arrancar el sistema de datos HDFS

lanzaremos el comando

**start-dfs.sh**

Para parar el sistema HDFS esta el comando

**stop-dfs.sh**



### 3. ARRANCAR HDFS

---

**Paso 4.** Ha ocurrido un error y es que desde dentro de hadoop es como sino reconociera la variable de entorno JAVA\_HOME

- ❖ En el fichero hadoop-env.sh, se debe de modificar la línea  
`export JAVA_HOME=${JAVA_HOME}`
- ❖ Para buscar este fichero ejecutamos  
`sudo find / -name hadoop-env.sh`

```
130 hadoop@nodo1:/opt/hadoop/sbin$ sudo find / -name hadoop-env.sh
/opt/hadoop_3.3.4/etc/hadoop/hadoop-env.sh
/opt/hadoop/etc/hadoop/hadoop-env.sh
find: '/run/user/1000/doc': Permission denied
find: '/run/user/1000/gvfs': Permission denied
1 hadoop@nodo1:/opt/hadoop/sbin$ cd ..
hadoop@nodo1:/opt/hadoop$ cd etc/hadoop
hadoop@nodo1:/opt/hadoop/etc/hadoop$ sudo nano hadoop-env.sh
```

```
File Edit View Search Terminal Help
GNU nano 6.2 hadoop-env.sh

# The java implementation to use.
#export JAVA_HOME=${JAVA_HOME}
export JAVA_HOME=/usr/lib/jvm/java-8-openjdk-amd64

# The jsvc implementation to use. Jsvc is required to run s
# that bind to privileged ports to provide authentication d
```

### 3. ARRANCAR HDFS

---

**Paso 5.** Volvemos a arrancar HDFS (la parte de datos) y ahora si hace todo lo que le hemos dicho en los ficheros de configuración.

- ❖ Luego tendremos que hacer la parte de procesos Hadoop mediante MapReduce u otras metodologías
- ❖ Va a arrancar 3 procesos:
  - El que gestiona **namenode**, es decir el que trabaja con los metadatos
  - el que gestiona **datanode**.
  - el que gestiona el **namenode secundario**

```
hadoop@nodo1:/opt/hadoop/sbin$ start-dfs.sh
Starting namenodes on [nodo1]
nodo1: starting namenode, logging to /opt/hadoop/logs/hadoop-hadoop-namenode-nodo1.out
localhost: starting datanode, logging to /opt/hadoop/logs/hadoop-hadoop-datanode-nodo1.out
Starting secondary namenodes [0.0.0.0]
0.0.0.0: starting secondarynamenode, logging to /opt/hadoop/logs/hadoop-hadoop-secondarynamenode-nodo1.out
hadoop@nodo1:/opt/hadoop/sbin$
```

### 3. ARRANCAR HDFS

---

**Paso 6.** Mediante el comando **jps** (dentro del JDK de Java) podemos ver qué procesos java tenemos en la máquina. Tenemos:

- el Datanode,
- el Secondary namenode
- y el namenode.

Con `jps -l` tenemos un poco mas de información:

```
hadoop@nodol:/opt/hadoop/sbin$ jps
95249 DataNode
95612 SecondaryNameNode
103487 Jps
95054 NameNode
hadoop@nodol:/opt/hadoop/sbin$ jps -l
106721 sun.tools.jps.Jps
95249 org.apache.hadoop.hdfs.server.datanode.DataNode
95612 org.apache.hadoop.hdfs.server.namenode.SecondaryNameNode
95054 org.apache.hadoop.hdfs.server.namenode.NameNode
hadoop@nodol:/opt/hadoop/sbin$
```

### 3. ARRANCAR HDFS

**Paso 7.** Mediante el comando **ps -ef | grep java** obtenemos lo mismo que nos dice el comando **jps** pero con más detalle

Sale un conjunto de 3 procesos java: namenode, datanode y otro proceso Secondary namenode (lo mismo que **jps** pero con mas datos)

```
File Edit View Search Terminal Help
-Dhadoop.log.dir=/opt/hadoop/logs -Dhadoop.log.file=hadoop-hadoop-datanode-nodo
1.log -Dhadoop.home.dir=/opt/hadoop -Dhadoop.id.str=hadoop -Dhadoop.root.logger=
INFO,RFA -Djava.library.path=/opt/hadoop/lib/native -Dhadoop.policy.file=hadoop-
policy.xml -Djava.net.preferIPv4Stack=true -server -Dhadoop.security.logger=ERRO
R,RFAS -Dhadoop.security.logger=ERROR,RFAS -Dhadoop.security.logger=ERROR,RFAS -
Dhadoop.security.logger=INFO,RFAS org.apache.hadoop.hdfs.server.datanode.DataNod
e
hadoop      95612      1  0 20:04 ?          00:00:22 /usr/lib/jvm/java-8-openjdk-
amd64/bin/java -Dproc_secondarynamenode -Xmx1000m -Djava.net.preferIPv4Stack=tru
e -Dhadoop.log.dir=/opt/hadoop/logs -Dhadoop.log.file=hadoop.log -Dhadoop.home.d
ir=/opt/hadoop -Dhadoop.id.str=hadoop -Dhadoop.root.logger=INFO,console -Djava.l
ibrary.path=/opt/hadoop/lib/native -Dhadoop.policy.file=hadoop-policy.xml -Djava
.net.preferIPv4Stack=true -Djava.net.preferIPv4Stack=true -Djava.net.preferIPv4S
tack=true -Dhadoop.log.dir=/opt/hadoop/logs -Dhadoop.log.file=hadoop-hadoop-seco
ndarynamenode-nodo1.log -Dhadoop.home.dir=/opt/hadoop -Dhadoop.id.str=hadoop -Dh
adoop.root.logger=INFO,RFA -Djava.library.path=/opt/hadoop/lib/native -Dhadoop.p
olicy.file=hadoop-policy.xml -Djava.net.preferIPv4Stack=true -Dhadoop.security.l
ogger=INFO,RFAS -Dhdfs.audit.logger=INFO,NullAppender -Dhadoop.security.logger=I
NFO,RFAS -Dhdfs.audit.logger=INFO,NullAppender -Dhadoop.security.logger=INFO,RFA
S -Dhdfs.audit.logger=INFO,NullAppender -Dhadoop.security.logger=INFO,RFAS org.a
pache.hadoop.hdfs.server.namenode.SecondaryNameNode
hadoop      168373     1767  0 21:24 pts/1    00:00:00 grep --color=auto java
hadoop@nodol:/opt/hadoop/sbin$
```

### 3. ARRANCAR HDFS

---

**Paso 8.** Podemos comprobar que en /datos/namenode/current tenemos una serie de ficheros dentro del directorio current

```
hadoop@nodol1:/opt/hadoop/sbin$ cd /datos/namenode
hadoop@nodol1:/datos/namenode$ ls
current  in_use.lock
hadoop@nodol1:/datos/namenode$ ls current
edits_00000000000000000001-00000000000000000002  fsimage_00000000000000000004.md5
edits_00000000000000000003-00000000000000000004  fsimage_00000000000000000006
edits_00000000000000000005-00000000000000000006  fsimage_00000000000000000006.md5
edits_inprogress_00000000000000000007             seen_txid
fsimage_00000000000000000004                       VERSION
hadoop@nodol1:/datos/namenode$
```

En /datos/datanode encontramos cosas nuevas. Anteriormente no habíamos arrancado ningún nodo de datos. Ahora ya sí. Tenemos un directorio current con un fichero con una serie de números

```
hadoop@nodol1:/datos/namenode$ cd /datos/datanode
hadoop@nodol1:/datos/datanode$ ls
current  in_use.lock
hadoop@nodol1:/datos/datanode$ ls current
BP-703191218-127.0.1.1-1673434148125  VERSION
hadoop@nodol1:/datos/datanode$ ls -l current
total 8
drwx----- 4 hadoop hadoop 4096 ene 14 20:04 BP-703191218-127.0.1.1-1673434148125
-rw-rw-r-- 1 hadoop hadoop 229 ene 14 20:04 VERSION
hadoop@nodol1:/datos/datanode$
```

# 4. WEB ADMINISTRACION HDFS

**Paso 1.** Para comprobar que nuestro entorno Hadoop ha arrancado correctamente a nivel de datos, en un navegador ponemos **localhost:50070** o **nodo1:50070** (nodo1 es el nombre real de la máquina donde acabamos de instalar y arrancar este clúster).

Esta web nos permite administrar todos los recursos iniciados y hacer un recorrido sobre toda la estructura de datos del clúster.

The screenshot shows the Hadoop web interface in a browser window titled 'Namenode information'. The address bar shows 'localhost:50070/dfshealth.html#tab-overview'. The interface has a green header with 'Hadoop' and a navigation menu with 'Overview' (selected), 'Datanodes', 'Datanode Volume Failures', 'Snapshot', and 'Startup Progress'. Below the header, the main content area is titled 'Overview 'nodo1:9000' (active)'. It contains a table with the following information:

<b>Started:</b>	Sat Jan 14 20:04:07 +0000 2023
<b>Version:</b>	2.10.2, r965fd380006fa78b2315668fbc7eb432e1d8200f
<b>Compiled:</b>	Tue May 24 22:35:00 +0000 2022 by ubuntu from branch-2.10.2
<b>Cluster ID:</b>	CID-18867706-b47d-445c-9ff2-53ed060664b6
<b>Block Pool ID:</b>	BP-703191218-127.0.1.1-1673434148125

The screenshot shows the same Hadoop web interface as the previous one, but the browser window title is 'Namenode information' and the address bar shows 'nodo1:50070/dfshealth.html#tab-overview'. The interface is identical, showing the 'Overview' tab for 'nodo1:9000' (active) with the same table of information:

<b>Started:</b>	Sat Jan 14 20:04:07 +0000 2023
<b>Version:</b>	2.10.2, r965fd380006fa78b2315668fbc7eb432e1d8200f
<b>Compiled:</b>	Tue May 24 22:35:00 +0000 2022 by ubuntu from branch-2.10.2
<b>Cluster ID:</b>	CID-18867706-b47d-445c-9ff2-53ed060664b6
<b>Block Pool ID:</b>	BP-703191218-127.0.1.1-1673434148125

## 4. WEB ADMINISTRACION HDFS

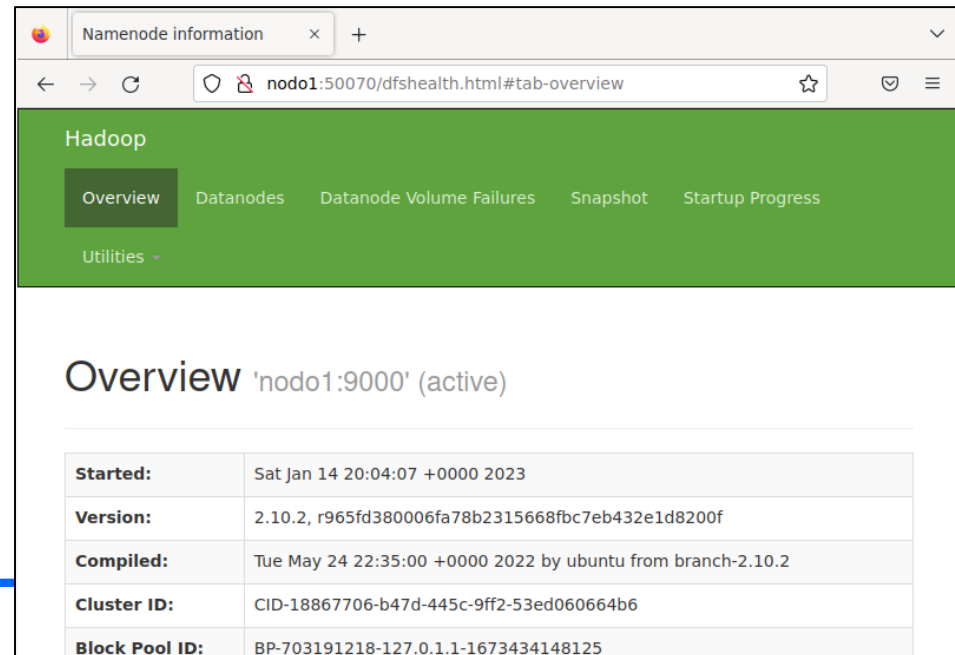
**Paso 2.** En la primera **pestaña Overview** podemos ver que la llamada a esta pagina web se ha realizado con los parámetros indicados en el fichero core-site.xml: **nodo1(maquina):9000(puerto)**

**NOTA:** Para acceder a la Web de Administración:

Versión 2 de Hadoop → **nodo1:50070**

Version 3 de Hadoop → **nodo1:9870**

```
<configuration>
  <property>
    <name>fs.defaultFS</name>
    <value>hdfs://nodo1:9000</value>
  </property>
</configuration>
```



The screenshot shows a web browser window with the title "Namenode information". The address bar displays "nodo1:50070/dfshealth.html#tab-overview". The page has a green header with the "Hadoop" logo and navigation tabs: "Overview" (selected), "Datanodes", "Datanode Volume Failures", "Snapshot", and "Startup Progress". Below the header, there is a section titled "Overview 'nodo1:9000' (active)". This section contains a table with the following information:

<b>Started:</b>	Sat Jan 14 20:04:07 +0000 2023
<b>Version:</b>	2.10.2, r965fd380006fa78b2315668fbc7eb432e1d8200f
<b>Compiled:</b>	Tue May 24 22:35:00 +0000 2022 by ubuntu from branch-2.10.2
<b>Cluster ID:</b>	CID-18867706-b47d-445c-9ff2-53ed060664b6
<b>Block Pool ID:</b>	BP-703191218-127.0.1.1-1673434148125

## 4. WEB ADMINISTRACION HDFS

---

**Paso 3.** La primera **pestaña Overview** muestra una serie de información bastante útil general de cómo está nuestro sistema:

- Cuando se inició
- Versión de Hadoop con la que estamos trabajando, etc

### Overview 'nodo1:9000' (active)

<b>Started:</b>	Sat Jan 14 20:04:07 +0000 2023
<b>Version:</b>	2.10.2, r965fd380006fa78b2315668fbc7eb432e1d8200f
<b>Compiled:</b>	Tue May 24 22:35:00 +0000 2022 by ubuntu from branch-2.10.2
<b>Cluster ID:</b>	CID-18867706-b47d-445c-9ff2-53ed060664b6
<b>Block Pool ID:</b>	BP-703191218-127.0.1.1-1673434148125



## 4. WEB ADMINISTRACION HDFS

---

**Paso 4.** La **pestaña Overview** también muestra un resumen de cómo está ahora HDFS:

- Tenemos 0 bloques (solo hemos arrancado sin subir nada).
- Memoria usada
- Capacidad de todo el disco (23,45 GB). A través de hdfs-site.xml se le puede poner un límite a la parte que tiene que ver con hadoop
- Tenemos 13 GB ocupado por el resto que no es DFS (S.O.)

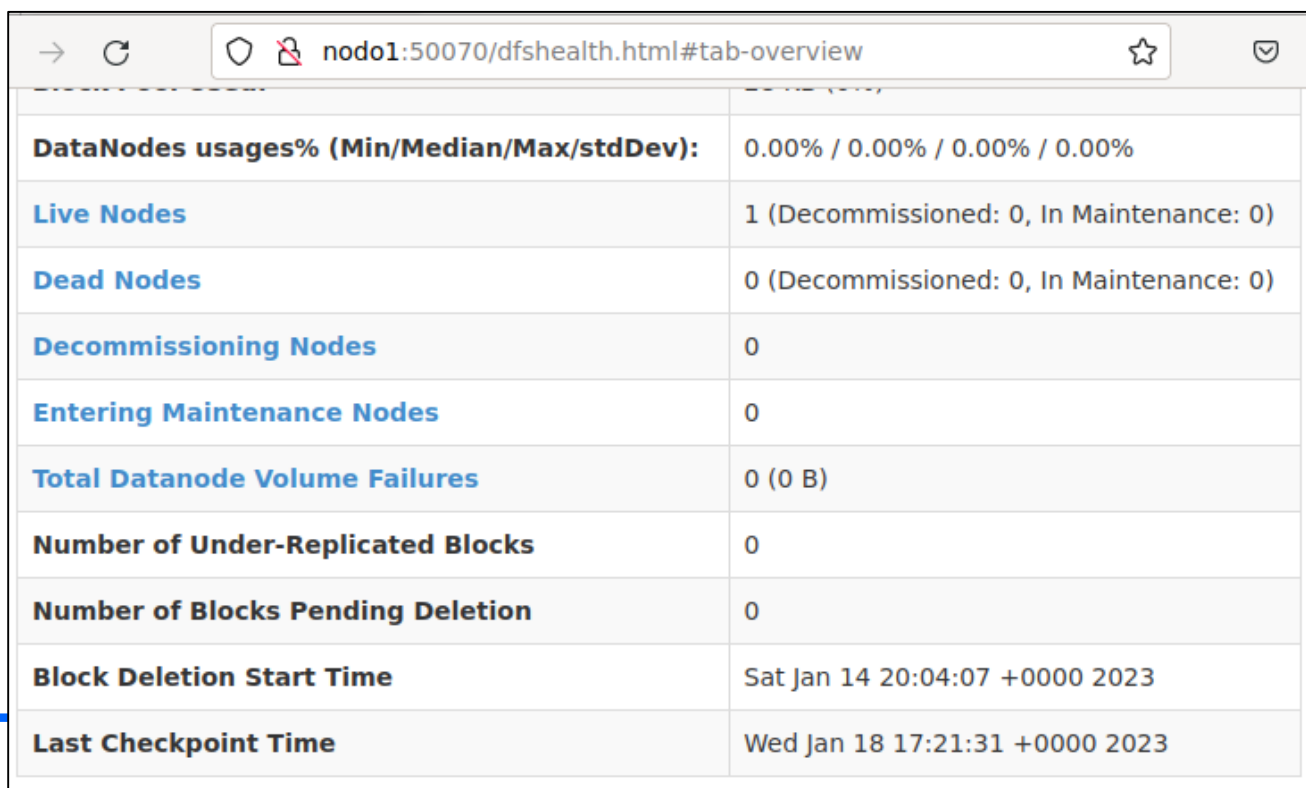
Summary	
Security is off.	
Safemode is off.	
1 files and directories, 0 blocks = 1 total filesystem object(s).	
Heap Memory used 105.2 MB of 189 MB Heap Memory. Max Heap Memory is 889 MB.	
Non Heap Memory used 50.33 MB of 51.52 MB Committed Non Heap Memory. Max Non Heap Memory is <unbounded>.	
Configured Capacity:	23.45 GB
DFS Used:	28 KB (0%)
Non DFS Used:	12.81 GB
DFS Remaining:	9.43 GB (40.19%)
Block Pool Used:	28 KB (0%)
DataNodes usages% (Min/Median/Max/stdDev):	0.00% / 0.00% / 0.00% / 0.00%

- Nos quedan 9 GB libres para poner datos Hadoop.
- Ahora tenemos 28kB

## 4. WEB ADMINISTRACION HDFS

Paso 5. La **pestaña Overview** también nos dice:

- Nodos vivos (solo tenemos un nodo vivo, en el que nos encontramos ahora mismo)
- Nodos con problemas o que estén muertos/apartados (ninguno)

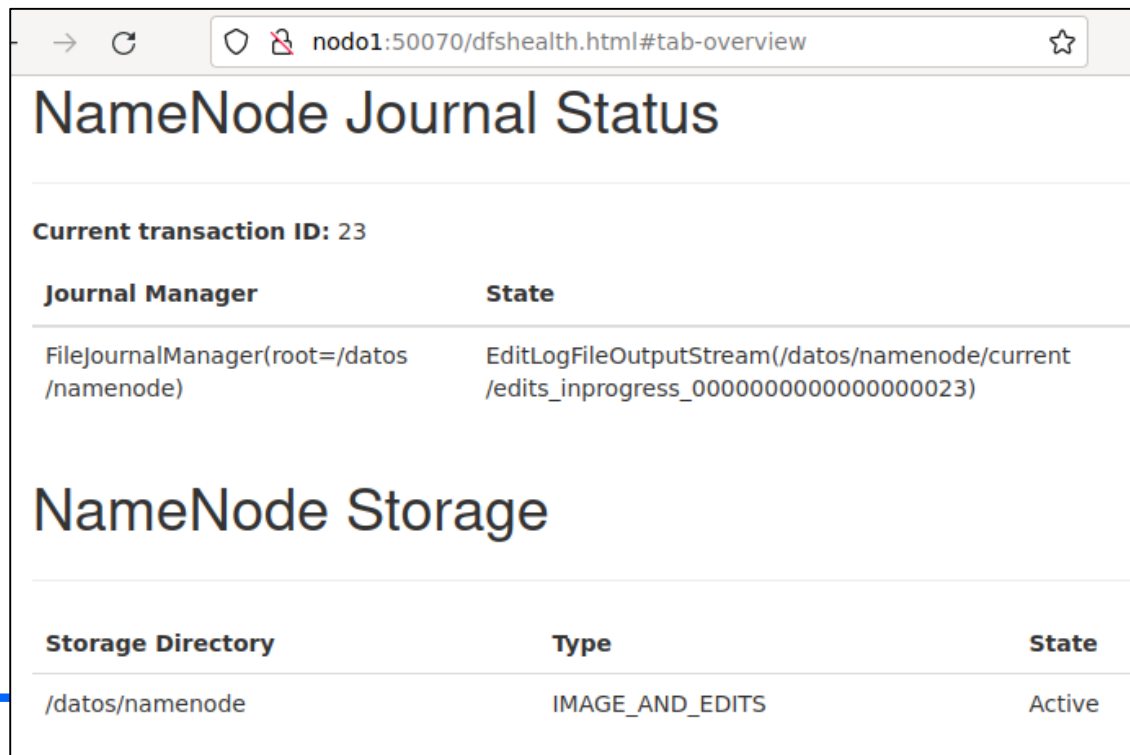


<b>DataNodes usages% (Min/Median/Max/stdDev):</b>	0.00% / 0.00% / 0.00% / 0.00%
<b>Live Nodes</b>	1 (Decommissioned: 0, In Maintenance: 0)
<b>Dead Nodes</b>	0 (Decommissioned: 0, In Maintenance: 0)
<b>Decommissioning Nodes</b>	0
<b>Entering Maintenance Nodes</b>	0
<b>Total Datanode Volume Failures</b>	0 (0 B)
<b>Number of Under-Replicated Blocks</b>	0
<b>Number of Blocks Pending Deletion</b>	0
<b>Block Deletion Start Time</b>	Sat Jan 14 20:04:07 +0000 2023
<b>Last Checkpoint Time</b>	Wed Jan 18 17:21:31 +0000 2023

## 4. WEB ADMINISTRACION HDFS

**Paso 6.** La **pestaña Overview** nos habla de dos ficheros que están en /datos/namenode que son los journals.

Son los ficheros que conforman los metadatos de HDFS es decir cómo él va guardando y gestionando la información de los datos según se van cargando.



The screenshot shows a web browser window with the URL `nodo1:50070/dfshealth.html#tab-overview`. The page is titled "NameNode Journal Status" and displays the "Current transaction ID: 23". Below this, there is a table with two columns: "Journal Manager" and "State". The table contains one row with the following data:

Journal Manager	State
FileJournalManager(root=/datos/namenode)	EditLogFileOutputStream(/datos/namenode/current/edits_inprogress_0000000000000000023)

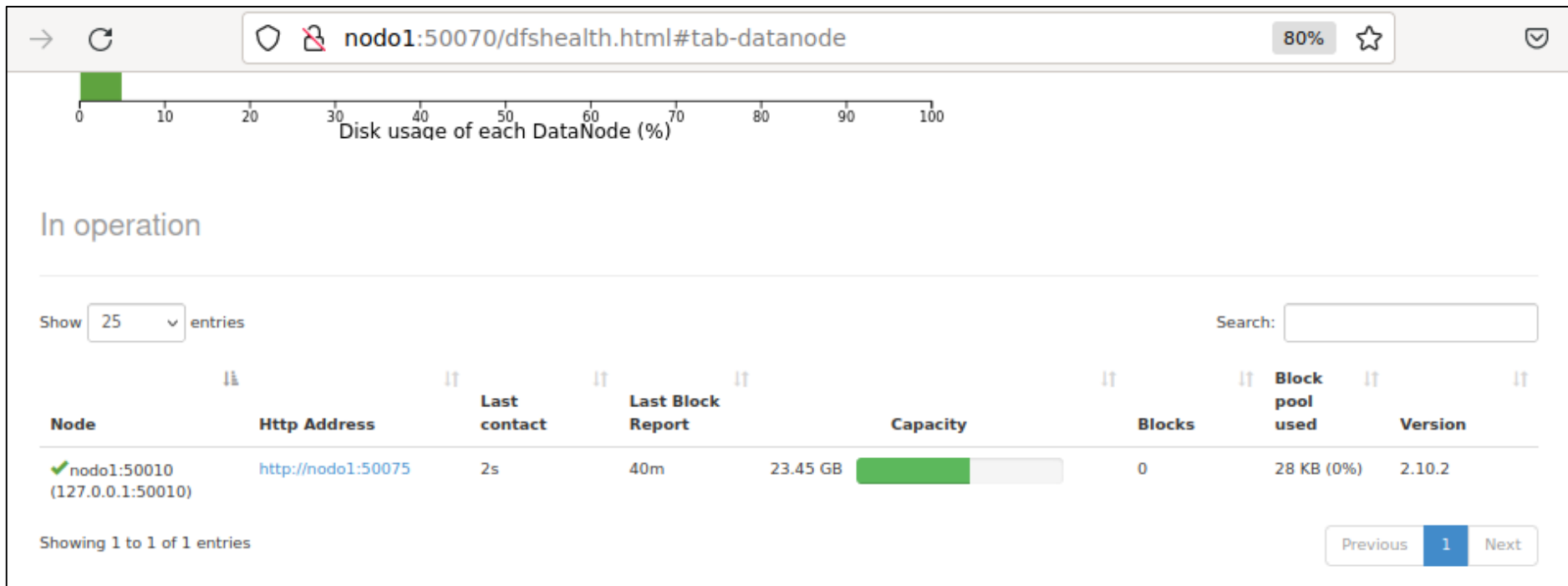
Below the first table, there is another section titled "NameNode Storage". It contains a table with three columns: "Storage Directory", "Type", and "State". The table contains one row with the following data:

Storage Directory	Type	State
/datos/namenode	IMAGE_AND_EDITS	Active

# 4. WEB ADMINISTRACION HDFS

**Paso 7.** En la **pestaña Datanodes** (Datanode Information) podemos comprobar:

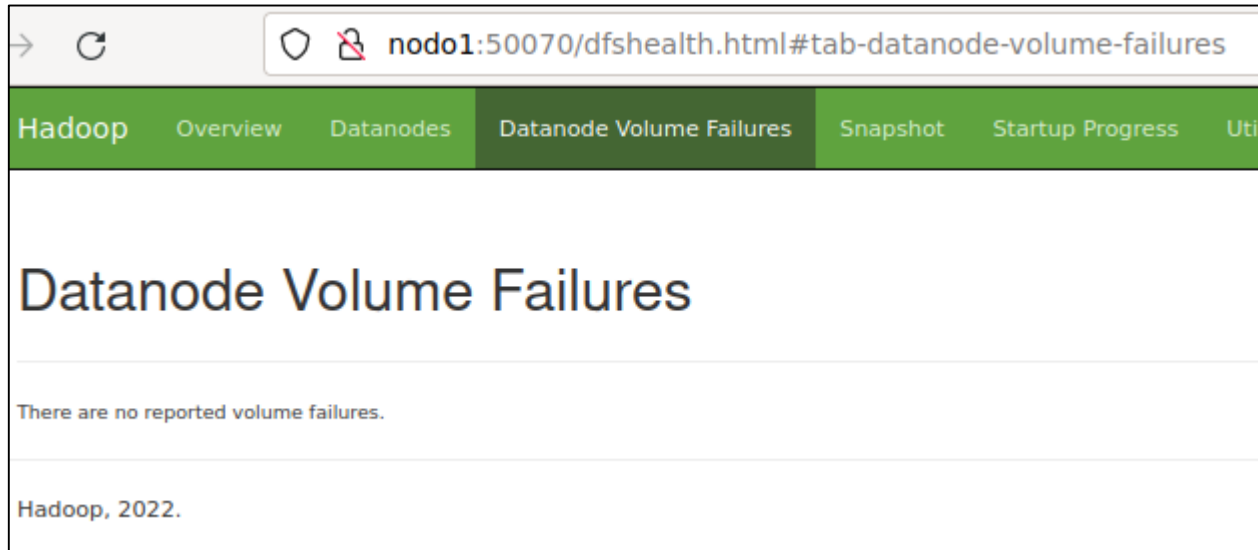
- los nodos que en este momento están activos (solo hay uno)
- la última vez que se le ha preguntado si esta vivo (si se le ha hecho ping)



## 4. WEB ADMINISTRACION HDFS

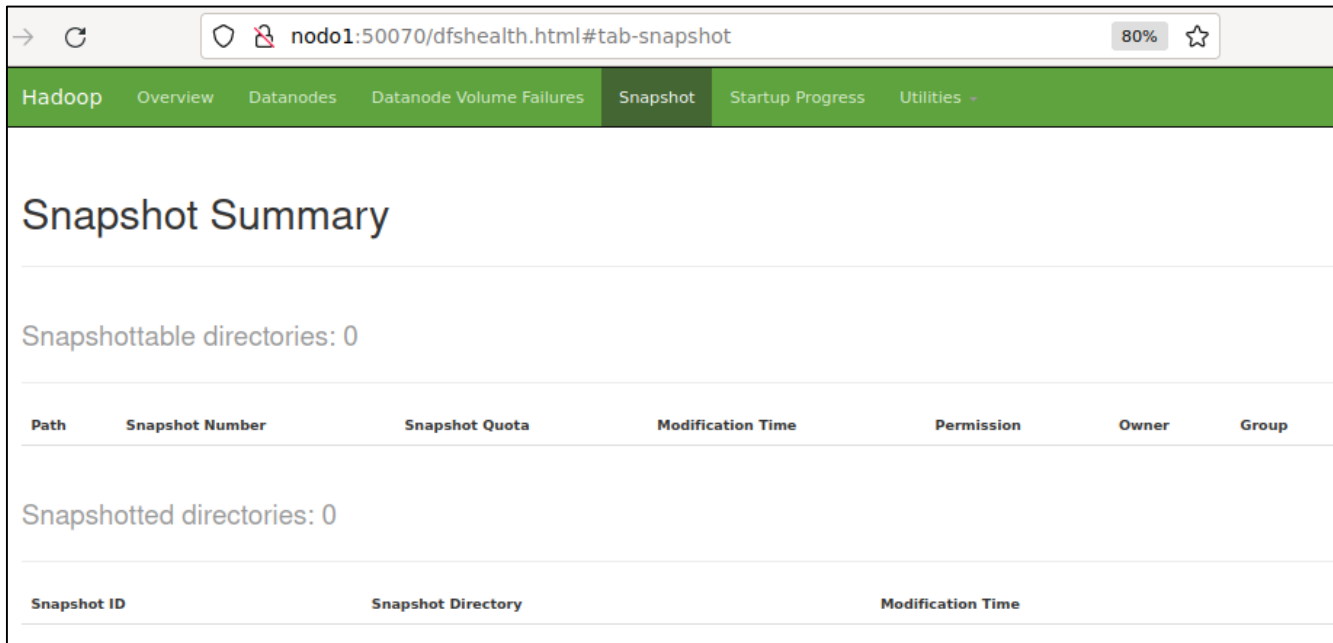
---

**Paso 8.** En la **pestaña Datanode Volume Failures** podemos ver si hay algún problema con algunos de los volúmenes de HFS que no es el caso



## 4. WEB ADMINISTRACION HDFS

**Paso 9.** En la **pestaña Snapshot** vemos los snapshots que tenemos del sistema: son como instantáneas o fotos que podemos tomar de nuestro sistema de ficheros



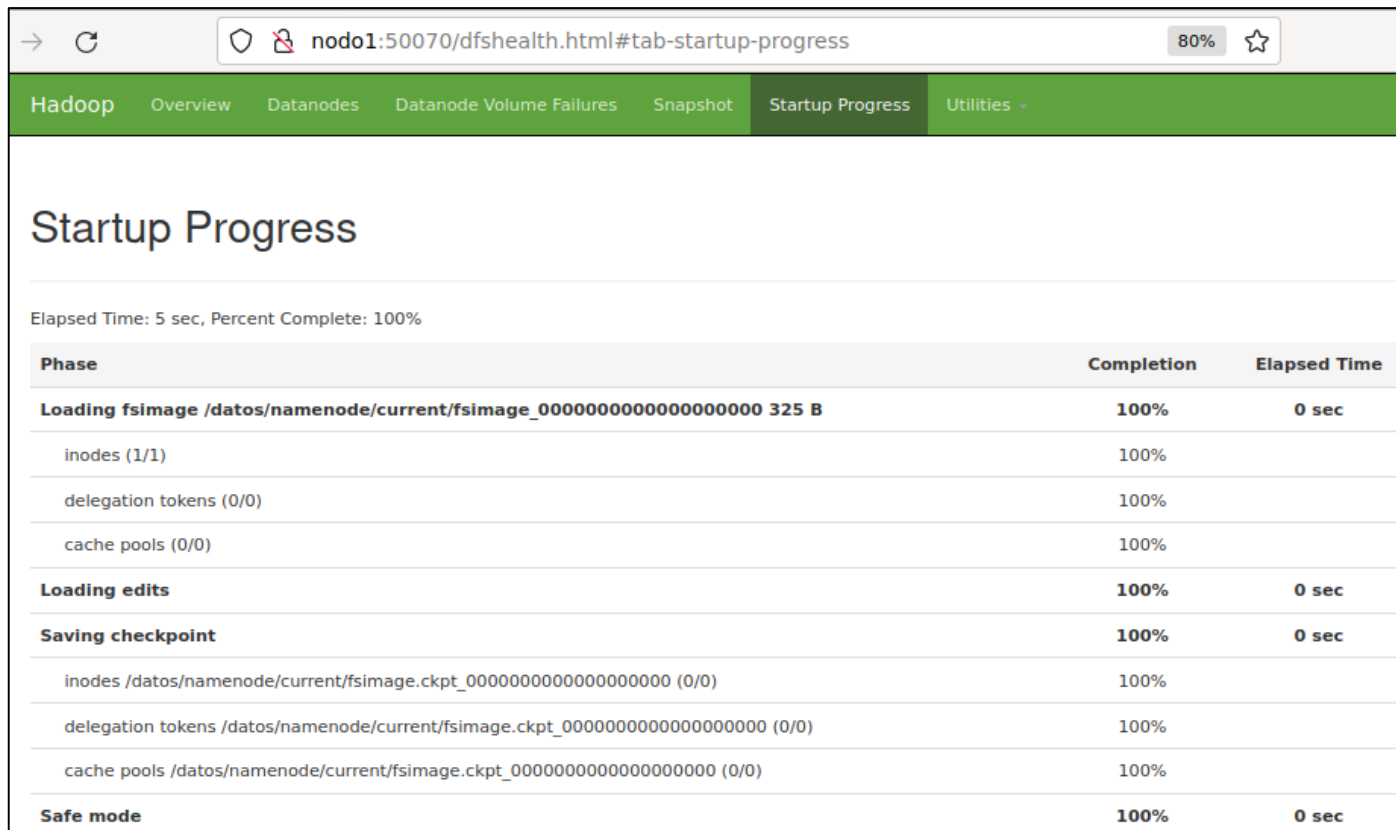
The screenshot shows a web browser window with the address bar displaying 'nodo1:50070/dfshealth.html#tab-snapshot'. The browser's address bar also shows a 80% zoom level and a star icon. The page has a green navigation bar with the following tabs: Hadoop, Overview, Datanodes, Datanode Volume Failures, Snapshot (selected), Startup Progress, and Utilities. The main content area is titled 'Snapshot Summary'. Below the title, there are two sections: 'Snapshottable directories: 0' and 'Snapshotted directories: 0'. Below these sections, there are two tables. The first table has columns: Path, Snapshot Number, Snapshot Quota, Modification Time, Permission, Owner, and Group. The second table has columns: Snapshot ID, Snapshot Directory, and Modification Time.

Path	Snapshot Number	Snapshot Quota	Modification Time	Permission	Owner	Group
Snapshottable directories: 0						

Snapshot ID	Snapshot Directory	Modification Time
Snapshotted directories: 0		

## 4. WEB ADMINISTRACION HDFS

Paso 10. En la **pestaña Startup Progress** vemos el progreso de arranque de nuestro sistema

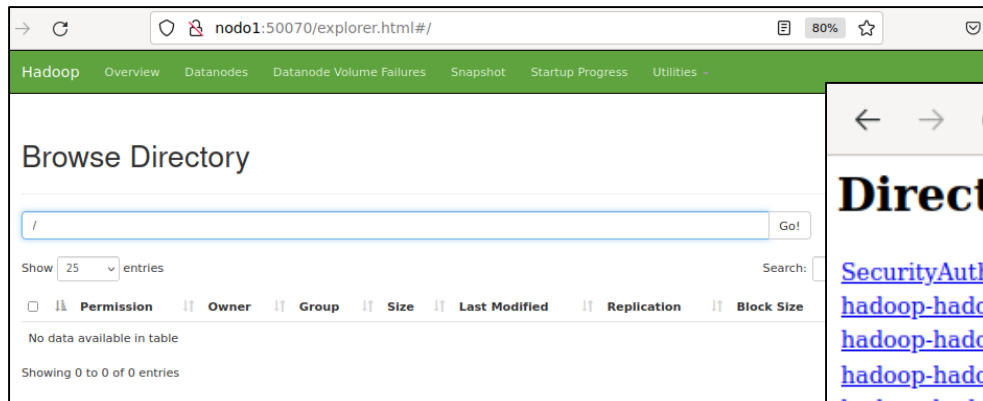
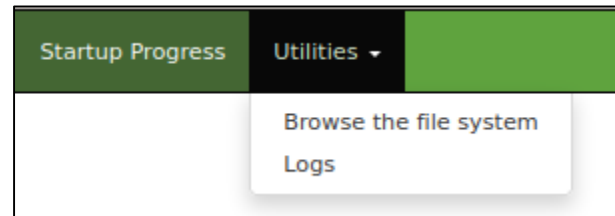


Phase	Completion	Elapsed Time
<b>Loading fsimage /datos/namenode/current/fsimage_00000000000000000000 325 B</b>	<b>100%</b>	<b>0 sec</b>
inodes (1/1)	100%	
delegation tokens (0/0)	100%	
cache pools (0/0)	100%	
<b>Loading edits</b>	<b>100%</b>	<b>0 sec</b>
<b>Saving checkpoint</b>	<b>100%</b>	<b>0 sec</b>
inodes /datos/namenode/current/fsimage.ckpt_00000000000000000000 (0/0)	100%	
delegation tokens /datos/namenode/current/fsimage.ckpt_00000000000000000000 (0/0)	100%	
cache pools /datos/namenode/current/fsimage.ckpt_00000000000000000000 (0/0)	100%	
<b>Safe mode</b>	<b>100%</b>	<b>0 sec</b>

# 4. WEB ADMINISTRACION HDFS

Paso 11. La **pestaña Utilities** nos permite dos cosas básicamente:

- Hacer un Browse del sistema de ficheros es decir poder mirar los ficheros que estén dentro de nuestro de nuestro HFS
- Ver los ficheros de log del datanode, namenode y del secondarynamenode





## 4. WEB ADMINISTRACION HDFS

---

**Paso 12.** En general, la web de administración HDFS nos sirve como administradores para poder comprobar el estado en el que se encuentra nuestro HDFS:

- Localizar información muy útil sobre el espacio que nos queda
- Ver posibles fallos
- Acceder al sistema de ficheros para ver el contenido que tenemos.

# 5. ESTRUCTURA DIRECTORIOS HDFS

---

**Paso 1.** Vamos a ver como es la estructura de directorios que tenemos dentro de los metadatos de Hadoop y cómo va gestionando. Vamos a **/datos/namenode** donde al ser maestro de datos está guardada la información de los metadatos.

- **current** → directorio donde están los metadatos
- **in\_use.lock** → indica que tenemos nuestro hadoop levantado

```
hadoop@nodo1:/datos/namenode$ pwd
/datos/namenode
hadoop@nodo1:/datos/namenode$ ls -l
total 8
drwxrwxr-x 2 hadoop hadoop 4096 ene 30 16:00 current
-rw-rw-r-- 1 hadoop hadoop  11 ene 14 20:04 in_use.lock
hadoop@nodo1:/datos/namenode$ jps
95249 DataNode
391278 Jps
95612 SecondaryNameNode
95054 NameNode
hadoop@nodo1:/datos/namenode$
```

Con **jps** comprobamos tenemos activos los 3 procesos datanode, secondarynamenode y el namenode

---

# 5. ESTRUCTURA DIRECTORIOS HDFS

**Paso 2.** En el directorio current tenemos una serie de ficheros que implementan la funcionalidad de trabajo habitual del cluster hadoop::

- edits + con números
- un unico edits\_inprogress y
- fsimage + números con y sin MD5.

Observamos que el edits\_inprogress tiene un determinado tamaño, fsimage y edit van por el nº 40 y edits\_inprogress va por el 41.

```
hadoop@nodo1:/datos/namenode/current$ ls -l
total 1128
-rw-rw-r-- 1 hadoop hadoop      42 ene 14 20:05 edits_000000000000000001-000000000000000002
-rw-rw-r-- 1 hadoop hadoop      42 ene 14 21:05 edits_000000000000000003-000000000000000004
-rw-rw-r-- 1 hadoop hadoop      42 ene 14 22:05 edits_000000000000000005-000000000000000006
-rw-rw-r-- 1 hadoop hadoop      42 ene 14 23:05 edits_000000000000000007-000000000000000008
-rw-rw-r-- 1 hadoop hadoop      42 ene 15 17:00 edits_000000000000000009-000000000000000010
-rw-rw-r-- 1 hadoop hadoop      42 ene 16 12:32 edits_000000000000000011-000000000000000012
-rw-rw-r-- 1 hadoop hadoop      42 ene 17 13:56 edits_000000000000000013-000000000000000014
-rw-rw-r-- 1 hadoop hadoop      42 ene 17 14:56 edits_000000000000000015-000000000000000016
-rw-rw-r-- 1 hadoop hadoop      42 ene 17 16:00 edits_000000000000000017-000000000000000018
-rw-rw-r-- 1 hadoop hadoop      42 ene 17 17:00 edits_000000000000000019-000000000000000020
-rw-rw-r-- 1 hadoop hadoop      42 ene 18 17:21 edits_000000000000000021-000000000000000022
-rw-rw-r-- 1 hadoop hadoop      42 ene 29 20:12 edits_000000000000000023-000000000000000024
-rw-rw-r-- 1 hadoop hadoop      42 ene 29 21:12 edits_000000000000000025-000000000000000026
-rw-rw-r-- 1 hadoop hadoop      42 ene 30 09:52 edits_000000000000000027-000000000000000028
-rw-rw-r-- 1 hadoop hadoop      42 ene 30 10:52 edits_000000000000000029-000000000000000030
-rw-rw-r-- 1 hadoop hadoop      42 ene 30 11:52 edits_000000000000000031-000000000000000032
-rw-rw-r-- 1 hadoop hadoop      42 ene 30 14:00 edits_000000000000000033-000000000000000034
-rw-rw-r-- 1 hadoop hadoop      42 ene 30 15:00 edits_000000000000000035-000000000000000036
-rw-rw-r-- 1 hadoop hadoop      42 ene 30 16:00 edits_000000000000000037-000000000000000038
-rw-rw-r-- 1 hadoop hadoop      42 ene 31 10:24 edits_000000000000000039-000000000000000040
-rw-rw-r-- 1 hadoop hadoop 1048576 ene 31 10:24 edits_inprogress_000000000000000041
-rw-rw-r-- 1 hadoop hadoop      325 ene 30 16:00 fsimage_000000000000000038
-rw-rw-r-- 1 hadoop hadoop      62 ene 30 16:00 fsimage_000000000000000038.md5
-rw-rw-r-- 1 hadoop hadoop      325 ene 31 10:24 fsimage_000000000000000040
-rw-rw-r-- 1 hadoop hadoop      62 ene 31 10:24 fsimage_000000000000000040.md5
-rw-rw-r-- 1 hadoop hadoop       3 ene 31 10:24 seen txid
-rw-rw-r-- 1 hadoop hadoop      213 ene 14 20:04 VERSION
hadoop@nodo1:/datos/namenode/current$
```

# 5. ESTRUCTURA DIRECTORIOS HDFS

---

**Paso 3.** El fichero VERSION contiene:

- el nombre del espacio
- el tipo de storage,
- el BlackpoolID (donde está dejando los bloques dentro de los datanode de los esclavos)

```
hadoop@nodo1:/datos/namenode/current$ cat VERSION
#Sat Jan 14 20:04:09 UTC 2023
namespaceID=2105103880
clusterID=CID-18867706-b47d-445c-9ff2-53ed060664b6
cTime=1673434148125
storageType=NAME_NODE
blockpoolID=BP-703191218-127.0.1.1-1673434148125
layoutVersion=-63
hadoop@nodo1:/datos/namenode/current$
```

# 5. ESTRUCTURA DIRECTORIOS HDFS

**Paso 4.** Paramos hdfs con el comando **stop-dfs.sh**. Va a parar el namenode, luego el datanode (solo tenemos uno) y por el ultimo el secundario namenode

```
hadoop@nodol:/datos/namenode/current$ stop-dfs.sh
Stopping namenodes on [nodol]
nodol: stopping namenode
localhost: stopping datanode
Stopping secondary namenodes [0.0.0.0]
0.0.0.0: stopping secondarynamenode
hadoop@nodol:/datos/namenode/current$
```

Si hacemos un `ls -l` de `current` esta tal y como estaba.

```
hadoop@nodol:/datos/namenode/current$ ls -l
total 1128
-rw-rw-r-- 1 hadoop hadoop 42 ene 14 20:05 edits_000000000000000001-000000000000000002
-rw-rw-r-- 1 hadoop hadoop 42 ene 14 21:05 edits_000000000000000003-000000000000000004
-rw-rw-r-- 1 hadoop hadoop 42 ene 14 22:05 edits_000000000000000005-000000000000000006
-rw-rw-r-- 1 hadoop hadoop 42 ene 14 23:05 edits_000000000000000007-000000000000000008
-rw-rw-r-- 1 hadoop hadoop 42 ene 15 17:00 edits_000000000000000009-000000000000000010
-rw-rw-r-- 1 hadoop hadoop 42 ene 16 12:32 edits_000000000000000011-000000000000000012
-rw-rw-r-- 1 hadoop hadoop 42 ene 17 13:56 edits_000000000000000013-000000000000000014
-rw-rw-r-- 1 hadoop hadoop 42 ene 17 14:56 edits_000000000000000015-000000000000000016
-rw-rw-r-- 1 hadoop hadoop 42 ene 17 16:00 edits_000000000000000017-000000000000000018
-rw-rw-r-- 1 hadoop hadoop 42 ene 17 17:00 edits_000000000000000019-000000000000000020
-rw-rw-r-- 1 hadoop hadoop 42 ene 18 17:21 edits_000000000000000021-000000000000000022
-rw-rw-r-- 1 hadoop hadoop 42 ene 29 20:12 edits_000000000000000023-000000000000000024
-rw-rw-r-- 1 hadoop hadoop 42 ene 29 21:12 edits_000000000000000025-000000000000000026
-rw-rw-r-- 1 hadoop hadoop 42 ene 30 09:52 edits_000000000000000027-000000000000000028
-rw-rw-r-- 1 hadoop hadoop 42 ene 30 10:52 edits_000000000000000029-000000000000000030
-rw-rw-r-- 1 hadoop hadoop 42 ene 30 11:52 edits_000000000000000031-000000000000000032
-rw-rw-r-- 1 hadoop hadoop 42 ene 30 14:00 edits_000000000000000033-000000000000000034
-rw-rw-r-- 1 hadoop hadoop 42 ene 30 15:00 edits_000000000000000035-000000000000000036
-rw-rw-r-- 1 hadoop hadoop 42 ene 30 16:00 edits_000000000000000037-000000000000000038
-rw-rw-r-- 1 hadoop hadoop 42 ene 31 10:24 edits_000000000000000039-000000000000000040
-rw-rw-r-- 1 hadoop hadoop 1048576 ene 31 10:24 edits_inprogress_000000000000000041
-rw-rw-r-- 1 hadoop hadoop 325 ene 30 16:00 fsimage_000000000000000038
-rw-rw-r-- 1 hadoop hadoop 62 ene 30 16:00 fsimage_000000000000000038.md5
-rw-rw-r-- 1 hadoop hadoop 325 ene 31 10:24 fsimage_000000000000000040
-rw-rw-r-- 1 hadoop hadoop 62 ene 31 10:24 fsimage_000000000000000040.md5
-rw-rw-r-- 1 hadoop hadoop 3 ene 31 10:24 seen.txid
-rw-rw-r-- 1 hadoop hadoop 213 ene 14 20:04 VERSION
hadoop@nodol:/datos/namenode/current$
```

# 5. ESTRUCTURA DIRECTORIOS HDFS

---

**Paso 5.** Volvemos a arrancar de nuevo el DSF **start-dfs.sh**.

- El arranque del Cluster se hace siempre desde el maestro. Es quien va dando comandos a través de SSH a los esclavos.
- Arranca el namenode, datanode y secondary namenode de nuevo.
- Dejará el log dentro de /opt/hadoop/logs dependiendo de si es el maestro o son los esclavos.

```
hadoop@nodo1:/datos/namenode/current$ start-dfs.sh
Starting namenodes on [nodo1]
nodo1: starting namenode, logging to /opt/hadoop/logs/hadoop-hadoop-namenode-nodo1.out
localhost: starting datanode, logging to /opt/hadoop/logs/hadoop-hadoop-datanode-nodo1.out
Starting secondary namenodes [0.0.0.0]
0.0.0.0: starting secondarynamenode, logging to /opt/hadoop/logs/hadoop-hadoop-secondarynamenode-nodo1.out
hadoop@nodo1:/datos/namenode/current$
```



## 5. ESTRUCTURA DIRECTORIOS HDFS

**Paso 6.** Si hacemos `ls -l` vemos que se ha producido un pequeño cambio. Edits y fsimage van ahora por el 43 y edits\_inprogress va ahora por el 44.

```
-rw-rw-r-- 1 hadoop hadoop 42 ene 14 20:05 edits_000000000000000001-000000000000000002
-rw-rw-r-- 1 hadoop hadoop 42 ene 14 21:05 edits_000000000000000003-000000000000000004
-rw-rw-r-- 1 hadoop hadoop 42 ene 14 22:05 edits_000000000000000005-000000000000000006
-rw-rw-r-- 1 hadoop hadoop 42 ene 14 23:05 edits_000000000000000007-000000000000000008
-rw-rw-r-- 1 hadoop hadoop 42 ene 15 17:00 edits_000000000000000009-000000000000000010
-rw-rw-r-- 1 hadoop hadoop 42 ene 16 12:32 edits_000000000000000011-000000000000000012
-rw-rw-r-- 1 hadoop hadoop 42 ene 17 13:56 edits_000000000000000013-000000000000000014
-rw-rw-r-- 1 hadoop hadoop 42 ene 17 14:56 edits_000000000000000015-000000000000000016
-rw-rw-r-- 1 hadoop hadoop 42 ene 17 16:00 edits_000000000000000017-000000000000000018
-rw-rw-r-- 1 hadoop hadoop 42 ene 17 17:00 edits_000000000000000019-000000000000000020
-rw-rw-r-- 1 hadoop hadoop 42 ene 18 17:21 edits_000000000000000021-000000000000000022
-rw-rw-r-- 1 hadoop hadoop 42 ene 29 20:12 edits_000000000000000023-000000000000000024
-rw-rw-r-- 1 hadoop hadoop 42 ene 29 21:12 edits_000000000000000025-000000000000000026
-rw-rw-r-- 1 hadoop hadoop 42 ene 30 09:52 edits_000000000000000027-000000000000000028
-rw-rw-r-- 1 hadoop hadoop 42 ene 30 10:52 edits_000000000000000029-000000000000000030
-rw-rw-r-- 1 hadoop hadoop 42 ene 30 11:52 edits_000000000000000031-000000000000000032
-rw-rw-r-- 1 hadoop hadoop 42 ene 30 14:00 edits_000000000000000033-000000000000000034
-rw-rw-r-- 1 hadoop hadoop 42 ene 30 15:00 edits_000000000000000035-000000000000000036
-rw-rw-r-- 1 hadoop hadoop 42 ene 30 16:00 edits_000000000000000037-000000000000000038
-rw-rw-r-- 1 hadoop hadoop 42 ene 31 10:24 edits_000000000000000039-000000000000000040
-rw-rw-r-- 1 hadoop hadoop 1048576 ene 31 10:24 edits_000000000000000041-000000000000000041
-rw-rw-r-- 1 hadoop hadoop 42 ene 31 11:14 edits_000000000000000042-000000000000000043
-rw-rw-r-- 1 hadoop hadoop 1048576 ene 31 11:14 edits_inprogress_000000000000000044
-rw-rw-r-- 1 hadoop hadoop 325 ene 31 10:24 fsimage_000000000000000040
-rw-rw-r-- 1 hadoop hadoop 62 ene 31 10:24 fsimage_000000000000000040.md5
-rw-rw-r-- 1 hadoop hadoop 325 ene 31 11:14 fsimage_000000000000000043
-rw-rw-r-- 1 hadoop hadoop 62 ene 31 11:14 fsimage_000000000000000043.md5
-rw-rw-r-- 1 hadoop hadoop 3 ene 31 11:14 seen_txid
-rw-rw-r-- 1 hadoop hadoop 213 ene 14 20:04 VERSION
hadoop@nodol1:/datos/namenode/current$
```

## 6. FUNCIONAMIENTO HDFS

---

**Paso 1.** Estudiaremos el funcionamiento de los metadatos de HDFS dentro de Hadoop. HDFS dispone de unos ficheros que gestionan los cambios que se producen en el cluster (por ejemplo que pasa cuando se sube un fichero, cómo se implementa esa información, etc)

Básicamente son 3 ficheros:

- Edits\_000xxxxxxx. Representa los cambios que se van produciendo dentro de la base de datos de HDFS, es decir los metadatos
- Edits\_inprogress\_xxxxx. Donde se están escribiendo los datos en este momento
- Fsimage\_00000xxxxxx. Contiene una especie de copia, representa una foto de un momento concreto del estado del sistema de ficheros HDFS



## 6. FUNCIONAMIENTO HDFS

---

**Paso 2.** ¿Que hace el sistema de ficheros hdfs cuando arranca?

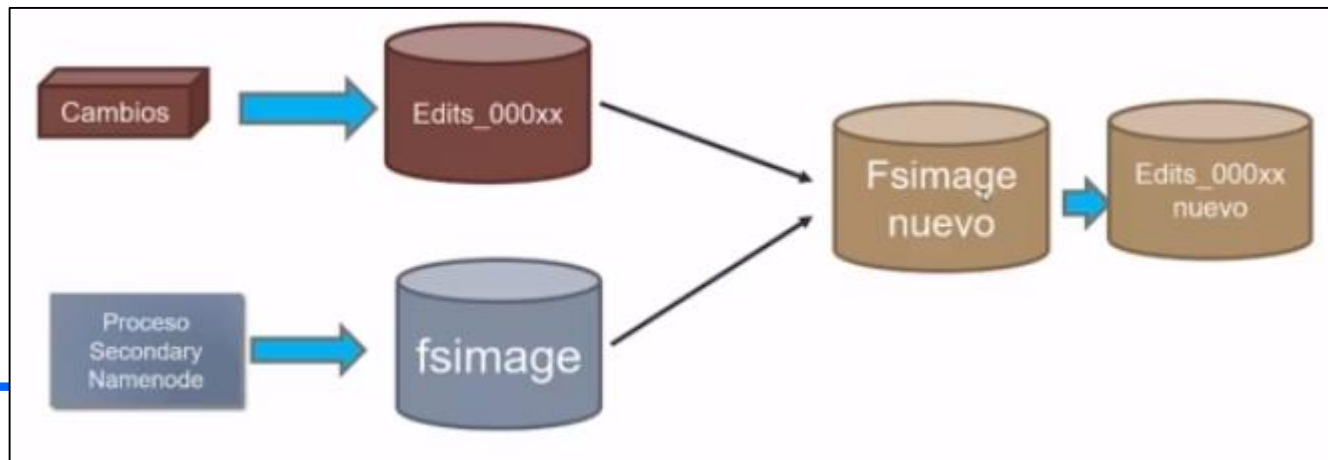
- 1) Carga en memoria el último fichero fsimage disponible, junto con los edits que no han sido procesados. fsimage solo es de lectura
- 2) Los cambios que se van produciendo en los datos se van almacenando en memoria en el fichero edits inprogres. Aquí se van guardando los bloques
- 3) Es inmanejable ir guardando en el fichero fsimage cada cosa que se hace el sistema.



## 6. FUNCIONAMIENTO HDFS

### Paso 3. Proceso de sincronización

- Cada cierto tiempo el proceso **Secondary namenode** (normalmente cada hora o cada vez que se llenan 128 megas de información que suelen equivaler a un bloque), sincroniza lo que se ha ido grabando en ese momento con lo que había en FSImage.
- Genera una **fsimage\_nueva** a partir de la foto (fsimage\_anterior) que había por la mañana junto con todos los cambios que se han hecho hasta la noche. La nueva imagen se deja registrada y se empieza con edits nuevos.



## 6. FUNCIONAMIENTO HDFS

---

**Paso 4.** Por lo tanto a lo largo del tiempo podremos llegar a tener:

- un `edits_inprogres` que sería el que en ese momento se está utilizando
- una serie de `edits` que son los puntos anteriores (checkpoints) por donde han ido pasando los cambios
- unos cuantos `fsimage` que son todas las fotos que se han ido haciendo a lo largo del trabajo.

El sistema de ficheros HDFS, es un sitio donde escribe HDFS automáticamente y por lo tanto no debemos de tocar manualmente.

# 7. OPERACIONES CON HDFS

---

**Paso 1.** Los comandos de HDFS funcionan de una manera muy parecida a los comandos de Linux: mkdir, ls, cat, mv..

El comando HDFS se encuentra dentro /opt/hadoop/bin

```
hadoop@nodo1:/datos/namenode/current$ hdfs
Usage: hdfs [--config confdir] [--loglevel loglevel] COMMAND
    where COMMAND is one of:
    dfs                run a filesystem command on the file systems supported in Hadoop
    classpath          prints the classpath
    namenode -format   format the DFS filesystem
    secondarynamenode run the DFS secondary namenode
    namenode           run the DFS namenode
    journalnode        run the DFS journalnode
    zkfc               run the ZK Failover Controller daemon
    datanode           run a DFS datanode
    debug              run a Debug Admin to execute HDFS debug commands
    dfsadmin           run a DFS admin client
    dfsrouter          run the DFS router
    dfsrouteradmin     manage Router-based federation
    haadmin            run a DFS HA admin client
    fsck               run a DFS filesystem checking utility
    balancer           run a cluster balancing utility
    jmxget             get JMX exported values from NameNode or DataNode.
    mover              run a utility to move block replicas across
                      storage types
    oiv                apply the offline fsimage viewer to an fsimage
    oiv_legacy         apply the offline fsimage viewer to an legacy fsimage
    oev               apply the offline edits viewer to an edits file
    fetchdt            fetch a delegation token from the NameNode
```

# 7. OPERACIONES CON HDFS

Paso 2. El formato de los comandos HDFS que interactúen con el sistema de ficheros es **hdfs dfs + mkdir, chmod, cat, df, rm, put**. Al tratarse de un sistema de ficheros, hadoop ha reescrito los de Linux.

```
hadoop@nodo1:/datos/namenode/current$ hdfs dfs
Usage: hadoop fs [generic options]
    [-appendToFile <localsrc> ... <dst>]
    [-cat [-ignoreCrc] <src> ...]
    [-checksum <src> ...]
    [-chgrp [-R] GROUP PATH...]
    [-chmod [-R] <MODE[,MODE]... | OCTALMODE> PATH...]
    [-chown [-R] [OWNER][:[GROUP]] PATH...]
    [-copyFromLocal [-f] [-p] [-l] [-d] <localsrc> ... <dst>]
    [-copyToLocal [-f] [-p] [-ignoreCrc] [-crc] <src> ... <localdst>]
    [-count [-q] [-h] [-v] [-t [<storage type>]] [-u] [-x] <path> ...]
    [-cp [-f] [-p | -p[topax]] [-d] <src> ... <dst>]
    [-createSnapshot <snapshotDir> [<snapshotName>]]
    [-deleteSnapshot <snapshotDir> <snapshotName>]
    [-df [-h] [<path> ...]]
    [-du [-s] [-h] [-x] <path> ...]
    [-expunge]
    [-getfacl [-R] <path>]
    [-help [cmd ...]]
    [-mv <src> ... <dst>]
    [-put [-f] [-p] [-l] [-d] <localsrc> ... <dst>]
    [-renameSnapshot <snapshotDir> <oldName> <newName>]
    [-rm [-f] [-r|-R] [-skipTrash] [-safely] <src> ...]
    [-rmdir [--ignore-fail-on-non-empty] <dir> ...]
    [-setfacl [-R] [{-b|-k} {-m|-x <acl_spec>} <path>]|[--set <acl_spec> <path>]]
    [-setfattr {-n name [-v value] | -x name} <path>]
    [-setrep [-R] [-w] <rep> <path> ...]
```

# 7. OPERACIONES CON HDFS

---

## Paso 3. Secuencia de carga de un simple fichero

- **hdfs dfs -ls /** → Indica lo que contiene el sistema de ficheros desde la raíz del propio sistema HDFS

```
hadoop@nodol:/datos/namenode/current$ hdfs dfs -ls /  
hadoop@nodol:/datos/namenode/current$
```

- No sale nada porque no tengo nada. También se puede ver en la pagina de configuración Utilities/Browse the file system.
- Cargaremos nuestro primer fichero prueba.txt

```
hadoop@nodol:~$ echo Hola >> prueba.txt  
hadoop@nodol:~$ cat prueba.txt  
Hola  
hadoop@nodol:~$
```

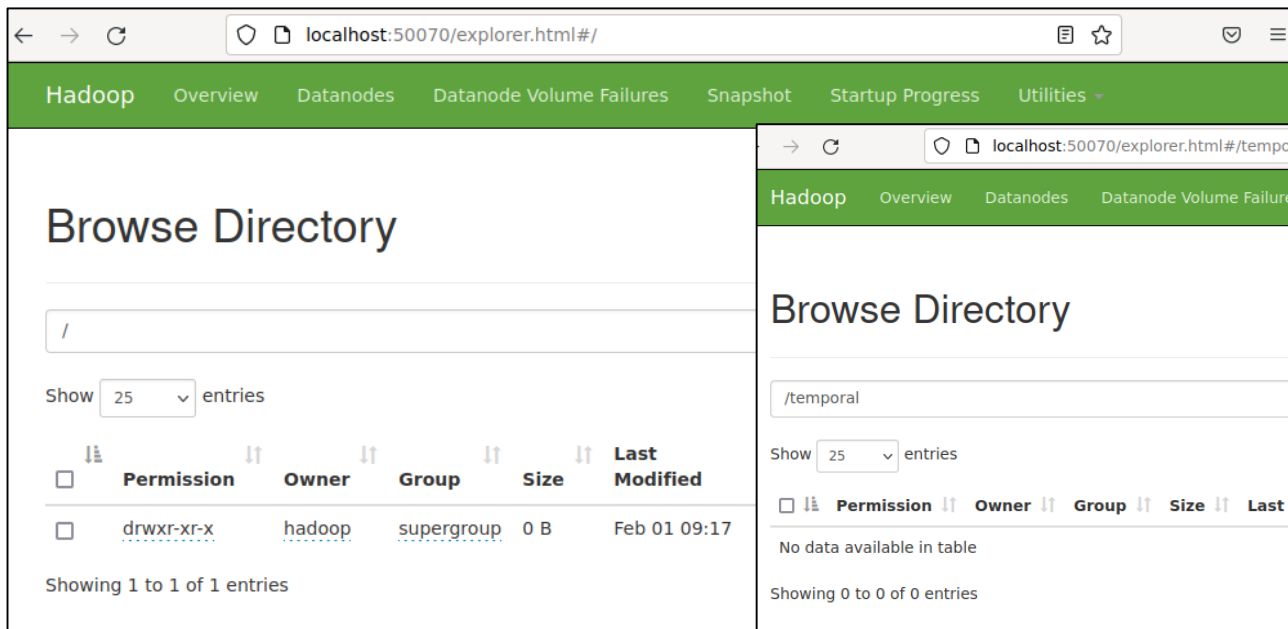
- **hdfs dfs -mkdir /temporal** → Creamos directorio en temporal en el sistema de ficheros HDFS.

```
hadoop@nodol:~$ hdfs dfs -mkdir /temporal  
hadoop@nodol:~$ hdfs dfs -ls /  
Found 1 items  
drwxr-xr-x - hadoop supergroup          0 2023-02-01 09:17 /temporal  
hadoop@nodol:~$
```

# 7. OPERACIONES CON HDFS

**Paso 4.** Si vamos a Utilities/Browse the file system vemos que ya tenemos el directorio temporal creado. Evidentemente el directorio está vacío porque no hemos subido ningún fichero aquí

Podemos ver que los permisos que tiene este directorio es muy similar a los permisos de Linux: con usuario, grupo y otros, el propietario, el grupo al que pertenece, el tamaño, etc



← → ↻ localhost:50070/explorer.html#/

Hadoop Overview Datanodes Datanode Volume Failures Snapshot Startup Progress Utilities ▾

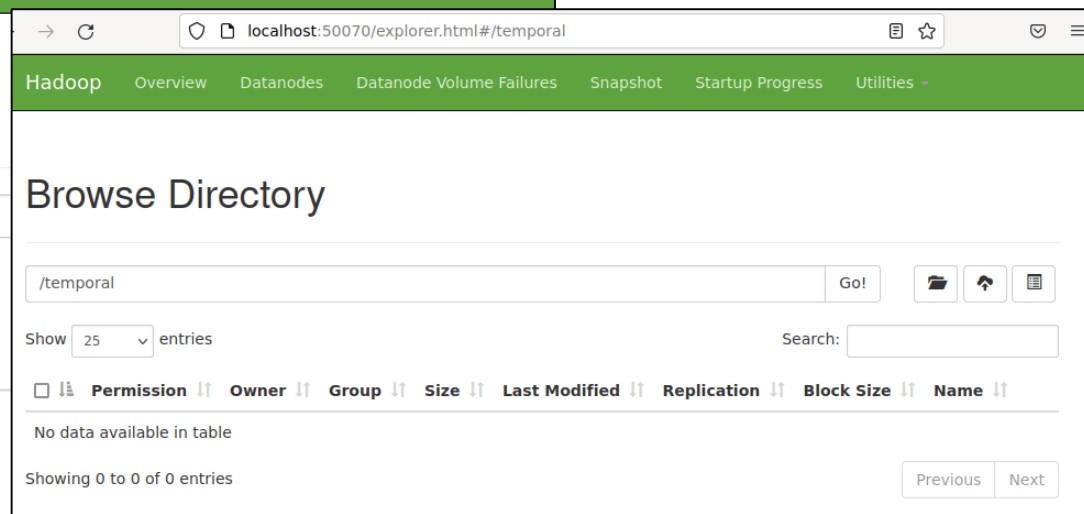
## Browse Directory

/

Show 25 ▾ entries

<input type="checkbox"/>	Permission	Owner	Group	Size	Last Modified
<input type="checkbox"/>	<a href="#">drwxr-xr-x</a>	<a href="#">hadoop</a>	<a href="#">supergroup</a>	0 B	Feb 01 09:17

Showing 1 to 1 of 1 entries






← → ↻ localhost:50070/explorer.html#/temporal

Hadoop Overview Datanodes Datanode Volume Failures Snapshot Startup Progress Utilities ▾

## Browse Directory

/temporal

Go!   

Show 25 ▾ entries Search:

<input type="checkbox"/>	Permission	Owner	Group	Size	Last Modified	Replication	Block Size	Name
No data available in table								

Showing 0 to 0 of 0 entries

Previous Next

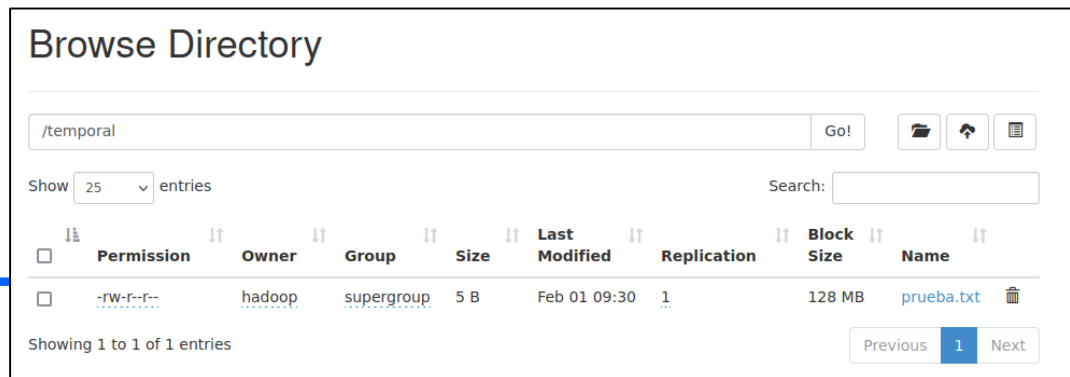
# 7. OPERACIONES CON HDFS

**Paso 5.** El comando PUT permite para subir ficheros de nuestro sistema Linux al sistema HDFS.

- **hdfs dfs -put prueba.txt /temporal** → Sube/carga el fichero llamado prueba.txt al directorio temporal.

```
hadoop@nodol1:~$ hdfs dfs -put prueba.txt /temporal
hadoop@nodol1:~$ hdfs dfs -ls /
Found 1 items
drwxr-xr-x   - hadoop supergroup          0 2023-02-01 09:30 /temporal
hadoop@nodol1:~$ hdfs dfs -ls /temporal
Found 1 items
-rw-r--r--   1 hadoop supergroup          5 2023-02-01 09:30 /temporal/prueba.txt
hadoop@nodol1:~$
```


- Dentro de Utilities/Browse the file System si accedemos el directorio temporal podemos observar el fichero prueba.txt.





## 8. UBICACION FISICA FICHEROS

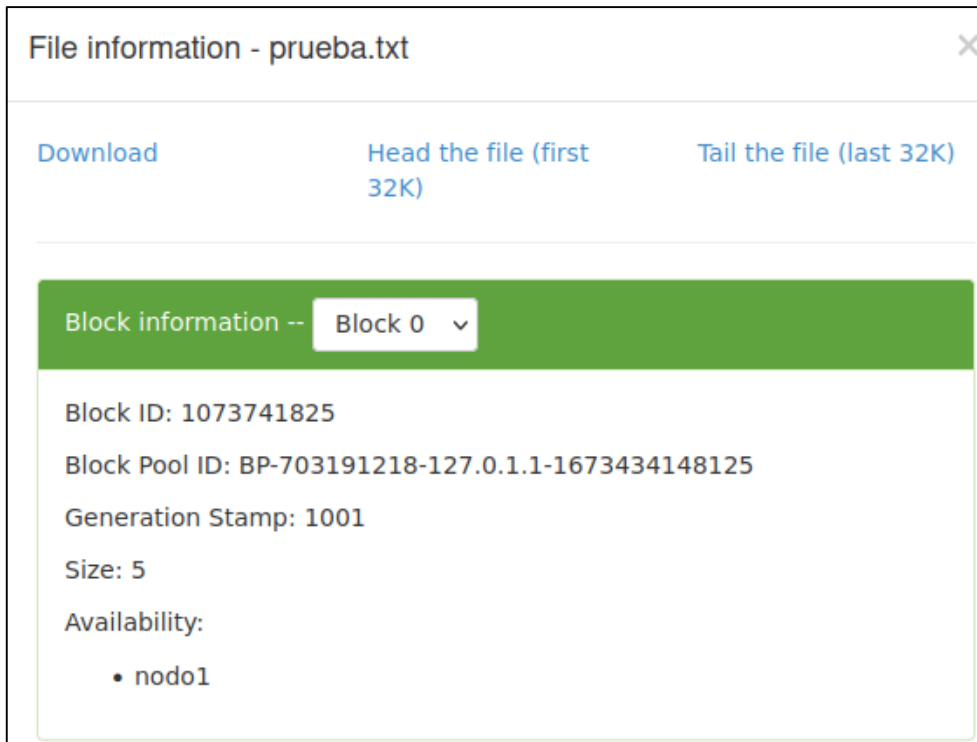
**Paso 1.** Una vez subido el fichero, podemos saber físicamente dónde se encuentra. En Browse the file System podemos ver la propiedades de prueba.txt (permisos, propietario, grupo de pertenencia, tamaño) igual a como si se tratara de un sistema de ficheros Linux tradicional pero con diferencias.

<input type="checkbox"/>	Permission	Owner	Group	Size	Last Modified	Replication	Block Size	Name	
<input type="checkbox"/>	<a href="#">-rw-r--r--</a>	<a href="#">hadoop</a>	<a href="#">supergroup</a>	5 B	Feb 01 09:30	<a href="#">1</a>	128 MB	<a href="#">prueba.txt</a>	

- **Parámetro Replicación: 1** → Como sólo contamos con un no sólo nodo, el 1 indica sin replicación, no podemos replicar ninguna vez
- **Tamaño bloque HDFS de hadoop: 128MB** → Muy grande comparado con el tamaño medio de bloque de un sistema de ficheros tradicional (500 kB - 0,5MB). Moveremos muchos ficheros de un gran tamaño.

## 8. UBICACION FISICA FICHEROS

**Paso 2.** Si hacemos click en pruebas.txt obtenemos la información de los bloques que tiene este fichero. Si tuviéramos un fichero de 1GB, Hadoop lo dividiría en bloques de 128 MB, saliendo unos 10 bloques. Este fichero sólo está en un bloque, el primero el Bloque 0



The screenshot shows a web interface titled "File information - prueba.txt". At the top, there are three links: "Download", "Head the file (first 32K)", and "Tail the file (last 32K)". Below these links is a green bar with the text "Block information --" and a dropdown menu showing "Block 0". Underneath the green bar, the following information is displayed:

- Block ID: 1073741825
- Block Pool ID: BP-703191218-127.0.1.1-1673434148125
- Generation Stamp: 1001
- Size: 5
- Availability:
  - nodo1

Nos indica:

- El ID del Bloque donde está
- Block Pool ID: directorio donde se encuentran los bloques
- Disponibilidad: en qué nodos se encuentra (solo el nodo 1)

## 8. UBICACION FISICA FICHEROS

**Paso 3.** En `/datos/datanode/current` se encuentra el Block Pool ID que hemos visto en Browse the File System. Ambos IDs coinciden. El fichero `prueba.txt` se encuentra en ese bloque por defecto.

Block Pool ID: BP-703191218-127.0.1.1-1673434148125

```
hadoop@nodo1:/datos/datanode/current$ ls -l
total 8
drwx----- 4 hadoop hadoop 4096 ene 31 11:13 BP-703191218-127.0.1.1-1673434148125
-rw-rw-r-- 1 hadoop hadoop 229 ene 31 11:13 VERSION
hadoop@nodo1:/datos/datanode/current$
```

Si profundizamos y entramos dentro del Block Pool:

```
hadoop@nodo1:/datos/datanode/current$ cd BP-703191218-127.0.1.1-1673434148125/
hadoop@nodo1:/datos/datanode/current/BP-703191218-127.0.1.1-1673434148125$ ls -l
total 12
drwxrwxr-x 4 hadoop hadoop 4096 ene 31 10:55 current
-rw-rw-r-- 1 hadoop hadoop 166 ene 14 20:04 scanner.cursor
drwxrwxr-x 2 hadoop hadoop 4096 ene 31 11:13 tmp
hadoop@nodo1:/datos/datanode/current/BP-703191218-127.0.1.1-1673434148125$ cd current/
hadoop@nodo1:/datos/datanode/current/BP-703191218-127.0.1.1-1673434148125/current$ ls -l
total 16
-rw-rw-r-- 1 hadoop hadoop 19 ene 31 10:55 dfsUsed
drwxrwxr-x 3 hadoop hadoop 4096 feb 1 09:30 finalized
drwxrwxr-x 2 hadoop hadoop 4096 feb 1 09:30 rbw
-rw-rw-r-- 1 hadoop hadoop 140 ene 31 11:13 VERSION
hadoop@nodo1:/datos/datanode/current/BP-703191218-127.0.1.1-1673434148125/current$
```

## 8. UBICACION FISICA FICHEROS

Paso 4. En **BP-XXX/current/finalized/subdir0/subdir0** encontramos nuestro bloque:

Block ID: 1073741825

Block Pool ID: BP-703191218-127.0.1.1-1673434148125

```
hadoop@nodo1:/datos/datanode/current/BP-703191218-127.0.1.1-1673434148125/current$ cd finalized/
hadoop@nodo1:/datos/datanode/current/BP-703191218-127.0.1.1-1673434148125/current/finalized$ ls -l
total 4
drwxrwxr-x 3 hadoop hadoop 4096 feb  1 09:30 subdir0
hadoop@nodo1:/datos/datanode/current/BP-703191218-127.0.1.1-1673434148125/current/finalized$ cd subdir0/
hadoop@nodo1:/datos/datanode/current/BP-703191218-127.0.1.1-1673434148125/current/finalized/subdir0$ ls -l
total 4
drwxrwxr-x 2 hadoop hadoop 4096 feb  1 09:30 subdir0
hadoop@nodo1:/datos/datanode/current/BP-703191218-127.0.1.1-1673434148125/current/finalized/subdir0$ cd subdir0/
hadoop@nodo1:/datos/datanode/current/BP-703191218-127.0.1.1-1673434148125/current/finalized/subdir0/subdir0$ ls -l
total 8
-rw-rw-r-- 1 hadoop hadoop  5 feb  1 09:30 blk_1073741825
-rw-rw-r-- 1 hadoop hadoop 11 feb  1 09:30 blk_1073741825_1001.meta
hadoop@nodo1:/datos/datanode/current/BP-703191218-127.0.1.1-1673434148125/current/finalized/subdir0/subdir0$
```

Es fácil de encontrar porque ahora mismo sólo tenemos un fichero y un directorio. Si se sube un fichero a HDFS lo trocea en bloques de 128 megas y lo distribuye a lo largo de este directorio con unos nombres similares a estos

## 8. UBICACION FISICA FICHEROS

---

**Paso 5.** En realidad no importa como se llamen y donde los ponga, salvo que alguna vez haya un problema de corrupción o que quiera hacer algo concreto. Habitualmente accederemos a HDFS a través de:

- El comando `hdfs dfs`
- Con alguna de las múltiples herramientas para acceder como es Scoop, Flavin, Hive o algún otro de los otros productos que forman parte del ecosistema HADOOP BIG DATA

## 8. UBICACION FISICA FICHEROS

---

**Paso 6.** Subiremos un fichero más grande, access.log.gz de 50MB

- Lo descomprimos y el tamaño del fichero obtenido es superior a 128MB (500MB aprox)
- Lo movemos al directorio /tmp

```
hadoop@nodol1:~/Downloads$ pwd
/home/hadoop/Downloads
hadoop@nodol1:~/Downloads$ ls -l
total 1137348
-rw-rw-r-- 1 hadoop hadoop 54547198 feb  5 13:19 access_log.gz
-rw-rw-r-- 1 hadoop hadoop 414624228 ene  4 12:09 hadoop-2.10.2.tar.gz
-rw-rw-r-- 1 hadoop hadoop 695457782 ene  2 12:11 hadoop-3.3.4.tar.gz
hadoop@nodol1:~/Downloads$ gunzip access_log.gz
hadoop@nodol1:~/Downloads$ ls -l
total 1577188
-rw-rw-r-- 1 hadoop hadoop 504941532 feb  5 13:19 access_log
-rw-rw-r-- 1 hadoop hadoop 414624228 ene  4 12:09 hadoop-2.10.2.tar.gz
-rw-rw-r-- 1 hadoop hadoop 695457782 ene  2 12:11 hadoop-3.3.4.tar.gz
hadoop@nodol1:~/Downloads$ mv access_log /tmp
hadoop@nodol1:~/Downloads$
```

## 8. UBICACION FISICA FICHEROS

**Paso 7.** Subimos el fichero de 500 MB al directorio /temporal de HDFS

- **hdfs dfs -put /tmp/access\_log /temporal**

```
hadoop@nodo1:~/Downloads$ hdfs dfs -put /tmp/access_log /temporal
hadoop@nodo1:~/Downloads$
```

- En Browse the file System observamos el directorio temporal y el fichero Access\_log de unos 481 MB.

/temporal

Go!

Show

25

▼

entries

Search:

<div><input type="checkbox"/></div>	<div><div><div></div></div>Permission</div>	<div><div><div></div></div>Owner</div>	<div><div><div></div></div>Group</div>	<div><div><div></div></div>Size</div>	<div><div><div></div></div>Last Modified</div>	<div><div><div></div></div>Replication</div>	<div><div><div></div></div>Block Size</div>	<div><div><div></div></div>Name</div>	<div><div><div></div></div></div>
<div><input type="checkbox"/></div>	<div><div>-rw-r--r--</div></div>	<div><div>hadoop</div></div>	<div><div>supergroup</div></div>	<div><div>481.55 MB</div></div>	<div><div>Feb 05 13:50</div></div>	<div><div>1</div></div>	<div><div>128 MB</div></div>	<div><div><a href="#">access_log</a></div></div>	<div><div><div></div></div></div>
<div><input type="checkbox"/></div>	<div><div>-rw-r--r--</div></div>	<div><div>hadoop</div></div>	<div><div>supergroup</div></div>	<div><div>5 B</div></div>	<div><div>Feb 01 09:30</div></div>	<div><div>1</div></div>	<div><div>128 MB</div></div>	<div><div><a href="#">prueba.txt</a></div></div>	<div><div><div></div></div></div>

Showing 1 to 2 of 2 entries

Previous

1

Next

# 8. UBICACION FISICA FICHEROS

**Paso 8.** Hacemos click en el fichero Access\_log y vemos que esta en 4 bloques de 128 MB

- Según se cambia de bloque, solo cambia el ID de bloque, todos se encuentran en el mismo Block Pool ID (el mismo de prueba.txt)
- Si tuviéramos más de un nodo veríamos estos bloques copiados en cada uno de los nodos (aquí solo tenemos 1 nodo)

Block Pool ID: BP-703191218-127.0.1.1-1673434148125

File information - access\_log

Download Head the file (first 32K) Tail the file (last 32K)

Block information -- Block 0

Block ID: 1073741826  
Block Pool ID: BP-703191218-127.0.1.1-1673434148125  
Generation Stamp: 1002  
Size: 134217728  
Availability: nodo1

File information - access\_log

Download Head the file (first 32K) Tail the file (last 32K)

Block information -- Block 1

Block ID: 1073741827  
Block Pool ID: BP-703191218-127.0.1.1-1673434148125  
Generation Stamp: 1003  
Size: 134217728  
Availability: nodo1

File information - access\_log

Download Head the file (first 32K) Tail the file (last 32K)

Block information -- Block 2

Block ID: 1073741828  
Block Pool ID: BP-703191218-127.0.1.1-1673434148125  
Generation Stamp: 1004  
Size: 134217728  
Availability: nodo1



## 8. UBICACION FISICA FICHEROS

---

**Paso 9.** La ubicación física de los bloques almacenados es:

```
hadoop@nodo1:/datos/datanode/current/BP-703191218-127.0.1.1-1673434148125/current/finalized/subdir0/subdir0$ ls -l
total 497000
-rw-rw-r-- 1 hadoop hadoop          5 feb  1 09:30 blk_1073741825
-rw-rw-r-- 1 hadoop hadoop         11 feb  1 09:30 blk_1073741825_1001.meta
-rw-rw-r-- 1 hadoop hadoop 134217728 feb  5 13:50 blk_1073741826
-rw-rw-r-- 1 hadoop hadoop  1048583 feb  5 13:50 blk_1073741826_1002.meta
-rw-rw-r-- 1 hadoop hadoop 134217728 feb  5 13:50 blk_1073741827
-rw-rw-r-- 1 hadoop hadoop  1048583 feb  5 13:50 blk_1073741827_1003.meta
-rw-rw-r-- 1 hadoop hadoop 134217728 feb  5 13:50 blk_1073741828
-rw-rw-r-- 1 hadoop hadoop  1048583 feb  5 13:50 blk_1073741828_1004.meta
-rw-rw-r-- 1 hadoop hadoop 102288348 feb  5 13:50 blk_1073741829
-rw-rw-r-- 1 hadoop hadoop   799135 feb  5 13:50 blk_1073741829_1005.meta
hadoop@nodo1:/datos/datanode/current/BP-703191218-127.0.1.1-1673434148125/current/finalized/subdir0/subdir0$
```

- Vemos que ha creado 4 bloques, 3 de 128 MB y el ultimo que coga el resto del fichero, junto con la información de metadatos que él utiliza para gestionar estos bloques.
- o de procesos el putt nos no es el que más se utiliza se utilizan otras herramientas como el Flumen el scoop Jive y algún otro producto adicional

## 9. OTROS COMANDOS HDFS

---

**Paso 1.** Ya hemos visto que el comando **hdfs dfs** nos brinda los comandos que podemos utilizar para hacer operaciones con el sistema de ficheros HDFS.

```
hadoop@nodol:~$ hdfs dfs
Usage: hadoop fs [generic options]
    [-appendToFile <localsrc> ... <dst>]
    [-cat [-ignoreCrc] <src> ...]
    [-checksum <src> ...]
    [-chgrp [-R] GROUP PATH...]
    [-chmod [-R] <MODE[,MODE]... | OCTALMODE> PATH...]
    [-chown [-R] [OWNER][:[GROUP]] PATH...]
    [-copyFromLocal [-f] [-p] [-l] [-d] <localsrc> ... <dst>]
    [-get [-f] [-p] [-ignoreCrc] [-crc] <src> ... <localdst>]
    [-getfacl [-R] <path>]
    [-getfattr [-R] {-n name | -d} [-e en] <path>]
    [-getmerge [-nl] [-skip-empty-file] <src> <localdst>]
    [-help [cmd ...]]
    [-ls [-C] [-d] [-h] [-q] [-R] [-t] [-S] [-r] [-u] [<path> ...]]
    [-mkdir [-p] <path> ...]
    [-moveFromLocal <localsrc> ... <dst>]
    [-moveToLocal <src> <localdst>]
    [-mv <src> ... <dst>]
    [-put [-f] [-p] [-l] [-d] <localsrc> ... <dst>]
```

## 9. OTROS COMANDOS HDFS

---

### Paso 2. Secuencia de comandos HDFS

- **hdfs dfs -ls /temporal** → Lista el contenido de ficheros hadoop

```
255 hadoop@nodol1:~$ hdfs dfs -ls /temporal
Found 2 items
-rw-r--r--    1 hadoop supergroup  504941532 2023-02-05 13:50 /temporal/access_log
-rw-r--r--    1 hadoop supergroup      5 2023-02-01 09:30 /temporal/prueba.txt
hadoop@nodol1:~$
```

- **hdfs dfs -cat /temporal/prueba.txt** → Muestra el contenido de un fichero

```
hadoop@nodol1:~$ hdfs dfs -cat /temporal/prueba.txt
Hola
hadoop@nodol1:~$
```

- **hdfs dfs -mkdir /temporal1** → Crea un directorio

```
hadoop@nodol1:~$ hdfs dfs -mkdir /temporal1
hadoop@nodol1:~$
```

- **hdfs dfs -cp /temporal/prueba.txt /temporal1/prueba1.txt** → Copia prueba.txt con otro nombre en el nuevo directorio temporal 1

```
hadoop@nodol1:~$ hdfs dfs -cp /temporal/prueba.txt /temporal1/prueba1.txt
hadoop@nodol1:~$
```

# 9. OTROS COMANDOS HDFS

## Paso 3. Borramos el fichero prueba.txt

- `hdfs dfs -rm /temporal/prueba.txt` → Borra el fichero original prueba.txt en /temporal

```
hadoop@nodo1:~$ hdfs dfs -rm /temporal/prueba.txt
Deleted /temporal/prueba.txt
hadoop@nodo1:~$
```

- En Utilities/Browse the file system podemos comprobar que en el directorio /temporal sólo tenemos el fichero access\_log y en el directorio /temporal1 el fichero prueba.txt

Browse Directory

/temporal

Go!

Show 25 entries

Search:

Permission	Owner	Group	Size	Last Modified	Replication	Block Size	Name
-rw-r--r--	hadoop	supergroup	481.55 MB	Feb 05 13:50	1	128 MB	access_log

Showing 1 to 1 of 1 entries

Browse Directory

/temporal1

Go!

Show 25 entries

Search:

Permission	Owner	Group	Size	Last Modified	Replication	Block Size	Name
-rw-r--r--	hadoop	supergroup	5 B	Feb 05 14:53	1	128 MB	prueba1.txt

Showing 1 to 1 of 1 entries

Previous 1 Next

## 9. OTROS COMANDOS HDFS

---

**Paso 4.** Descargamos/bajamos un fichero del sistema de ficheros HDFS

- **hdfs dfs -get /temporal1/prueba1.txt /tmp/test.txt** →  
Permite extraer/bajar el fichero /temporal1/prueba1.txt del sistema hdfs al sistema Linux, cambiando el nombre

```
hadoop@nodo1:~$ hdfs dfs -get /temporal1/prueba1.txt /tmp/test.txt
hadoop@nodo1:~$ ls /tmp/t*
/tmp/test.txt

/tmp/tmux-1000:
default
hadoop@nodo1:~$
```

- **Hive, Scoop, Flumen**, etc son herramientas adicionales que hay en el ecosistema de Hadoop que permiten realizar operaciones con el sistema de ficheros y no directamente con instrucciones hdfs dfs

# 10. COMANDOS ADMINISTRACION HDFS

---

**Paso 1.** Veremos comandos enfocados en la parte de administración de HDFS.

- Empiezan con `hdfs dfsadmin`
- Permiten hacer distintas operaciones de administración como:
  - poner el cluster en modo safemode (parecido al de sólo lectura),
  - refrescar los nodos para comprobar si se ha añadido alguno más

```
hadoop@nodol:~$ hdfs dfsadmin
Usage: hdfs dfsadmin
Note: Administrative commands can only be run as the HDFS superuser.
    [-report [-live] [-dead] [-decommissioning] [-enteringmaintenance] [-inmaintenance]]
    [-safemode <enter | leave | get | wait>]
    [-saveNamespace]
    [-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]
```

# 10. COMANDOS ADMINISTRACION HDFS

## Paso 2. Obtener un report de HDFS

- **hdfs hdf -report** → Permite sacar una especie de resumen similar al que nos aparece en la página web apartado Overview:
  - qué capacidad tenemos disponible,
  - qué capacidad tenemos disponible para HDFS y no HDFS.

```
255 hadoop@nodol1:~$ hdfs dfsadmin -report
Configured Capacity: 25180848128 (23.45 GB)
Present Capacity: 9068249088 (8.45 GB)
DFS Remaining: 8559280128 (7.97 GB)
DFS Used: 508968960 (485.39 MB)
DFS Used%: 5.61%
Under replicated blocks: 0
Blocks with corrupt replicas: 0
Missing blocks: 0
Missing blocks (with replication factor 1): 0
Pending deletion blocks: 0

-----
Live datanodes (1):

Name: 127.0.0.1:50010 (localhost)
Hostname: nodol1
Decommission Status : Normal
Configured Capacity: 25180848128 (23.45 GB)
DFS Used: 508968960 (485.39 MB)
Non DFS Used: 14807543808 (13.79 GB)
DFS Remaining: 8559280128 (7.97 GB)
DFS Used%: 2.02%
DFS Remaining%: 33.99%
Configured Cache Capacity: 0 (0 B)
Cache Used: 0 (0 B)
Cache Remaining: 0 (0 B)
Cache Used%: 100.00%
Cache Remaining%: 0.00%
```

localhost:50070/dfshealth.html#tab-overview	
Summary	
Security is off.	
Safemode is off.	
5 files and directories, 5 blocks = 10 total filesystem object(s).	
Heap Memory used 80.31 MB of 219 MB Heap Memory. Max Heap Memory is 889 MB.	
Non Heap Memory used 61.56 MB of 63.28 MB Committed Non Heap Memory. Max Non Heap Memory is <unbounded>.	
Configured Capacity:	23.45 GB
DFS Used:	485.39 MB (2.02%)
Non DFS Used:	13.79 GB
DFS Remaining:	7.97 GB (33.99%)
Block Pool Used:	485.39 MB (2.02%)
DataNodes usages% (Min/Median/Max/stdDev):	2.02% / 2.02% / 2.02% / 0.00%

# 10. COMANDOS ADMINISTRACION

## HDFS

---

- **Under Replicated blocks:** → Número de bloques que están por debajo de su posible réplica. Por ejemplo si tenemos que el nivel de replicas es 3 y existe algún bloque que tiene menos de tres réplicas
- Hay un porcentaje a partir del cual hadoop decide ponerse en modo Safe (en modo lectura) si ve que el número de bloques a replicar es superior a un porcentaje. Piensa que existe algún problema adicional



# 10. COMANDOS ADMINISTRACION HDFS

Paso 3. Para chequear el sistema de ficheros

- **hdfs fsck /** → Permite chequear el sistema de ficheros HDFS y ver si tenemos algún pequeño/grande problema aquí
- Nos dice que está saludable: tenemos 3 directorios, dos ficheros, número total de bloques 5, Bloques mínimamente replicados 5, etc

```
255 hadoop@nodol:~$ hdfs fsck /
Connecting to namenode via http://nodol:50070/fsck?ugi=hadoop&path=%2F
FSCK started by hadoop (auth:SIMPLE) from /127.0.0.1 for path / at Sun Feb 05 17:22:53 UTC 2023
..Status: HEALTHY
Total size:      504941537 B
Total dirs:      3
Total files:     2
Total symlinks:   0
Total blocks (validated): 5 (avg. block size 100988307 B)
Minimally replicated blocks: 5 (100.0 %)
Over-replicated blocks: 0 (0.0 %)
Under-replicated blocks: 0 (0.0 %)
Mis-replicated blocks: 0 (0.0 %)
Default replication factor: 1
Average block replication: 1.0
Corrupt blocks: 0
Missing replicas: 0 (0.0 %)
Number of data-nodes: 1
Number of racks: 1
FSCK ended at Sun Feb 05 17:22:54 UTC 2023 in 9 milliseconds
```

```
The filesystem under path '/' is HEALTHY
hadoop@nodol:~$
```

# 10. COMANDOS ADMINISTRACION

## HDFS

---

### Paso 2. Comando de topologia más avanzado

- **hdfs dfsadmin -printTopology** → Permite identificar el nº de nodos que tenemos dentro de la máquina y a que rack pertenecen

```
hadoop@nodo1:~$ hdfs dfsadmin -printTopology
Rack: /default-rack
      127.0.0.1:50010 (localhost)
hadoop@nodo1:~$
```

- Hadoop es rack-aware, es decir es consciente de los rack físicos que pertenecen a la máquina. Se lo tenemos que decir nosotros.
  - Si tuviéramos dentro de nuestro CPD varios Rack físicos con cada una de las máquinas dentro podríamos decir que nodos pertenecen a rack.
  - Esto le permite distribuir correctamente los bloques de forma que se repliquen siempre en rack distintos. Si se cae un rack entero, nos podríamos quedar sin todas las réplicas
-


# 11. SNAPSHOTS

**Paso 1.** Un snapshot es una foto de un sistema de ficheros en un momento determinado. Permite utilizarlo en Backups, recuperaciones,

- Crearemos un fichero, después un Snapshot y lo utilizaremos para recuperarle un fichero que hemos borrado accidentalmente.

```
hadoop@nodo1:~$ hdfs dfs -mkdir /datos
hadoop@nodo1:~$ echo Esto es una prueba > f1.txt
hadoop@nodo1:~$ cat f1.txt
Esto es una prueba
hadoop@nodo1:~$ hdfs dfs -put f1.txt /datos
hadoop@nodo1:~$ hdfs dfs -ls /datos
Found 1 items
-rw-r--r--    1 hadoop supergroup          19 2023-02-05 18:59 /datos/f1.txt
hadoop@nodo1:~$
```

- En el entorno de administración Utilities/Browse the file system podemos ver el directorio /datos y dentro el fichero f1.txt.

<input type="checkbox"/>	Permission	Owner	Group	Size	Last Modified	Replication	Block Size	Name	<input type="checkbox"/>
<input type="checkbox"/>	-rw-r--r--	<a href="#">hadoop</a>	<a href="#">supergroup</a>	19 B	Feb 05 18:59	<a href="#">1</a>	128 MB	<a href="#">f1.txt</a>	

Showing 1 to 1 of 1 entries

Previous1Next

# 11. SNAPSHOTS

**Paso 2.** Podemos localizar el fichero creado de dos maneras:

- Haciendo click encima del fichero en Browse the File System
- Con comando **hdfs fsck /datos/f1.txt -blocks -locations -files**

File information - f1.txt

Download

Head the file (first 32K)

Tail the file

Block information -- Block 0 ▾

Block ID: 1073741831

Block Pool ID: BP-703191218-127.0.1.1-1673434148125

Generation Stamp: 1007

Size: 19

Availability:

- nodo1

```
hadoop@nodo1:~$ hdfs fsck /datos/f1.txt -blocks -locations -files
Connecting to namenode via http://nodo1:50070/fsck?ugi=hadoop&blocks=1&locations=1&file
s=1&path=%2Fdatos%2Ff1.txt
FSCK started by hadoop (auth:SIMPLE) from /127.0.0.1 for path /datos/f1.txt at Sun Feb
05 19:44:04 UTC 2023
/datos/f1.txt 19 bytes, 1 block(s): OK
0. BP-703191218-127.0.1.1-1673434148125:blk_1073741831_1007 len=19 Live_repl=1 [Datanod
eInfoWithStorage[127.0.0.1:50010,DS-0577e7de-93fa-4d42-a251-adadacbc99a2,DISK]]

Status: HEALTHY
Total size:      19 B
Total dirs:      0
Total files:     1
Total symlinks:  0
Total blocks (validated): 1 (avg. block size 19 B)
Minimally replicated blocks: 1 (100.0 %)
Over-replicated blocks: 0 (0.0 %)
Under-replicated blocks: 0 (0.0 %)
Mis-replicated blocks: 0 (0.0 %)
Default replication factor: 1
Average block replication: 1.0
Corrupt blocks: 0
Missing replicas: 0 (0.0 %)
Number of data-nodes: 1
Number of racks: 1
FSCK ended at Sun Feb 05 19:44:04 UTC 2023 in 3 milliseconds

The filesystem under path '/datos/f1.txt' is HEALTHY
hadoop@nodo1:~$
```

# 11. SNAPSHOTS

---

Paso 3. Activamos y creamos un snapshot

- **hdfs dfsadmin -allowSnapshot /datos** → Permite activar el Snapshot en el directorio /datos

```
hadoop@nodo1:~$ hdfs dfsadmin -allowSnapshot /datos
Allowing snapshot on /datos succeeded
hadoop@nodo1:~$
```

- **hdfs dfs -createSnapshot /datos snap1** → Crea un snapshot (foto en un momento determinado) del /datos, de nombre snap1

```
hadoop@nodo1:~$ hdfs dfs -createSnapshot /datos snap1
Created snapshot /datos/.snapshot/snap1
hadoop@nodo1:~$ hdfs dfs -ls /datos/.snapshot/snap1
Found 1 items
-rw-r--r--    1 hadoop supergroup          19 2023-02-05 18:59 /datos/.snapshot/snap1/f1.txt
hadoop@nodo1:~$
```

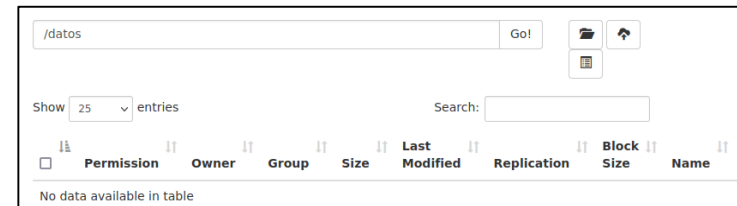
- Desde la web de administración no aparece el Snapshot. Sólo aparecen los ficheros de datos. Los Snapshots solo se pueden ver desde línea de comandos.

# 11. SNAPSHOTS

**Paso 4.** Borrarnos el fichero y lo recuperamos a partir del snapshot:

- **`hdf dfs -rm /datos/f1.txt`** → Borrarnos el fichero `f1.txt`

```
hadoop@nodol1:~$ hdfs dfs -rm /datos/f1.txt
Deleted /datos/f1.txt
hadoop@nodol1:~$ hdfs dfs -ls /datos
hadoop@nodol1:~$
```



- **`hdfs dfs -cp /datos/.snapshot/snap1/f1.txt /datos`** → Recuperaremos el fichero a partir del Snapshot.

```
hadoop@nodol1:~$ hdfs dfs -ls /datos
hadoop@nodol1:~$ hdfs dfs -cp /datos/.snapshot/snap1/f1.txt /datos
hadoop@nodol1:~$ hdfs dfs -ls /datos
Found 1 items
-rw-r--r--    1 hadoop supergroup          19 2023-02-05 20:49 /datos/f1.txt
hadoop@nodol1:~$
```

- Los Snatshots nos permiten hacer operaciones de recuperación , para mantener a salvo nuestro sistema de ficheros.