



**Escuela Técnica Superior de Ingeniería**

Ingeniería de computadores

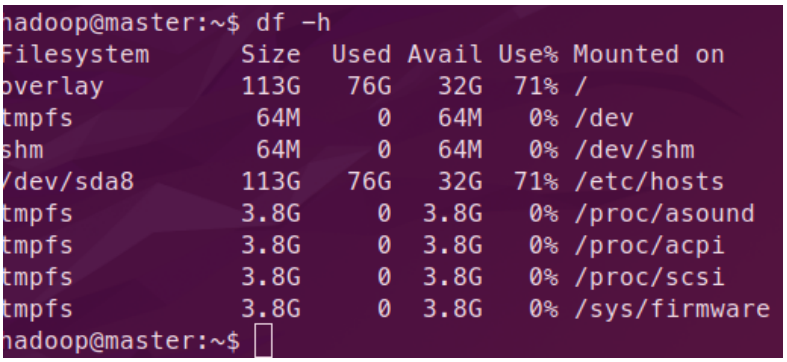
# Práctica 2: HDFS-Contenedores Docker

Nuria Gómez Sánchez del Valle
Fernando Fraile Mulas

Octubre 2023

**Instalación de HDFS en un solo nodo:**

- Utilizando el comando `update-alternatives` se obtiene que java se encuentra instalado en `/usr/lib/jvm/java-8-openjdk-amd64`.
- Ejecutando el comando `ssh-keygen -t rsa -P '' -f  /.ssh/id_rsa` se crea una clave sin contraseña, por lo que no es necesario poner contraseña al realizar el `ssh`. Si se indicara la contraseña en el comando entre comillas habría sido necesario.
- El parámetro `dfs.replication` indica la cantidad de réplicas que se deben mantener para cada bloque de datos almacenado en HDFS. Al ser 1, se está indicando que solamente habrá una copia.
- Los servicios en ejecución son: NameNode, DataNode y SecondaryNameNode.
- Al levantar el docker se está indicando `-p 9870`, por tanto se está indicando que se va a mapear un puerto en el host al puerto dentro del contenedor. Esto permite que los servicios o aplicaciones dentro del contenedor sean accesibles desde fuera del contenedor a través del puerto especificado en el host.
- La capacidad del HDFS es 112.81G. Esto se debe a que es el tamaño del disco virtual del contenedor, es decir, Hadoop tiene asignado todo el espacio del disco del contenedor, como se puede observar en la siguiente figura.




- La capacidad es prácticamente del 0 %. Esto se debe a que todavía no se han introducido archivos en el sistema de archivos distribuidos. La capacidad disponible para DFS es de 31.66GB (28.07 %), ya que en la partición en la que está montado (indicado en `configuration` del HDFS, sistema de ficheros local) ya están ocupados 75.38GB (Non DFS Used) y lo que falta está reservado para datos de configuración.
- Captura de pantalla de la interfaz web.

## Summary

Security is off.

Safemode is off.

1 files and directories, 0 blocks (0 replicated blocks, 0 erasure coded block groups) = 1 total filesystem object(s).

Heap Memory used 112.02 MB of 274.5 MB Heap Memory. Max Heap Memory is 1.68 GB.

Non Heap Memory used 49.53 MB of 51.05 MB Commited Non Heap Memory. Max Non Heap Memory is <unbounded>.

| | |
|---|---|
| **Configured Capacity:** | 112.81 GB |
| **Configured Remote Capacity:** | 0 B |
| **DFS Used:** | 28 KB (0%) |
| **Non DFS Used:** | 75.38 GB |
| **DFS Remaining:** | 31.66 GB (28.07%) |
| **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 |

## Construcción de un cluster HDFS:

- Dockerfile

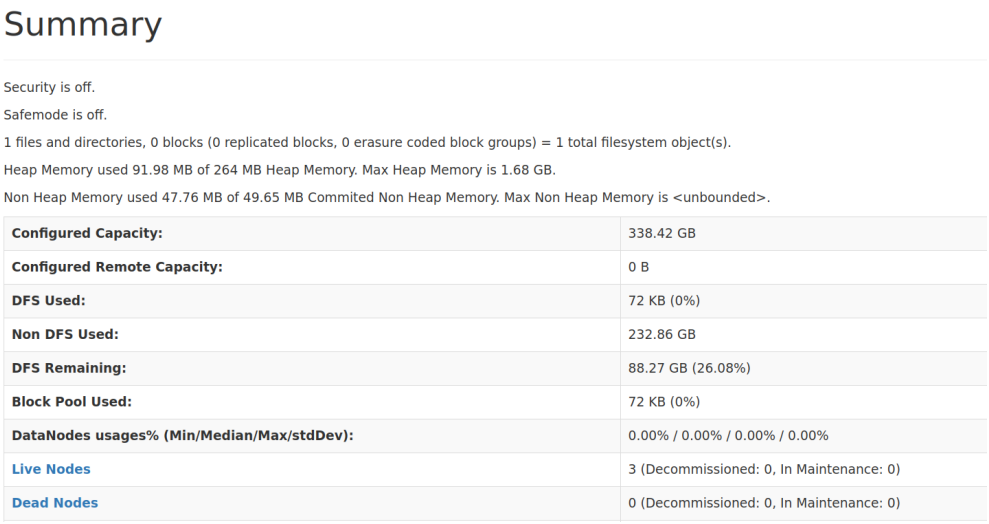
```
    # syntax=docker/dockerfile:1.3-labs
FROM ubuntu:20.04

RUN export DEBIAN_FRONTEND=noninteractive && \
apt-get update && \
apt install -y openjdk-8-jdk wget ssh vim

RUN <<EOF
wget https://downloads.apache.org/hadoop/common
/hadoop-3.3.1/hadoop-3.3.1.tar.gz

tar -xf hadoop-3.3.1.tar.gz
mv hadoop-3.3.1 /usr/local/hadoop

useradd -m hadoop -s /usr/bin/bash
chown -R hadoop:hadoop /usr/local/hadoop

ssh-keygen -A
mkdir /run/sshd

echo 'export HADOOP_HOME=/usr/local/hadoop' >> /etc/bash.bashrc
echo 'export PATH=${PATH}:${HADOOP_HOME}/bin:${HADOOP_HOME}/sbin' >>
/etc/bash.bashrc
echo 'export JAVA_HOME=/usr/lib/jvm/java-8-openjdk-amd64' >>
/etc/bash.bashrc
echo 'export JAVA_HOME=/usr/lib/jvm/java-8-openjdk-amd64' >>
/usr/local/hadoop/etc/hadoop/hadoop-env.sh
EOF
```

```
29
30 USER hadoop
31
32 COPY <<'EOF' /usr/local/hadoop/etc/hadoop/core-site.xml
33 <?xml version="1.0" encoding="UTF-8"?>
34 <?xml-stylesheet type="text/xsl" href="configuration.xsl"?>
35
36 <configuration>
37         <property>
38         <name>fs.defaultFS</name>
39         <value>hdfs://master:9000</value>
40         </property>
41 </configuration>
42
43 EOF
44
45 COPY <<'EOF' /usr/local/hadoop/etc/hadoop/hdfs-site.xml
46 <?xml version="1.0" encoding="UTF-8"?>
47 <?xml-stylesheet type="text/xsl" href="configuration.xsl"?>
48
49 <configuration>
50         <configuration>
51         <property>
52         <name>dfs.replication</name>
53         <value>2</value>
54         </property>
55         <property>
56         <name>dfs.namenode.name.dir</name>
57         <value>file:///usr/local/hadoop/hdfs/namenode</value>
58         </property>
59         <property>
60         <name>dfs.datanode.data.dir</name>
61         <value>file:///usr/local/hadoop/hdfs/datanode</value>
62         </property>
63         </configuration>
64         </configuration>
65
66 EOF
67
68 RUN ssh-keygen -t rsa -P '' -f ~/.ssh/id_rsa && \
69 cat ~/.ssh/id_rsa.pub >> ~/.ssh/authorized_keys
70
71 USER root
```

- Las diferencias en los parámetros Docker para lanzar el master respecto al apartado anterior son las siguientes:

- En el apartado anterior el nombre del contenedor era hdfs y ahora es master, indicado en la opción `--name`.
- En este caso se ha añadido la opción `--network hdfs`, haciendo que el contenedor se conecte a la red "hdfs"creada anteriormente.
- En el caso anterior se utiliza una imagen de Ubuntu (ubuntu:20.04) para crear el contenedor y en este caso se está utilizando la imagen `hdfs-base`, creada a partir del Dockerfile.
- En este caso se vinculando el directorio `/data/0` del host con el directorio `/usr/local/hadoop/hdfs` del contenedor, de manera que si se pierde el contenedor no se perderán los datos.

- La capacidad es 338.42GB. Esto se debe a que ahora se tienen tres Datanodes y cada uno de ellos, como se indicó en el apartado anterior, tiene una capacidad de 112.81GB.



Summary

Security is off.

Safemode is off.

1 files and directories, 0 blocks (0 replicated blocks, 0 erasure coded block groups) = 1 total filesystem object(s).

Heap Memory used 91.98 MB of 264 MB Heap Memory. Max Heap Memory is 1.68 GB.

Non Heap Memory used 47.76 MB of 49.65 MB Commited Non Heap Memory. Max Non Heap Memory is <unbounded>.

| | |
|---|---|
| **Configured Capacity:** | 338.42 GB |
| **Configured Remote Capacity:** | 0 B |
| **DFS Used:** | 72 KB (0%) |
| **Non DFS Used:** | 232.86 GB |
| **DFS Remaining:** | 88.27 GB (26.08%) |
| **Block Pool Used:** | 72 KB (0%) |
| **DataNodes usages% (Min/Median/Max/stdDev):** | 0.00% / 0.00% / 0.00% / 0.00% |
| **Live Nodes** | 3 (Decommissioned: 0, In Maintenance: 0) |
| **Dead Nodes** | 0 (Decommissioned: 0, In Maintenance: 0) |

- Podremos usar la mitad de la capacidad disponible indicada, ya que todos los archivos van a estar replicados una vez. Además, en la práctica, no podremos usar toda la capacidad indicada ya que realmente los tres nodos están ejecutándose en la misma máquina, y el espacio disponible de esta se están multiplicando por tres.

**Manejo de ficheros:**

- Como no hemos indicado un directorio a la hora de subir los ficheros, se han subido a la carpeta de usuario de hadoop, `/user/hadoop`.
- El tamaño de bloque es de 128MB. Como el fichero `sample.txt` es de 1GB, tiene 8 bloques. Como el tamaño del fichero `11-0.txt` es de 170.23KB, menor que el tamaño de un bloque, tiene un solo bloque. Además, todos los bloques estarán replicados una vez. No todos los bloques están almacenados en los mismos nodos. Las réplicas de un bloque estarán en un nodo distinto, para que exista redundancia. Además, los bloques correspondientes a un mismo fichero no tienen por qué estar en los mismos nodos, es decir, un fichero

podría tener un bloque en el master y nodo 1 y otro bloque en el nodo 1 y nodo 2. Sin embargo, en este caso, al subir los ficheros desde el master y al estar este vacío, siempre se está quedando una réplica de todos los bloques para aprovechar la localidad y reducir la carga de realizar copias, pero en general no tendría porqué pasar.

- Uno, ya que se tienen dos réplicas de todos los bloques en diferentes nodos. No tendrían porque perderse todos los ficheros. Por ejemplo, en caso de que el fichero `11-0.txt`, que ocupa solo un bloque, esté en el master y en el nodo 1, y se pierdan dos nodos, siendo estos el nodo 1 y el nodo 2, el fichero aún se encontraría en el master, por tanto no se perdería.