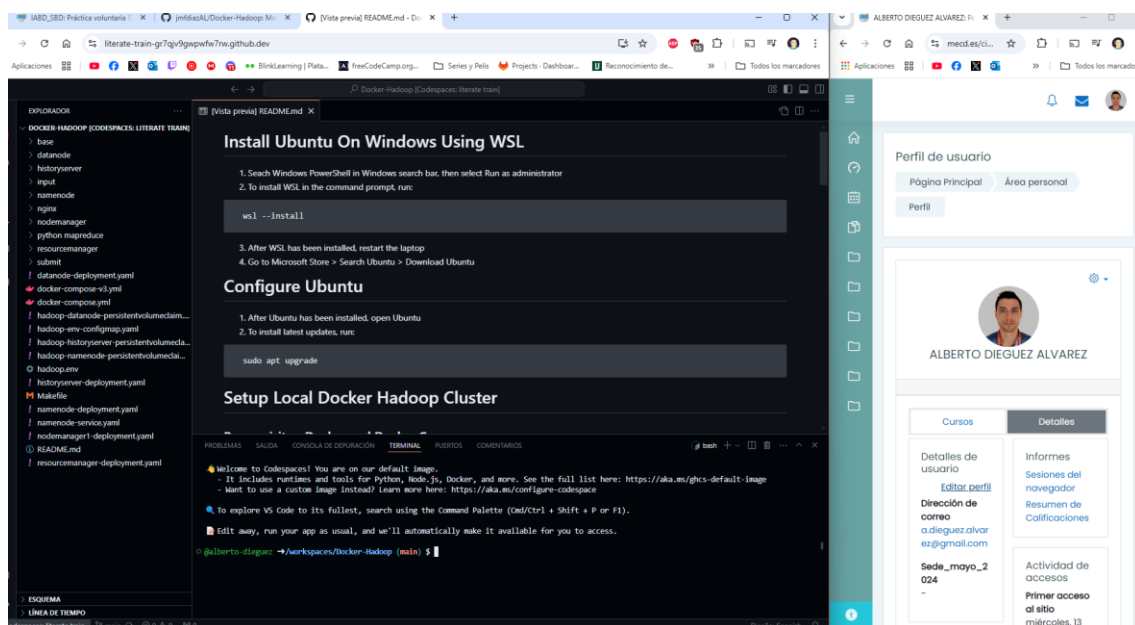


Práctica voluntaria SBD02.a

I. Preparación del entorno

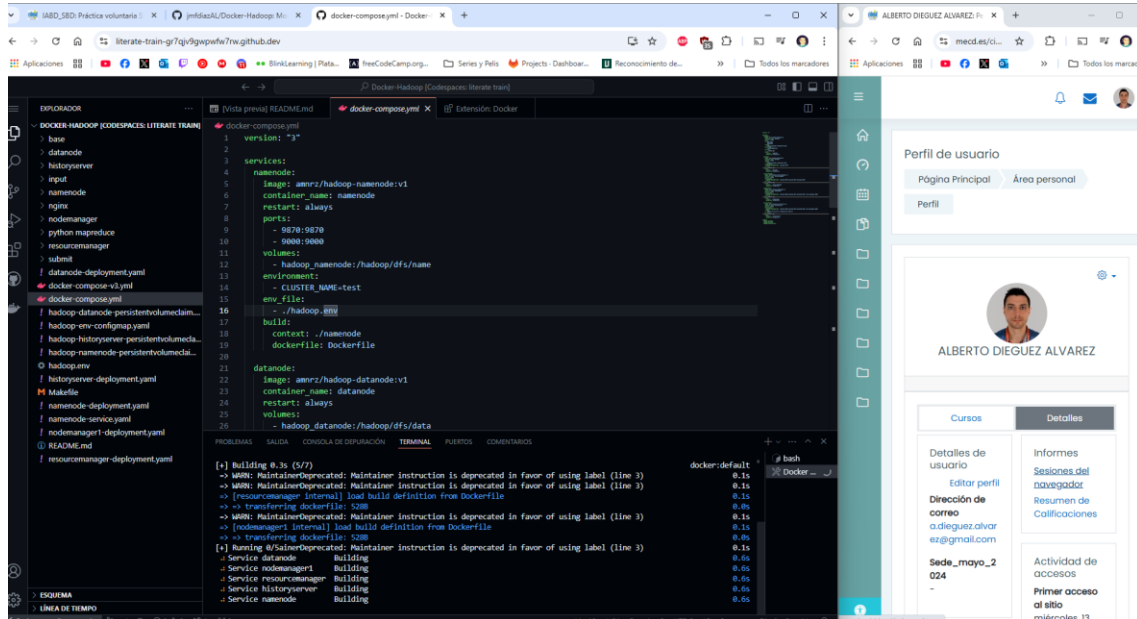
1. Lo primero que haremos será hacer un *fork* del siguiente repositorio: <https://github.com/jmfidiazAL/Docker-Hadoop>, de esta manera, podemos conservar nuestros cambios. (En caso de que hayamos decidido usar una instalación local, clonaremos el repositorio anterior.)
2. Desde nuestro repositorio, pulsaremos en el botón <> **Code**, y luego seleccionaremos la pestaña **Codespaces**:
3. Ahora pulsamos en el botón **Create codespace on master**. Se nos abrirá una instancia en la nube con VS Code. (En el caso de usar un entorno local, simplemente abriremos el repositorio local con VS Code.)



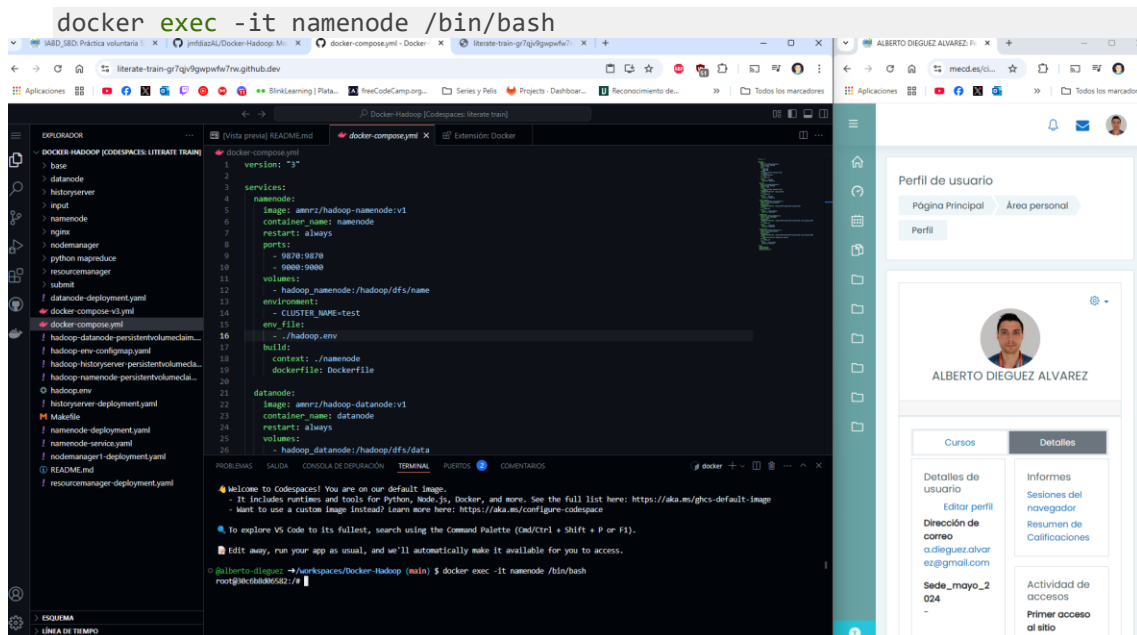
4. Si abrimos bien el archivo **docker-compose.yml** o **docker-compose-v3.yml**, se nos propone instalar el complemento de Docker, pulsamos el botón Instalar.
5. Como hemos mencionado en el punto anterior, disponemos de 2 archivos para Docker Compose:
 1. **docker-compose-v3.yml**, que utiliza las imágenes clásicas de [Big Data Europe](#).
 2. **docker-compose.yml**, que utiliza unas imágenes basadas en las anteriores pero que nos van a permitir añadir Python como veremos en la última parte.

6. A continuación, pulsamos sobre el archivo **docker-compose.yml** con el botón derecho, y en el menú contextual seleccionaremos **Compose Up**.
7. Se descargarán las imágenes correspondientes, y se levantará la pila de contenedores.

Se ejecuta compose up



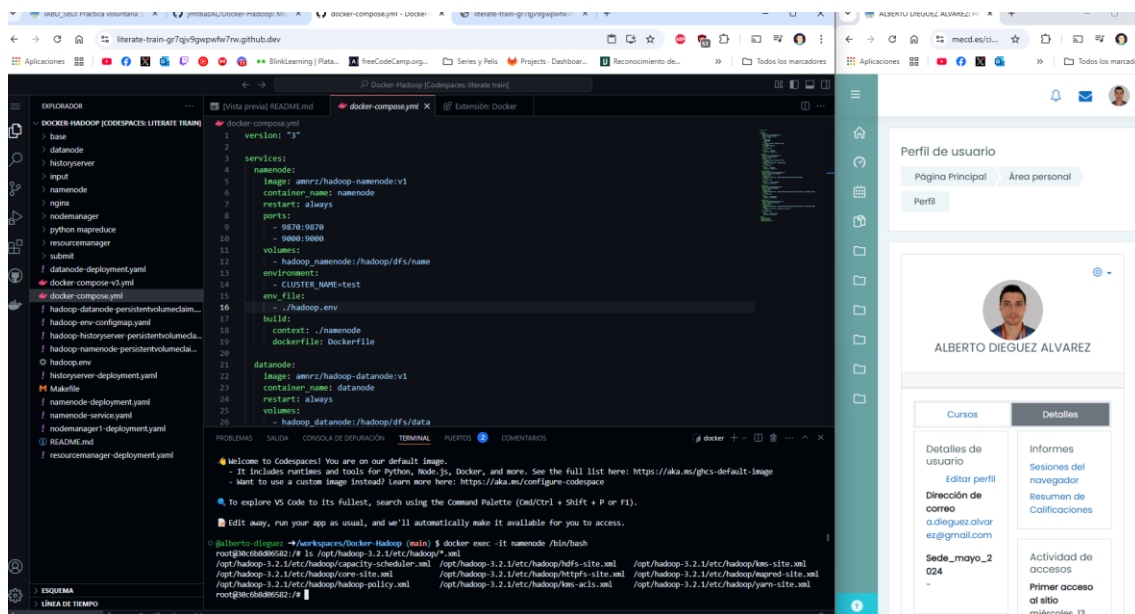
8. Ahora, en la ventana inferior de **Terminal**, introduciremos el comando siguiente para abrir un **shell** en el contenedor:



A partir de ahora, podemos entrar en la consola del clúster.

9. Con el siguiente comando, podemos ver todos los archivos de configuración:

`ls /opt/hadoop-3.2.1/etc/hadoop/*.xml`



II. Crear un archivo en HDFS

En esta parte vamos a subir un archivo a HDFS y vamos a consultarlo desde la línea de comandos.

1. Creamos el directorio `user/root/input`:

`hdfs dfs -mkdir -p /user/root/input`

```

Welcome to Codespaces! You are on our default image.
- It includes runtimes and tools for Python, Node.js, Docker, and more. See the full list here: https://aka.ms/ghcs-default-image
- Want to use a custom image instead? Learn more here: https://aka.ms/configure-codespace

To explore VS Code to its fullest, search using the Command Palette (Cmd/Ctrl + Shift + P or F1).

Edit away, run your app as usual, and we'll automatically make it available for you to access.

@alberto-dieguez →/workspaces/Docker-Hadoop (main) $ docker exec -it namenode /bin/bash
root@30c6b8d06582:/# ls /opt/hadoop-3.2.1/etc/hadoop/*.xml
/opt/hadoop-3.2.1/etc/hadoop/capacity-scheduler.xml /opt/hadoop-3.2.1/etc/hadoop/hdfs-site.xml /opt/hadoop-3.2.1/etc/hadoop/kms-site.xml
/opt/hadoop-3.2.1/etc/hadoop/core-site.xml /opt/hadoop-3.2.1/etc/hadoop/httpfs-site.xml /opt/hadoop-3.2.1/etc/hadoop/mapred-site.xml
/opt/hadoop-3.2.1/etc/hadoop/hadoop-policy.xml /opt/hadoop-3.2.1/etc/hadoop/kms-acls.xml /opt/hadoop-3.2.1/etc/hadoop/yarn-site.xml
root@30c6b8d06582:/# hdfs dfs -mkdir -p /user/root/input
root@30c6b8d06582:/#

```

2. Copiamos los archivos xml de configuración en el directorio de entrada:

`hdfs dfs -put $HADOOP_HOME/etc/hadoop/*.xml /user/root/input`

```

/opt/hadoop-3.2.1/etc/hadoop/capacity-scheduler.xml /opt/hadoop-3.2.1/etc/hadoop/hdfs-site.xml /opt/hadoop-3.2.1/etc/hadoop/kms-site.xml
/opt/hadoop-3.2.1/etc/hadoop/core-site.xml /opt/hadoop-3.2.1/etc/hadoop/httpfs-site.xml /opt/hadoop-3.2.1/etc/hadoop/mapred-site.xml
/opt/hadoop-3.2.1/etc/hadoop/hadoop-policy.xml /opt/hadoop-3.2.1/etc/hadoop/kms-acls.xml /opt/hadoop-3.2.1/etc/hadoop/yarn-site.xml

root@30c6b8d06582:/# hdfs dfs -mkdir -p /user/root/input
root@30c6b8d06582:/# hdfs dfs -put $HADOOP_HOME/etc/hadoop/*.xml /user/root/input
2024-12-23 00:18:57,965 INFO sasl.SaslDataTransferClient: SASL encryption trust check: localHostTrusted = false, remoteHostTrusted = false
2024-12-23 00:18:58,484 INFO sasl.SaslDataTransferClient: SASL encryption trust check: localHostTrusted = false, remoteHostTrusted = false
2024-12-23 00:18:58,508 INFO sasl.SaslDataTransferClient: SASL encryption trust check: localHostTrusted = false, remoteHostTrusted = false
2024-12-23 00:18:58,928 INFO sasl.SaslDataTransferClient: SASL encryption trust check: localHostTrusted = false, remoteHostTrusted = false
2024-12-23 00:18:59,349 INFO sasl.SaslDataTransferClient: SASL encryption trust check: localHostTrusted = false, remoteHostTrusted = false
2024-12-23 00:18:59,369 INFO sasl.SaslDataTransferClient: SASL encryption trust check: localHostTrusted = false, remoteHostTrusted = false
2024-12-23 00:18:59,788 INFO sasl.SaslDataTransferClient: SASL encryption trust check: localHostTrusted = false, remoteHostTrusted = false
2024-12-23 00:18:59,808 INFO sasl.SaslDataTransferClient: SASL encryption trust check: localHostTrusted = false, remoteHostTrusted = false
2024-12-23 00:19:00,228 INFO sasl.SaslDataTransferClient: SASL encryption trust check: localHostTrusted = false, remoteHostTrusted = false
root@30c6b8d06582:/#

```

3. Creamos el archivo **data.txt**: en el directorio actual, y lo cargamos con valores de ejemplo:

curl https://raw.githubusercontent.com/jimfdiazAL/ooxwv-docker_hadoop/master/SampleMapReduce.txt --output data.txt

```
2024-12-23 00:18:58,508 INFO sasl.SaslDataTransferClient: SASL encryption trust check: localhostTrusted = false, remoteHostTrusted = false
2024-12-23 00:18:58,928 INFO sasl.SaslDataTransferClient: SASL encryption trust check: localhostTrusted = false, remoteHostTrusted = false
2024-12-23 00:18:59,349 INFO sasl.SaslDataTransferClient: SASL encryption trust check: localhostTrusted = false, remoteHostTrusted = false
2024-12-23 00:18:59,369 INFO sasl.SaslDataTransferClient: SASL encryption trust check: localhostTrusted = false, remoteHostTrusted = false
2024-12-23 00:18:59,788 INFO sasl.SaslDataTransferClient: SASL encryption trust check: localhostTrusted = false, remoteHostTrusted = false
2024-12-23 00:18:59,808 INFO sasl.SaslDataTransferClient: SASL encryption trust check: localhostTrusted = false, remoteHostTrusted = false
2024-12-23 00:19:00,228 INFO sasl.SaslDataTransferClient: SASL encryption trust check: localhostTrusted = false, remoteHostTrusted = false
root@30c6b8d06582:/# curl https://raw.githubusercontent.com/jimfdiazAL/ooxwv-docker_hadoop/master/SampleMapReduce.txt --output data.txt
% Total % Received % Xferd Average Speed Time Time Time Current
Dload Upload Total Spent Left Speed
100 6858 100 6858 0 0 30807 0 --:--:-- --:--:-- --:--:-- 30891
root@30c6b8d06582:/# ls
KEYS boot dev etc hadoop-data lib media opt root run.sh srv tmp var
bin data.txt entrypoint.sh hadoop home lib64 mnt proc run sbin sys usr
root@30c6b8d06582:/#
```

4. Copiamos el archivo **data.txt** en el directorio **user/root**:

hdfs dfs -put data.txt /user/root/

```
2024-12-23 00:18:59,349 INFO sasl.SaslDataTransferClient: SASL encryption trust check: localhostTrusted = false, remoteHostTrusted = false
2024-12-23 00:18:59,369 INFO sasl.SaslDataTransferClient: SASL encryption trust check: localhostTrusted = false, remoteHostTrusted = false
2024-12-23 00:18:59,788 INFO sasl.SaslDataTransferClient: SASL encryption trust check: localhostTrusted = false, remoteHostTrusted = false
2024-12-23 00:18:59,808 INFO sasl.SaslDataTransferClient: SASL encryption trust check: localhostTrusted = false, remoteHostTrusted = false
2024-12-23 00:19:00,228 INFO sasl.SaslDataTransferClient: SASL encryption trust check: localhostTrusted = false, remoteHostTrusted = false
root@30c6b8d06582:/# curl https://raw.githubusercontent.com/jimfdiazAL/ooxwv-docker_hadoop/master/SampleMapReduce.txt --output data.txt
% Total % Received % Xferd Average Speed Time Time Time Current
Dload Upload Total Spent Left Speed
100 6858 100 6858 0 0 30807 0 --:--:-- --:--:-- --:--:-- 30891
root@30c6b8d06582:/# ls
KEYS boot dev etc hadoop-data lib media opt root run.sh srv tmp var
bin data.txt entrypoint.sh hadoop home lib64 mnt proc run sbin sys usr
root@30c6b8d06582:/# hdfs dfs -put data.txt /user/root/
2024-12-23 00:19:36,188 INFO sasl.SaslDataTransferClient: SASL encryption trust check: localhostTrusted = false, remoteHostTrusted = false
root@30c6b8d06582:/#
```

5. Verificamos que el contenido se ha copiado en HDFS:

hdfs dfs -cat /user/root/data.txt

```
PROBLEMAS SALIDA CONSOLA DE DEPURACIÓN TERMINAL PUERTOS 2 COMENTARIOS
root@30c6b8d06582:/# hdfs dfs -cat /user/root/data.txt
2024-12-23 00:19:53,140 INFO sasl.SaslDataTransferClient: SASL encryption trust check: localhostTrusted = false, remoteHostTrusted = false
What is big data? More than volume, velocity and variety -

Get your head around the topic of big data
By J. Steven Perry
Published May 22, 2017

You've heard of Big Data, right? We're all supposed to say yes, and Big Data is one of those topics I thought I understood until I tried explain
ing it. I realized that I needed to get my head around it at a high level. If you're like I was, then this blog post is for you.

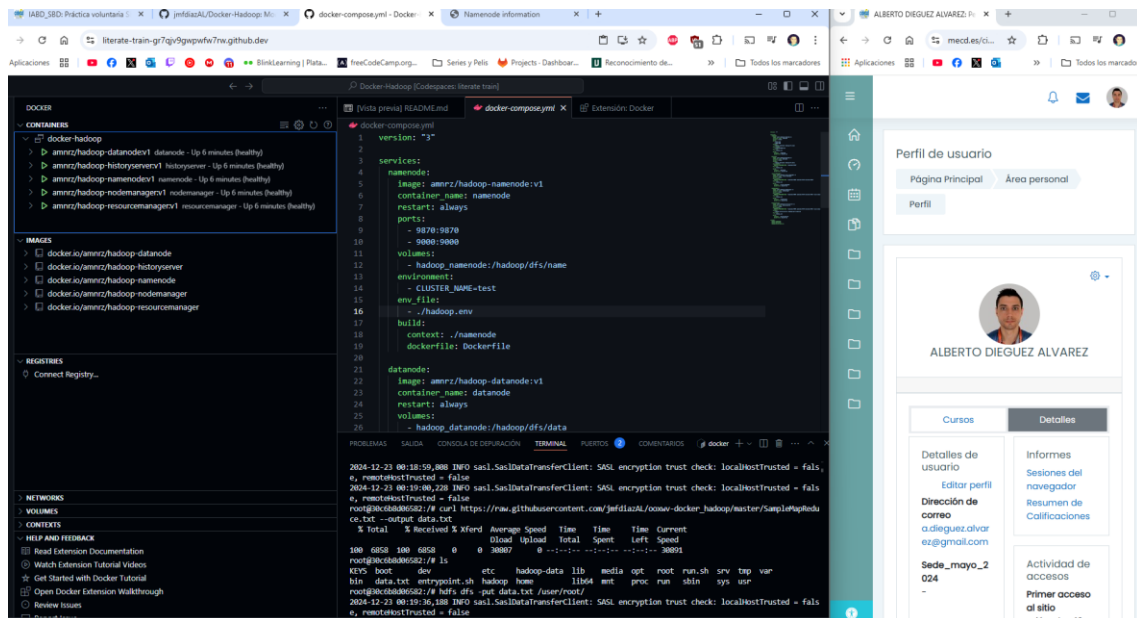
The Problem(s)
Any technology is only useful if it solves a problem (or problems). So what problem(s) does Big Data solve?

As we all know, there is data, lots of it: historical data, sure, but also new data generated from social media apps, click stream data from web
```

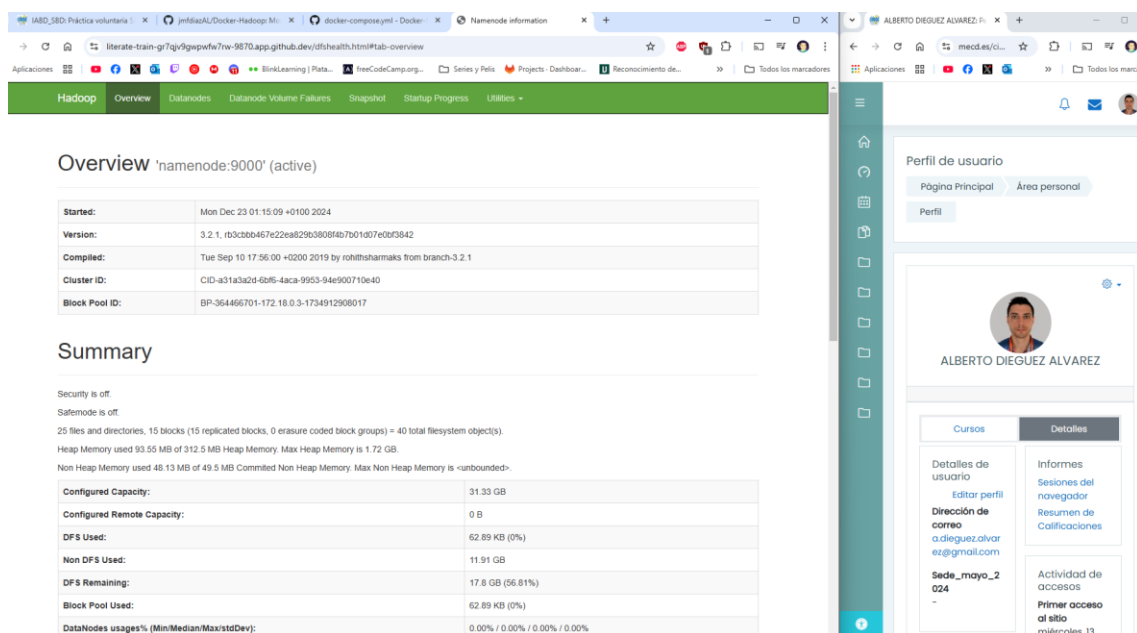
III. Ver en HDFS

En la tercera parte, vamos a ver los archivos subidos a HDFS en la interfaz gráfica.

1. Pulsamos en la pestaña lateral en la extensión Docker, se nos mostrará la pila de contenedores ejecutándose y, pasando el ratón por encima del contenedor de nombre **namenode**, se abrirá un recuadro con los puertos disponibles y seleccionaremos el puerto 9870. También podemos abrir el menú contextual que se despliega con el botón derecho, sobre el mismo contenedor y pulsar la opción Open in Browser, y elegir el mismo puerto. Se nos abrirá una nueva ventana con la interfaz gráfica de nuestro clúster.



The screenshot shows the Docker Desktop interface. On the left, the 'CONTAINERS' list shows several containers, including 'namenode'. The 'docker-compose.yml' file is open in the center, showing the configuration for the 'namenode' service. The 'namenode' service is configured with the image 'amrzt/hadoop-namenode:v1', container name 'namenode', and ports 9870-9870 and 9000-9000. The 'namenode' container is highlighted in the 'CONTAINERS' list. The 'TERMINAL' tab at the bottom shows the output of the 'docker-compose up' command, indicating that the 'namenode' container is running successfully.



The screenshot shows the Hadoop Overview page. The 'Overview' tab is selected, showing the 'namenode:9000' (active) status. The page displays various metrics and a summary of the cluster's health. The 'Summary' section shows that the cluster is healthy, with 25 files and directories, 15 blocks (15 replicated blocks, 0 erasure coded block groups) = 40 total filesystem object(s). The 'DataNodes' section shows that the cluster is healthy, with 1 data node (1/1) and 1 block pool (1/1).

Started:	Mon Dec 23 01:15:09 +0100 2024
Version:	3.2.1, r03cbb467e22ca829c3808f4b7b01d07e09f5842
Compiled:	Tue Sep 10 17:56:00 +0200 2019 by rohitsharmak from branch-3.2.1
Cluster ID:	CID-a31a3a2d-62b5-4aca-9953-94e900710e40
Block Pool ID:	BP-364466701-172.18.0.3-1734912908017

Summary

Security is off.
Safemode is off.
25 files and directories, 15 blocks (15 replicated blocks, 0 erasure coded block groups) = 40 total filesystem object(s).
Heap Memory used 93.55 MB of 312.5 MB Heap Memory. Max Heap Memory is 1.72 GB.
Non Heap Memory used 48.13 MB of 49.5 MB Committed Non Heap Memory. Max Non Heap Memory is 'unbounded'.

Configured Capacity:	31.33 GB
Configured Remote Capacity:	0 B
DFS Used:	62.89 KB (0%)
Non DFS Used:	11.91 GB
DFS Remaining:	17.8 GB (56.81%)
Block Pool Used:	62.89 KB (0%)
DataNodes usages% (Min/Median/Max/stdDev):	0.00% / 0.00% / 0.00% / 0.00%

2. A continuación, pulsamos en la opción Utilities y en Browse the file system:

The screenshot shows the Hadoop web interface with the 'Browse Directory' view. The address bar shows the path '/'. The table below lists the contents of the root directory:

Permission	Owner	Group	Size	Last Modified	Replication	Block Size	Name
drwxr-xr-x	root	supergroup	0 B	Dec 23 01:15	0	0 B	rmstate
drwxr-xr-x	root	supergroup	0 B	Dec 23 01:18	0	0 B	user

Showing 1 to 2 of 2 entries

Hadoop, 2019.

3. Pulsamos en **user** y en **root**:

The screenshot shows the Hadoop web interface with the 'Browse Directory' view. The address bar shows the path '/user/root'. The table below lists the contents of the '/user/root' directory:

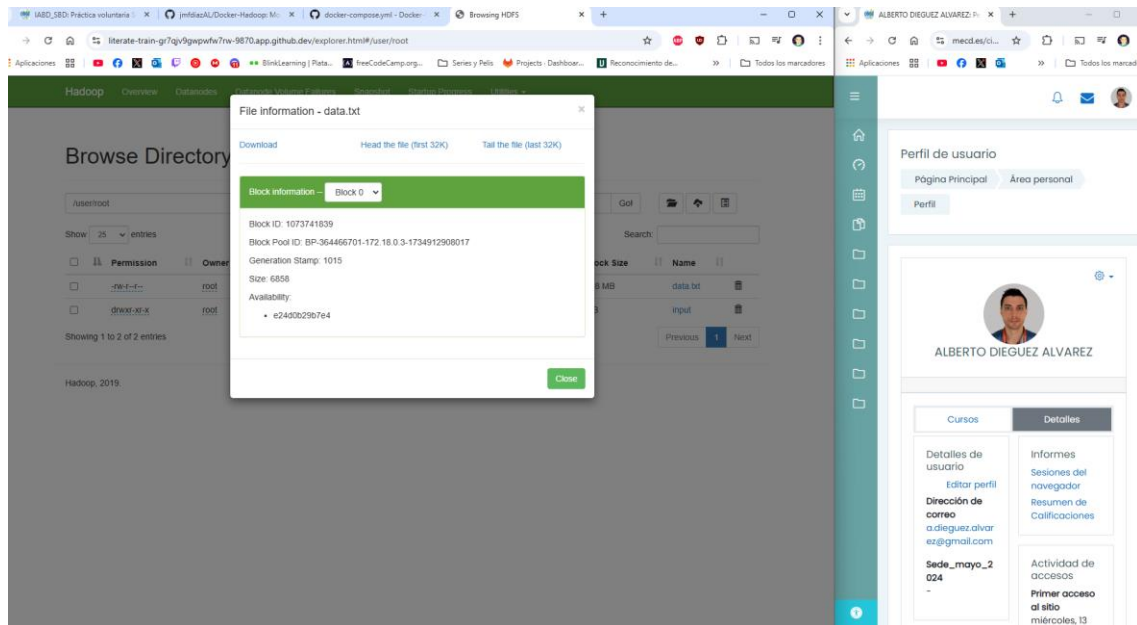
Permission	Owner	Group	Size	Last Modified	Replication	Block Size	Name
-rw-r--r--	root	supergroup	6.7 KB	Dec 23 01:19	3	128 MB	data.txt
drwxr-xr-x	root	supergroup	0 B	Dec 23 01:19	0	0 B	input

Showing 1 to 2 of 2 entries

Hadoop, 2019.

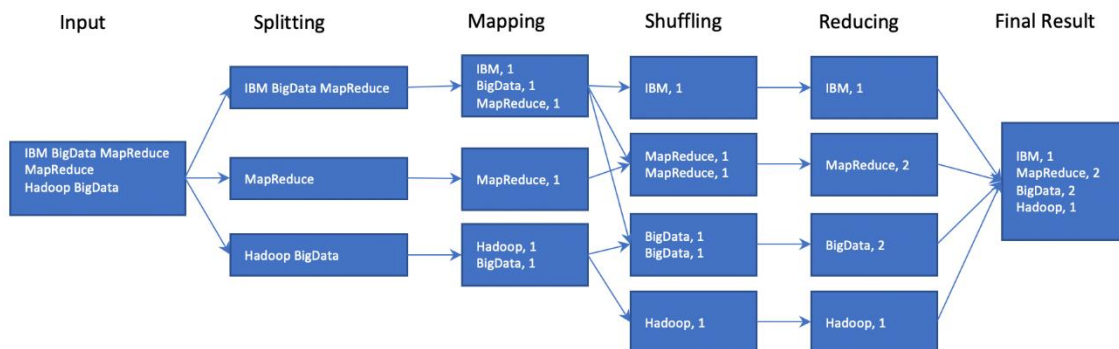
4. Observa que el tamaño del bloque es de 128 MB, aunque el tamaño del archivo es en realidad mucho más pequeño. Esto se debe a que el tamaño de bloque predeterminado utilizado por HDFS es 128 MB.

5. Puedes hacer click en el archivo para revisarlo. Nos da información sobre el archivo en términos de número de bytes, identificación de bloque, etc.



IV. Ejecutar un ejemplo MapReduce para contar palabras

En este apartado vamos a ver una operación sencilla MapReduce sobre un archivo de ejemplo. En la siguiente imagen se puede ver un resumen del procesamiento.



Para ello seguiremos los pasos:

1. Nos movemos al directorio **opt/hadoop-3.2.1**:

cd opt/hadoop-3.2.1

4. Cargamos el archivo **sample.txt** en HDFS:

hdfs dfs -put sample.txt /user/root/

```
Source code repository https://gitbox.apache.org/repos/asf/hadoop.git -r b3cbbb467e22ea829b3808f4b7b01d07e0bf3
842
Compiled by rohithsharmaks on 2019-09-10T15:56Z
Compiled with protoc 2.5.0
From source with checksum 776eaf9eee9c0ffc370bcbcb1888737
This command was run using /opt/hadoop-3.2.1/share/hadoop/common/hadoop-common-3.2.1.jar
root@30c6b8d06582:/opt/hadoop-3.2.1# curl https://cf-courses-data.s3.us.cloud-object-storage.appdomain.cloud/I
BM-BD0225EN-SkillsNetwork/labs/data/data.txt --output sample.txt
  % Total    % Received % Xferd  Average Speed   Time    Time     Time  Current
                                 Dload  Upload  Total   Spent    Left     Speed
100    47    100    47    0    0    50      0  0:00:01  0:00:01  0:00:00  50
root@30c6b8d06582:/opt/hadoop-3.2.1# hdfs dfs -put sample.txt /user/root/
2024-12-23 00:27:42,738 INFO sasl.SaslDataTransferClient: SASL encryption trust check: localHostTrusted = fals
e, remoteHostTrusted = false
root@30c6b8d06582:/opt/hadoop-3.2.1#
```

5. Ejecutamos el archivo Java compilado de ejemplo para contar palabras con MapReduce, y copiamos la salida en el directorio **/user/root/output**:

bin/hadoop jar share/hadoop/mapreduce/hadoop-mapreduce-examples-3.2.1.jar wordcount sample.txt output

6. Una vez haya terminado la ejecución, vemos los archivos generados en el directorio de salida:

hdfs dfs -ls /user/root/output

8. Nos mostrará una salida parecida a la siguiente:

```
2024-12-20 20:52:01,009 INFO sasI.SaslDataTransferClient: SASL encryption trust check:
localhostTrusted = false, remoteHostTrusted = false
```

BigData 2

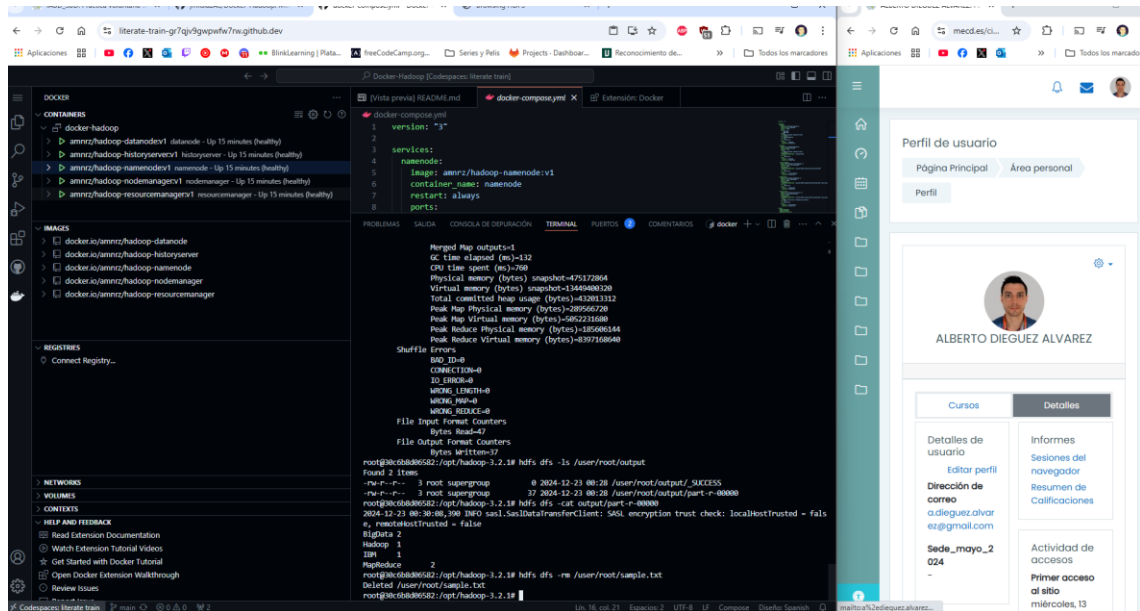
Hadoop 1

IBM 1

MapReduce 2

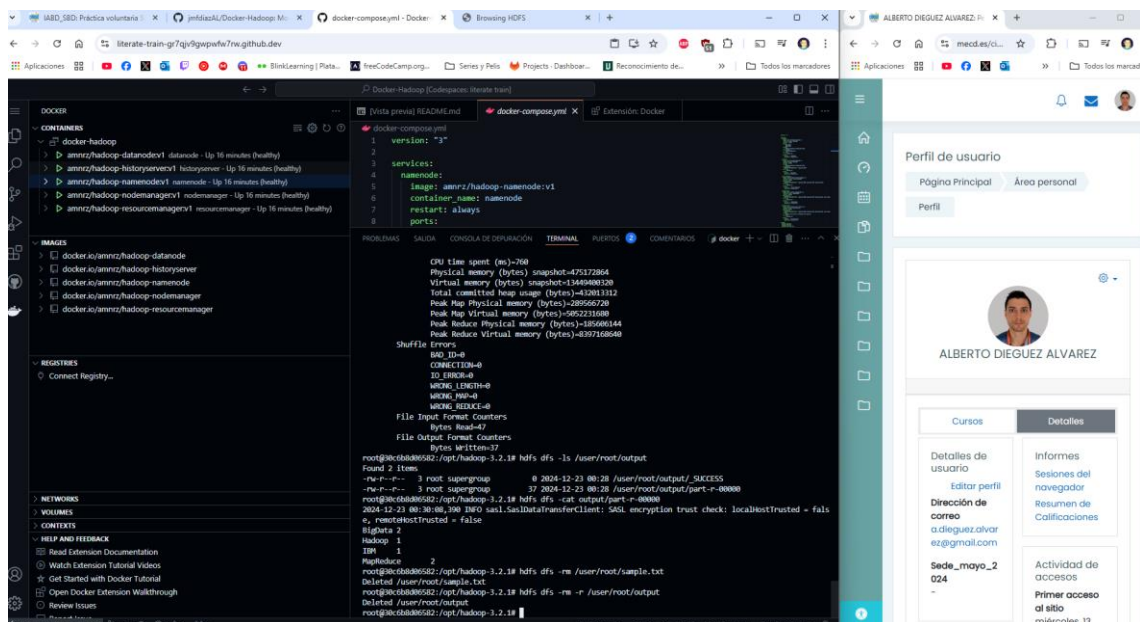
9. Para finalizar, podemos eliminar el archivo **sample.txt**:

```
hdfs dfs -rm /user/root/sample.txt
```



10. Y el directorio **/user/root/output**:

```
hdfs dfs -rm -r /user/root/output
```

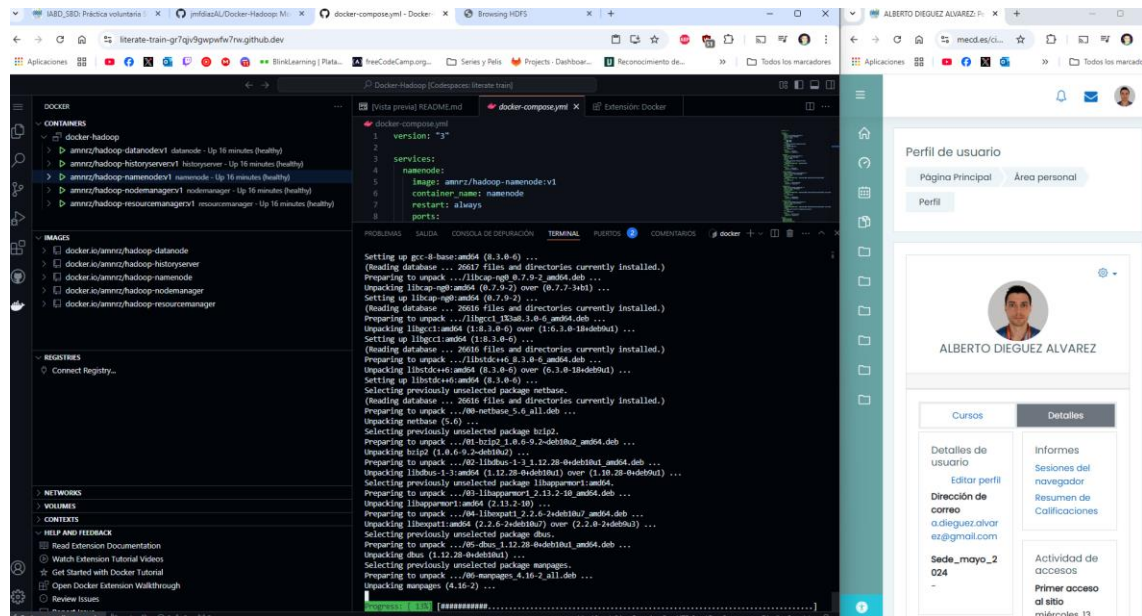


V. Acceso mediante Python

En este apartado vamos a usar el paquete **snakebite** para acceder a HDFS mediante Python:

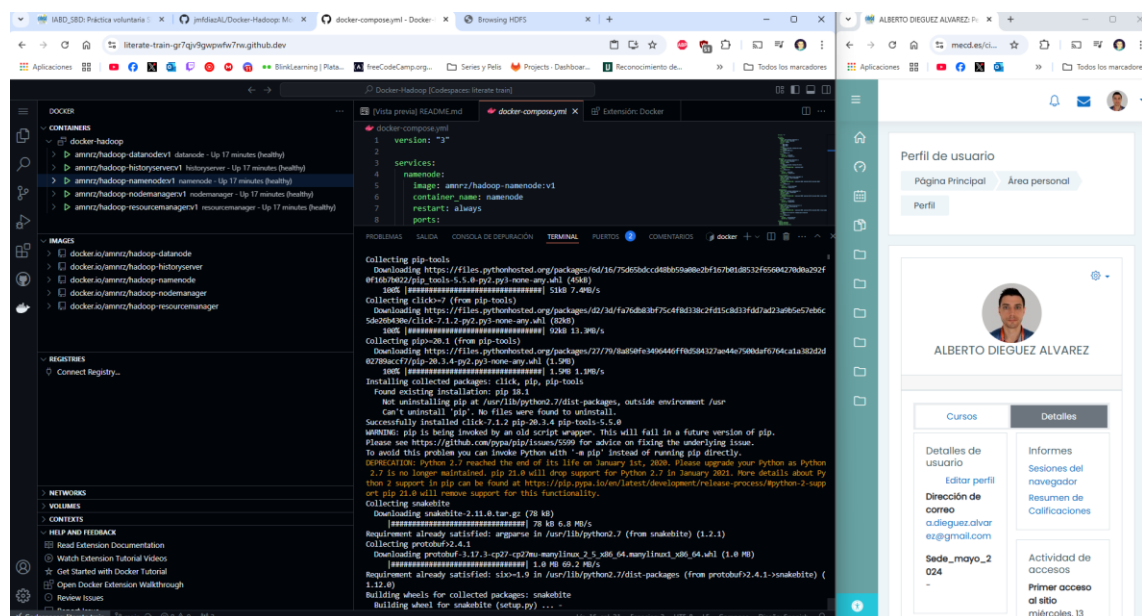
1. En la consola del contenedor **namenode** que hemos venido utilizando hasta ahora, escribiremos:

apt update && apt install python && apt install python-pip -y



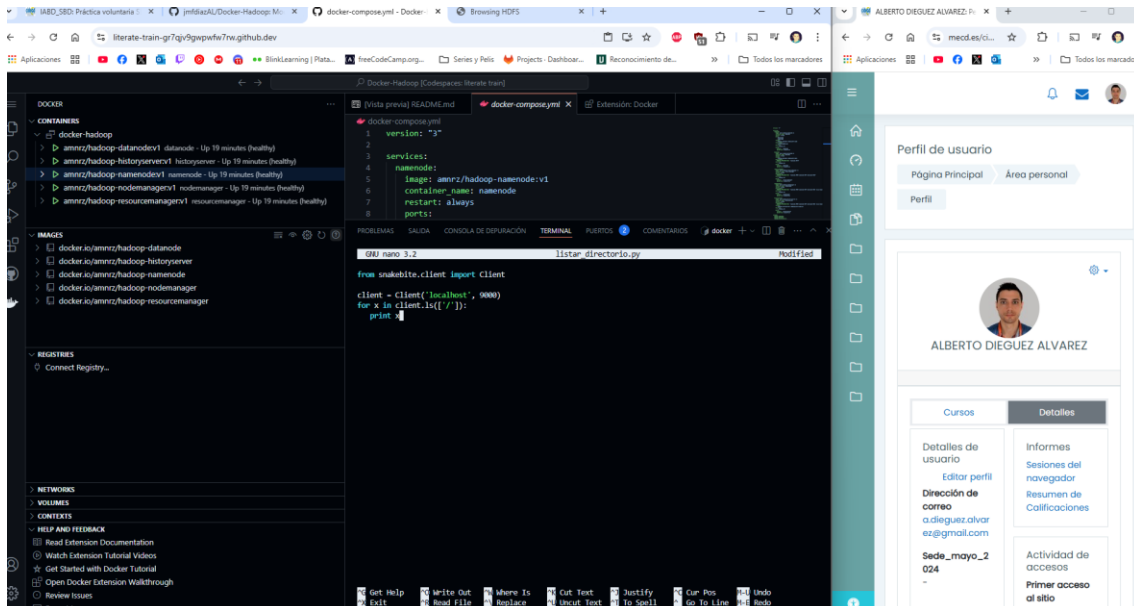
2. Instalamos con **pip** los paquetes necesarios:

pip install pip-tools && pip install snakebite



3. Nos movemos al directorio **/root** por ejemplo y creamos el fichero **listar_directorio.py** con el siguiente contenido:

```
from snakebite.client import Client
client = Client('localhost', 9000)
for x in client.ls(['/']):
    print x
```



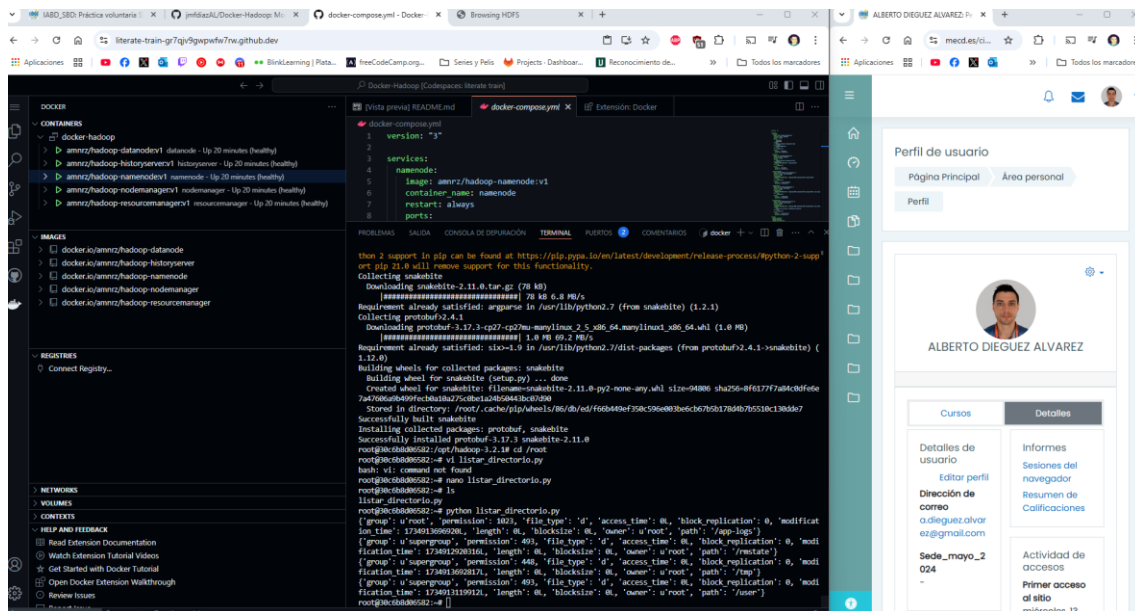
```
Requirement already satisfied: six>=1.9 in /usr/lib/python2.7/dist-packages (from protobuf>2.4.1->snakebite) (1.12.0)
Building wheels for collected packages: snakebite
Building wheel for snakebite (setup.py) ... done
Created wheel for snakebite: filename=snakebite-2.11.0-py2-none-any.whl size=94806 sha256=8f6177f7a84c0dfe6e7a47606a9b499fecb0a10a275c0be1a24b58443bc07d90
Stored in directory: /root/.cache/pip/wheels/86/db/ed/f66b449ef350c596e003be6cb67b5b178d4b7b5510c130dde7
Successfully built snakebite
Installing collected packages: protobuf, snakebite
Successfully installed protobuf-3.17.3 snakebite-2.11.0
root@30c6b8d06582:/opt/hadoop-3.2.1# cd /root
root@30c6b8d06582:~# vi listar_directorio.py
bash: vi: command not found
root@30c6b8d06582:~# nano listar_directorio.py
root@30c6b8d06582:~# ls
listar_directorio.py
root@30c6b8d06582:~#
```

4. Si ejecutamos el archivo anterior, podremos ver la siguiente salida:

```
root@83e2d8f60697:~# python listar_directorio.py
```

```
{'group': u'supergroup', 'permission': 493, 'file_type': 'd', 'access_time': 0L, 'block_replication': 0, 'modification_time': 1734798473139L, 'length': 0L, 'blocksize': 0L, 'owner': u'root', 'path': '/rmstate'}
```

```
{'group': u'supergroup', 'permission': 493, 'file_type': 'd', 'access_time': 0L, 'block_replication': 0, 'modification_time': 1734800493043L, 'length': 0L, 'blocksize': 0L, 'owner': u'root', 'path': '/user'}
```

- Podemos probar el resto de código que aparece en la unidad para crear directorios, eliminar ficheros y directorios, copiar ficheros de HDFS a local y mostrar el contenido de ficheros HDFS.

Con esta práctica hemos realizado una pequeña introducción a HDFS.