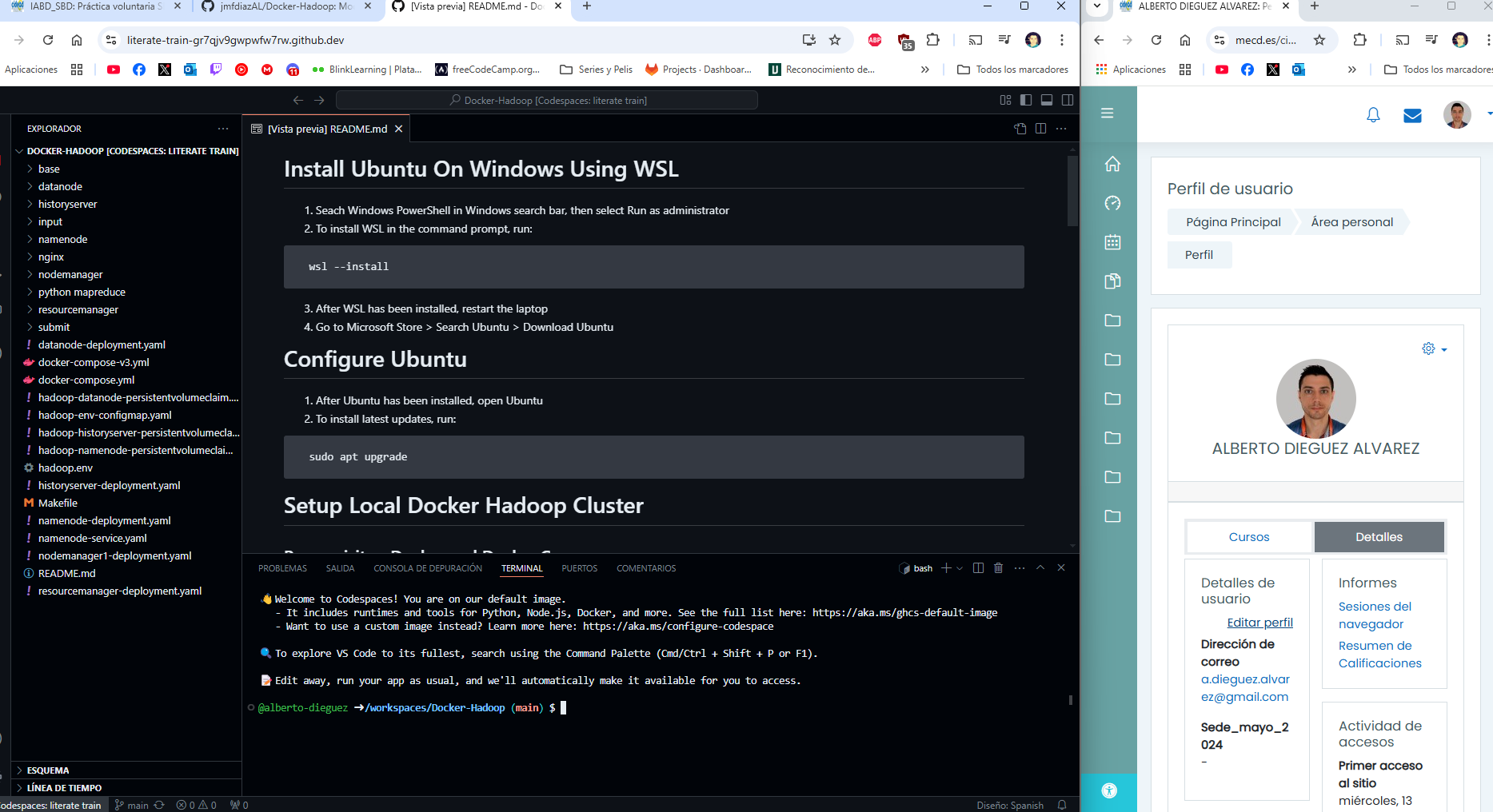
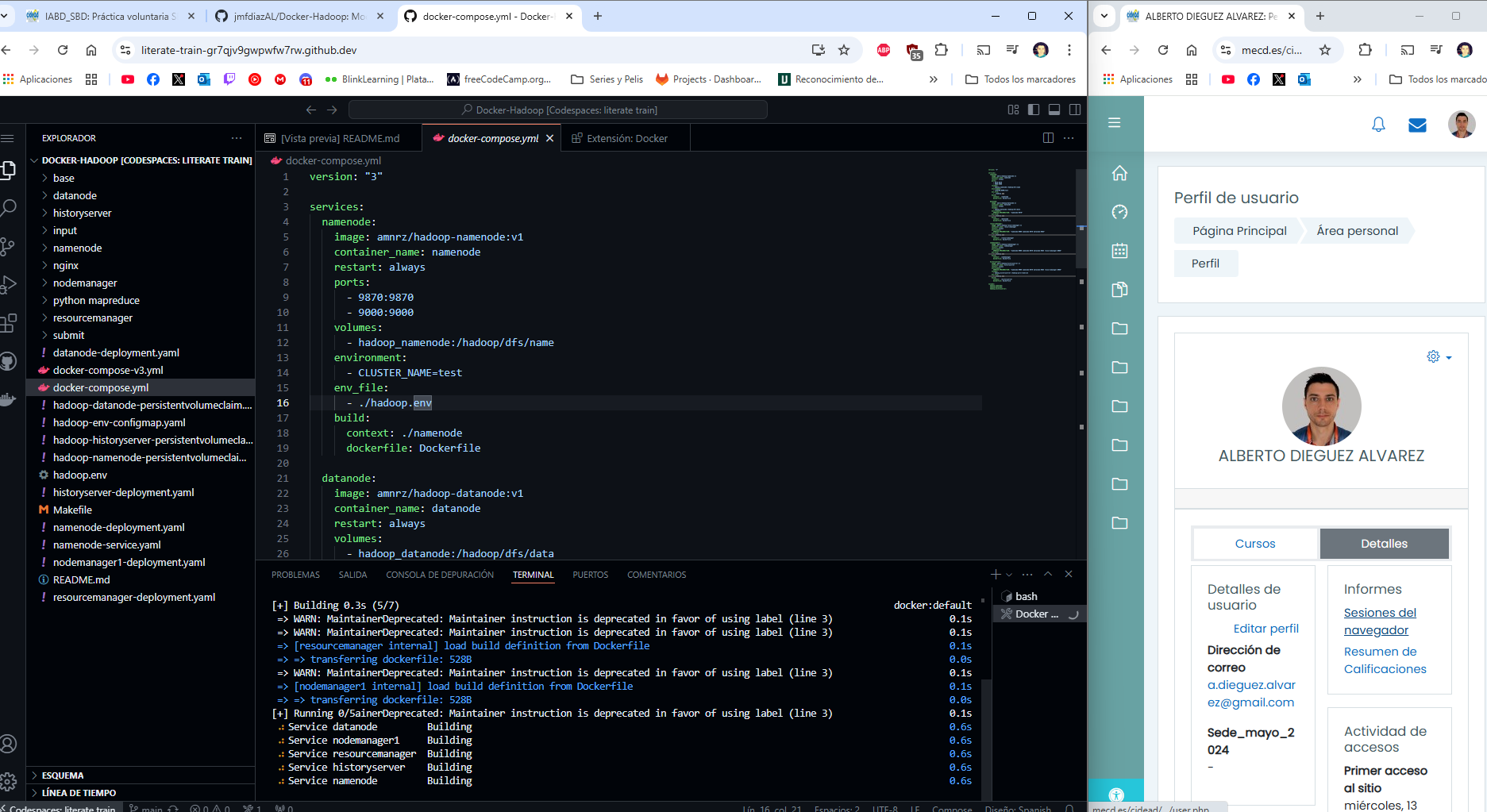
Práctica voluntaria SBD02.a

# I. Preparación del entorno

1. Lo primero que haremos será hacer un *fork* del siguiente repositorio: [**https://github.com/jmfdiazAL/Docker-Hadoop**](https://github.com/jmfdiazAL/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.)

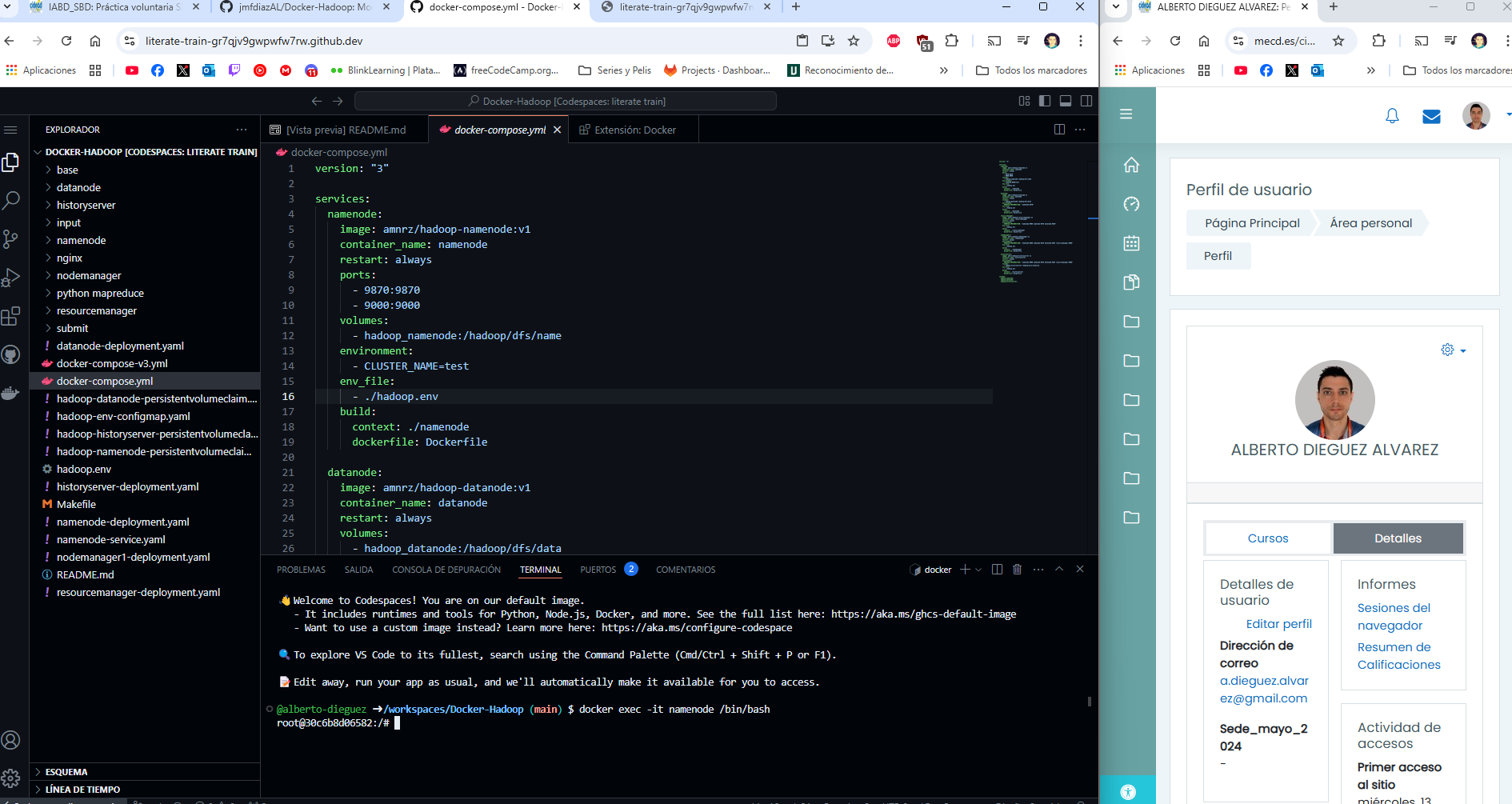


1. Si abrimos bien el archivo **docker-compose.yml** o **docker-compose-v3.yml**,  se nos propone instalar el complemento de Docker, pulsamos el botón Instalar.
2. Como hemos mencionado en el punto anterior, disponemos de 2 archivos para Docker Compose:
   1. **docker-compose-v3.yml**, que utiliza las imágenes clásicas de [**Big Data Europe**](https://hub.docker.com/u/bde2020).
   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 parteé.
3. A continuación, pulsamos sobre el archivo **docker-compose.yml** con el botón derecho, y en el menú contextual seleccionaremos **Compose Up**.
4. Se descargarán las imágenes correspondientes, y se levantará la pila de contenedores.

Se ejecuta compose up  


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

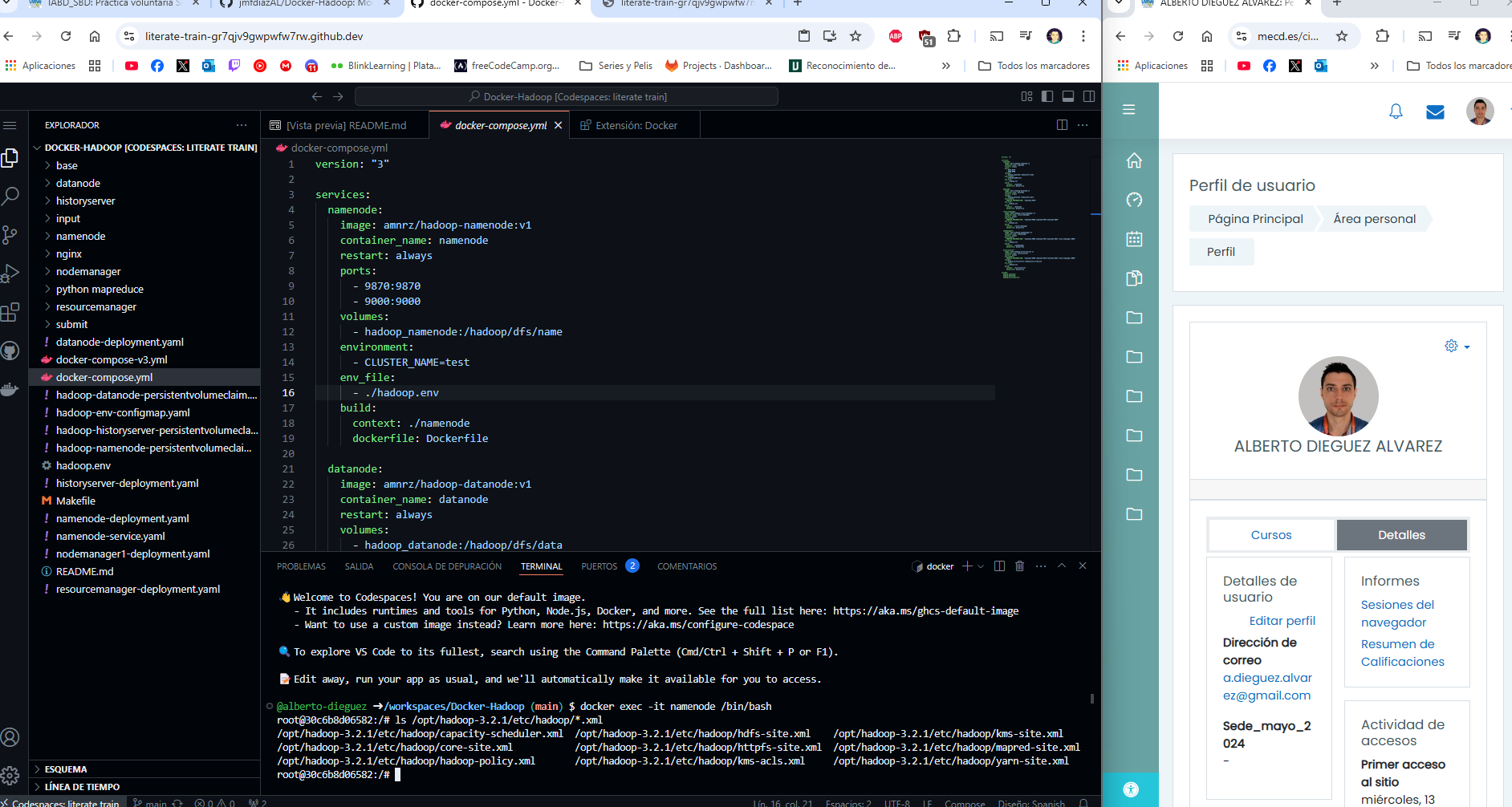
docker exec -it namenode /bin/bash



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

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

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

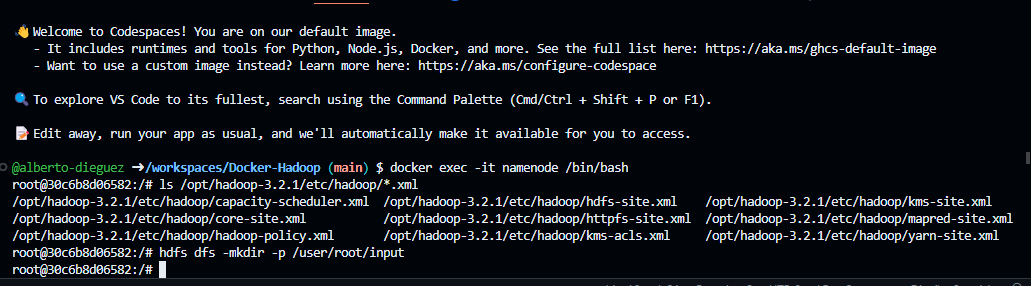


# II. Crear un archivo en HDFS

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

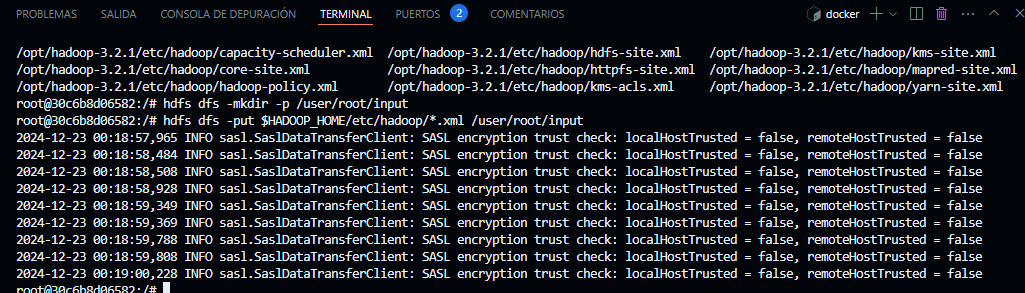
1. Creamos el directorio **user/root/input**:

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



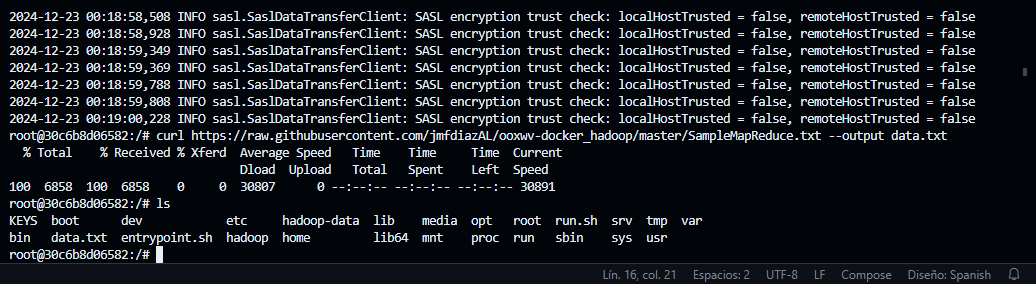
1. Copiamos los archivos xml de configuración en el directorio de entrada:

hdfs dfs -put $HADOOP\_HOME/etc/hadoop/\*.**xml** **/user**/root/input



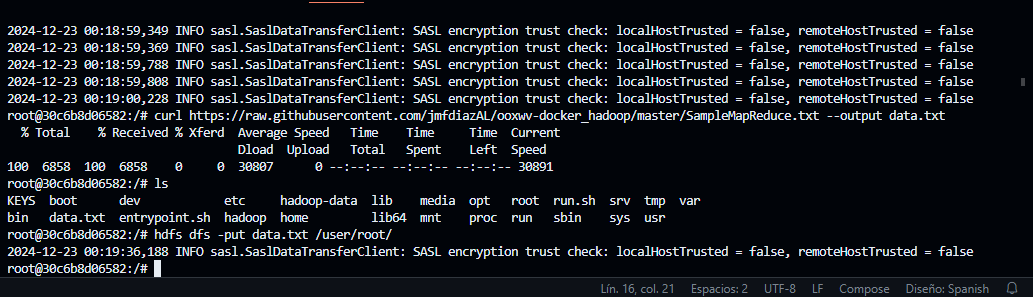
1. Creamos el archivo **data.txt**: en el directorio actual, y lo cargamos con valores de ejemplo:

**curl** [https://raw.githubusercontent.com/jmfdiazAL/ooxwv-docker\_hadoop/master/SampleMapReduce.txt --output data.txt](https://raw.githubusercontent.com/jmfdiazAL/ooxwv-docker_hadoop/master/SampleMapReduce.txt%20--output%20data.txt)



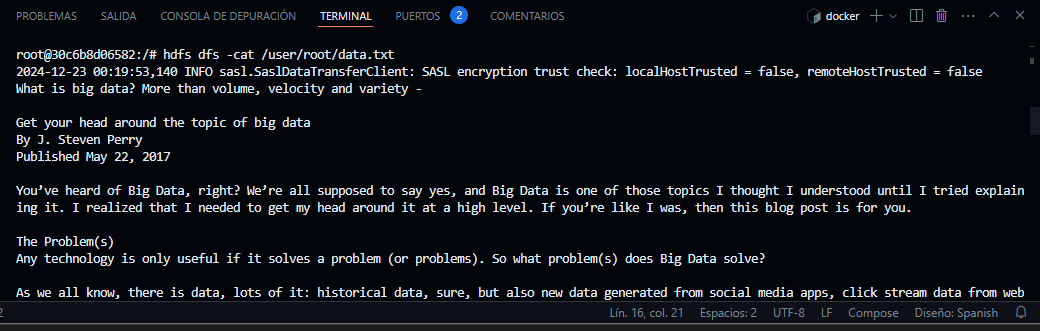
1. Copiamos el archivo **data.txt** en el directorio **user/root**:

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



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

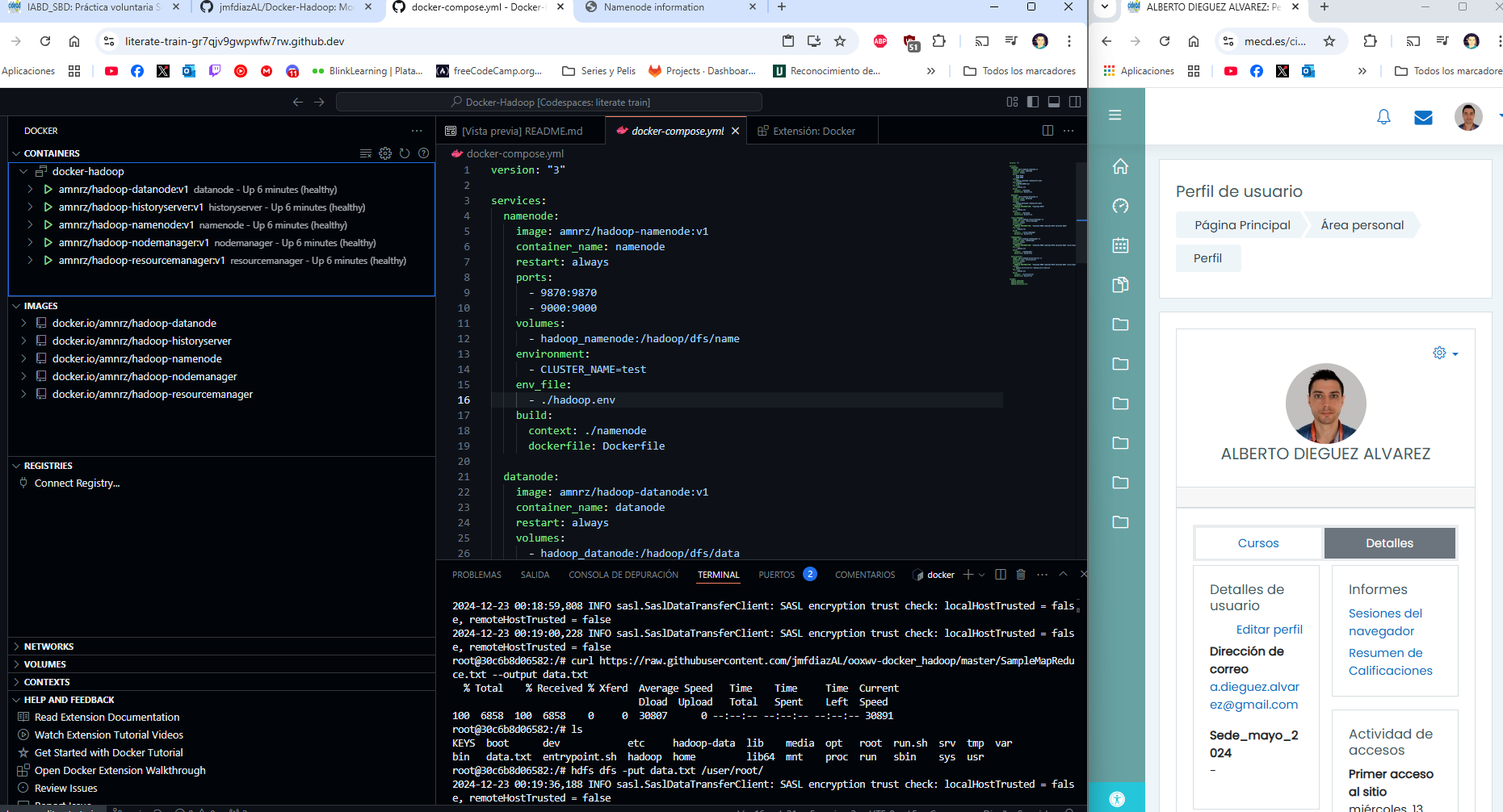
**hdfs** dfs -cat /user/root/**data**.txt

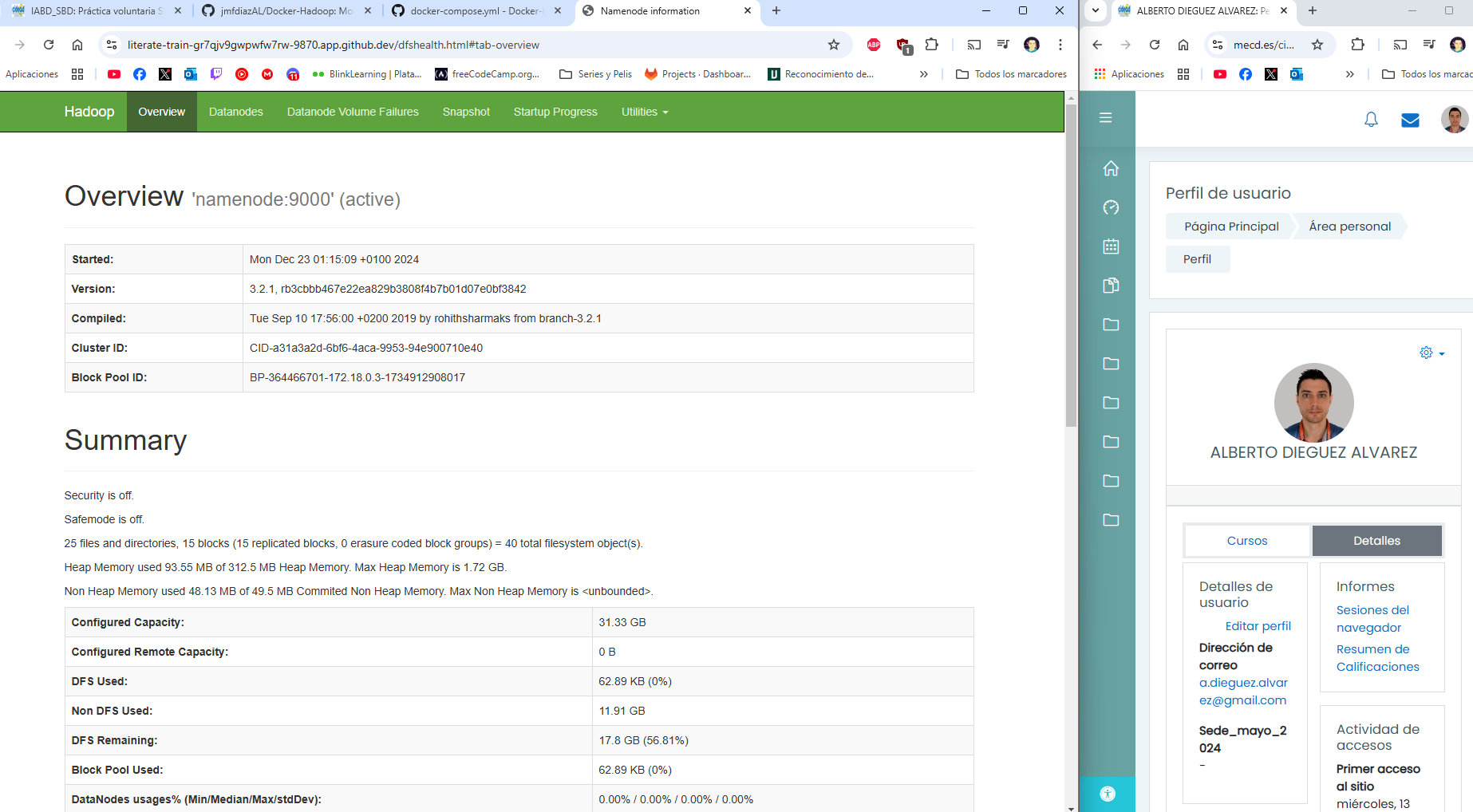


# III. Ver en HDFS

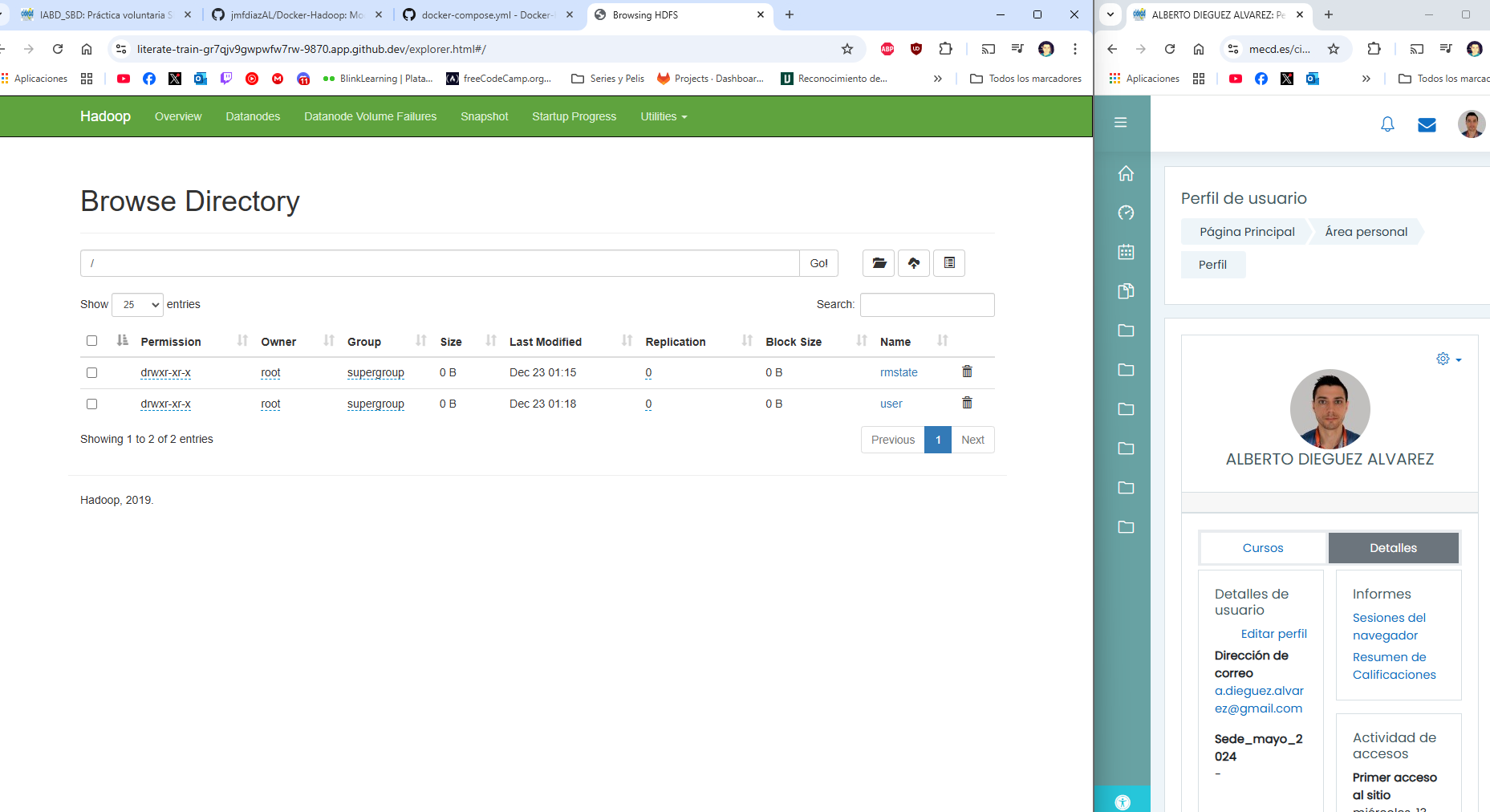
En la tercera parte, vamos a ver los archivos subidos a HDFS en la interfaz gráfica.

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.

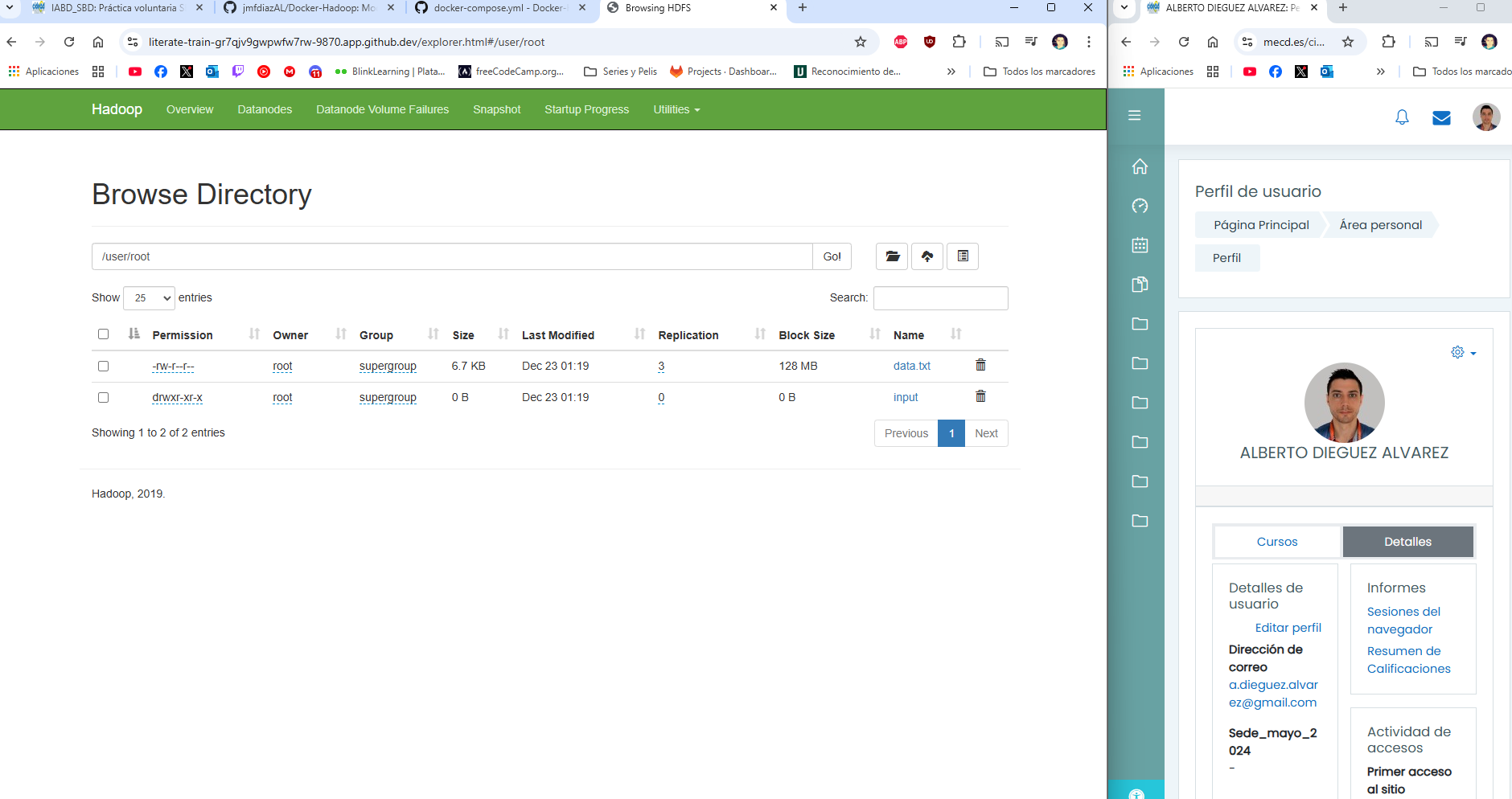




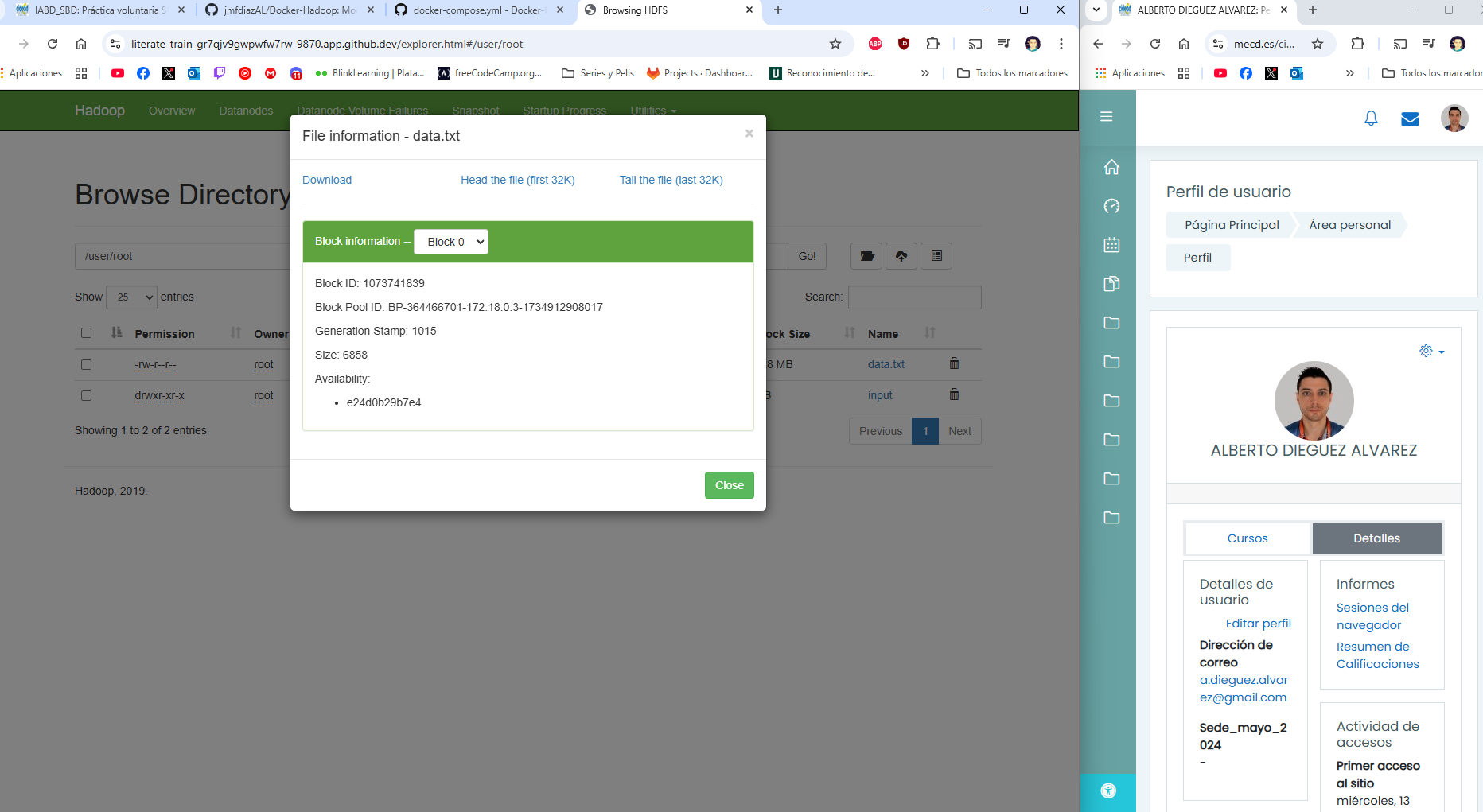
1. A continuación, pulsamos en la opción Utilities y en Browse the file system:



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

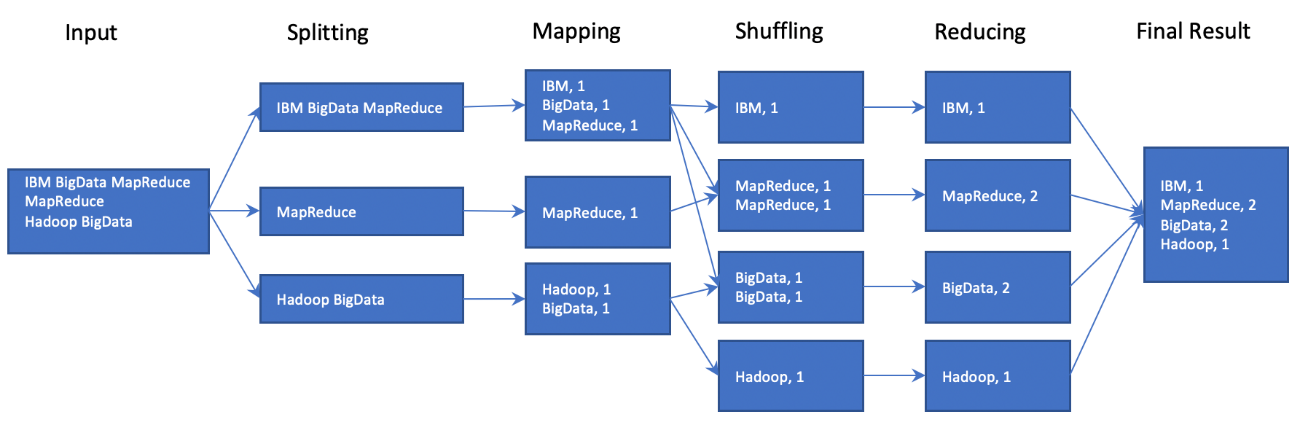


1. Observa que el tamaño del bloque es de 128 MB, aunque el tamaño del archivo es en realidad mucho más pequeño. Esto se debe a que el tamaño de bloque predeterminado utilizado por HDFS es 128 MB.
2. Puedes hacer click en el archivo para revisarlo. Nos da información sobre el archivo en términos de número de bytes, identificación de bloque, etc.



# IV. Ejecutar un ejemplo MapReduce para contar palabras

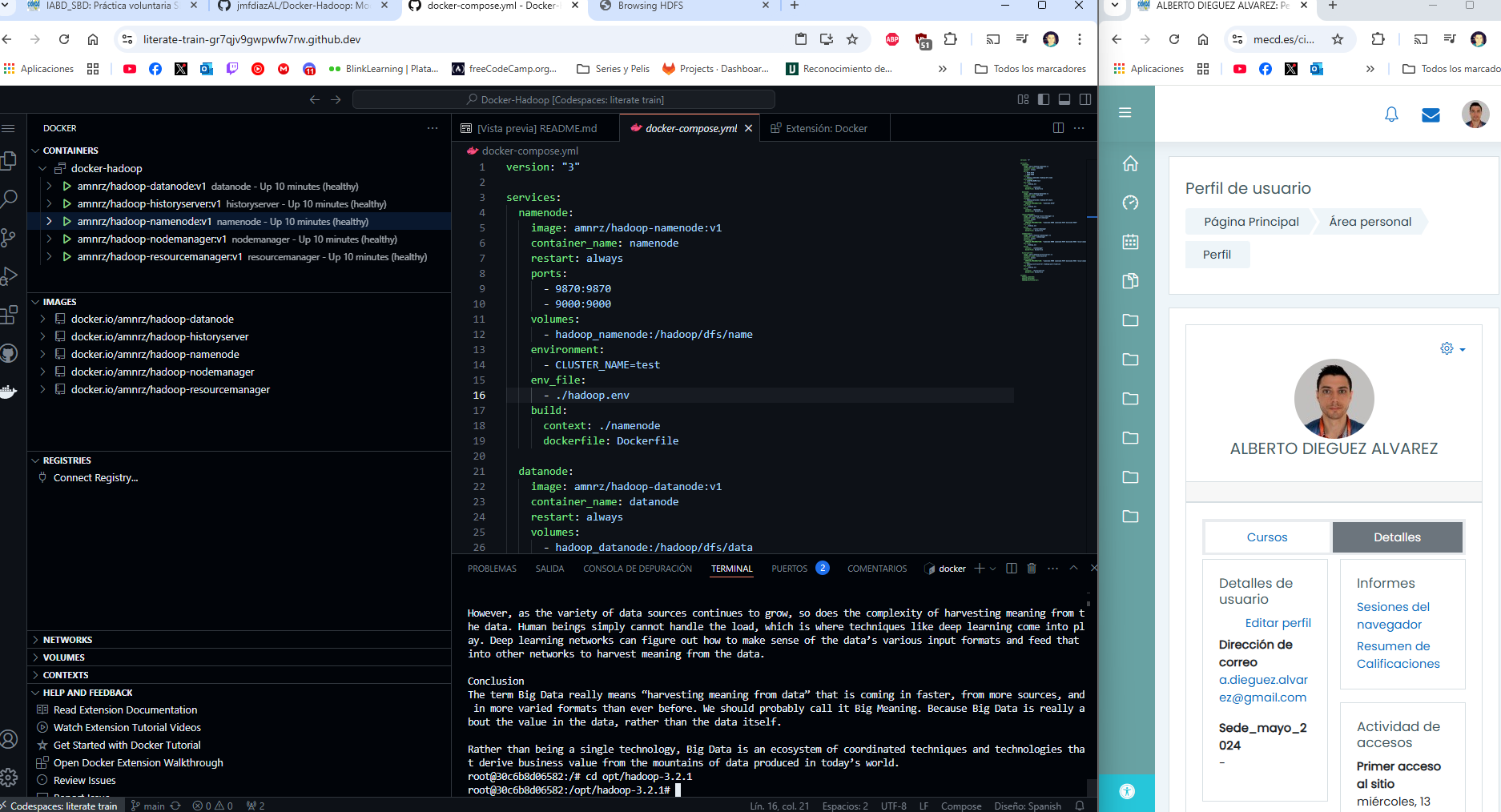
En este apartado vamos a ver una operación sencilla MapReduce sobre un archivo de ejemplo. En la siguiente imagen se puede ver un resumen del procesamiento.



Para ello seguiremos los pasos:

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

**cd** **opt**/hadoop-3.2.1



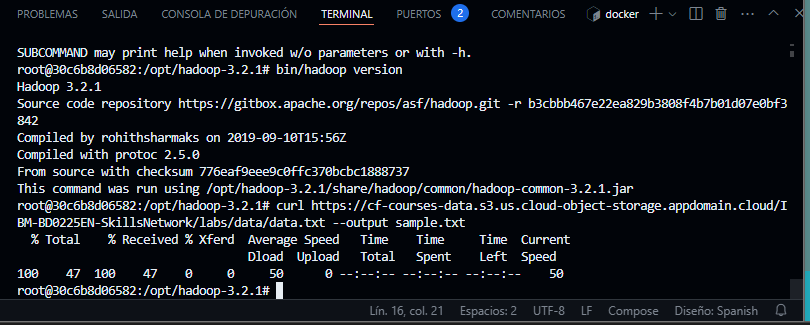
1. Chequeamos la versión de Hadoop:

bin/hadoop versión



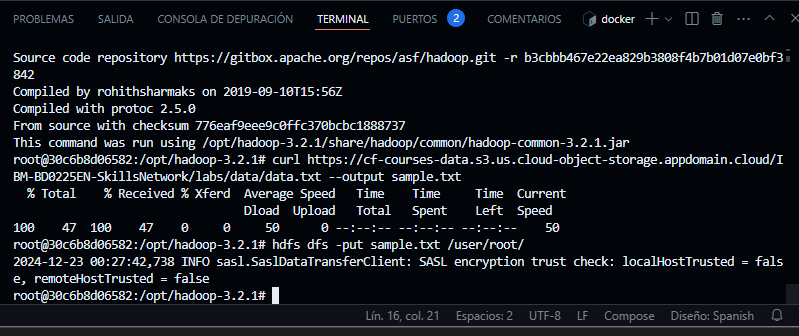
1. Nos bajamos un archivo de prueba **sample.txt**:

**curl** [https://cf-courses-**data**.s3.us.cloud-object-storage.appdomain.cloud/IBM-BD0225EN-SkillsNetwork/labs/**data**/**data**.txt --output sample.txt](https://cf-courses-data.s3.us.cloud-object-storage.appdomain.cloud/IBM-BD0225EN-SkillsNetwork/labs/data/data.txt%20--output%20sample.txt)



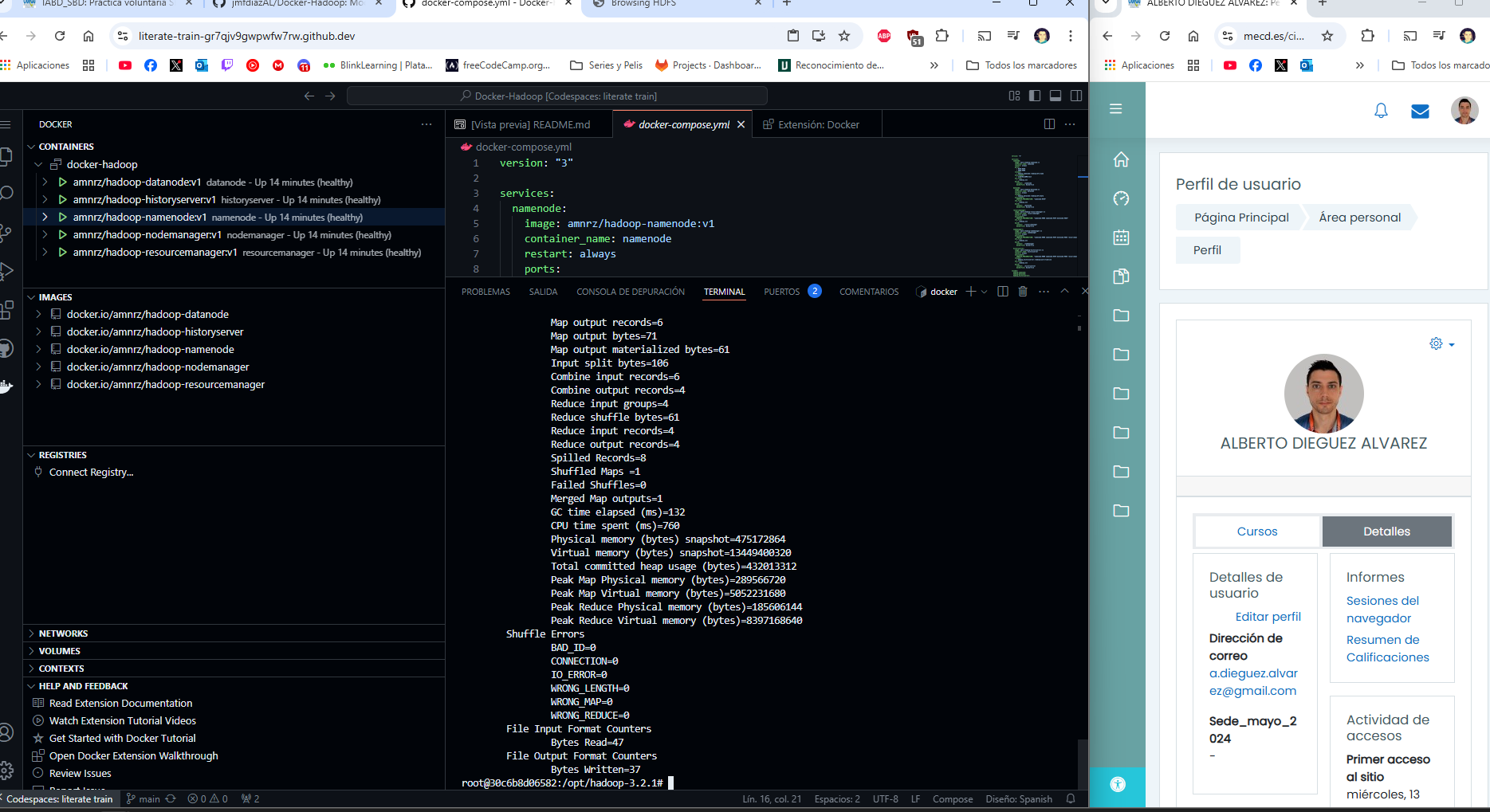
1. Cargamos el archivo **sample.txt** en HDFS:

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



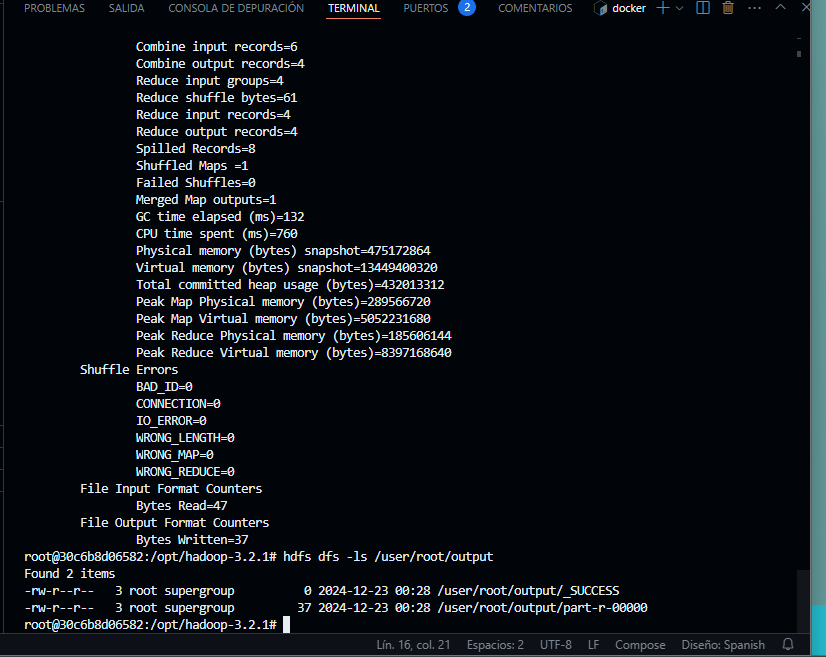
1. Ejecutamos el archivo Java compilado de ejemplo para contar palabras con MapReduce, y copiamos la salida en el directorio **/user/root/output**:

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



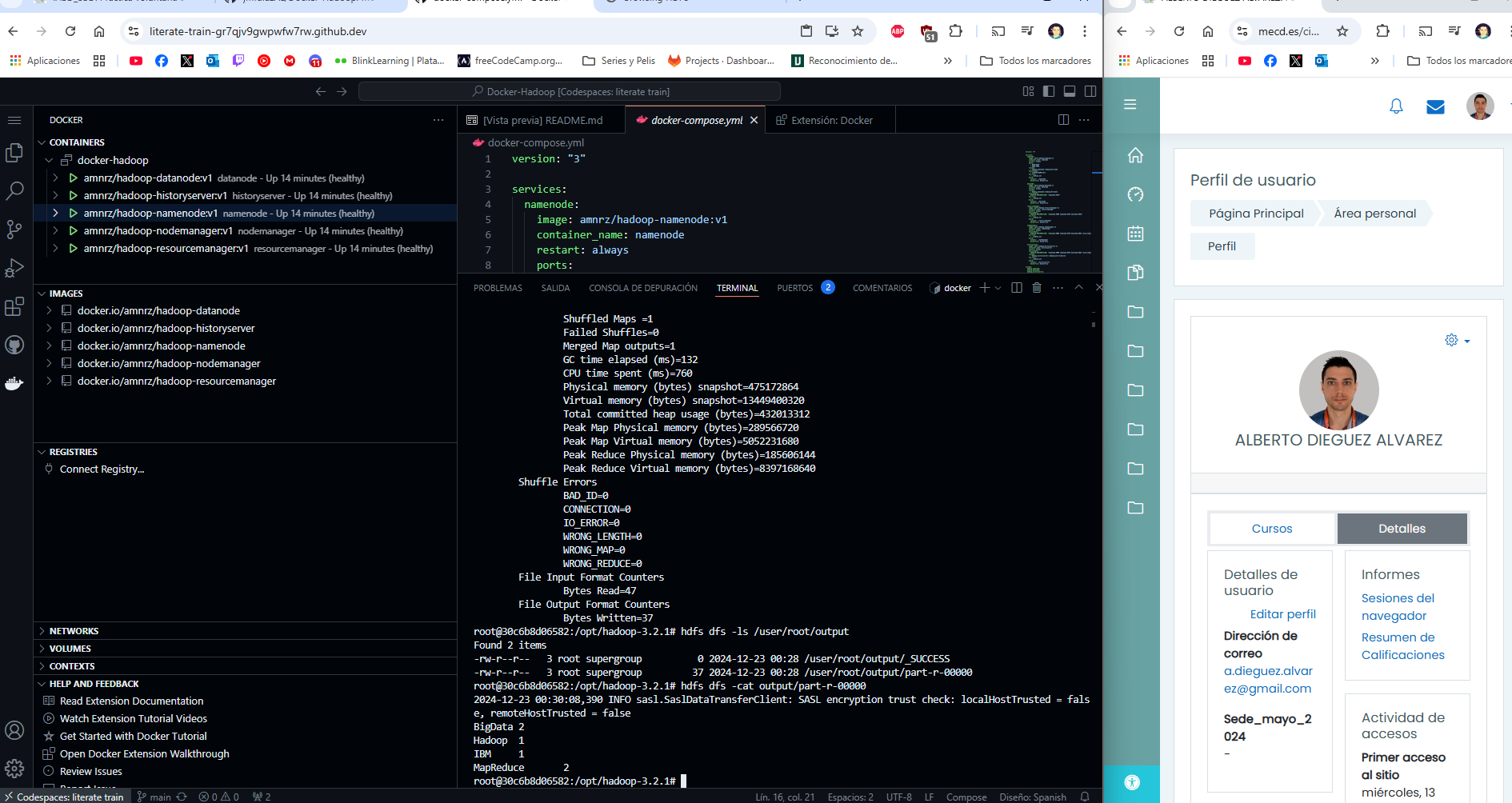
1. Una vez haya terminado la ejecución, vemos los archivos generados en el directorio de salida:

hdfs dfs -ls /**user**/root/output



1. Vemos la salida consultando el archivo **part-r-00000**:

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



1. Nos mostrará una salida parecida a la siguiente:

2024-12-20 20:52:01,009 INFO sasl.SaslDataTransferClient: SASL encryption trust check: localHostTrusted = false, remoteHostTrusted = false

BigData 2

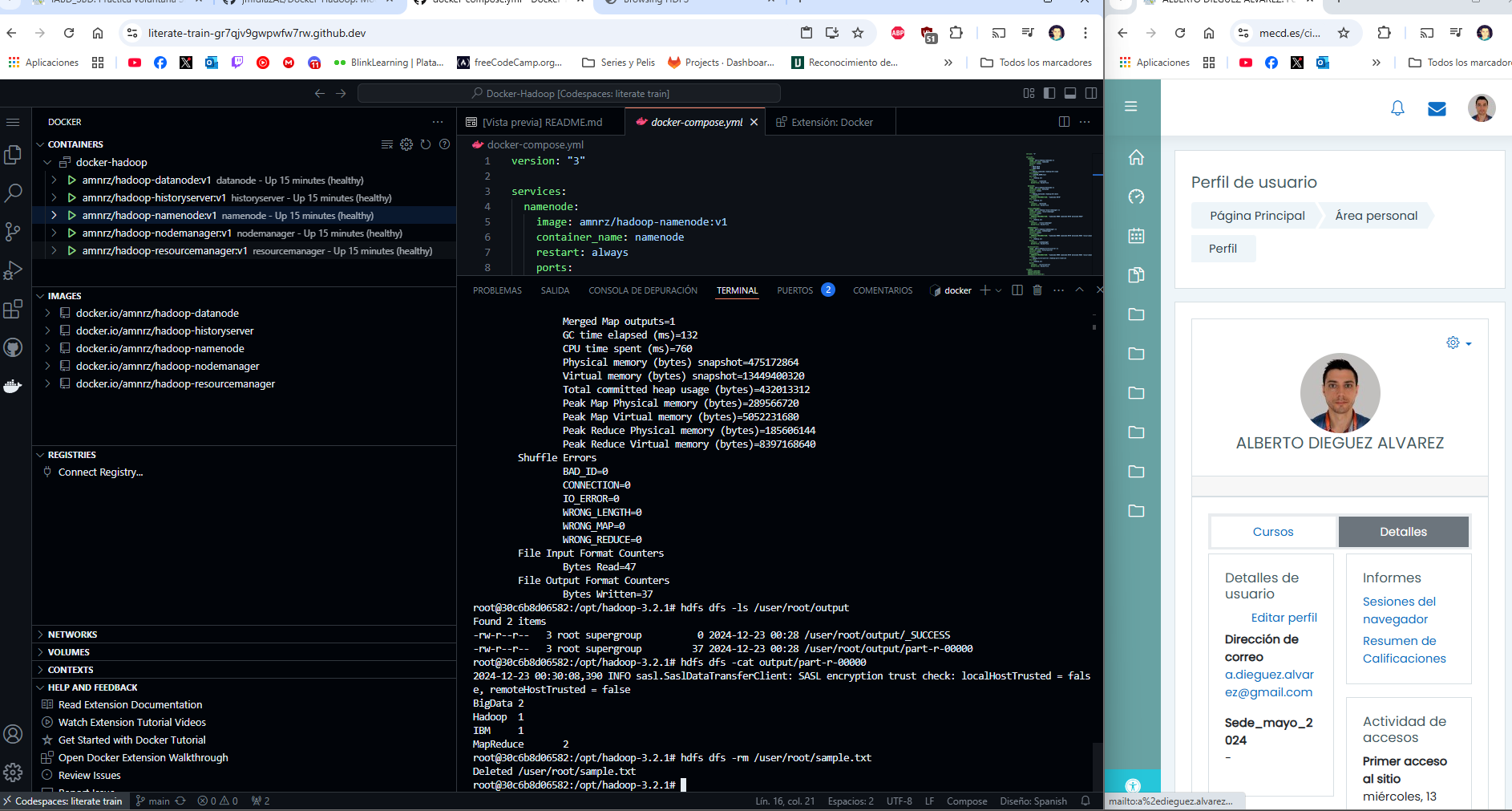
Hadoop 1

IBM 1

MapReduce 2

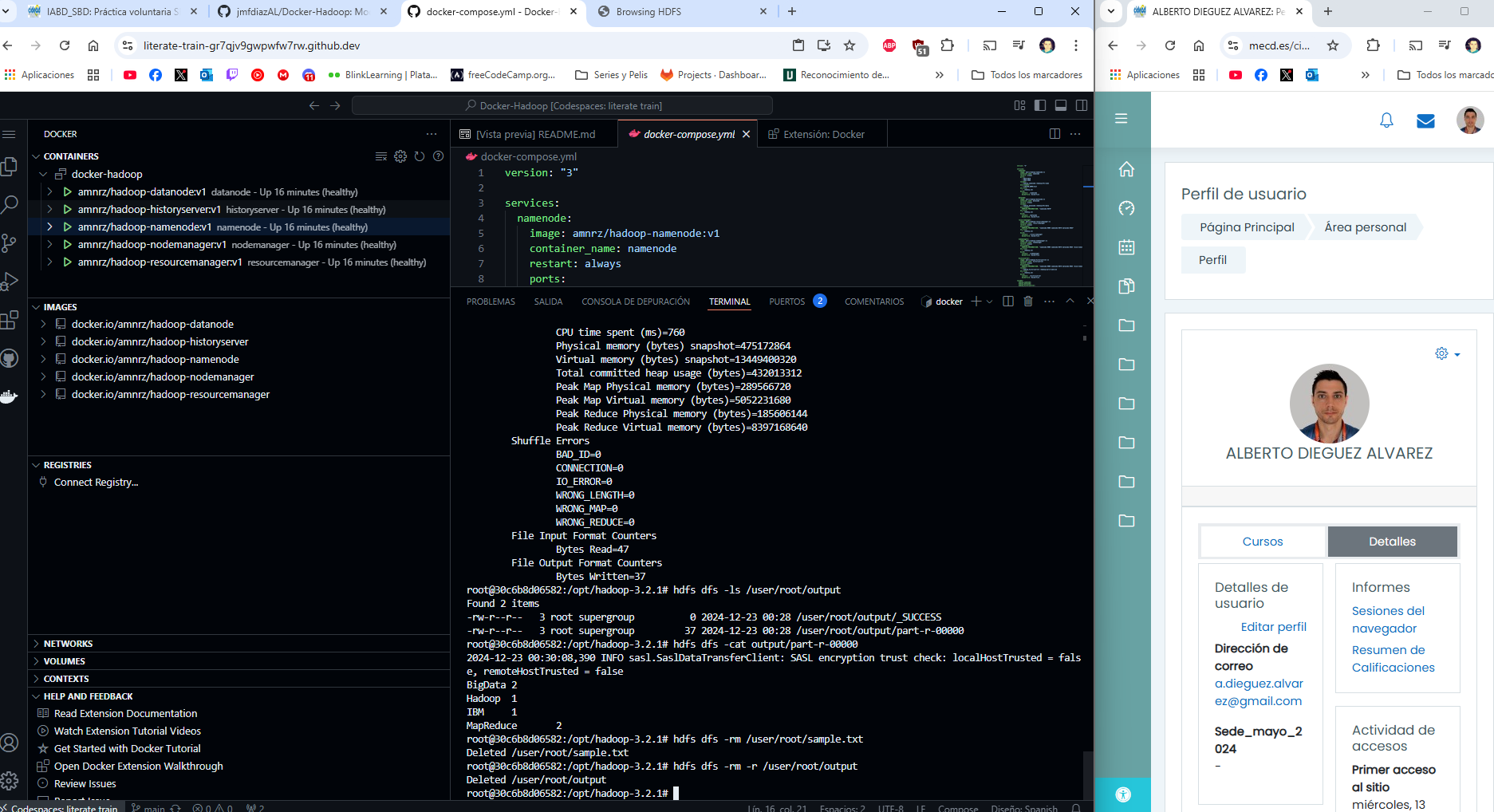
1. Para finalizar, podemos eliminar el archivo **sample.txt**:

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



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

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

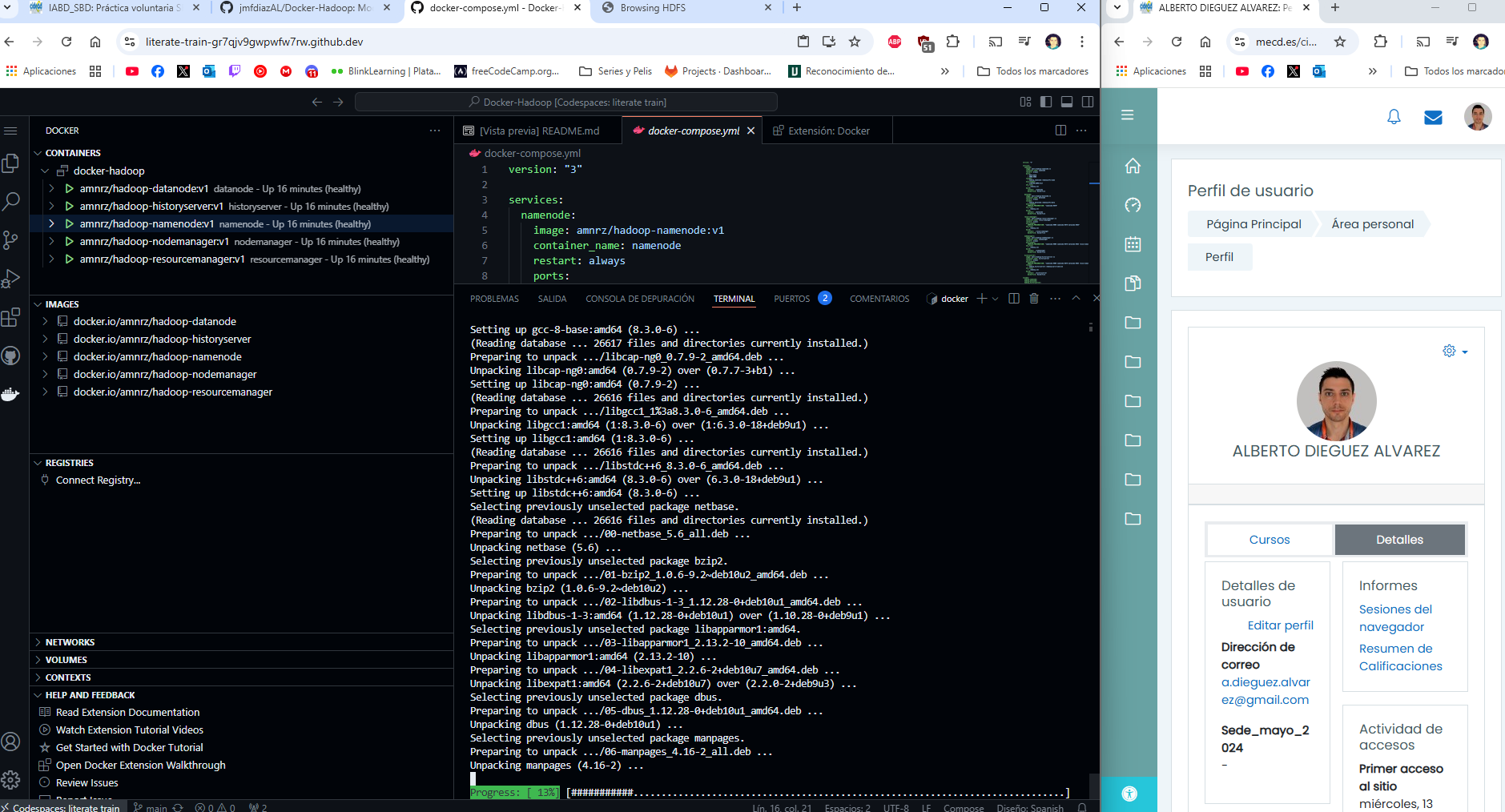


# V. Acceso mediante Python

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

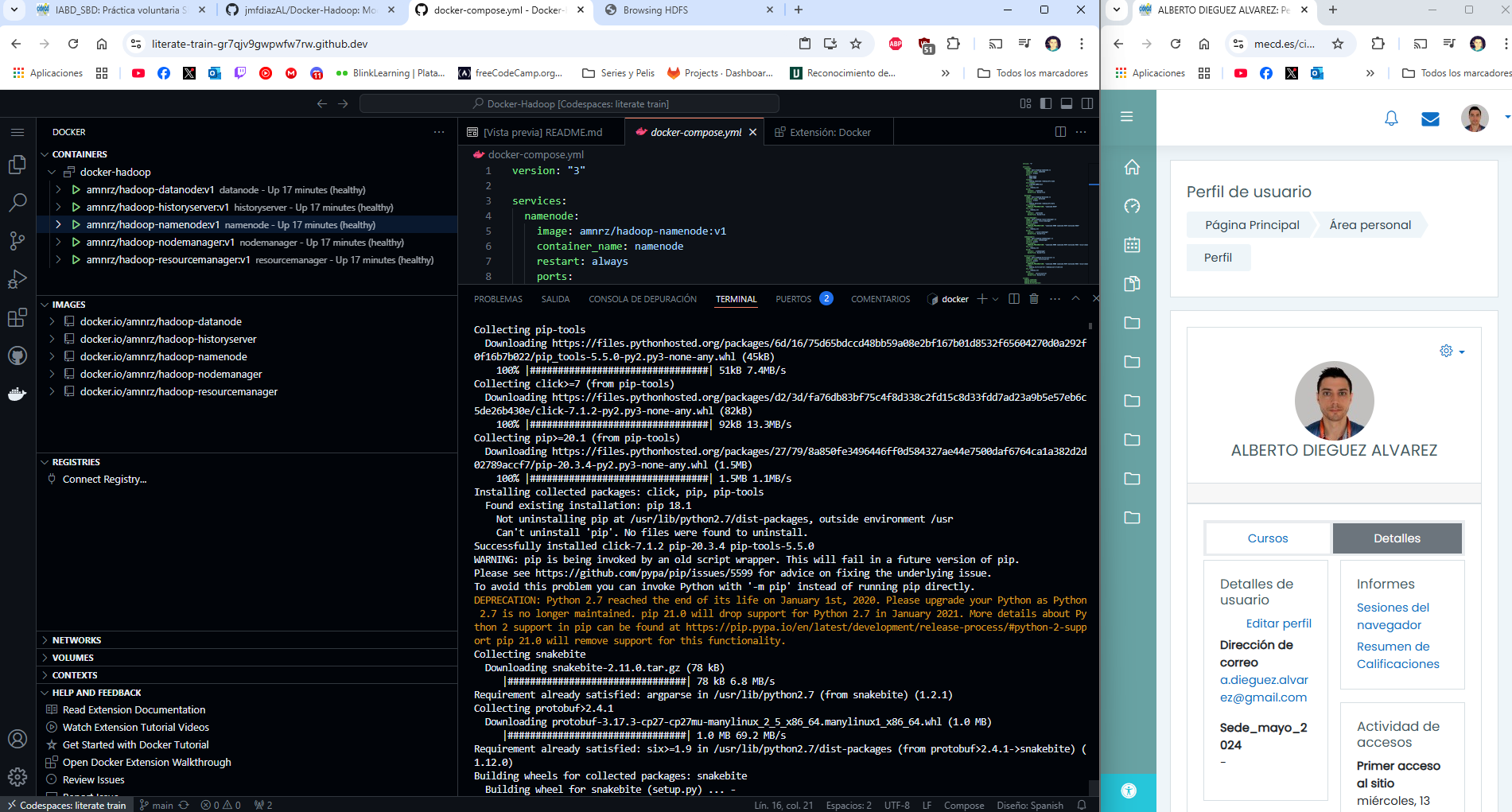
1. En la consola del contenedor **namenode** que hemos venido utilizando hasta ahora, escribiremos:

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



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

pip **install** pip-tools && pip **install** snakebite



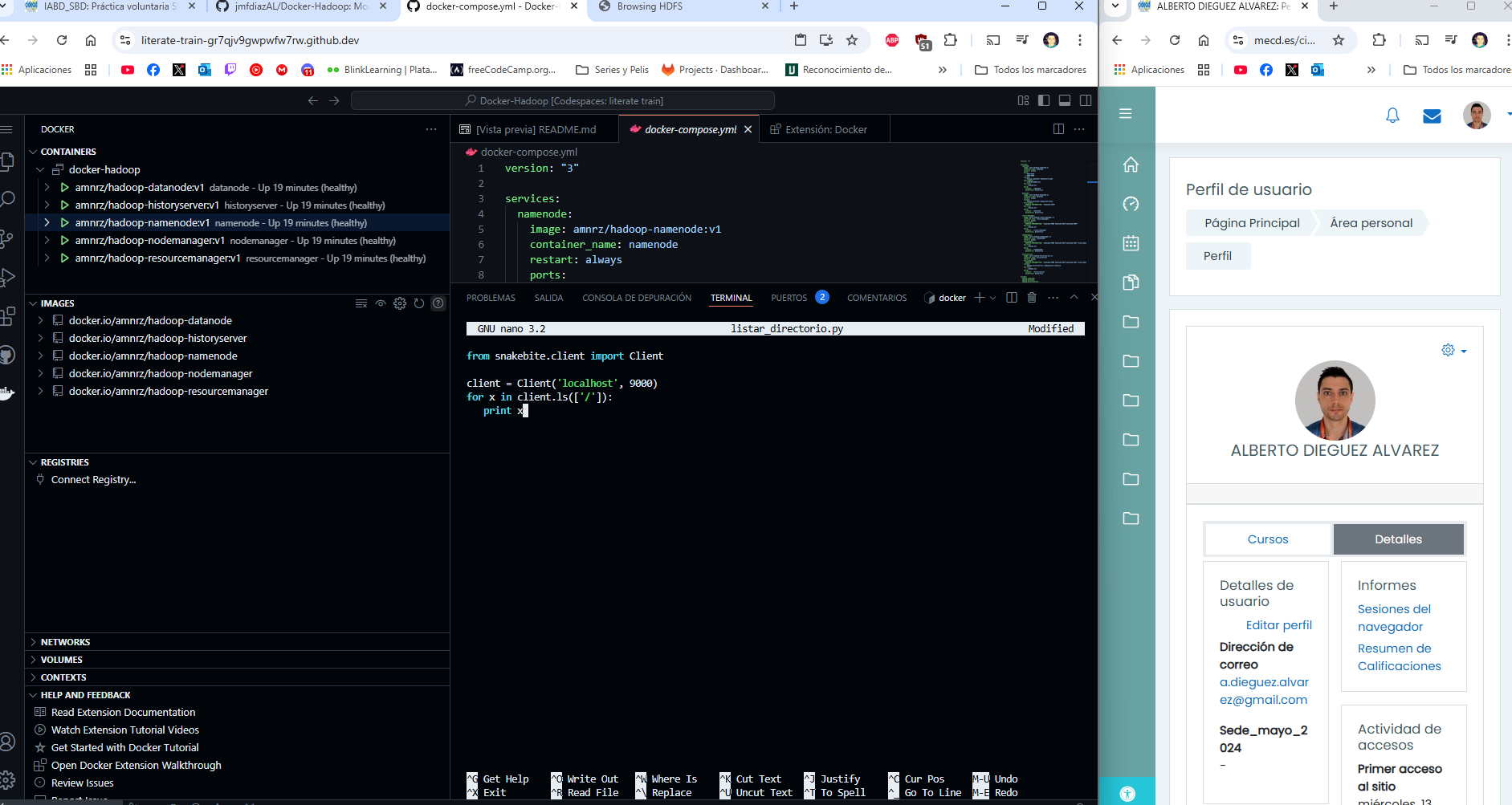
1. Nos movemos al directorio **/root** por ejemplo y creamos el fichero **listar\_directorio.py** con el siguiente contenido:

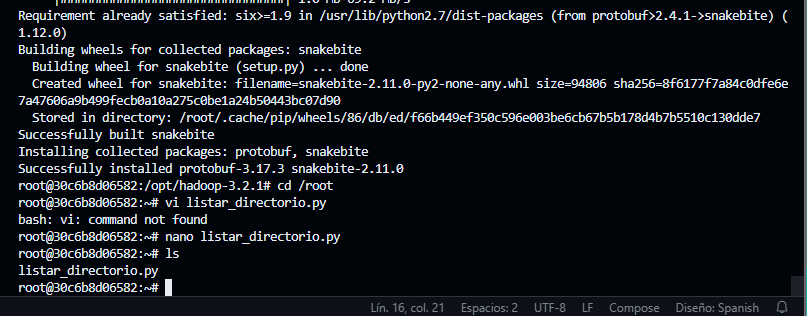
**from** snakebite.client **import** Client

client = Client('localhost', 9000)

**for** x **in** client.ls(['/']):

**print** x



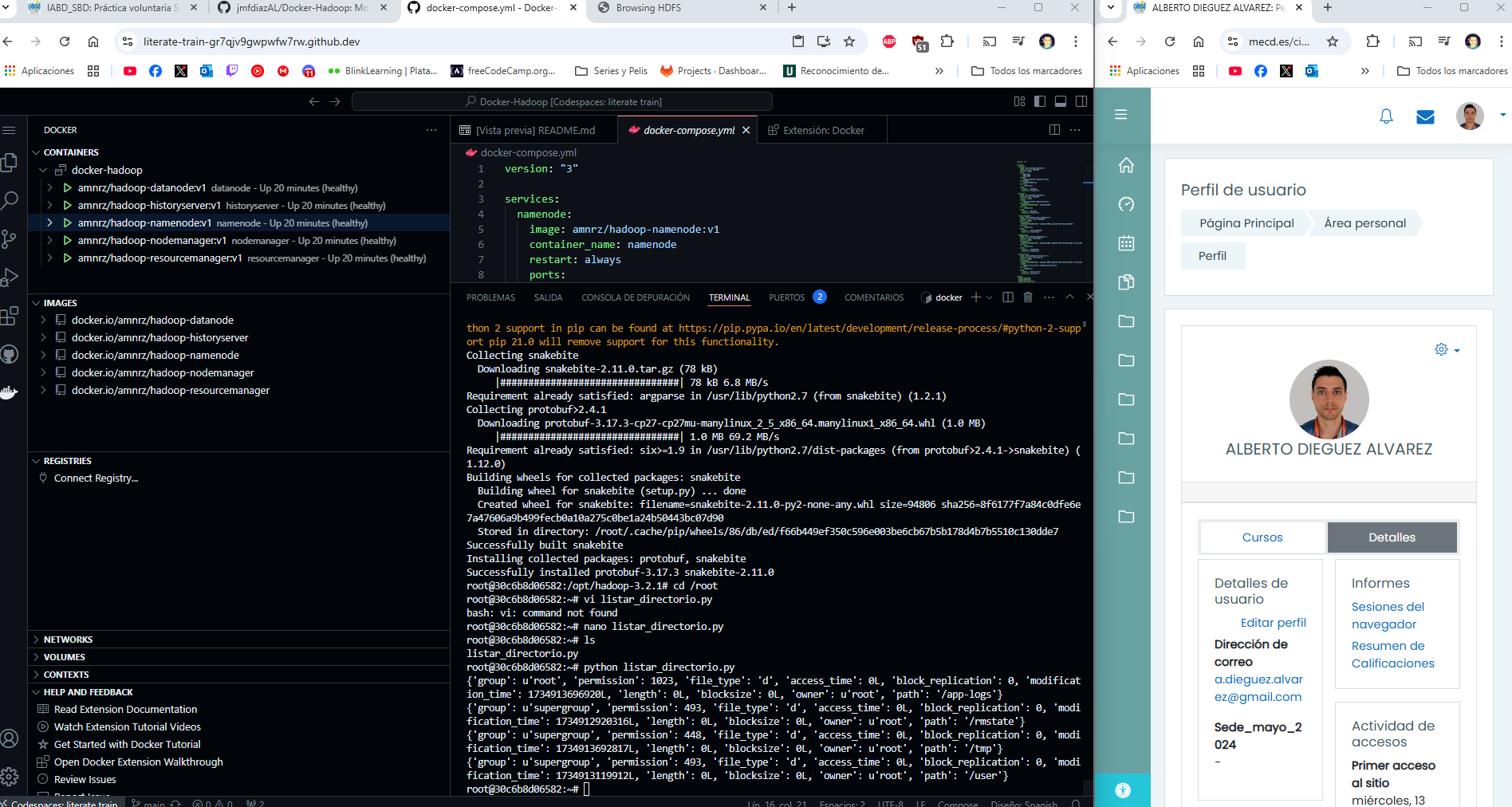


1. Si ejecutamos el archivo anterior, podremos ver la siguiente salida:

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'}



1. 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.