**Máster Universitario en Big Data y Ciencia de Datos**

**ASIGNATURA: 03MBID *Procesamiento de datos masivos***

*Instalación Spark en Docker (Linux)*

Alumno: **Bru Montes, Israel**

Edición **Octubre 2024 – Grupo A** a 24/12/2024

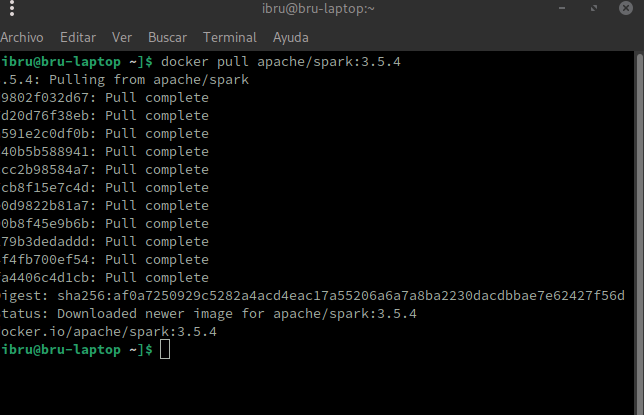
# Introducción.

En el siguiente documento vamos a documentar como hemos instalado spark y docker dentro de un Sistema Operativo Linux (Manjaro).

Vamos a partir de la instalación y configuración del servidor que ya tenemos creado de hadoop.

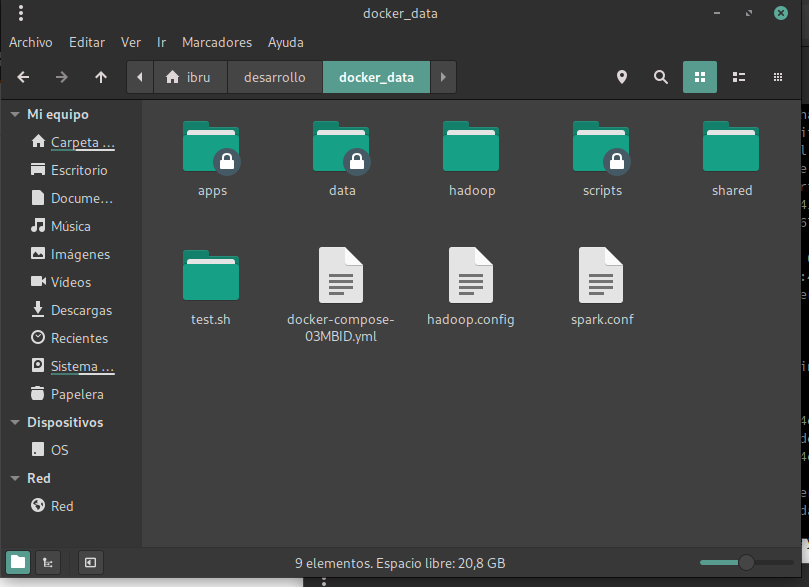
Nos bajamos la última versión que existe en <https://hub.docker.com/> de la imagen que ofrece apache:

[ibru@bru-laptop ~]$ docker pull apache/spark:3.5.4



[ibru@bru-laptop desarrollo]$ mkdir docker\_data

[ibru@bru-laptop desarrollo]$ mkdir docker\_data\shared



Definimos la configuración tanto del docker (docker-compose-03MBID.yml) como su configuración por defecto. Detallamos aquí el archivo, aunque lo adjunto en la documentación:

spark.config:

*spark.master=spark://spark-master:7077*

*spark.eventLog.enabled=true*

*spark.eventLog.dir=/opt/spark/logs*

*spark.history.fs.logDirectory=/opt/spark/logs*

*spark.hadoop.fs.defaultFS=hdfs://namenode:8020*

*docker-compose-03MBID.yml:*

services:

namenode:

image: apache/hadoop:3.4.1

hostname: hadoop

container\_name: hadoop

user: root

command: ["hdfs", "namenode"]

ports:

- 9870:9870

environment:

ENSURE\_NAMENODE\_DIR: "/tmp/hadoop-root/dfs/name"

volumes:

- ./shared:/opt/hadoop/input\_files

env\_file:

- ./hadoop.config

networks:

- hdfs\_net

datanode\_1:

image: apache/hadoop:3.4.1

command: ["hdfs", "datanode"]

env\_file:

- ./hadoop.config

networks:

- hdfs\_net

datanode\_2:

image: apache/hadoop:3.4.1

command: ["hdfs", "datanode"]

env\_file:

- ./hadoop.config

networks:

- hdfs\_net

resourcemanager:

image: apache/hadoop:3.4.1

command: ["yarn", "resourcemanager"]

ports:

- 8088:8088

env\_file:

- ./hadoop.config

networks:

- hdfs\_net

nodemanager:

image: apache/hadoop:3.4.1

command: ["yarn", "nodemanager"]

environment:

ENSURE\_NODEMANAGER\_DIR: "tmp/hadoop-hadoop/nm-local-dir/"

networks:

- hdfs\_net

env\_file:

- ./hadoop.config

spark-master:

image: apache/spark:3.5.4

command: /opt/spark/bin/spark-class org.apache.spark.deploy.master.Master

hostname: spark

container\_name: spark

ports:

- 8080:8080

- 7077:7077

environment:

SPARK\_HOME: /opt/spark

HADOOP\_USER\_NAME: root

volumes:

- ./spark.conf:/opt/spark/conf/spark-defaults.conf

- ./shared:/opt/spark/work-dir

networks:

- hdfs\_net

spark-worker-1:

image: apache/spark:3.5.4

command: /opt/spark/bin/spark-class org.apache.spark.deploy.worker.Worker spark://spark-master:7077

depends\_on:

- spark-master

environment:

SPARK\_MODE: worker

SPARK\_WORKER\_CORES: 1

SPARK\_WORKER\_MEMORY: 1g

SPARK\_MASTER\_URL: spark://spark-master:7077

SPARK\_HOME: /opt/spark

HADOOP\_CONF\_DIR: hdfs://resourcemanager:8088

volumes:

- ./spark.conf:/opt/spark/conf/spark-defaults.conf

networks:

- hdfs\_net

spark-worker-2:

image: apache/spark:3.5.4

command: /opt/spark/bin/spark-class org.apache.spark.deploy.worker.Worker spark://spark-master:7077

depends\_on:

- spark-master

environment:

SPARK\_MODE: worker

SPARK\_WORKER\_CORES: 1

SPARK\_WORKER\_MEMORY: 1g

SPARK\_MASTER\_URL: spark://spark-master:7077

SPARK\_HOME: /opt/spark

volumes:

- ./spark.conf:/opt/spark/conf/spark-defaults.conf

networks:

- hdfs\_net

volumes:

datanode1:

datanode2:

namenode:

spark-logs:

networks:

hdfs\_net:

# Specify driver options

driver: bridge

driver\_opts:

com.docker.network.bridge.host\_binding\_ipv4: "127.0.0.1"

driver: bridge

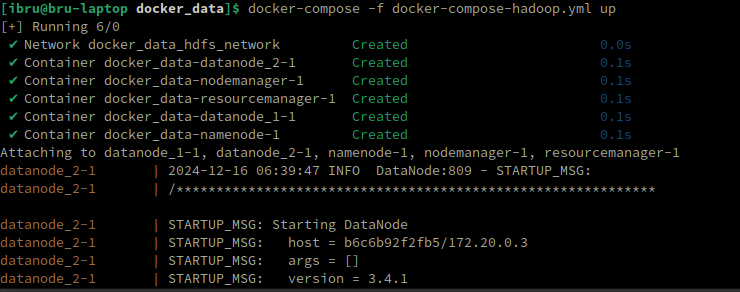
driver\_opts:

*com.docker.network.bridge.host\_binding\_ipv4: "127.0.0.1"*

Nos movemos hasta la carpeta que hemos creado para alojar tanto los archivos de configuración como la carpeta “shared” para hacer el intercambio de archivos entre el contenedor principal y nuestro sistema operativo. A continuación iniciamos docker con el siguiente comando:

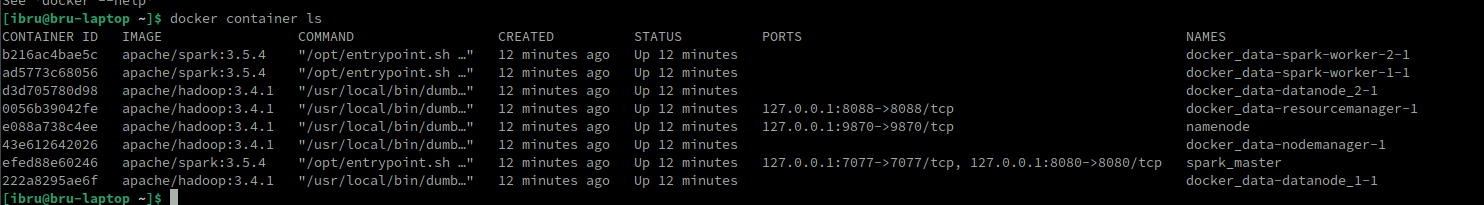
[ibru@bru-laptop ~]$ docker-compose -f docker-compose-03MBID.yml up

Al arrancar nos muestra la siguiente información:



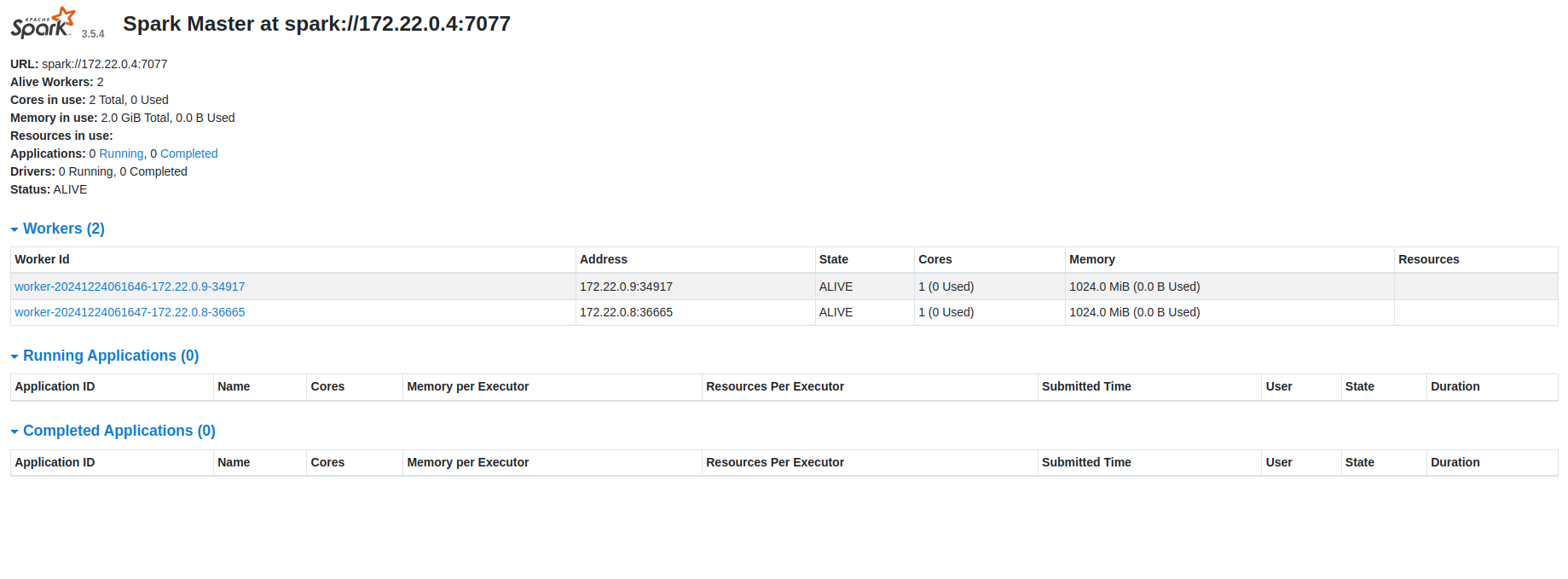
Abrimos un terminal nuevo, siempre y cuando tengamos bloqueada la terminal actual y lanzamos el siguiente comando para ver los contenedores y el id que tienen asignado.

bru@bru-laptop ~]$ docker container ls

Y con el siguiente comando nos permite conectarnos dentro del contenedor que tiene

[ibru@bru-laptop desarrollo]$ docker exec -it 606b7d5e29bc bash

También podemos comprobar la información del sistema de hadoop accediendo a la siguiente url: http://localhost:8080



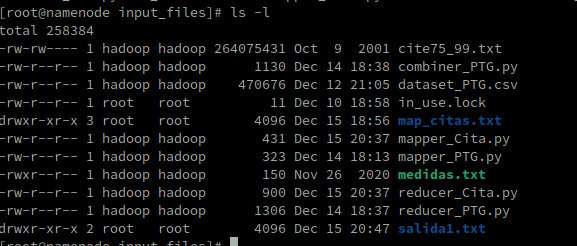
Creamos en el sistema hdfs, tanto la carpeta user como la carpeta root, y subimos un archivo para poder probar que funciona todo el sistema.

Para ello nos dirigimos al “namenode” de hadoop:

[ibru@bru-laptop ~]$ docker exec -it namenode bash

[root@namenode input\_files]# hadoop fs -mkdir /user

[root@namenode input\_files]# hadoop fs -mkdir /user/root



Nos movemos a la carpeta que tenemos compartida desde nuestro sistema para subir a hdfs el archivo del quijote:

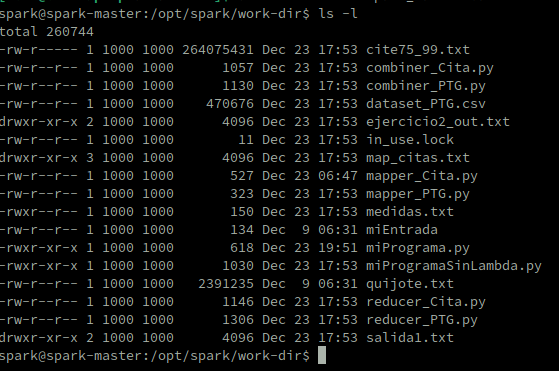
[root@namenode hadoop]# cd input\_files/

Subimos el archivo a hdfs para poder ejecutar con posterioridad.

Abrimos un nuevo terminal, y nos conectamos a la máquina de spark:

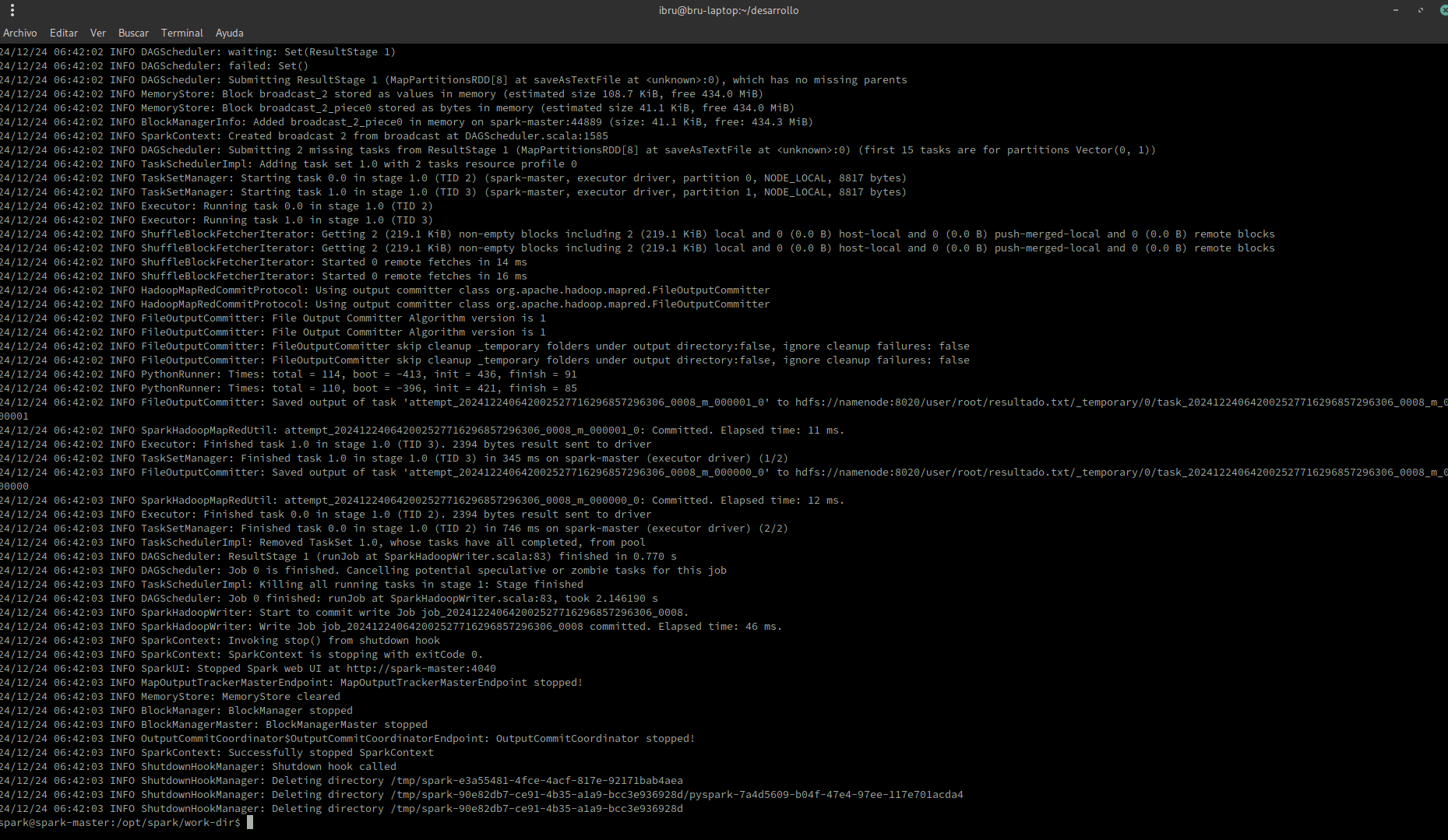
[ibru@bru-laptop desarrollo]$ docker exec -it spark-master bash

Comprobamos que tenemos el contenido de la carpeta “shared” visible en la carpeta de trabajo por defecto:



Como anteriormente hemos subido el fichero que queremos utilizar a hdfs, vamos a proceder a procesarlo:

spark@spark-master:/opt/spark/work-dir$ ../bin/spark-submit miPrograma.py quijote.txt resultado.txt



A continuación podemos descargarnos de hdfs el resultado del procesamiento de los datos que existen en “quijote.txt” y la salida resultado.txt

[root@namenode input\_files]# hdfs dfs -get resultado.txt .

[root@namenode input\_files]# cat resultado.txt/part-00000 |more

('The', 848)

('Project', 80)

('of', 12868)

('Don', 2541)

('Quixote,', 626)

('Miguel', 9)

('', 7961)

('is', 3505)

('use', 64)

('anyone', 82)

('anywhere', 10)

('in', 6864)

('United', 15)

('States', 8)

('other', 528)

('world', 184)

('at', 1928)

('no', 1227)

('restrictions', 4)

('whatsoever.', 2)

('may', 773)

('it,', 523)

('give', 546)

('away', 181)

('re-use', 2)

('this', 2291)

('online', 4)

('are', 1278)

('have', 2534)

('check', 10)