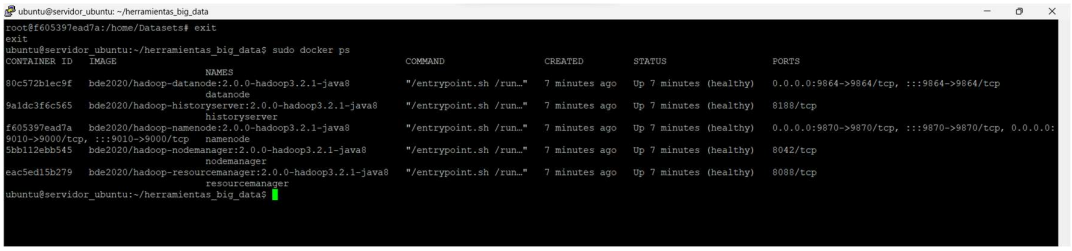
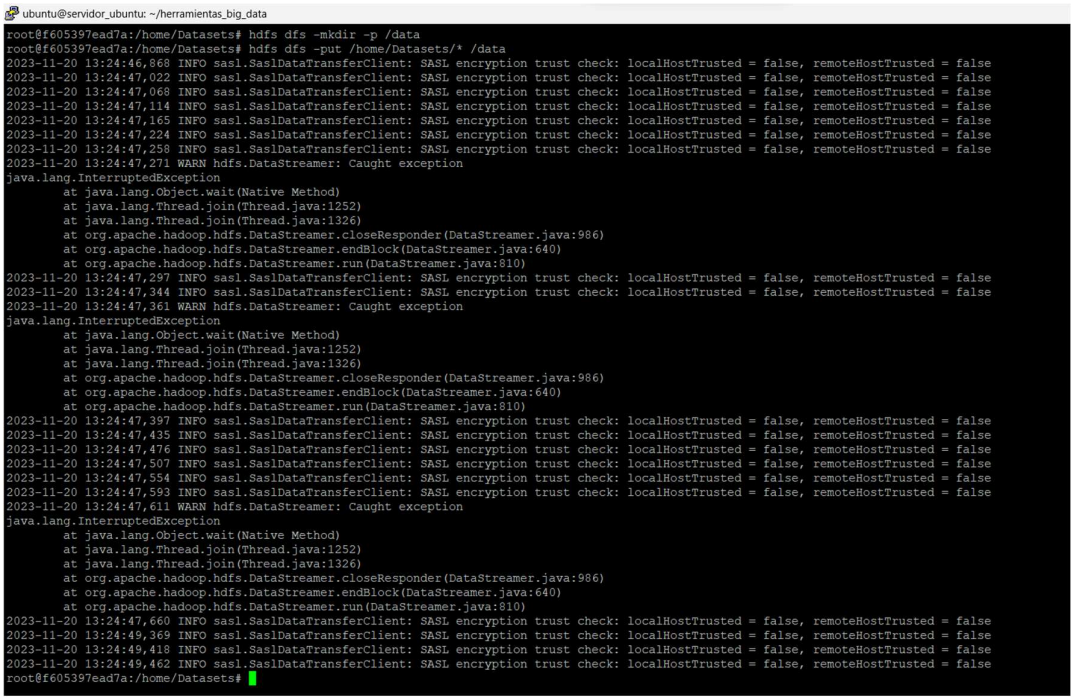
Alumno: Alter Caimi – DATA-PT06

Parte 1

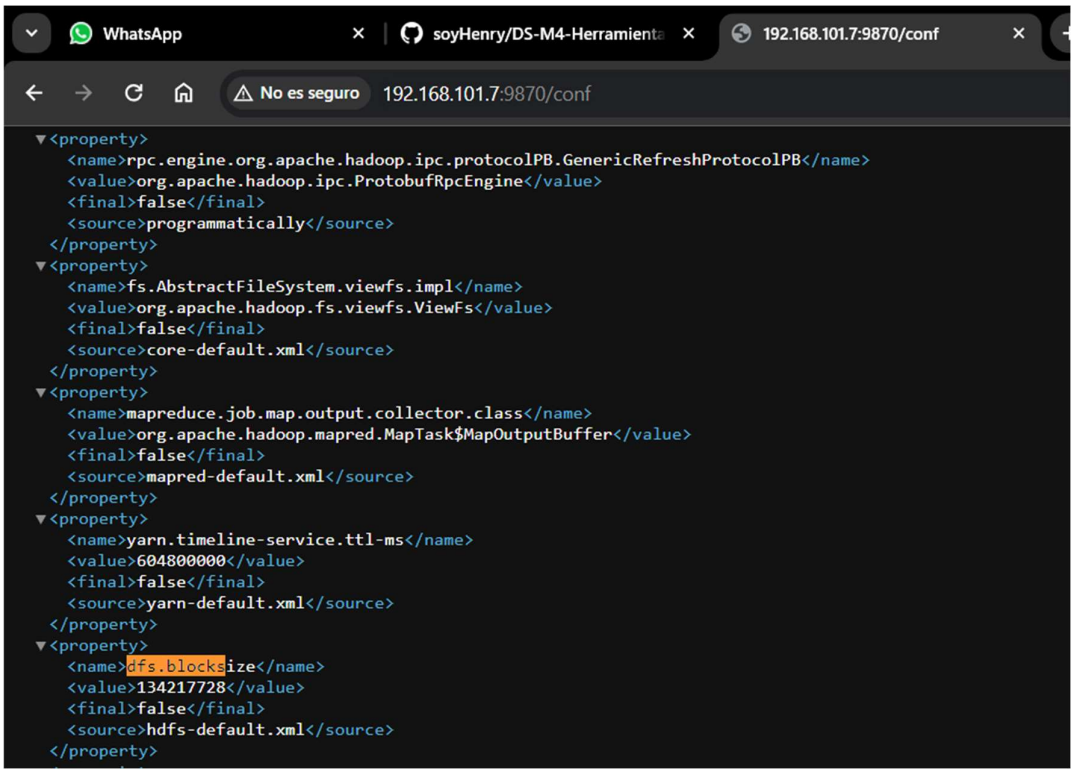


Archivos cargados en el HDFS de Hadoop

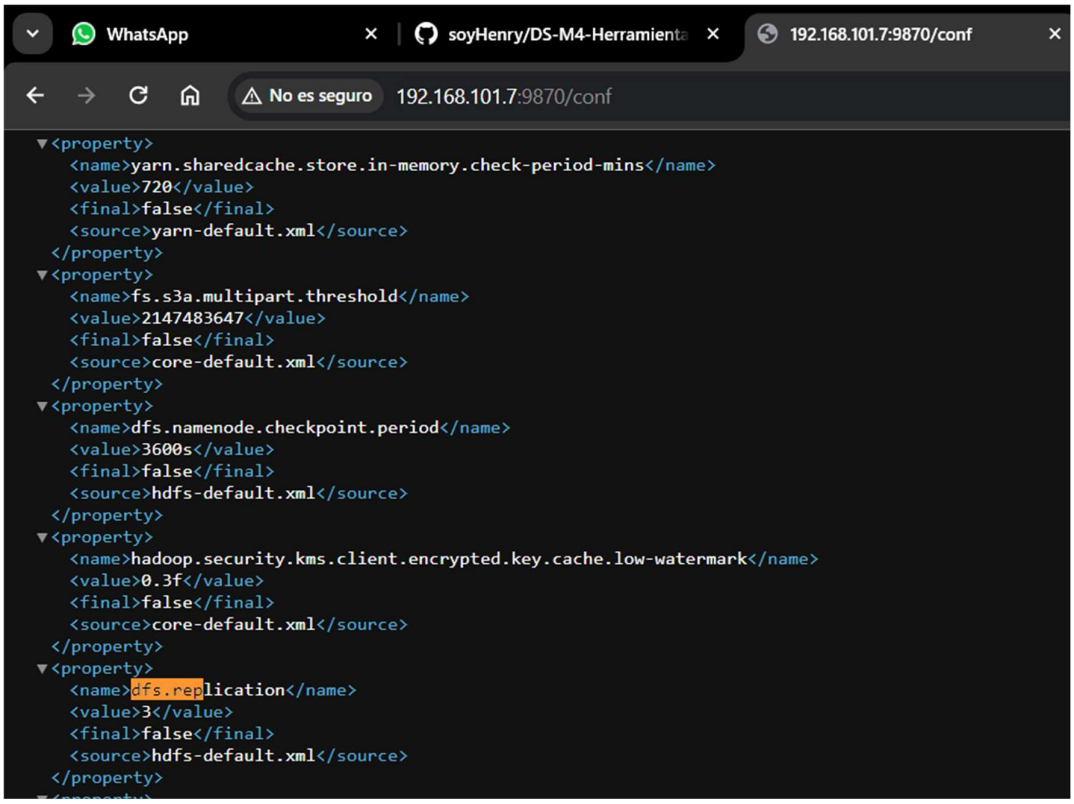


```
ubuntu@servidor_ubuntu: ~/herramientas_big_data
ubuntu@servidor_ubuntu:~/herramientas_big_data$ sudo docker exec -it namenode bash
root@f605397ead7a:/# hdfs dfs -ls /data
Found 17 items
-rw-r--r--   3 root supergroup      16308 2023-11-20 13:24 /data/airports.csv
drwxr-xr-x   - root supergroup          0 2023-11-20 13:24 /data/calendario
drwxr-xr-x   - root supergroup          0 2023-11-20 13:24 /data/canaldeventa
drwxr-xr-x   - root supergroup          0 2023-11-20 13:24 /data/cliente
drwxr-xr-x   - root supergroup          0 2023-11-20 13:24 /data/compra
drwxr-xr-x   - root supergroup          0 2023-11-20 13:24 /data/data_nvo
drwxr-xr-x   - root supergroup          0 2023-11-20 13:24 /data/empleado
drwxr-xr-x   - root supergroup          0 2023-11-20 13:24 /data/gasto
-rw-r--r--   3 root supergroup       4813 2023-11-20 13:24 /data/iris.csv
-rw-r--r--   3 root supergroup      15802 2023-11-20 13:24 /data/iris.json
-rw-r--r--   3 root supergroup         94 2023-11-20 13:24 /data/personal.csv
drwxr-xr-x   - root supergroup          0 2023-11-20 13:24 /data/producto
drwxr-xr-x   - root supergroup          0 2023-11-20 13:24 /data/proveedor
-rw-r--r--   3 root supergroup   69772435 2023-11-20 13:24 /data/raw-flight-data.csv
drwxr-xr-x   - root supergroup          0 2023-11-20 13:24 /data/sucursal
drwxr-xr-x   - root supergroup          0 2023-11-20 13:24 /data/tipodegasto
drwxr-xr-x   - root supergroup          0 2023-11-20 13:24 /data/venta
root@f605397ead7a:/#
```

Tamaño de bloque:

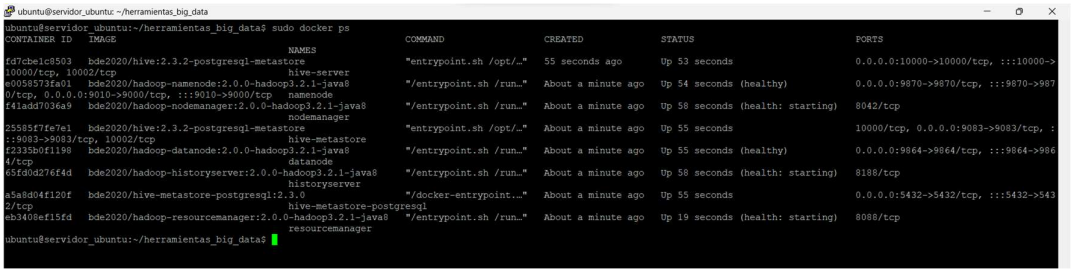


Factor de réplica



Parte 2

Containers creados:



Ejecutando script de creación de tablas:

```
ubuntu@servidor_ubuntu: ~/herramientas_big_data
root@fd7cbe1c8503:/opt# hive -f Paso02.hql
SLF4J: Class path contains multiple SLF4J bindings.
SLF4J: Found binding in [jar:file:/opt/hive/lib/log4j-slf4j-impl-2.6.2.jar!/org/slf4j/impl/StaticLoggerBinder.class]
SLF4J: Found binding in [jar:file:/opt/hadoop-2.7.4/share/hadoop/common/lib/slf4j-log4j12-1.7.10.jar!/org/slf4j/impl/StaticLoggerBinder.class]
SLF4J: See http://www.slf4j.org/codes.html#multiple_bindings for an explanation.
SLF4J: Actual binding is of type [org.apache.logging.slf4j.Log4jLoggerFactory]

Logging initialized using configuration in file:/opt/hive/conf/hive-log4j2.properties Async: true
Could not open input file for reading. (File file:/opt/Paso02.hql does not exist)
root@fd7cbe1c8503:/opt# ls -l
total 16
drwxr-xr-x 1 20415 input 4096 Feb  5  2018 hadoop-2.7.4
drwxr-xr-x 1 root  root  4096 Nov 20 14:16 hive
root@fd7cbe1c8503:/opt# cd hive
root@fd7cbe1c8503:/opt/hive# ls
LICENSE  NOTICE  Paso02.hql  RELEASE_NOTES.txt  bin  binary-package-licenses  conf  examples  hcatalog  jdbc  lib  scripts
root@fd7cbe1c8503:/opt/hive# ls -l
total 84
-rw-r--r-- 1 root staff 20798 Nov  9  2017 LICENSE
-rw-r--r-- 1 root staff   230 Nov  9  2017 NOTICE
-rw-rw-r-- 1 1000  1000  4679 Nov 20 13:08 Paso02.hql
-rw-r--r-- 1 root staff  1979 Nov  9  2017 RELEASE_NOTES.txt
drwxr-xr-x 3 root root   4096 Feb  5  2018 bin
drwxr-xr-x 2 root root   4096 Feb  5  2018 binary-package-licenses
drwxr-xr-x 1 root root   4096 Nov 20 13:48 conf
drwxr-xr-x 4 root root   4096 Feb  5  2018 examples
drwxr-xr-x 7 root root   4096 Feb  5  2018 hcatalog
drwxr-xr-x 2 root root   4096 Feb  5  2018 jdbc
drwxr-xr-x 4 root root  16384 Feb  5  2018 lib
drwxr-xr-x 4 root root   4096 Feb  5  2018 scripts
root@fd7cbe1c8503:/opt/hive# clear
root@fd7cbe1c8503:/opt/hive# hive -f Paso02.hql
SLF4J: Class path contains multiple SLF4J bindings.
SLF4J: Found binding in [jar:file:/opt/hive/lib/log4j-slf4j-impl-2.6.2.jar!/org/slf4j/impl/StaticLoggerBinder.class]
SLF4J: Found binding in [jar:file:/opt/hadoop-2.7.4/share/hadoop/common/lib/slf4j-log4j12-1.7.10.jar!/org/slf4j/impl/StaticLoggerBinder.class]
SLF4J: See http://www.slf4j.org/codes.html#multiple_bindings for an explanation.
SLF4J: Actual binding is of type [org.apache.logging.slf4j.Log4jLoggerFactory]

Logging initialized using configuration in file:/opt/hive/conf/hive-log4j2.properties Async: true
OK
Time taken: 1.924 seconds
OK
Time taken: 0.085 seconds
OK
Time taken: 0.18 seconds
OK
Time taken: 0.383 seconds
OK
Time taken: 0.016 seconds
OK
Time taken: 0.076 seconds
OK
```

Tablas creadas:

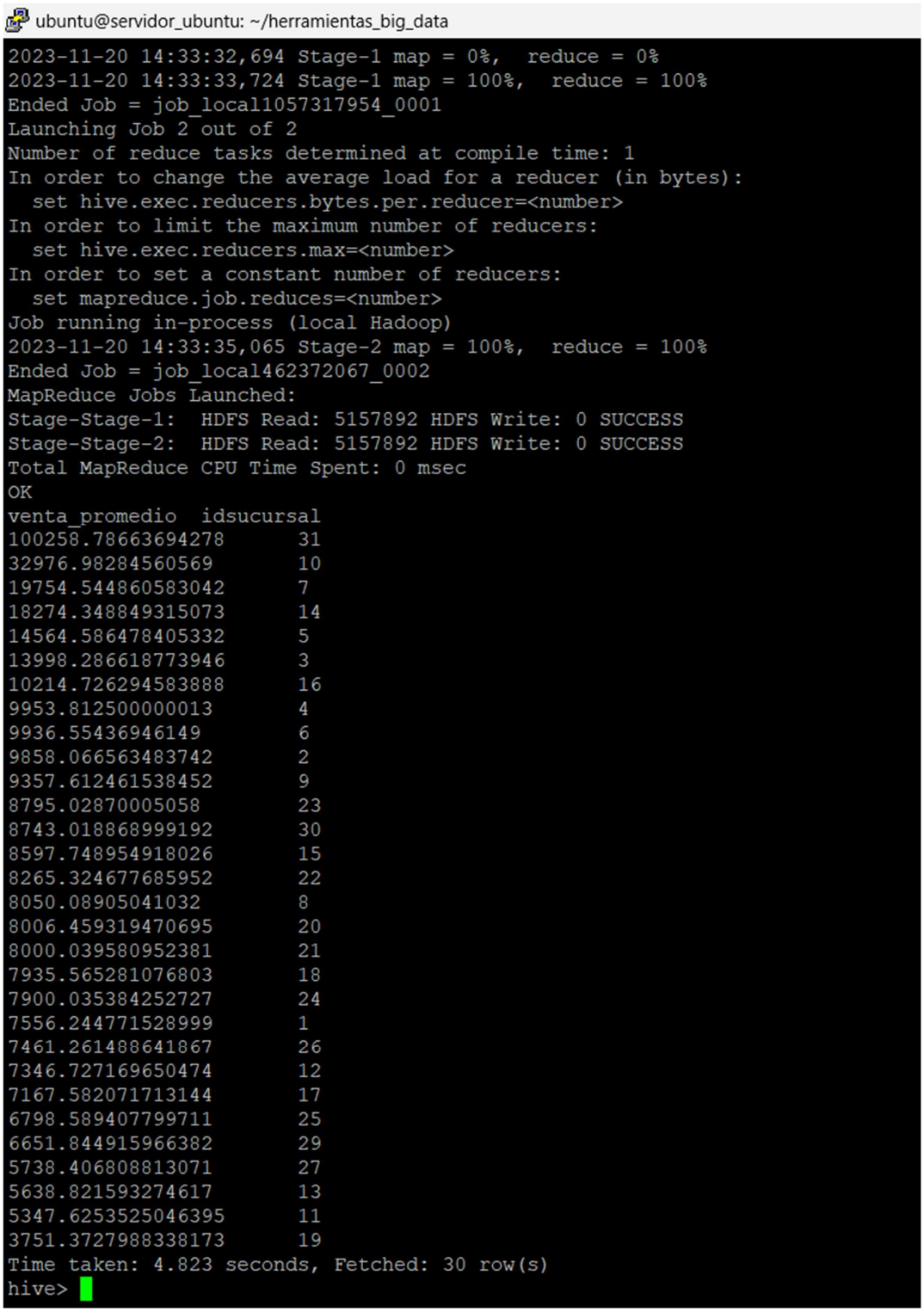
```
ubuntu@servidor_ubuntu: ~/herramientas_big_data
root@fd7cbe1c8503:/opt/hive# hive
SLF4J: Class path contains multiple SLF4J bindings.
SLF4J: Found binding in [jar:file:/opt/hive/lib/log4j-slf4j-impl-2.6.2.jar!/org/slf4j/impl/StaticLoggerBinder.class]
SLF4J: Found binding in [jar:file:/opt/hadoop-2.7.4/share/hadoop/common/lib/slf4j-log4j12-1.7.10.jar!/org/slf4j/impl/StaticLoggerBinder.class]
SLF4J: See http://www.slf4j.org/codes.html#multiple_bindings for an explanation.
SLF4J: Actual binding is of type [org.apache.logging.slf4j.Log4jLoggerFactory]

Logging initialized using configuration in file:/opt/hive/conf/hive-log4j2.properties Async: true
Hive-on-MR is deprecated in Hive 2 and may not be available in the future versions. Consider using a different execution engine (i.e. spark, tez) or using Hive 1.X releases.
hive> use integrador;
OK
Time taken: 0.586 seconds
hive> show tables;
OK
calendario
canal_venta
cliente
compra
empleado
gasto
producto
proveedor
sucursal
tipo_gasto
venta
Time taken: 0.132 seconds, Fetched: 11 row(s)
hive>
```

Consulta:

SELECT AVG(precio * cantidad) AS venta_promedio, idsucursal FROM venta

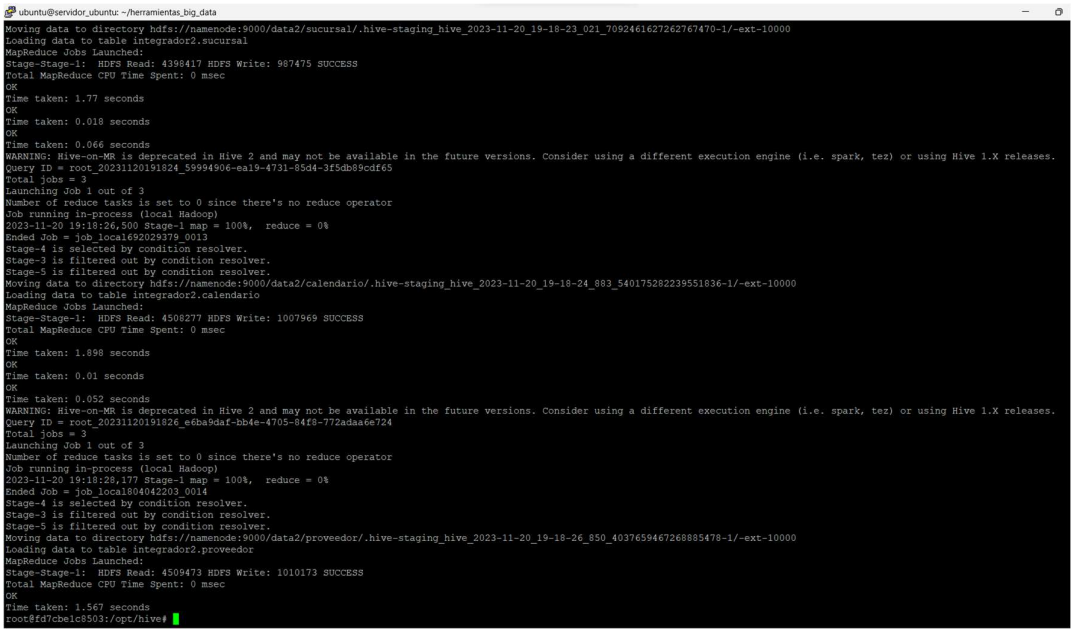
GROUP BY idsucursal

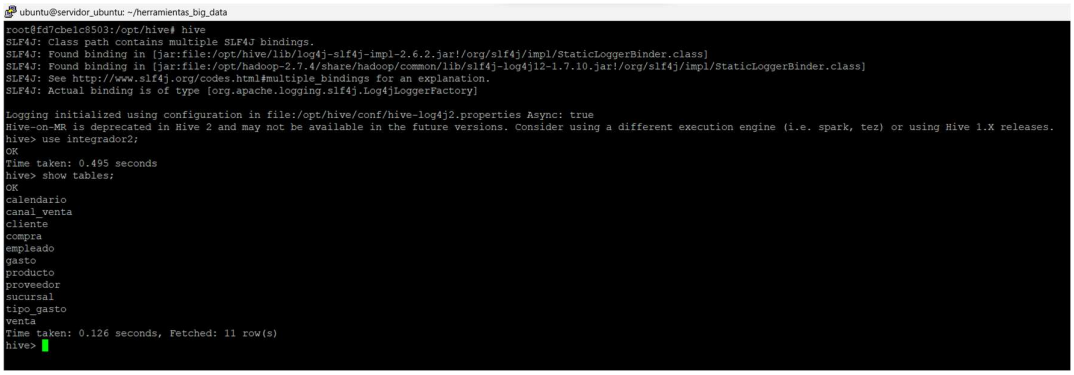
ORDER BY venta_promedio DESC;



Paso 03

Creación de tablas Parquet-Snappy

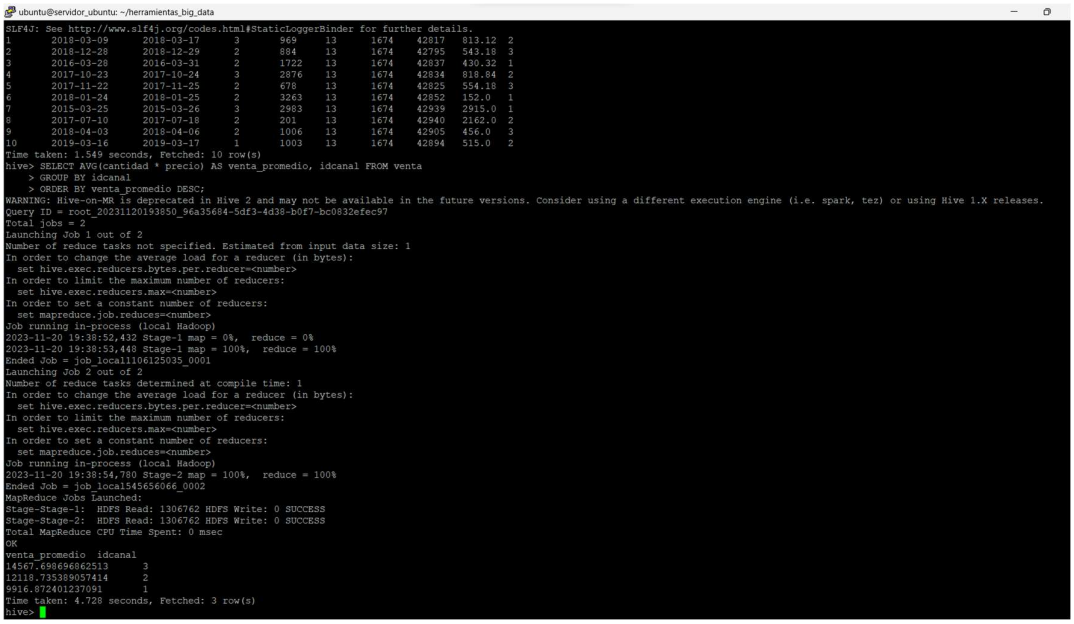




SET hive.cli.print.header=true; (comando para que muestre los nombres de columna).

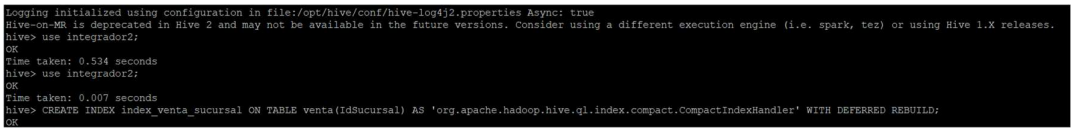
Ejecutamos la QUERY:

```
hive> SELECT AVG(cantidad * precio) AS venta_promedio, idcanal FROM venta
    > GROUP BY idcanal
    > ORDER BY venta_promedio DESC;
```



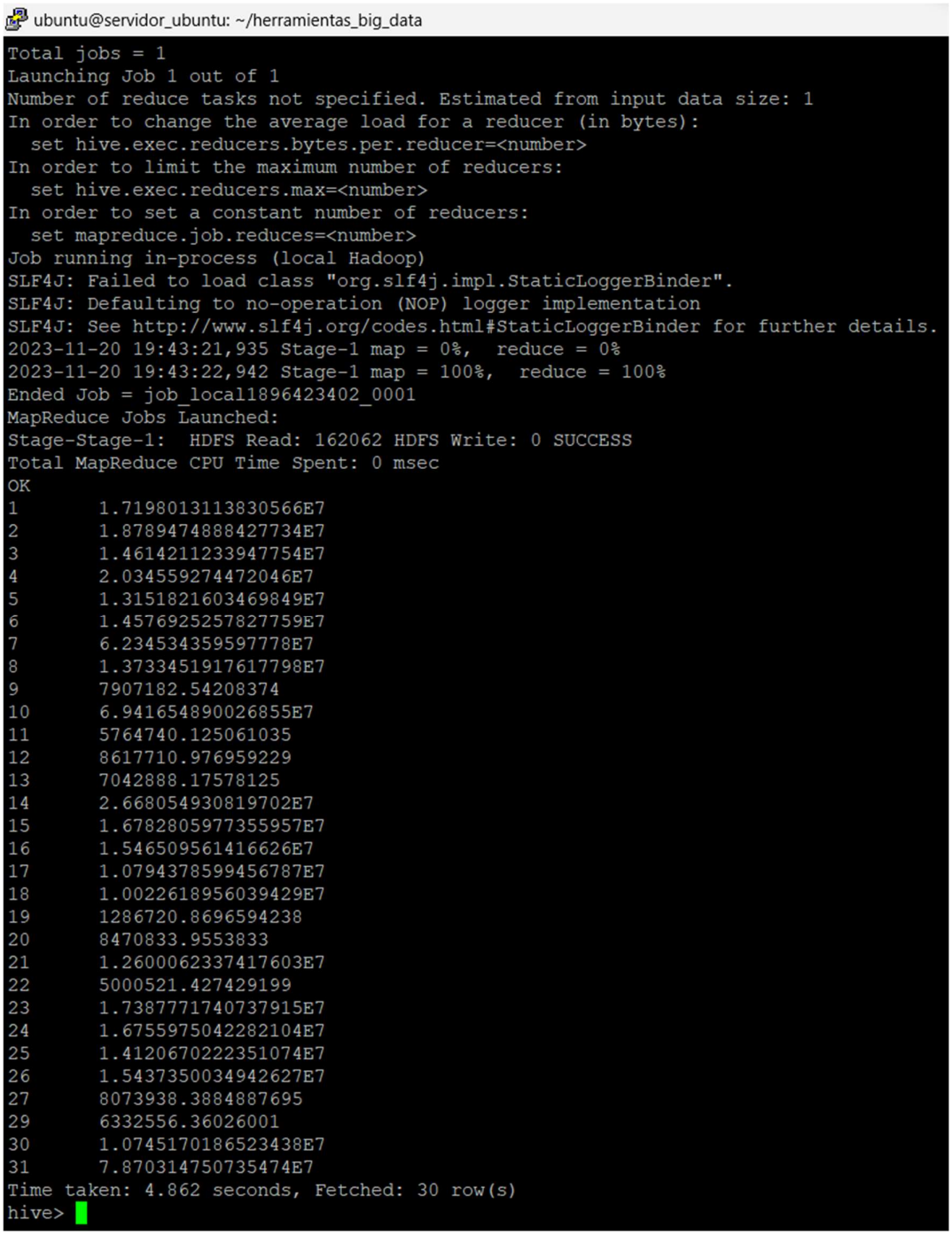
Paso 04

Creamos el índice



Ejecutamos query

hive> select idsucursal, sum(precio * cantidad) from venta group by idsucursal;

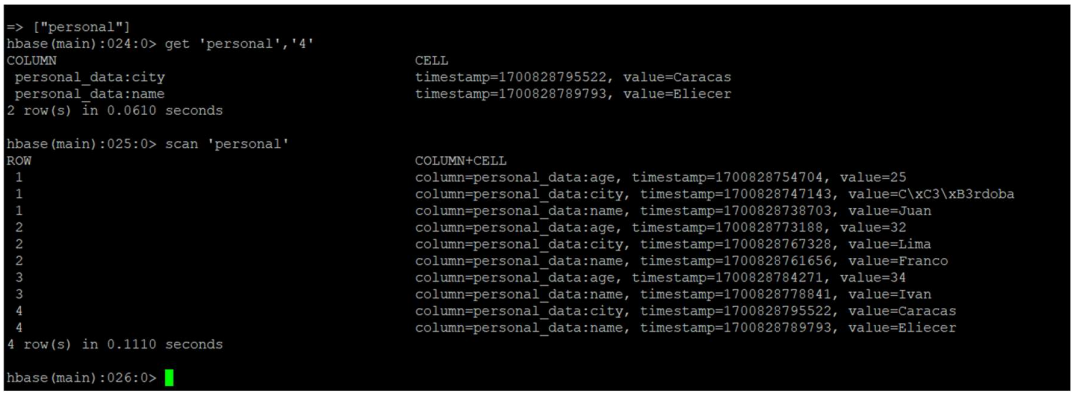


Parte 5

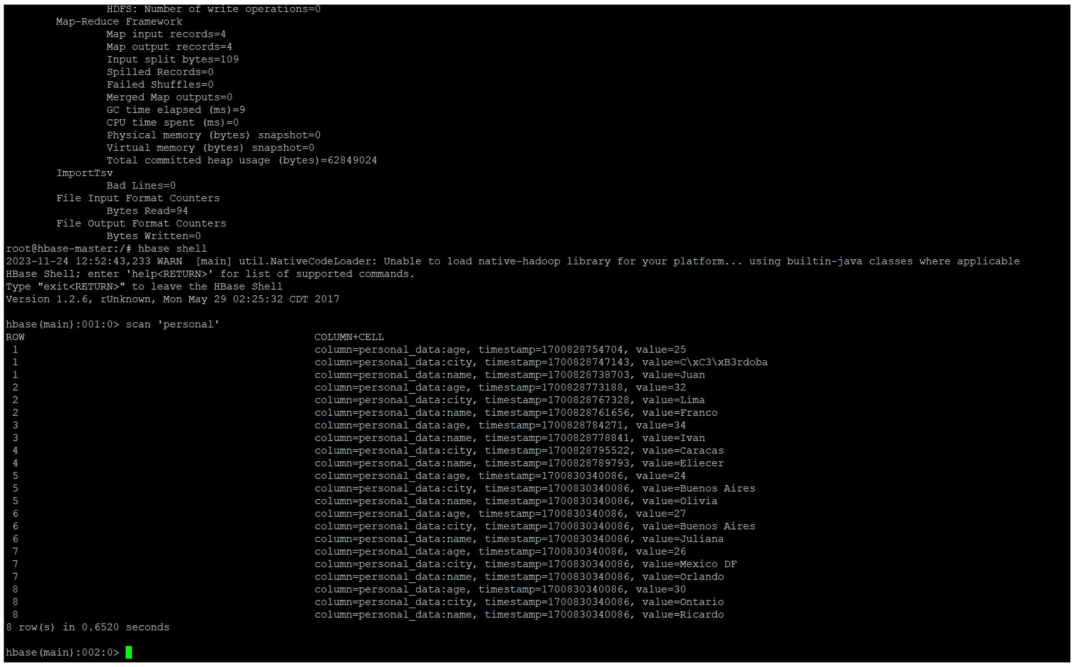
HBase

Contenedores levantados:

Comando: scan ‘personal’. Muestra los datos de la tabla creada



Luego de pasar el archivo personal.csv al HDFS, y de ejecutar la importación de los datos de personal.csv a una tabla Hbase con el comando ImportTsv. Luego mostramos la tabla con el comando scan ‘personal’.



Creamos la tabla ‘album’, poblamos con datos, y mostramos la familia de columnas ‘label1’:

```
hbase(main):002:0> create 'album','label','image'
0 row(s) in 1.4240 seconds

=> Hbase::Table - album
hbase(main):003:0> put 'album','label1','label:size','10'
0 row(s) in 0.1370 seconds

hbase(main):004:0> put 'album','label1','label:color','255:255:255'
0 row(s) in 0.0460 seconds

hbase(main):005:0> put 'album','label1','label:text','Family album'
0 row(s) in 0.0330 seconds

hbase(main):006:0> put 'album','label1','image:name','holiday'
0 row(s) in 0.0150 seconds

hbase(main):007:0> put 'album','label1','image:source','/tmp/pic1.jpg'
0 row(s) in 0.0150 seconds

hbase(main):008:0> get 'album','label1'
COLUMN                                          CELL
 image:name                                     timestamp=1700831080983, value=holiday
 image:source                                   timestamp=1700831086689, value=/tmp/pic1.jpg
 label:color                                    timestamp=1700831067882, value=255:255:255
 label:size                                     timestamp=1700831056007, value=10
 label:text                                     timestamp=1700831074342, value=Family album
5 row(s) in 0.0630 seconds

hbase(main):009:0>
```

MongoDB

Para esta parte tuve que editar el .yml para usar la versión mongodb:4.4.6

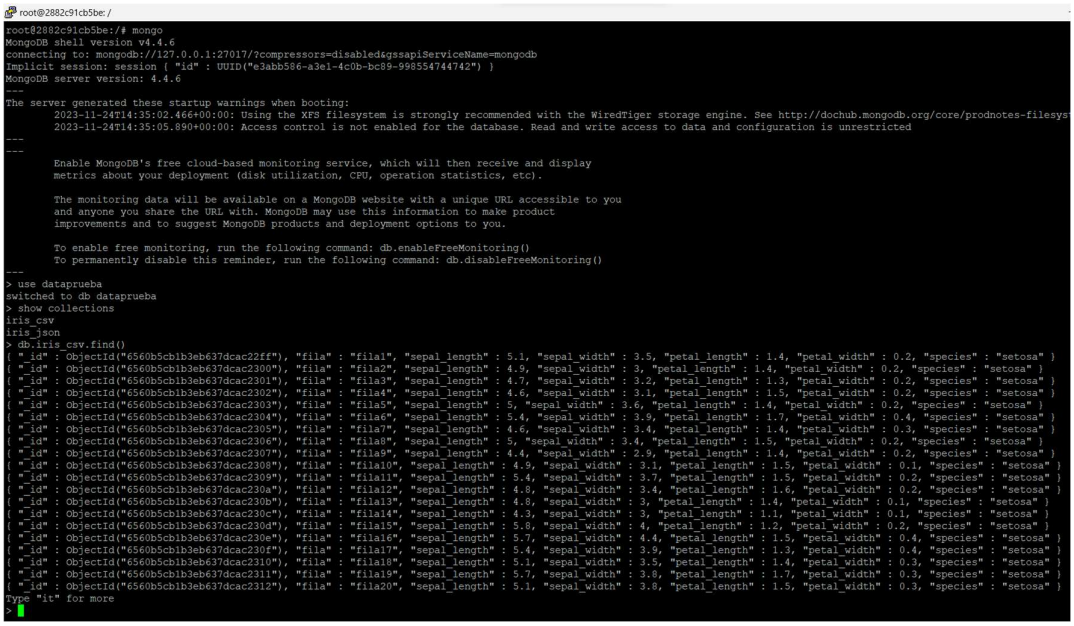
Una vez levantados los contenedores, ejecutamos un mongoimport

```
root@2882c91cb5be: /
ubuntu@servidor_ubuntu:~/herramientas_big_data/Datasets$ sudo docker exec -it mongodb bash
root@2882c91cb5be:/# mongoimport /data/iris.csv --type csv --headerline -d dataprueba -c iris_csv
2023-11-24T14:40:11.474+0000    connected to: mongodb://localhost/
2023-11-24T14:40:11.540+0000    150 document(s) imported successfully. 0 document(s) failed to import.
root@2882c91cb5be:/#
```

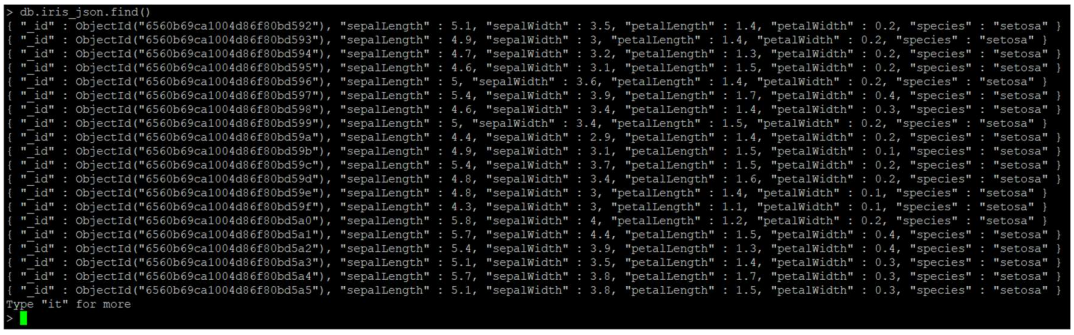
Luego otro mongoimport desde un json:

```
root@2882c91cb5be: /
root@2882c91cb5be:/# mongoimport --db dataprueba --collection iris_json --file /data/iris.json --jsonArray
2023-11-24T14:43:40.377+0000    connected to: mongodb://localhost/
2023-11-24T14:43:40.424+0000    150 document(s) imported successfully. 0 document(s) failed to import.
root@2882c91cb5be:/#
```

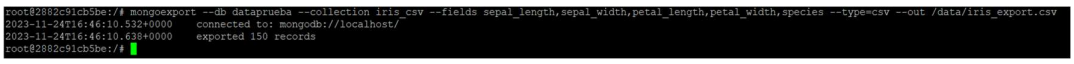
En lugar de usar mongosh, usamos la línea de comandos de mongo, debido a que no reconoce el comando mongosh. Luego usamos la base de datos creada y mostramos los documentos presentes en iris_csv



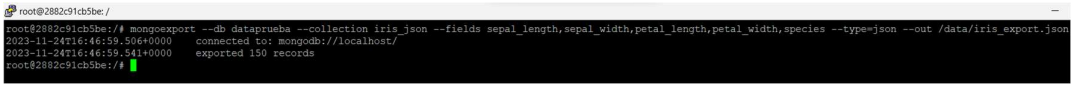
Mostramos los documentos de iris_json:



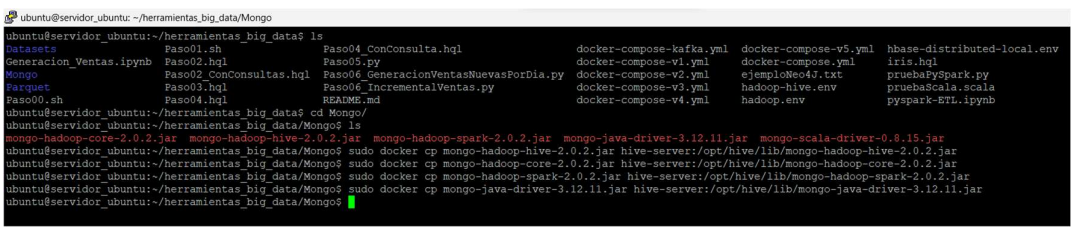
Aplicamos un primero mongoexport:



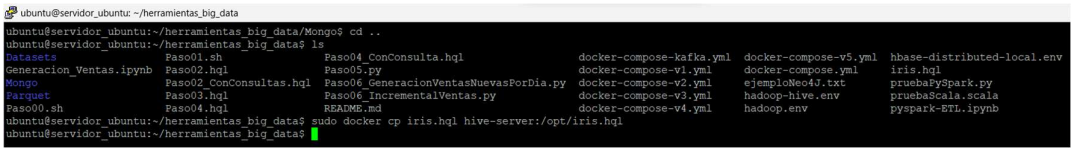
Aplicamos un segundo mongoexport:



Copiamos los archivos .jar dentro de hive-server:

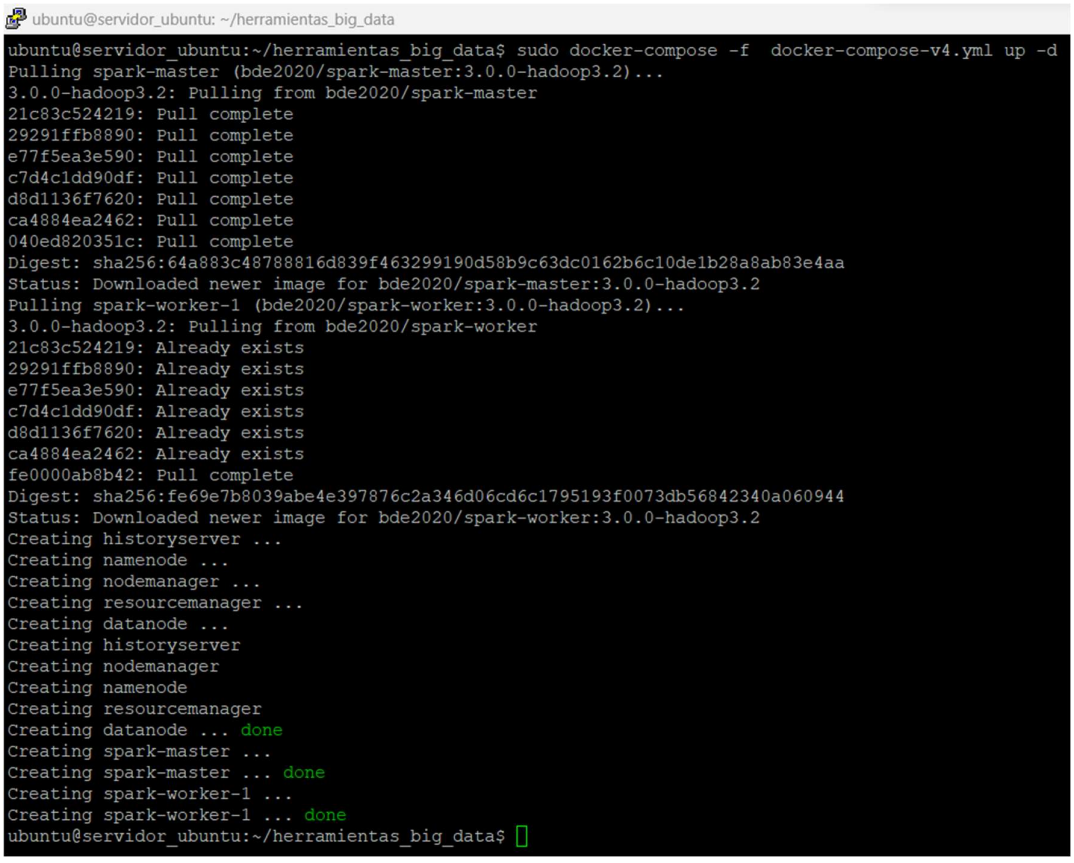


Copiamos iris.hql en hive-server:

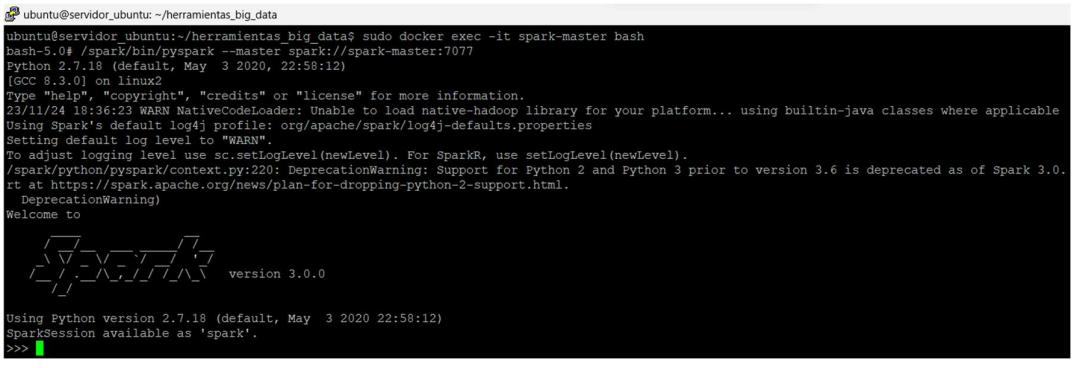


Paso 06

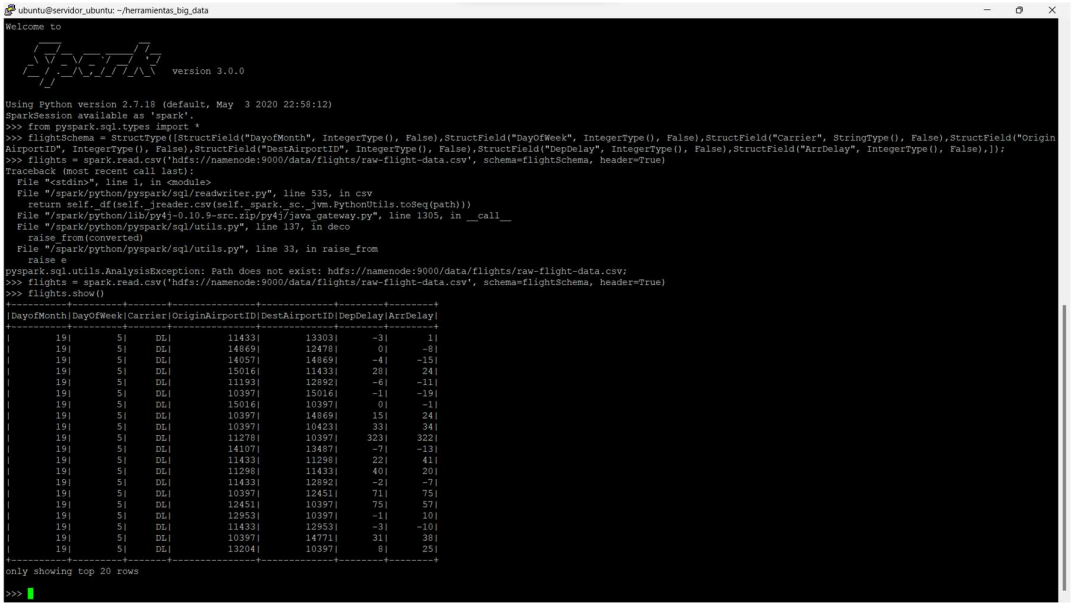
Generamos el entorno compose-v4.



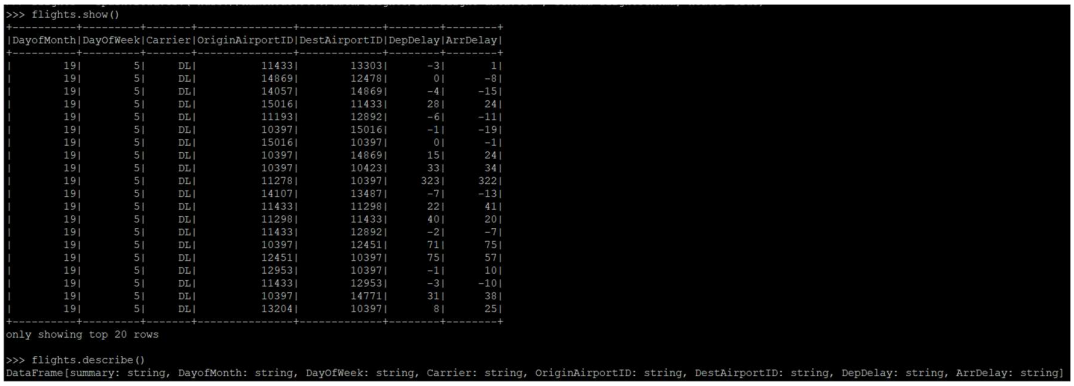
Entramos al bash de spark-master y ejecutamos pyspark



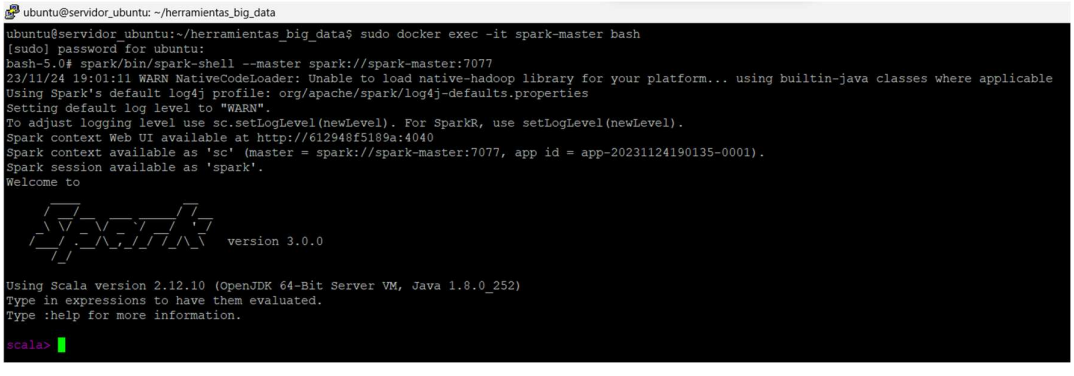
Importamos pyspark.sql.types *, luego definimos el esquema y cargamos los datos del csv (previamente colocado en HDFS) en spark. Luego mostramos la tabla:



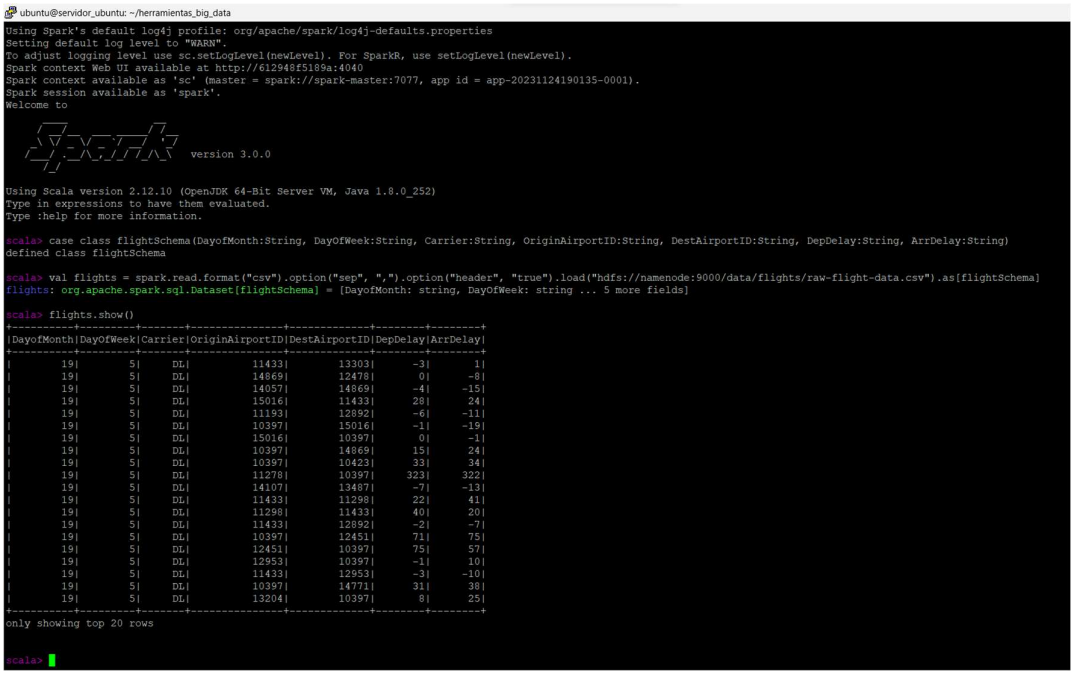
Ahora ejecutamos flights.describe():



Nos colocamos en la línea de comandos de spark-master y comenzamos scala:

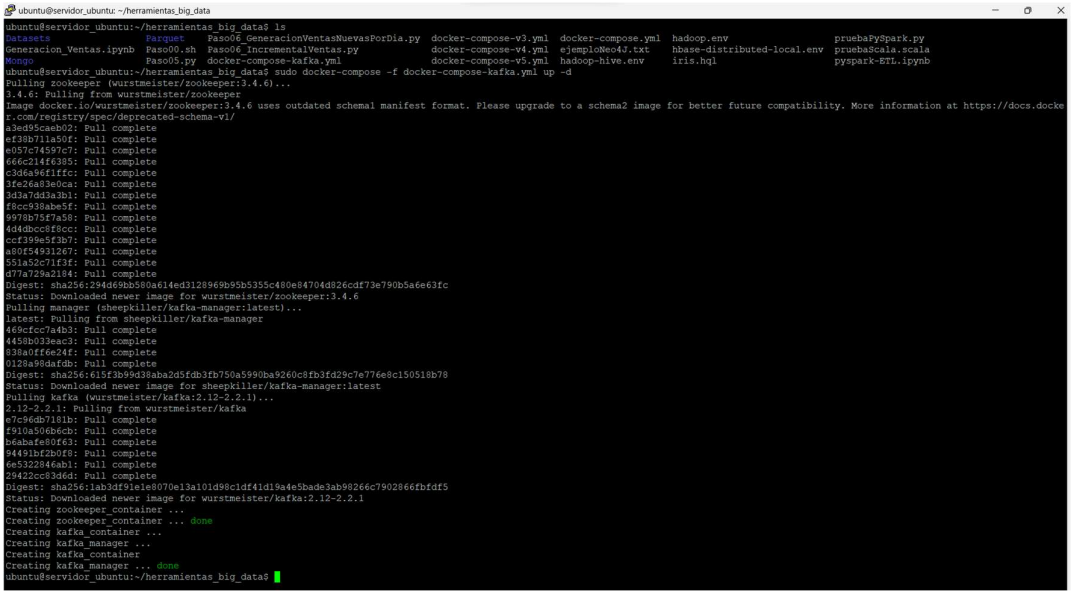


Creamos la clase flightschema, cargamos los datos desde el csv almacenado en el hdfs y luego mostramos la tabla:

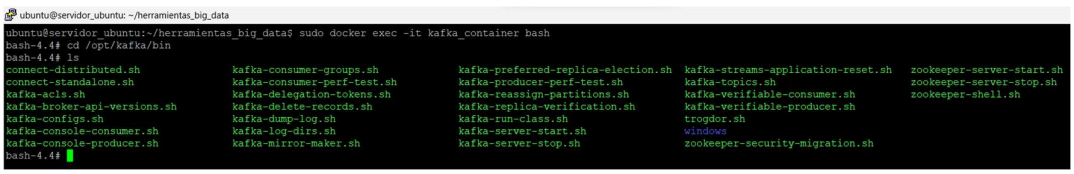


KAFKA

Levantamos el entorno con Docker-compose-kafka:



Entramos a Kafka_container:



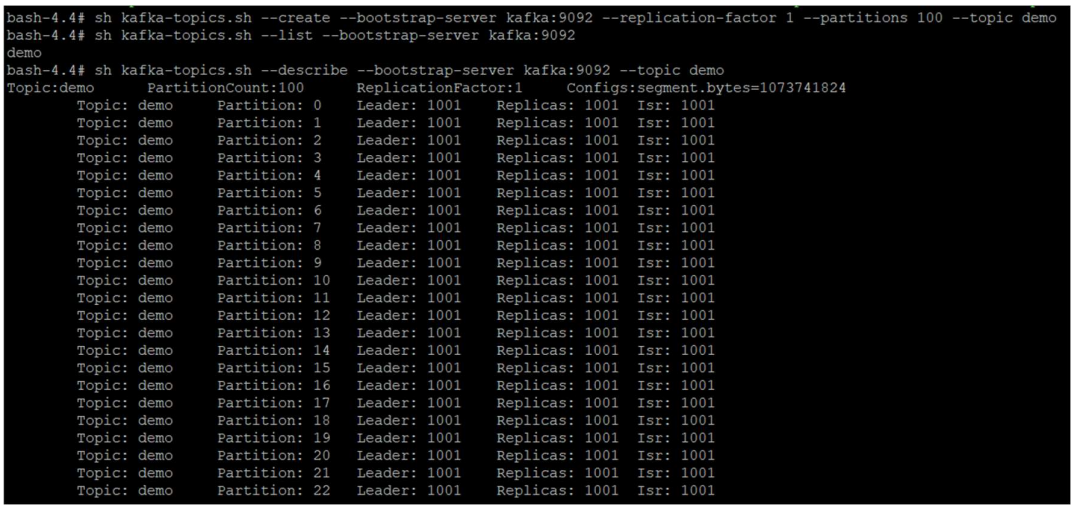
sh kafka-topics.sh --create --bootstrap-server kafka:9092 --replication-factor 1 --partitions 100 --topic demo

Con el commando anterior creamos un nuevo tópico llamado topic demo. Con el siguiente comando listamos los tópicos creados para ver si efectivamente se creó:

sh kafka-topics.sh --list --bootstrap-server kafka:9092

Luego con el commando siguiente describimos el tópico topic demo, donde vemos que hay 100 particiones (imagen cortada)

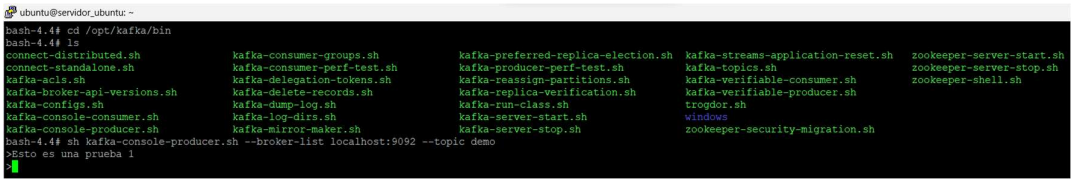
sh kafka-topics.sh --describe --bootstrap-server kafka:9092 --topic demo



sh kafka-console-consumer.sh --bootstrap-server kafka:9092 --topic demo --from-beginning

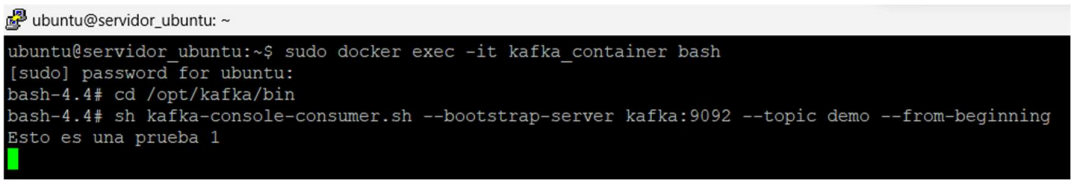
Con este comando indicamos consumir mensajes del tópico topic demo, desde el principio

En una consola aparte abrimos el bash de cafkca_container y ejecutamos el productor:

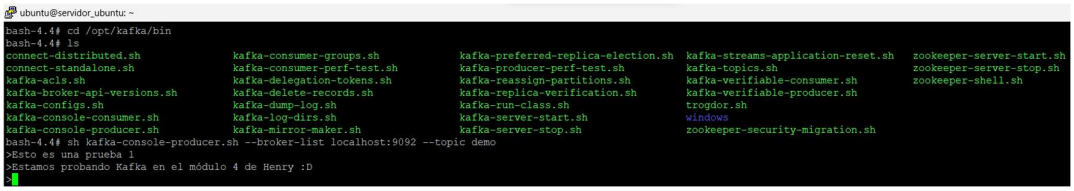


En la imagen anterior se observa que enviamos el mensaje ‘Esto es una prueba 1’

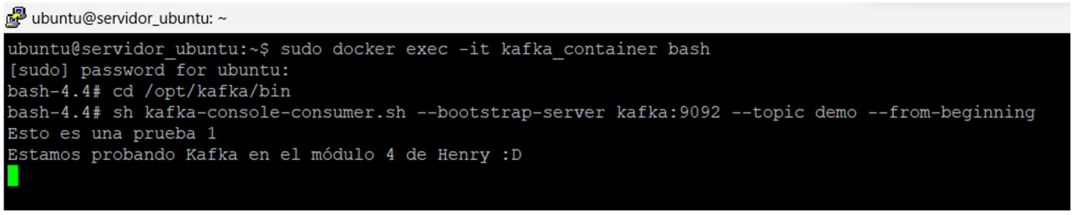
Ahora vamos del lado del consumidor y observamos que aparece el mensaje:



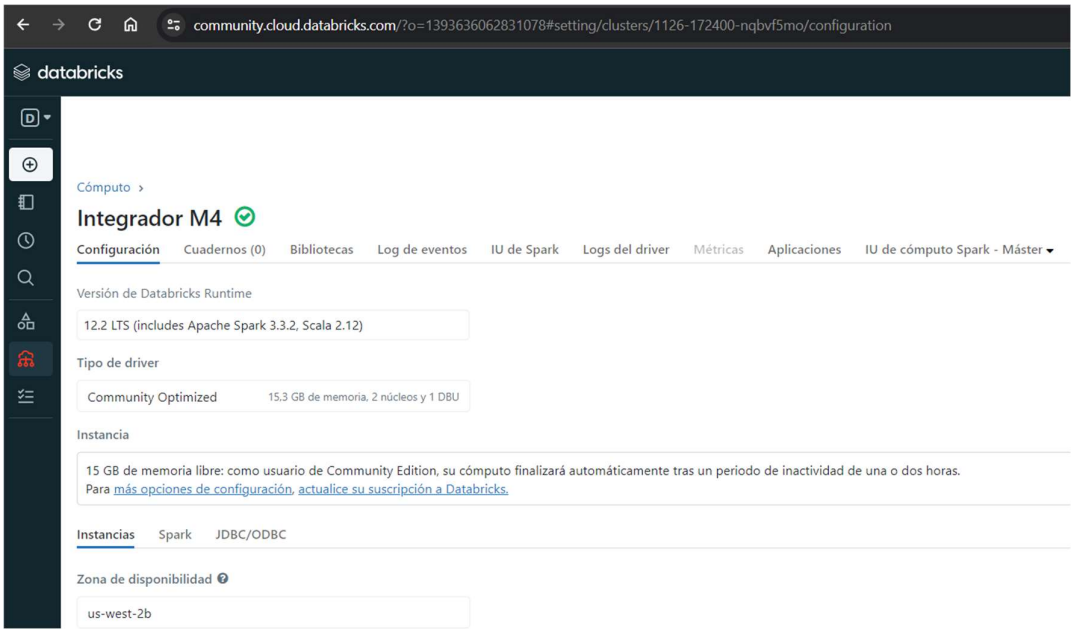
Enviamos un nuevo mensaje:



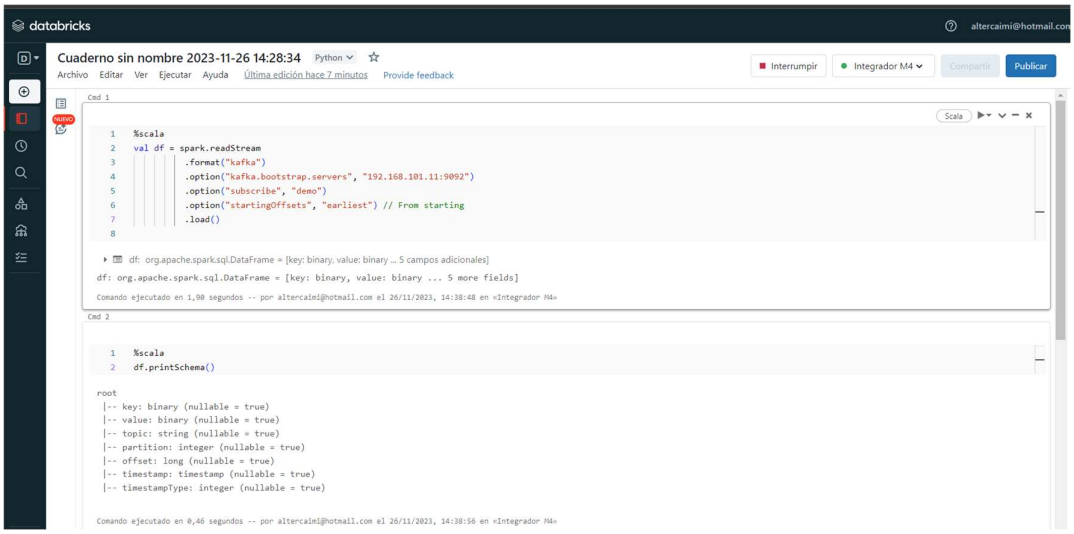
Vemos que aparece del lado del consumidor:



Ahora creamos un cluster en Databricks para poder usar spark:



Creamos el stream y vemos el esquema:


databricks
altercaimi@hotmail.com
Cuaderno sin nombre 2023-11-26 14:28:34
Python
Archivo Editar Ver Ejecutar Ayuda
Última edición hace 7 minutos
Provide feedback
Interrumpir
Integrador M4
Compartir
Publicar
Cmd 1
Scala
%scala
val df = spark.readStream
.format("kafka")
.option("kafka.bootstrap.servers", "192.168.101.11:9092")
.option("subscribe", "demo")
.option("startingOffsets", "earliest") // From starting
.load()
df: org.apache.spark.sql.DataFrame = [key: binary, value: binary ... 5 campos adicionales]
df: org.apache.spark.sql.DataFrame = [key: binary, value: binary ... 5 more fields]
Comando ejecutado en 1,90 segundos -- por altercaimi@hotmail.com el 26/11/2023, 14:38:48 en «Integrador M4»
Cmd 2
%scala
df.printSchema()
root
|-- key: binary (nullable = true)
|-- value: binary (nullable = true)
|-- topic: string (nullable = true)
|-- partition: integer (nullable = true)
|-- offset: long (nullable = true)
|-- timestamp: timestamp (nullable = true)
|-- timestampType: integer (nullable = true)
Comando ejecutado en 0,46 segundos -- por altercaimi@hotmail.com el 26/11/2023, 14:38:56 en «Integrador M4»