

PROYECTO INTEGRADOR

1) HDFS

Paso 1: Ingresamos al contendor

```
ubuntu@servidor_ubuntu: ~/herramientas_big_data
ubuntu@servidor_ubuntu:~/herramientas_big_data$ sudo docker exec -it namenode bash
root@2b65ee097865:/# hdfs dfs -put /home/Datasets/* /data
2024-08-25 23:00:02,985 INFO sasl.SaslDataTransferClient: SASL encryption trust check: localHost
Trusted = false, remoteHostTrusted = false
2024-08-25 23:00:03,272 INFO sasl.SaslDataTransferClient: SASL encryption trust check: localHost
Trusted = false, remoteHostTrusted = false
2024-08-25 23:00:03,340 INFO sasl.SaslDataTransferClient: SASL encryption trust check: localHost
Trusted = false, remoteHostTrusted = false
2024-08-25 23:00:03,410 INFO sasl.SaslDataTransferClient: SASL encryption trust check: localHost
Trusted = false, remoteHostTrusted = false
2024-08-25 23:00:03,473 INFO sasl.SaslDataTransferClient: SASL encryption trust check: localHost
Trusted = false, remoteHostTrusted = false
2024-08-25 23:00:03,562 INFO sasl.SaslDataTransferClient: SASL encryption trust check: localHost
Trusted = false, remoteHostTrusted = false
2024-08-25 23:00:03,617 INFO sasl.SaslDataTransferClient: SASL encryption trust check: localHost
Trusted = false, remoteHostTrusted = false
2024-08-25 23:00:03,694 INFO sasl.SaslDataTransferClient: SASL encryption trust check: localHost
Trusted = false, remoteHostTrusted = false
2024-08-25 23:00:03,723 WARN hdfs.DataStreamer: Caught exception
java.lang.InterruptedExcepcion
    at java.lang.Object.wait(Native Method)
    at java.lang.Thread.join(Thread.java:1252)
    at java.lang.Thread.join(Thread.java:1326)
    at org.apache.hadoop.hdfs.DataStreamer.closeResponder(DataStreamer.java:986)
    at org.apache.hadoop.hdfs.DataStreamer.closeInternal(DataStreamer.java:847)
    at org.apache.hadoop.hdfs.DataStreamer.run(DataStreamer.java:843)
2024-08-25 23:00:03,811 INFO sasl.SaslDataTransferClient: SASL encryption trust check: localHost
Trusted = false, remoteHostTrusted = false
2024-08-25 23:00:03,894 INFO sasl.SaslDataTransferClient: SASL encryption trust check: localHost
Trusted = false, remoteHostTrusted = false
2024-08-25 23:00:03,956 INFO sasl.SaslDataTransferClient: SASL encryption trust check: localHost
Trusted = false, remoteHostTrusted = false
```

Paso 2: Se copia los Datasets en el contenedor

```
ubuntu@servidor_ubuntu:~/herramientas_big_data$ sudo docker cp ./Datasets namenode:/home/Datasets
ubuntu@servidor_ubuntu:~/herramientas_big_data$ ls
Datasets
Generacion_Ventas.ipynb
Mongo
Parquet
Paso00.sh
Paso01.sh
Paso02.hql
Paso02_ConConsultas.hql
Paso03.hql
Paso04.hql
Paso04_ConConsulta.hql
Paso05.py
Paso06_GeneracionVentasNuevasPorDia.py
Paso06_IncrementalVentas.py
README.md
ubuntu@servidor_ubuntu:~/herramientas_big_data$ sudo docker cp Paso02.hql hive-s
```

Paso 3: Utilizamos el puerto para observar los datos

No es seguro192.168.1.83:9870/explorer.html#

Hadoop

Overview

Datanodes

Datanode Volume Failures

Snapshot

Startup Progress

Utilities

Browse Directory

/

Go!

Show25entries

Search

	Permission	Owner	Group	Size	Last Modified	Replication	Block Size	Name	
<input type="checkbox"/>	drwxr-xr-x	root	supergroup	0 B	Aug 25 16:00	9	0 B	data	<div></div>
<input type="checkbox"/>	drwxr-xr-x	root	supergroup	0 B	Aug 25 17:23	9	0 B	rmstate	<div></div>

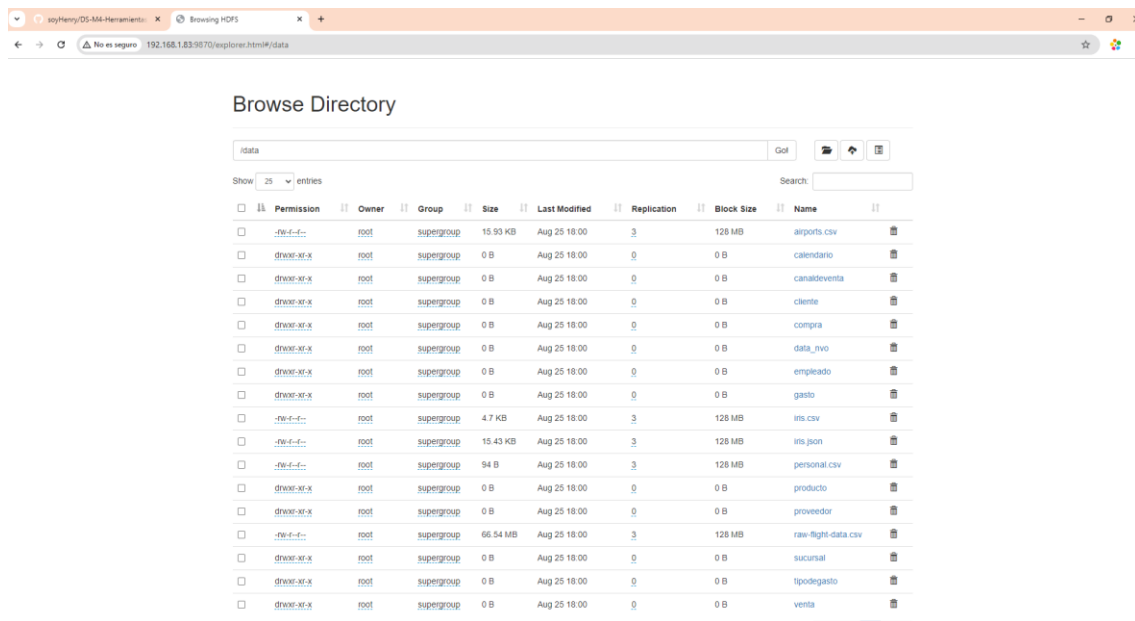
Showing 1 to 2 of 2 entries

Previous

1

Next

Hadoop, 2019



## 2) Hive

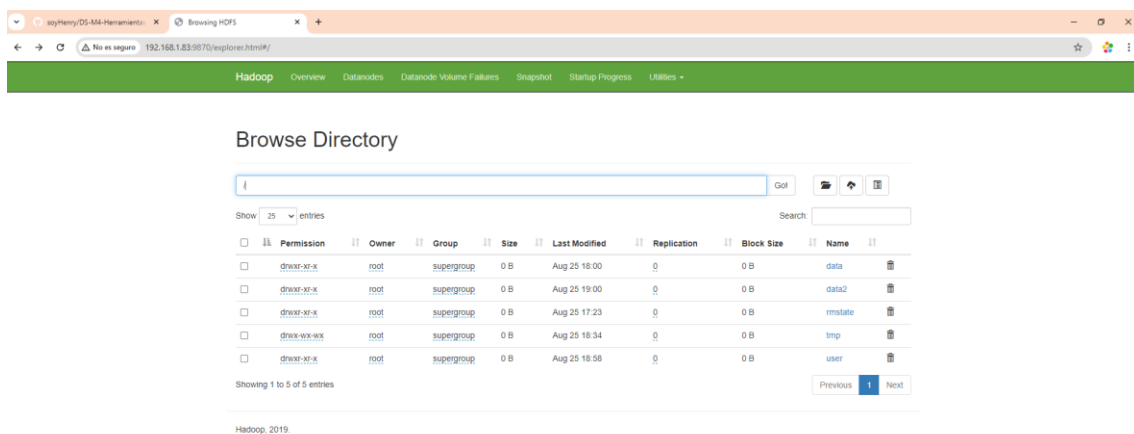
Paso 1: Utilizamos el entorno sudo docker-compose -f docker-compose-v2.yml up -d

Paso 2: Copiamos los archivos hacia el contenedor

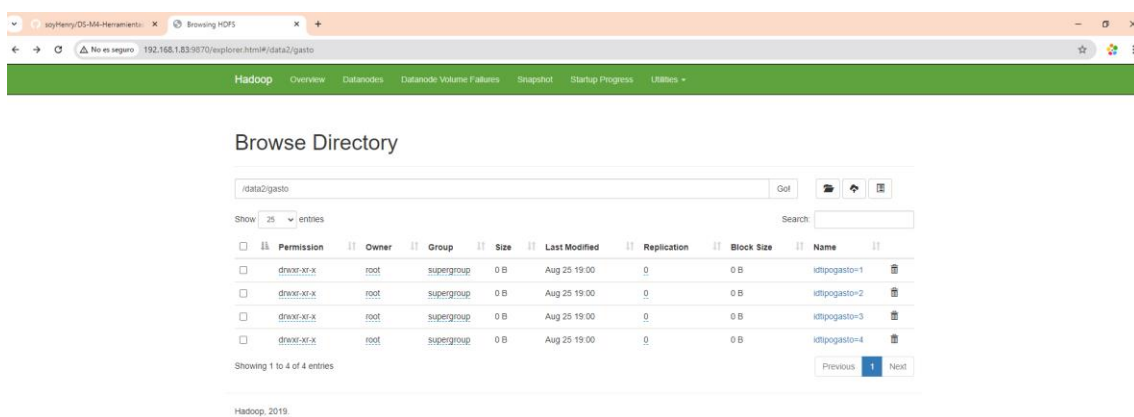
Paso 3: Ingresamos al contenedor sudo docker exec -it hive-server bash, confirmamos los archivos que se encuentran dentro del contenedor y luego ejecutamos los archivos Paso2, Paso3 y Paso 4, los cuales van a comprimir los archivos.

```
ubuntu@servidor_ubuntu: ~/herramientas_big_data
ubuntu@servidor_ubuntu:~/herramientas_big_data$ ls
Datasets                                Paso04_ConConsulta.hql                docker-compose.yml
Generacion_Ventas.ipynb                 Paso05.py                             ejemploNeo4J.txt
Mongo                                   Paso06_GeneracionVentasNuevasPorDia.py  hadoop-hive.env
Parquet                                 Paso06_IncrementalVentas.py           hadoop.env
Paso00.sh                               README.md                             hbase-distributed-local.env
Paso01.sh                               docker-compose-kafka.yml              iris.hql
Paso02.hql                              docker-compose-v1.yml                 pruebaPySpark.py
Paso02_ConConsultas.hql                 docker-compose-v2.yml                 pruebaScala.scala
Paso03.hql                              docker-compose-v3.yml                 pyspark-ETL.ipynb
Paso04.hql                              docker-compose-v4.yml
ubuntu@servidor_ubuntu:~/herramientas_big_data$ sudo docker cp Paso02.hql hive-server:/opt/
ubuntu@servidor_ubuntu:~/herramientas_big_data$ sudo docker cp Paso02.hql hive-server:/opt/
ubuntu@servidor_ubuntu:~/herramientas_big_data$ sudo docker cp Paso02.hql hive-server:/opt/
ubuntu@servidor_ubuntu:~/herramientas_big_data$ sudo docker cp Paso03.hql hive-server:/opt/
ubuntu@servidor_ubuntu:~/herramientas_big_data$ sudo docker cp Paso04.hql hive-server:/opt/
root@9f5d10178ed8:/opt# ls
Paso02.hql Paso03.hql Paso04.hql hadoop-2.7.4 hive
root@9f5d10178ed8:/opt# hive -f Paso02.hql
SLF4J: Class path contains multiple SLF4J bindings
```

Paso 2: Mostramos la carpeta data2 que se han generado con los archivos comprimidos.



Se muestra que la Tabla gato se ha particionado por tipo de gasto.



### 3) Formatos de Almacenamiento

Paso 1: creamos un nuevo archivo con la partición de las compras Particion\_compra.hql

Paso 2: como el winzip llevamos el archivo dentro de la carpeta “herramientas\_big\_data”

Paso 3: Luego llevamos el archivo dentro del contenedor

Sudo Docker cp Partición\_compra.hql hive-server:/opt/

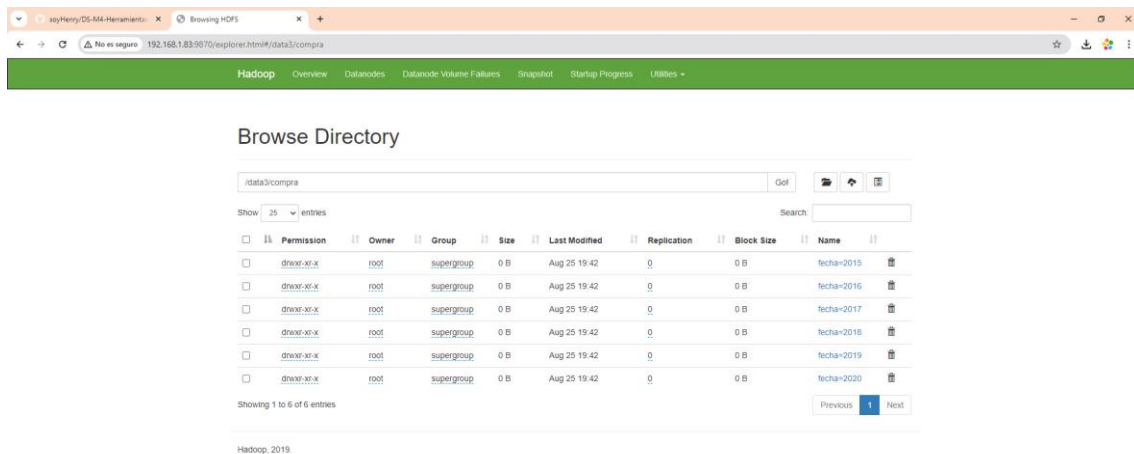
Paso 4: Ingresamos al contenedor sudo docker exec -it hive-server bash

Paso 5: Dentro del contenedor ejecutamos el archivo

root@9f5d10178ed8:/opt# hive -f Particion\_compra.hql

```
ubuntu@servidor_ubuntu: ~/herramientas_big_data
Loading data to table integrador3.compra partition (fecha=2019)
MapReduce Jobs Launched:
Stage-Stage-1: HDFS Read: 1954656 HDFS Write: 103929 SUCCESS
Total MapReduce CPU Time Spent: 0 msec
OK
Time taken: 3.147 seconds
WARNING: 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.
Query ID = root_20240826004255_7a97ce4e-d965-42d7-9c7a-77a721029312
Total jobs = 3
Launching Job 1 out of 3
Number of reduce tasks is set to 0 since there's no reduce operator
Job running in-process (local Hadoop)
2024-08-26 00:42:57,836 Stage-1 map = 100%, reduce = 0%
Ended Job = job_local1429551787_0006
Stage-4 is selected by condition resolver.
Stage-3 is filtered out by condition resolver.
Stage-5 is filtered out by condition resolver.
Moving data to directory hdfs://namenode:9000/data3/compra/fecha=2020/.hive-staging_hive_2024-08-26_00-42-55_621_8607605660891471158-1/-ext-10000
Loading data to table integrador3.compra partition (fecha=2020)
MapReduce Jobs Launched:
Stage-Stage-1: HDFS Read: 2345605 HDFS Write: 132925 SUCCESS
Total MapReduce CPU Time Spent: 0 msec
OK
Time taken: 2.828 seconds
root@9f5d10178ed8:/opt#
```

```
ubuntu@servidor_ubuntu: ~/herramientas_big_data
SLF4J: Defaulting to no-operation (NOP) logger implementation
SLF4J: See http://www.slf4j.org/codes.html#StaticLoggerBinder for further details.
1 42832 13 560.51 12 2015
2 42833 11 497.58 7 2015
3 42834 1 588.5 6 2015
4 42835 9 567.66 14 2015
5 42839 14 231.31 2 2015
6 42840 14 232.07 13 2015
7 42841 8 236.98 4 2015
8 42842 4 255.33 4 2015
9 42845 5 578.61 12 2015
10 42855 1 809.04 6 2015
Time taken: 3.238 seconds, Fetched: 10 row(s)
hive> select * from compra where fecha = 2020 limit 10;
OK
8945 42832 4 523.27 13 2020
8946 42833 6 621.58 9 2020
8947 42834 6 538.44 9 2020
8948 42835 11 585.74 10 2020
8949 42836 11 553.44 11 2020
8950 42837 14 287.34 8 2020
8951 42839 14 236.09 14 2020
8952 42841 5 221.7 4 2020
8953 42844 18 1367.73 3 2020
8954 42845 24 574.65 5 2020
Time taken: 1.301 seconds, Fetched: 10 row(s)
hive>
```



#### 4) SQL

**Paso 1:** Ingresamos al contenedor :~/herramientas\_big\_data\$ sudo docker exec -it hive-server bash

**Paso 2:** dentro del contenedor ingresamos a root@9f5d10178ed8:/opt# hive

**Paso 3:** dentro de hive> show databases; observamos:

```
hive> show databases;
OK
default
integrador
integrador2
integrador3
Time taken: 1.283 seconds, Fetched: 4 row(s)
hive> use integrador2;
OK
Time taken: 0.049 seconds
```

**Paso 3:** usamos el database integrador2 para crear el índice en la tabla venta:

```
hive> use integrador2;
OK
Time taken: 0.049 seconds
hive> CREATE INDEX index_fechaentrega ON TABLE venta(Fecha_Entrega)
> AS 'org.apache.hadoop.hive.q1.index.compact.CompactIndexHandler'
> WITH DEFERRED REBUILD;
OK
Time taken: 0.389 seconds
```

**Paso 4:** observamos los índices de la tabla venta y la descripción de la misma.

```
ubuntu@servidor_ubuntu: ~/herramientas_big_data
hive>
hive> show INDEXES ON venta;
OK
index_fechaentrega      venta      fecha_entrega      integrador2__venta_index_fechaentrega__
compact
index_venta_sucursal     venta      idsucursal          integrador2__venta_index_venta_sucursal
compact
Time taken: 0.108 seconds, Fetched: 2 row(s)
hive> DESCRIBE FORMATTED venta;
OK
# col_name              data_type      comment
idventa                 int
fecha                  date
fecha_entrega           date
idcanal                 int
idcliente              int
idsucursal              int
idempleado              int
idproducto              int
precio                 float
cantidad               int

# Detailed Table Information
Database:                integrador2
Owner:                   root
CreateTime:              Mon Aug 26 00:00:15 UTC 2024
LastAccessTime:          UNKNOWN
Retention:               0
Location:                hdfs://namenode:9000/data2/venta
Table Type:              MANAGED_TABLE
Table Parameters:
```