

Análisis aerolínea

Práctica obligatoria

Antonio Cabrera

Trabajo para el doble grado de
Ingeniería del Software y Matemática Computacional



Asignatura de procesamiento de datos

U-tad

España

Mayo 2024

Contents

1	Preparación del entorno de la máquina virtual	3
1.1	Preparación de Cassandra	3
1.2	Desplegar HDFS	5
1.3	Desplegar PostgreSQL	6
1.4	Desplegar Cassandra	6
1.5	Desplegar clúster de Spark Standalone	7
1.6	Desplegar una shell de Spark	10

List of Figures

1.1	Actualización del sistema operativo	3
1.2	Instalación de Python 2	4
1.3	Descarga de Cassandra	4
1.4	Descompresión de Cassandra	4
1.5	Eliminación del archivo .tar.gz	5
1.6	Despliegue de HDFS	5
1.7	Comprobación de HDFS	5
1.8	Comando de HDFS	5
1.9	Comprobación de Postgres	6
1.10	Comando de prueba de Postgres	6
1.11	Despliegue de Cassandra	7
1.12	Consola de Cassandra	7
1.13	Creación de keypace en Cassandra	7
1.14	Cambio de directorio a Spark	7
1.15	Arranque del Master de Spark	8
1.16	URL del Master de Spark	8
1.17	Interfaz web del Master de Spark	8
1.18	URL de los Workers de Spark	9
1.19	Configuración de los Workers de Spark	9
1.20	Arranque de los Workers de Spark	9
1.21	Interfaz web del Master de Spark con los Workers conectados . .	10

Chapter 1

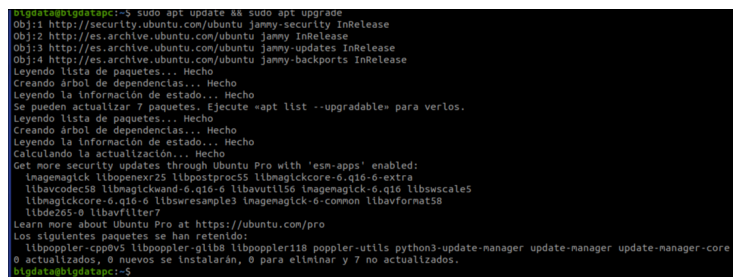
Preparación del entorno de la máquina virtual

Partiremos de la máquina virtual proporcionada por el profesor, la cual tiene instalado el sistema operativo Ubuntu 22.04.3 LTS.

1.1 Preparación de Cassandra

Primero instalaremos Cassandra, para ello primero actualizaremos el sistema operativo. El comando `sudo apt update` actualiza la lista de paquetes disponibles y sus versiones, mientras que el comando `sudo apt upgrade` instala las actualizaciones disponibles.

```
1 sudo apt update && sudo apt upgrade
```

A terminal window showing the output of the command 'sudo apt update && sudo apt upgrade'. The output includes updates for security, jammy-security, jammy-backports, and various system packages like libmagick, libavcodec, and libavformat. It also lists packages that have been held back and provides information on how to get more security updates through Ubuntu Pro.

```
bigdata@bigdatapc:~$ sudo apt update && sudo apt upgrade
Obj:1 http://security.ubuntu.com/ubuntu jammy-security InRelease
Obj:2 http://es.archive.ubuntu.com/ubuntu jammy InRelease
Obj:3 http://es.archive.ubuntu.com/ubuntu jammy-updates InRelease
Obj:4 http://es.archive.ubuntu.com/ubuntu jammy-backports InRelease
Leyendo lista de paquetes... Hecho
Creando árbol de dependencias... Hecho
Leyendo la información de estado... Hecho
Se pueden actualizar 7 paquetes. Ejecute «apt list --upgradable» para verlos.
Leyendo lista de paquetes... Hecho
Creando árbol de dependencias... Hecho
Leyendo la información de estado... Hecho
Calculando la actualización... Hecho
Get more security updates through Ubuntu Pro with 'esm-apps' enabled:
inagemagick libopenexr25 libpostproc55 libmagickcore-6.q16-6-extra
libavcodec58 libmagickwand-6.q16-6 libavutil56 inagemagick-6.q16 libswscales
libmagickcore-6.q16-6 libswresample3 inagemagick-6-common libavformat58
libdecor-0 libavfilter7
Learn more about Ubuntu Pro at https://ubuntu.com/pro
Los siguientes paquetes se han retenido:
  libpoppler-cpp05 libpoppler-glib8 libpoppler118 poppler-utils python3-update-manager update-manager update-manager-core
0 actualizados, 0 nuevos se instalarán, 0 para eliminar y 7 no actualizados.
bigdata@bigdatapc:~$
```

Figure 1.1: Actualización del sistema operativo

Ahora instalaremos Python 2, ya que Cassandra requiere esta versión de Python. Para ello, ejecutamos el siguiente comando:

```
1 sudo apt install python2
```

```
bigdata@bigdatapc:~$ sudo apt install python2
Leyendo lista de paquetes... Hecho
Creando árbol de dependencias... Hecho
Leyendo la información de estado... Hecho
python2 ya está en su versión más reciente (2.7.18-3).
0 actualizados, 0 nuevos se instalarán, 0 para eliminar y 7 no actualizados.
bigdata@bigdatapc:~$
```

Figure 1.2: Instalación de Python 2

Después, descargaremos el archivo .tar.gz de Cassandra desde la página oficial de Apache. Para ello, ejecutamos el siguiente comando:

```
1 wget https://dlcdn.apache.org/cassandra/3.11.16/apache-cassandra-3.11.16-bin.tar.gz
```

```
bigdata@bigdatapc:~$ wget https://dlcdn.apache.org/cassandra/3.11.16/apache-cassandra-3.11.16-bin.tar.gz
--2024-05-10 20:03:05-- https://dlcdn.apache.org/cassandra/3.11.16/apache-cassandra-3.11.16-bin.tar.gz
Resolviendo dlcdn.apache.org (dlcdn.apache.org)... 151.101.2.132, 2a01:4ee2::1644
Conectando con dlcdn.apache.org (dlcdn.apache.org)[151.101.2.132]:443... conectado.
Petición HTTP enviada, esperando respuesta... 200 OK
Longitud: 31111361 (30M) [application/x-gzip]
Guardando como: 'apache-cassandra-3.11.16-bin.tar.gz.1'
apache-cassandra-3.11.16-bin.tar.gz 100%[=====] 29,67M 9,33MB/s en 3,2s
2024-05-10 20:03:08 (9,33 MB/s) - 'apache-cassandra-3.11.16-bin.tar.gz.1' guardado [31111361/31111361]
bigdata@bigdatapc:~$
```

Figure 1.3: Descarga de Cassandra

Descomprimos el archivo .tar.gz con el siguiente comando:

```
1 tar -xvzf apache-cassandra-3.11.16-bin.tar.gz
```

```
bigdata@bigdatapc:~$ tar -xvzf apache-cassandra-3.11.16-bin.tar.gz
apache-cassandra-3.11.16/bin/
apache-cassandra-3.11.16/conf/
apache-cassandra-3.11.16/conf/triggers/
apache-cassandra-3.11.16/doc/
apache-cassandra-3.11.16/doc/cql3/
apache-cassandra-3.11.16/interface/
apache-cassandra-3.11.16/lib/
apache-cassandra-3.11.16/lib/sigar-bin/
apache-cassandra-3.11.16/pylib/
apache-cassandra-3.11.16/pylib/cqlshlib/
apache-cassandra-3.11.16/pylib/cqlshlib/test/
apache-cassandra-3.11.16/pylib/cqlshlib/test/config/
apache-cassandra-3.11.16/tools/
apache-cassandra-3.11.16/tools/bin/
apache-cassandra-3.11.16/tools/lib/
apache-cassandra-3.11.16/CASSANDRA-14092.txt
apache-cassandra-3.11.16/CHANGES.txt
apache-cassandra-3.11.16/LICENSE.txt
apache-cassandra-3.11.16/NEWS.txt
```

Figure 1.4: Descompresión de Cassandra

Por último, eliminamos el archivo .tar.gz con el siguiente comando:

```
1 rm apache-cassandra-3.11.16-bin.tar.gz
```

```
bigdata@bigdatapc:~$ rm apache-cassandra-3.11.16-bin.tar.gz
bigdata@bigdatapc:~$
```

Figure 1.5: Eliminación del archivo .tar.gz

1.2 Desplegar HDFS

HDFS ya está instalado por defecto en la máquina virtual en la carpeta `/hadoop-3.3.6`. Para desplegar HDFS ejecutamos el siguiente comando:

```
1 ~/hadoop-3.3.6/sbin/start-dfs.sh
```

```
bigdata@bigdatapc:~$ ./hadoop-3.3.6/sbin/start-dfs.sh
Starting namenodes on [localhost]
Starting datanodes
Starting secondary namenodes [bigdatapc]
```

Figure 1.6: Despliegue de HDFS

Ahora para comprobar que HDFS se ha desplegado correctamente, ejecutamos el siguiente comando:

```
1 jps
```

```
bigdata@bigdatapc:~$ jps
81604 DataNode
82038 Jps
81834 SecondaryNameNode
81466 NameNode
bigdata@bigdatapc:~$
```

Figure 1.7: Comprobación de HDFS

Ahora ya podemos ejecutar comandos de HDFS, como por ejemplo el siguiente comando que muestra los archivos en el sistema de archivos HDFS:

```
1 ~/hadoop-3.3.6/bin/hdfs dfs -ls /
```

```
bigdata@bigdatapc:~$ ~/hadoop-3.3.6/bin/hdfs dfs -ls /
Found 1 items
drwx-wx-wx - bigdata supergroup 0 2024-01-29 01:29 /tmp
```

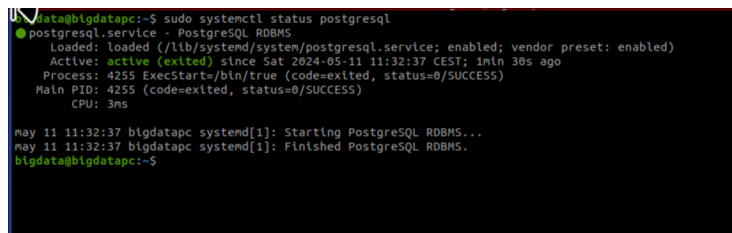
Figure 1.8: Comando de HDFS

1.3 Desplegar PostgreSQL

Postgres también está instalado por defecto en la máquina virtual, además se arranca por defecto al iniciar la sesión en la máquina virtual. El motivo por el que arranca por defecto es que se ha configurado como un servicio de systemd, por lo que se inicia automáticamente al arrancar el sistema.

Para comprobar que Postgres se ha desplegado correctamente, ejecutamos el siguiente comando:

```
1 sudo systemctl status postgresql
```



```
bigdata@bigdatapc:~$ sudo systemctl status postgresql
● postgresql.service - PostgreSQL RDBMS
   Loaded: loaded (/lib/systemd/system/postgresql.service; enabled; vendor preset: enabled)
   Active: active (exited) since Sat 2024-05-11 11:32:37 CEST; 1min 30s ago
     Process: 4255 ExecStart=/bin/true (code=exited, status=0/SUCCESS)
    Main PID: 4255 (code=exited, status=0/SUCCESS)
      CPU: 3ms

may 11 11:32:37 bigdatapc systemd[1]: Starting PostgreSQL RDBMS...
may 11 11:32:37 bigdatapc systemd[1]: Finished PostgreSQL RDBMS.
bigdata@bigdatapc:~$
```

Figure 1.9: Comprobación de Postgres

Ahora para asegurarnos de que todo funciona correctamente, nos conectamos a la consola de Postgres y ejecutamos un comando de prueba. Para ello, ejecutamos el siguiente comando:

```
1 sudo -u postgres psql
```

```
1 SELECT version();
```



```
bigdata@bigdatapc:~$ sudo -u postgres psql
psql (14.11 (Ubuntu 14.11-0ubuntu0.22.04.1))
Type "help" for help.

postgres=# SELECT version();
               version
-----
PostgreSQL 14.11 (Ubuntu 14.11-0ubuntu0.22.04.1) on x86_64-pc-linux-gnu, compiled by gcc (Ubuntu 11.4.0-1ubuntu1-22.04) 11.4.0, 64-bit
(1 row)

postgres=#
```

Figure 1.10: Comando de prueba de Postgres

1.4 Desplegar Cassandra

Primero nos moveremos a la carpeta de Cassandra con el siguiente comando:

```
1 cd ~/apache-cassandra-3.11.16
```

Acto seguido, arrancamos el servicio de Cassandra con el siguiente comando:

```
1 bin/cassandra
```

```
bigdata@bigdatapc:~$ cd ~/apache-cassandra-3.11.16/
bigdata@bigdatapc:~/apache-cassandra-3.11.16$ bin/cassandra
bigdata@bigdatapc:~/apache-cassandra-3.11.16$ OpenJDK 64-Bit Server VM warning: Cannot open file bin/./logs/gc.log due to No such file or
r directory

CompilerOracle: dontinline org/apache/cassandra/db/ColumnsSerializer.deserializeLargeSubset (Lorg/apache/cassandra/io/Util/DataInputPlus
;Lorg/apache/cassandra/db/Columns;I)Lorg/apache/cassandra/db/Columns;
CompilerOracle: dontinline org/apache/cassandra/db/ColumnsSerializer.serializeLargeSubset (Ljava/util/Collection;Lorg/apache/cassandra/
db/Columns;I)Lorg/apache/cassandra/io/Util/DataOutputPlus;V
CompilerOracle: dontinline org/apache/cassandra/db/ColumnsSerializer.serializeLargeSubsetSize (Ljava/util/Collection;Lorg/apache/cassan
dra/db/Columns;I)I
CompilerOracle: dontinline org/apache/cassandra/db/commitlog/AbstractCommitLogSegmentManager.advanceAllocatingFrom (Lorg/apache/cassand
```

Figure 1.11: Despliegue de Cassandra

Ahora inciamos la consola de Cassandra con el siguiente comando:

```
1 bin/cqlsh
```

```
bigdata@bigdatapc:~/apache-cassandra-3.11.16$ bin/cqlsh
Connected to Test Cluster at 127.0.0.1:9042.
[cqlsh 5.0.1 | Cassandra 3.11.16 | CQL spec 3.4.4 | Native protocol v4]
Use HELP for help.
cqlsh>
```

Figure 1.12: Consola de Cassandra

Por último, tendremos que generar un keyspace que nos servirá más adelante.

```
1 CREATE KEYSPACE IF NOT EXISTS practica WITH REPLICATION = {'class':
  'SimpleStrategy', 'replication_factor': 1};
```

Salimos de la consola de Cassandra con el siguiente comando:

```
1 exit
```

```
bigdata@bigdatapc:~/apache-cassandra-3.11.16$ bin/cqlsh
Connected to Test Cluster at 127.0.0.1:9042.
[cqlsh 5.0.1 | Cassandra 3.11.16 | CQL spec 3.4.4 | Native protocol v4]
Use HELP for help.
cqlsh> CREATE KEYSPACE IF NOT EXISTS practica WITH REPLICATION = {'class': 'SimpleStrategy', 'replication_factor': 1};
cqlsh> exit
bigdata@bigdatapc:~/apache-cassandra-3.11.16$
```

Figure 1.13: Creación de keyspace en Cassandra

1.5 Desplegar clúster de Spark Standalone

Vamos a ver como configurar y arrancar un despliegue de 1 nodo Master y 2 nodos Worker de Spark Standalone.

Primero, nos movemos a la carpeta de Spark con el siguiente comando:

```
1 cd ~/spark-3.2.0-bin-hadoop3.2
```

```
bigdata@bigdatapc:~$ cd spark-3.3.3-bin-hadoop3/
bigdata@bigdatapc:~/spark-3.3.3-bin-hadoop3$ ls
bin  data  examples  kubernetes  licenses  NOTICE.R  RELEASE  spark-warehouse
conf  derby.log  jars  LICENSE  metastore_db  python  README.md  sbin  yarn
bigdata@bigdatapc:~/spark-3.3.3-bin-hadoop3$
```

Figure 1.14: Cambio de directorio a Spark

Una vez que estamos en la carpeta de Spark, arrancamos el Master con el siguiente comando:

```
1 sbin/start-master.sh
```

```
bigdata@bigdatapc:~/spark-3.3.3-bin-hadoop3$ sbin/start-master.sh
starting org.apache.spark.deploy.master.Master, logging to /home/bigdata/spark-3.3.3-bin-hadoop3/logs/spark-bigdata-org.apache.spark.depl
oy.master.Master-1-bigdata.out
bigdata@bigdatapc:~/spark-3.3.3-bin-hadoop3$
```

Figure 1.15: Arranque del Master de Spark

En la ejecución del comando anterior, se nos muestra el archivo de los logs del Master, en este caso el archivo es `/home/bigdata/spark-3.3.3-bin-hadoop3/logs/spark-bigdata-org.apache.spark.deploy.master.Master-1-bigdata.out`. Con un par de comandos sacaremos la URL del Master, que es la dirección que usaremos para conectarnos a la interfaz web del Master.

```
1 cat /home/bigdata/spark-3.3.3-bin-hadoop3/logs/spark-bigdata-org.
  apache.spark.deploy.master.Master-1-bigdata.out | grep 'http://' |
  awk '{print $NF}'
```

```
bigdata@bigdatapc:~/spark-3.3.3-bin-hadoop3$ cat /home/bigdata/spark-3.3.3-bin-hadoop3/logs/spark-bigdata-org.apache.spark.deploy.master.
Master-1-bigdata.out | grep 'http://' | awk '{print $NF}'
http://10.0.2.15:8080
bigdata@bigdatapc:~/spark-3.3.3-bin-hadoop3$
```

Figure 1.16: URL del Master de Spark

En nuestro caso, si nos conectamos a la URL `http://10.0.2.15:8080/` podremos ver la interfaz web del Master de Spark.

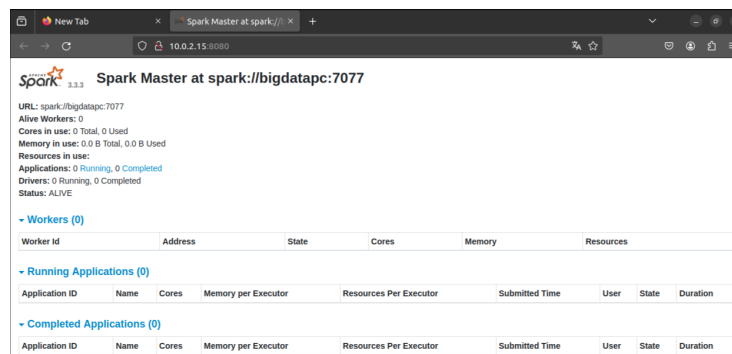


Figure 1.17: Interfaz web del Master de Spark

Para los workes, también necesitaremos una URL que encontraremos en los logs del Master. Para ello, ejecutamos el siguiente comando:

```
1 cat /home/bigdata/spark-3.3.3-bin-hadoop3/logs/spark-bigdata-org.
  apache.spark.deploy.master.Master-1-bigdata.out | grep 'spark://'
  | awk '{print $NF}'
```

```
bigdata@bigdatapc:~/spark-3.3.3-bin-hadoop3$ cat /home/bigdata/spark-3.3.3-bin-hadoop3/logs/spark-bigdata-org.apache.spark.deploy.master.  
master-1-bigdatapc.out | grep 'spark://' | awk '{print $6}'  
spark://bigdatapc:7077  
bigdata@bigdatapc:~/spark-3.3.3-bin-hadoop3$
```

Figure 1.18: URL de los Workers de Spark

Antes de desplegar los Workres, para poder tener dos Workers en una misma máquina vamos a modificar la configuración del archivo `conf/spark-env.sh`. Para ello, ejecutamos el siguiente comando:

```
1 vim conf/spark-env.sh
```

Y añadimos las 3 siguientes líneas al final del archivo:

```
1 SPARK_WORKER_INSTANCES=2  
2 SPARK_WORKER_CORES=2  
3 SPARK_WORKER_MEMORY=1g
```

```
SPARK_WORKER_CORES=2  
SPARK_WORKER_MEMORY=1g  
SPARK_WORKER_INSTANCES=2  
"conf/spark-env.sh.template" 81L, 4576B escritos
```

Figure 1.19: Configuración de los Workers de Spark

Ahora arrancaremos dos Workers con 1GB de memoria y 2 cores (la configuración que hemos especificado). Es necesario especificar la memoria y los cores ya que por defecto los Workers usarán toda la memoria y todos los cores disponibles.

```
1 sbin/start-slave.sh spark://bigdatapc:7077
```

```
bigdata@bigdatapc:~/spark-3.3.3-bin-hadoop3$ sbin/start-slave.sh spark://bigdatapc:7077  
This script is deprecated, use start-worker.sh  
starting org.apache.spark.deploy.worker.Worker, logging to /home/bigdata/spark-3.3.3-bin-hadoop3/logs/spark-bigdata-org.apache.spark.depl  
oy.worker.worker-1-bigdatapc.out  
starting org.apache.spark.deploy.worker.Worker, logging to /home/bigdata/spark-3.3.3-bin-hadoop3/logs/spark-bigdata-org.apache.spark.depl  
oy.worker.worker-2-bigdatapc.out  
bigdata@bigdatapc:~/spark-3.3.3-bin-hadoop3$
```

Figure 1.20: Arranque de los Workers de Spark

Si nos vamos a la interfaz web del Master de Spark, podremos ver los Workers conectados. En esta interfaz se nos muestra el id del nodo Worker, la dirección IP, el número de cores, la memoria disponible, la carga de trabajo, la memoria utilizada y el estado del nodo. Además, arriba se nos muestra un resumen de todos los recursos usados y de las aplicaciones en ejecución.

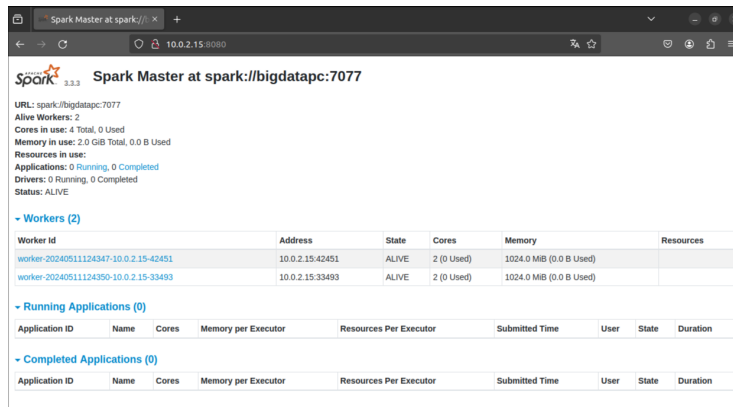


Figure 1.21: Interfaz web del Master de Spark con los Workers conectados

1.6 Desplegar una shell de Spark

El primer paso será descargar los conectores de Postgres y Cassandra. Primero nos moveremos a la carpeta *spark-3.3.3-bin-hadoop3/jars* y a continuación descargaremos los conectores con los siguientes comandos:

```

1 cd ~/spark-3.3.3-bin-hadoop3/jars
2 wget https://jdbc.postgresql.org/download/postgresql-42.7.3.jar
3 wget https://repo1.maven.org/maven2/com/datastax/spark/spark-
  cassandra-connector_2.12/3.3.0/spark-cassandra-connector_2
  .12-3.3.0.jar

```