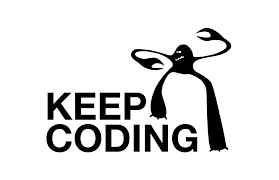
**Alumno……César Blanco Fernández**



# Objetivos

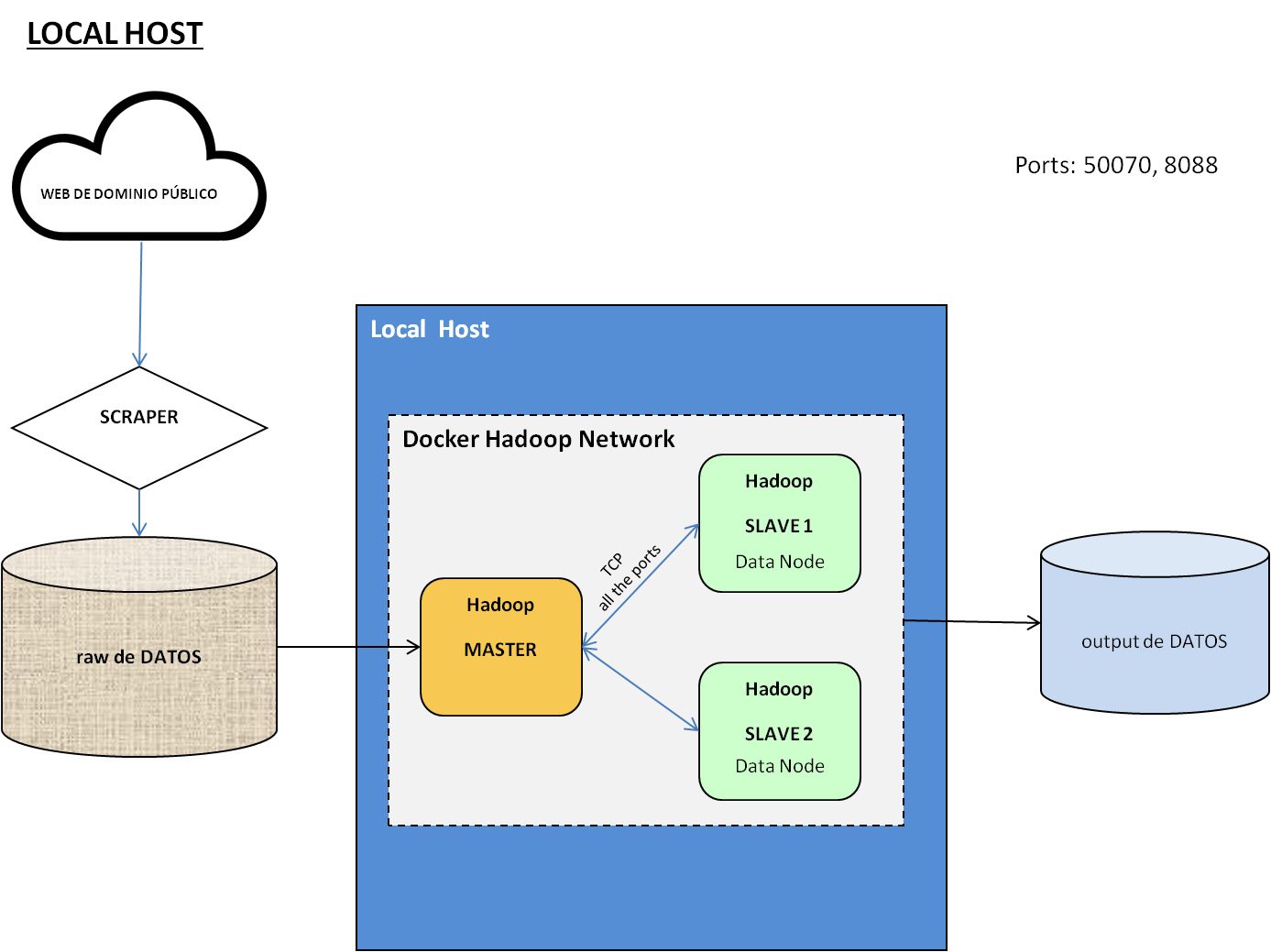
* Asegurar nociones básicas de arquitectura big data
* Conocer operaciones básicas de control de recursos de Hadoop
* Asegurar nociones básicas de administración de instancias en arquitectura Hadoop.

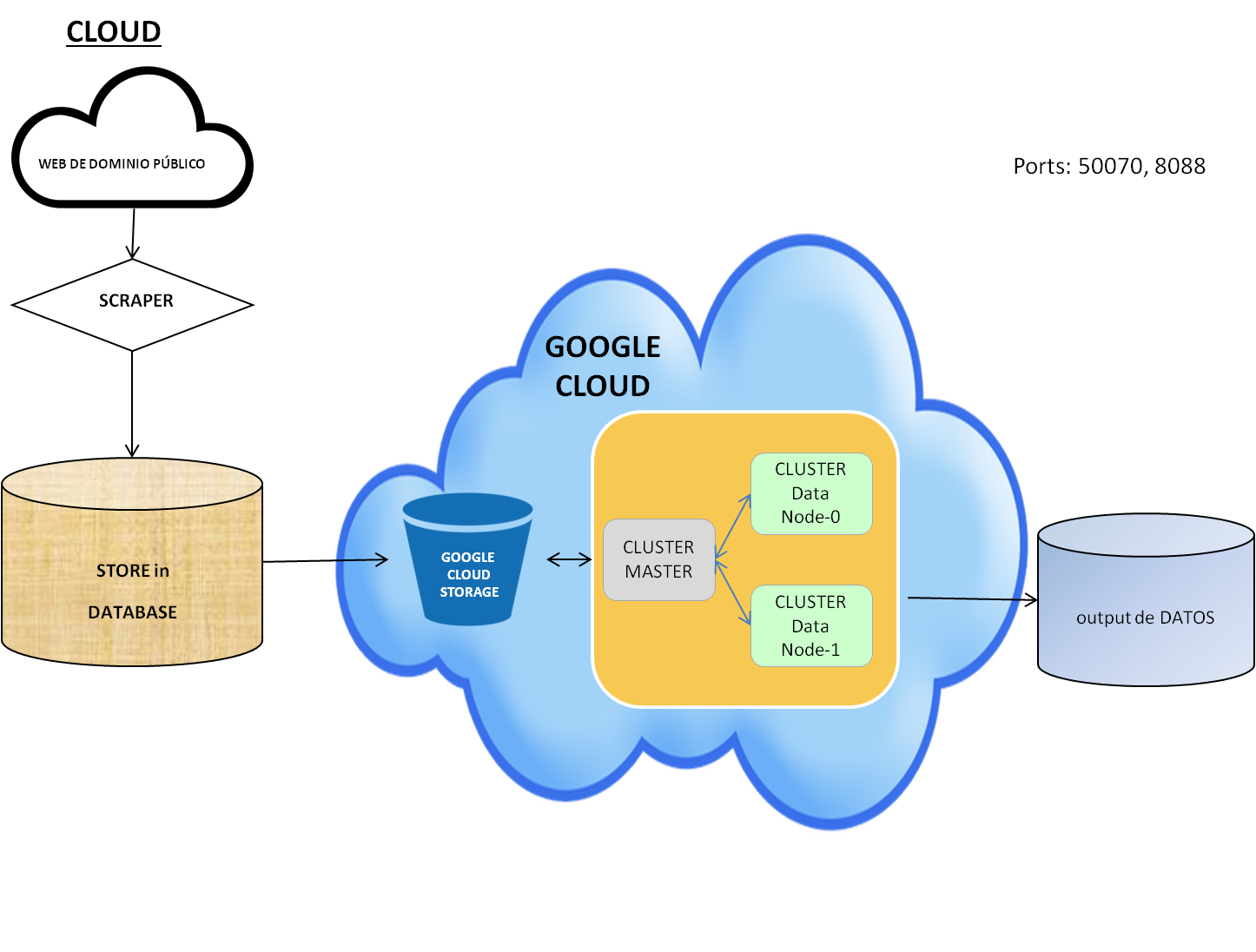
# Enunciado

Diseñar, especificar y desplegar un cluster de Hadoop para el procesamiento de datos estructurados extraídos mediante técnicas de scraping de sitios de dominio público.

## Sprint 1

Utilizar una herramienta de diagramado como DIA para diseñar y especificar el flujo de datos y herramientas utilizadas





## Sprint 2

Crear un scraper en Google Collaboratory a partir de un API o de un crawler con scrapy, que descargue los datos a un archivo de formato estructurado.

*Seguir el siguiente enlace del Notebook de Google Collaboratory para ver el resultado del Scrapy:*

<https://drive.google.com/open?id=17y0HV78WlFY8MXwNwxIKby4Plnvwe-bv>

## Sprint 3

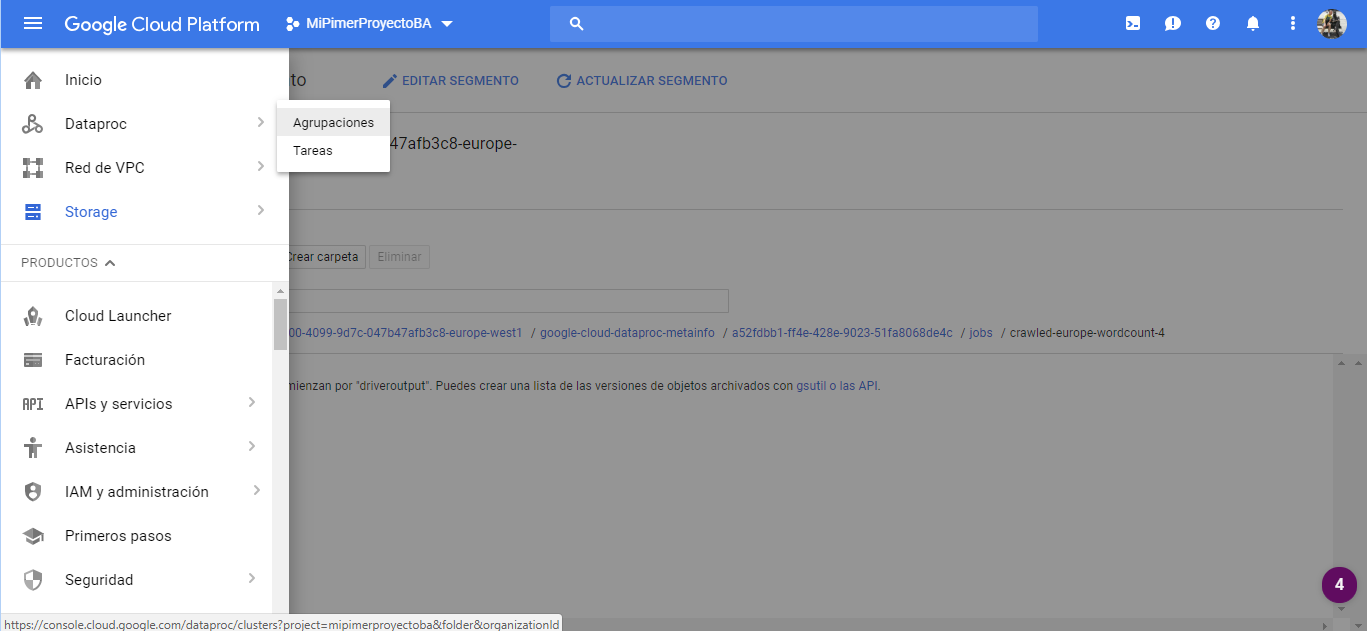
* Utilizar repositorio de docker de la asignatura para montar un cluster de al menos 2 contenedores configurados correctamente

ó

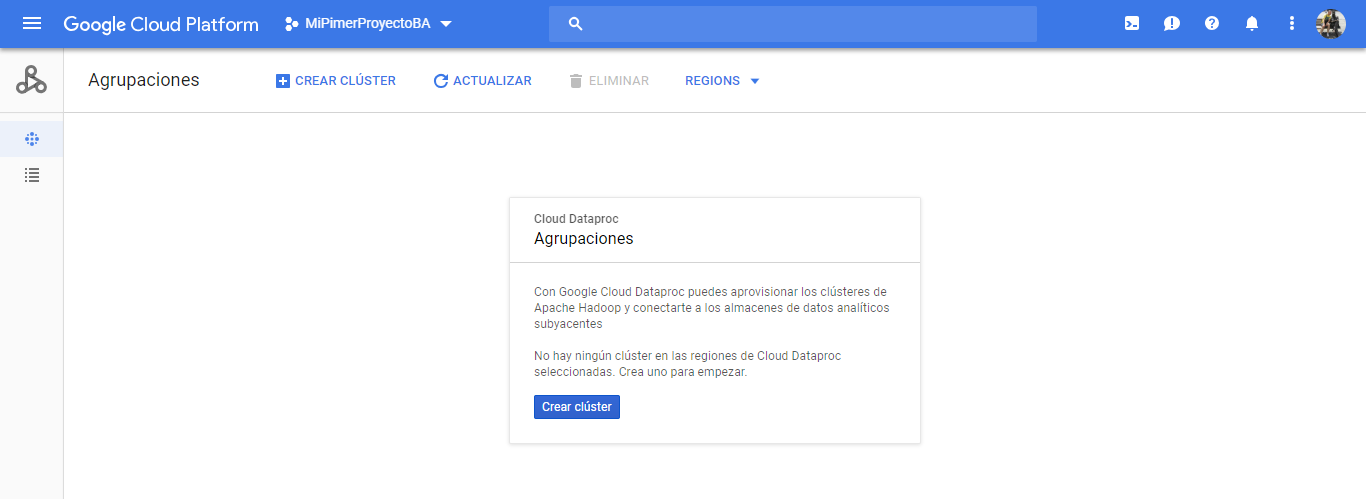
* Utilizar un proveedor de Cloud para montar un cluster de al menos 3 contenedores configurados correctamente

He elegido realizar el ejercicio montando un cluster formado por 3 contenedores en Google Cloud.

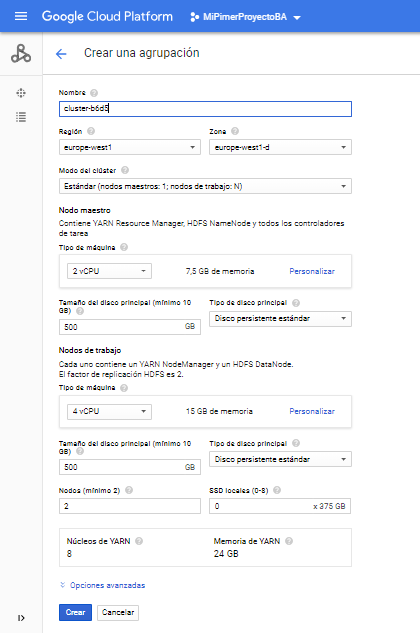
1. Creamos un proyecto con nombre: MiPrimerProyectoBA
2. Creamos un cluster formado por: 1 contenedor master y 2 contenedores. Para ello nos vamos en el Menú a Dataproc > Agrupaciones



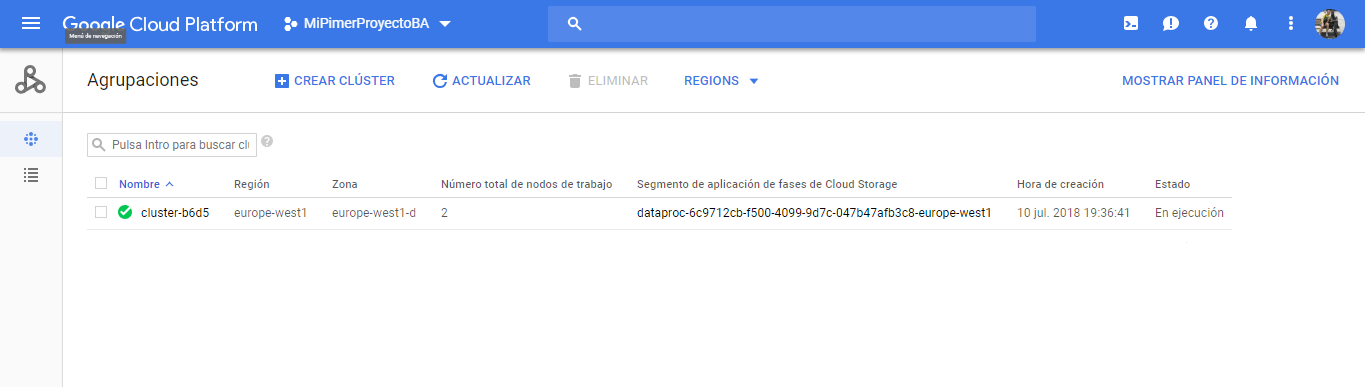
Pulsamos en Crear clúster:



Configuramos aquí la Región, Zona y el Tipo de máquina:

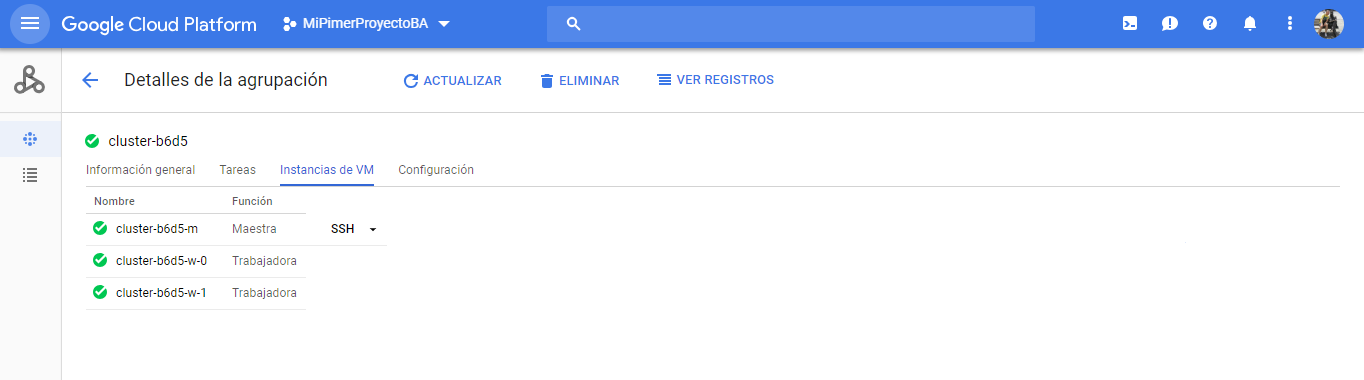


Ya temenos configurado el clúster:

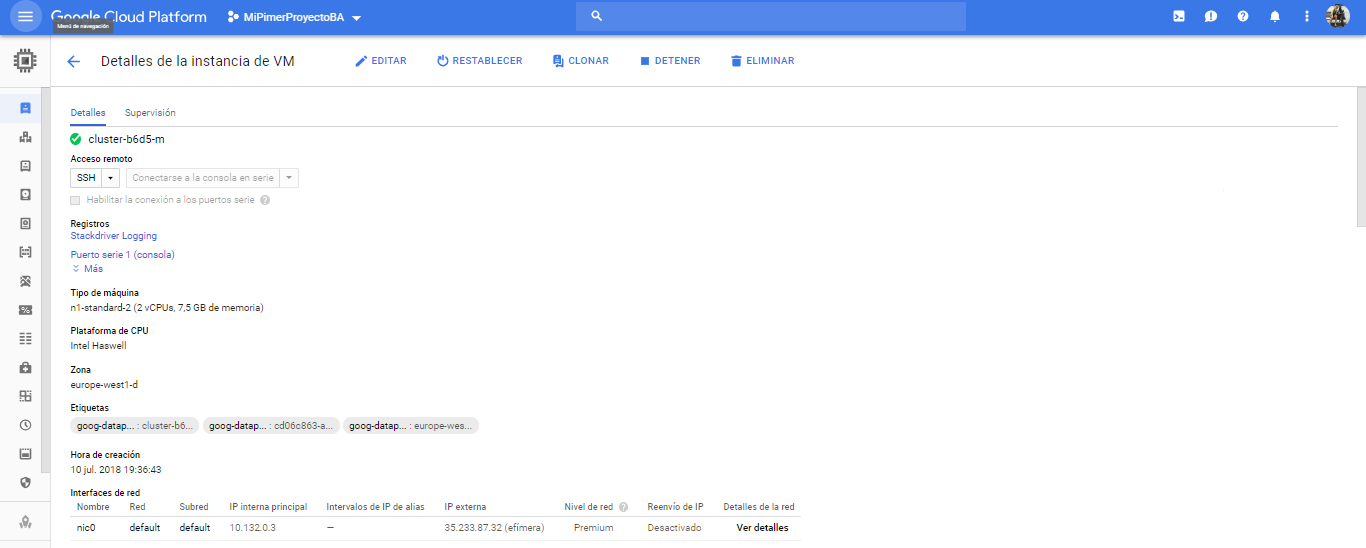


Nos metemos en: clúster-b6d5 y nos vamos a la pestaña Instancias de VM. Allí observamos que tiene 3 contenedores:

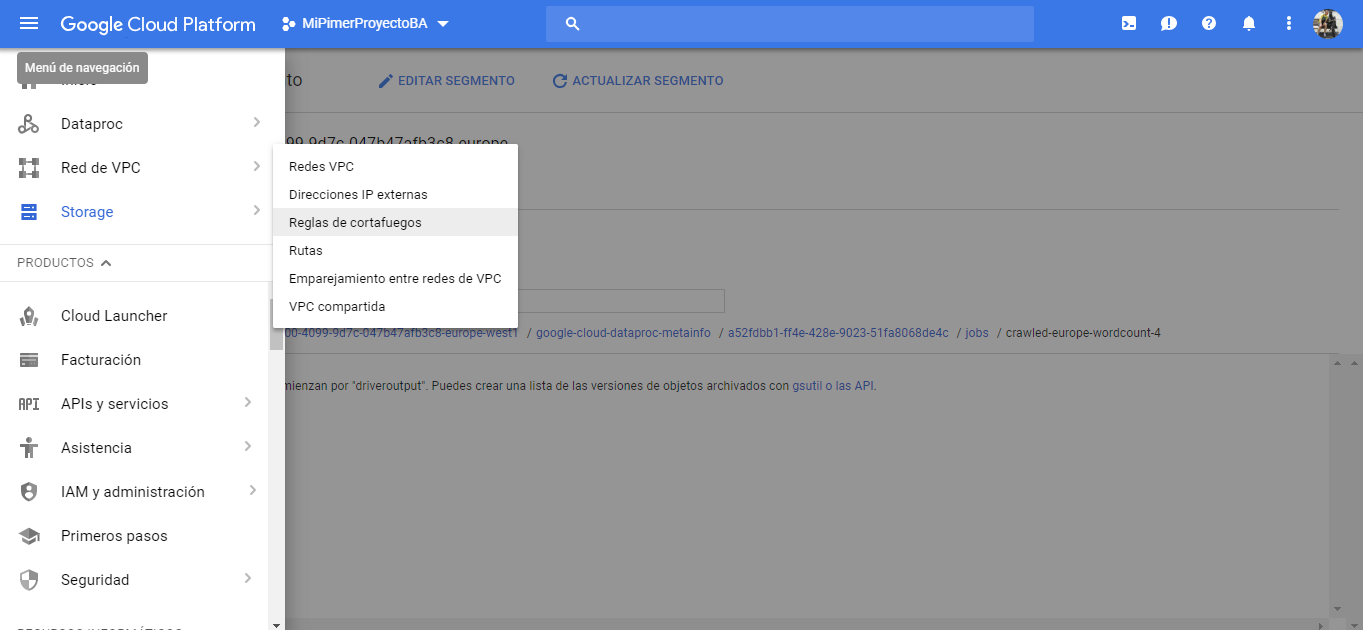
* 1 Máster: cluster-b6d5-m
* 2 Data nodes: cluster-b6d5-w-0 y cluster- b6d5-w-1



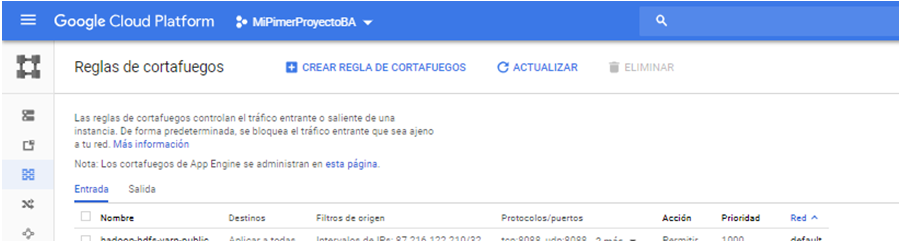
Nos metemos en cluster-b6d5-m y observamos los detalles de la instancia de VM de cluster-b6d5-m. Aquí tomamos el valor de la IP externa: *35.233.87.32* que nos permitirá entrar a la máquina de Yarn a través del puerto 9870 y también nos permitirá entrar a la máquina de Hadoop a través del puerto 8088.



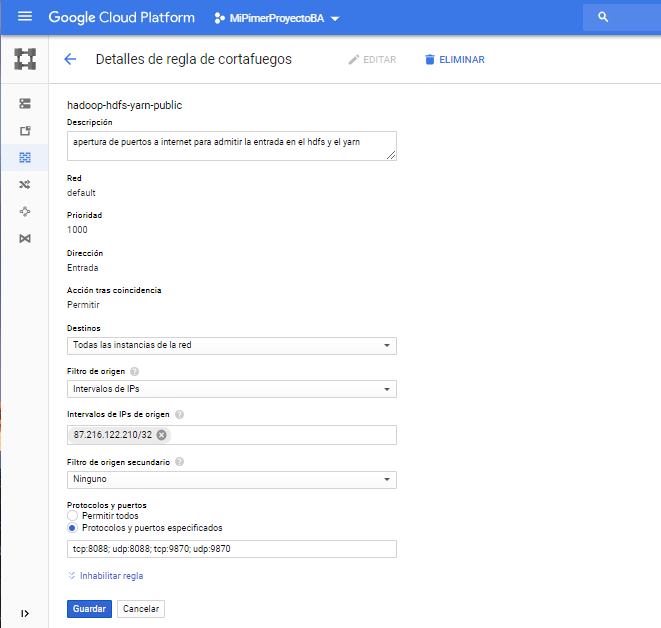
Vamos a configurar las reglas del cortafuego. Para ello nos vamos a: Menú > Red de VPC > Reglas de cortafuegos:



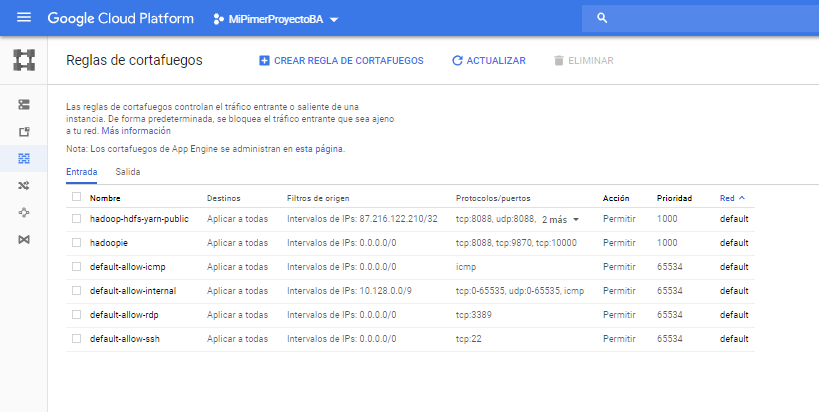
Pulsamos en el botón de CREAR REGLA DE CORTAFUEGOS:



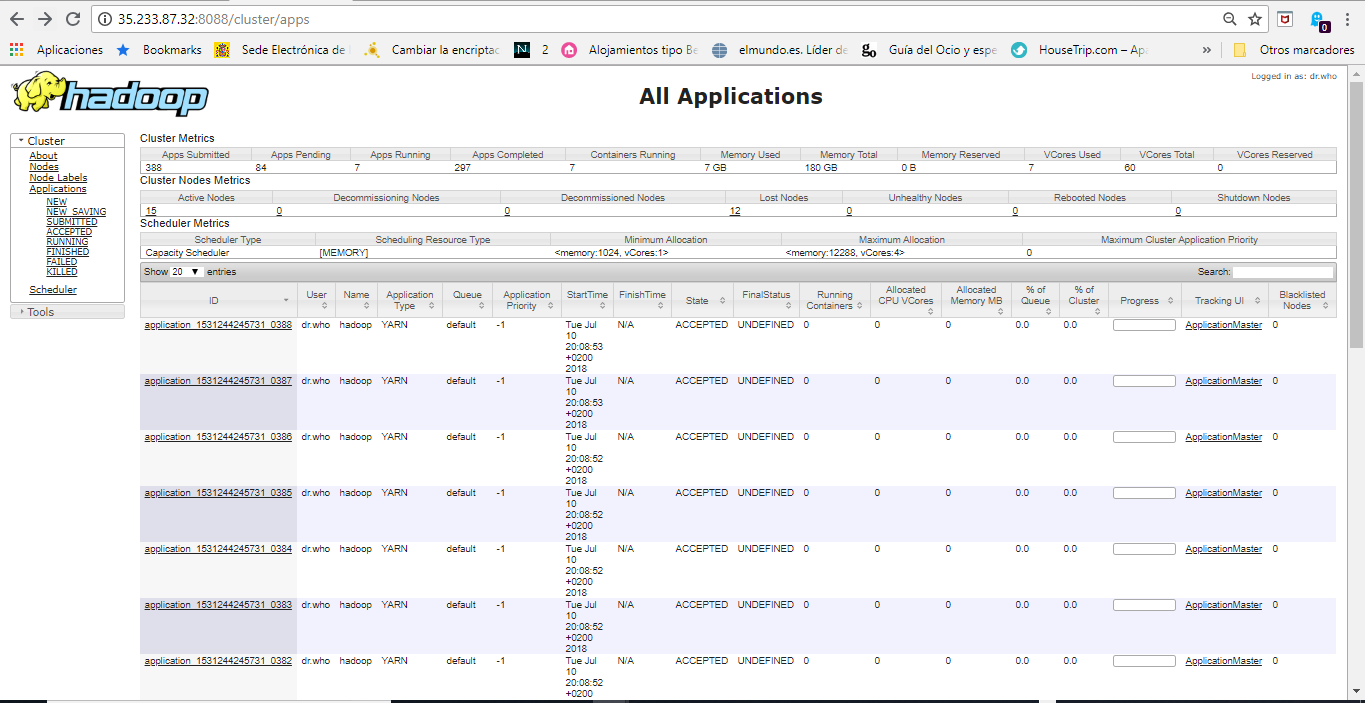
Creamos una nueva regla en el cortafuegos con nombre: hadoop-hdfs-yarn-public. También, definimos los Protocolos y puertos especificados: tcp:8088;udp:8088;tcp:9870;udp:9870. También, ponemos una descripción.



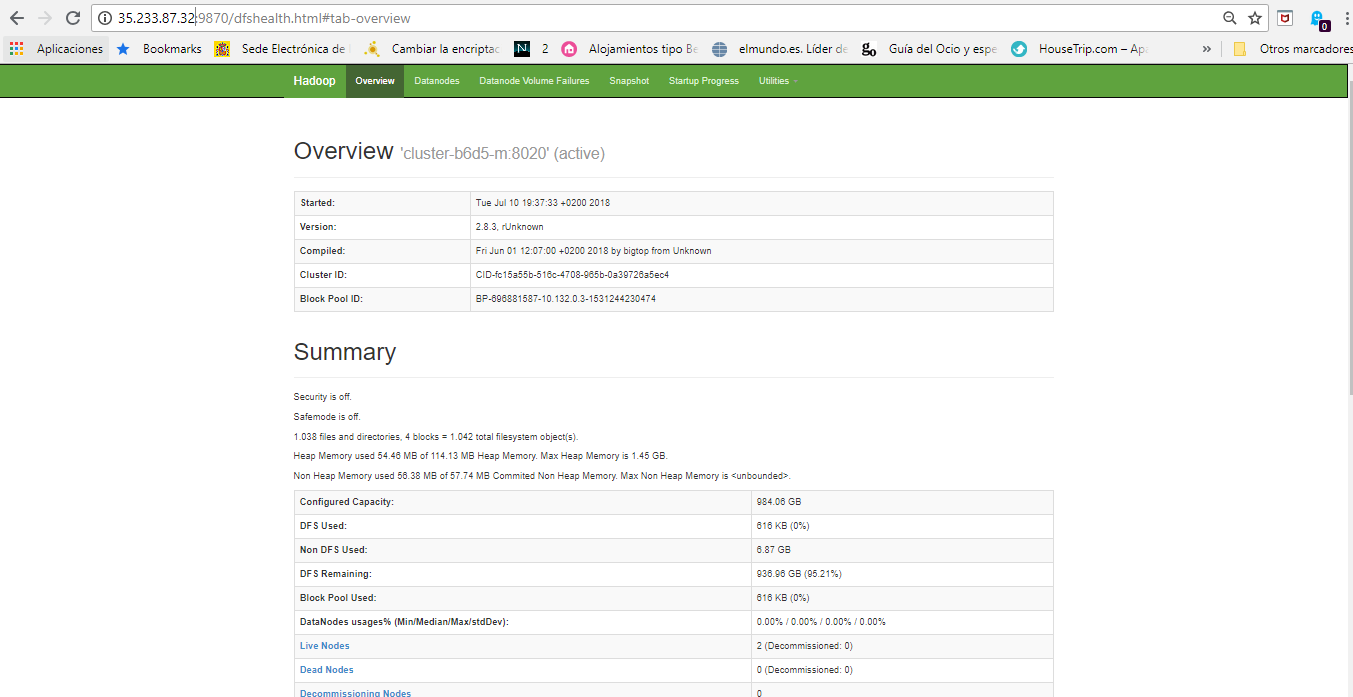
Observamos que se ha creado la nueva regla del cortafuegos:



Comprobamos que tenemos acceso a la máquina de Hadoop:



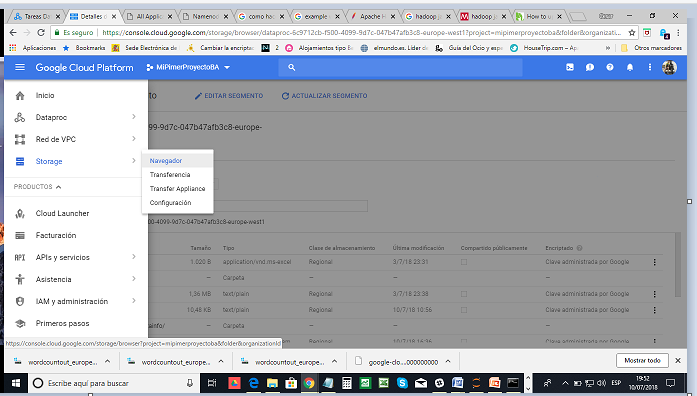
Comprobamos que tenemos acceso a la máquina de Yarn:



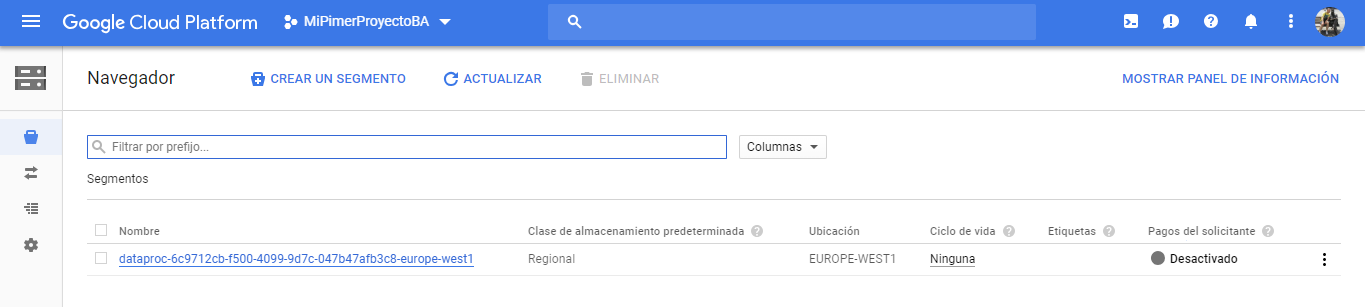
## Sprint 4

* Proveer los resultados de una tarea de procesamiento (wordcount por ejemplo) junto con los pasos para su ejecución, la copia de los resultados (incluidos los comandos de hadoop), y el log de la ejecución

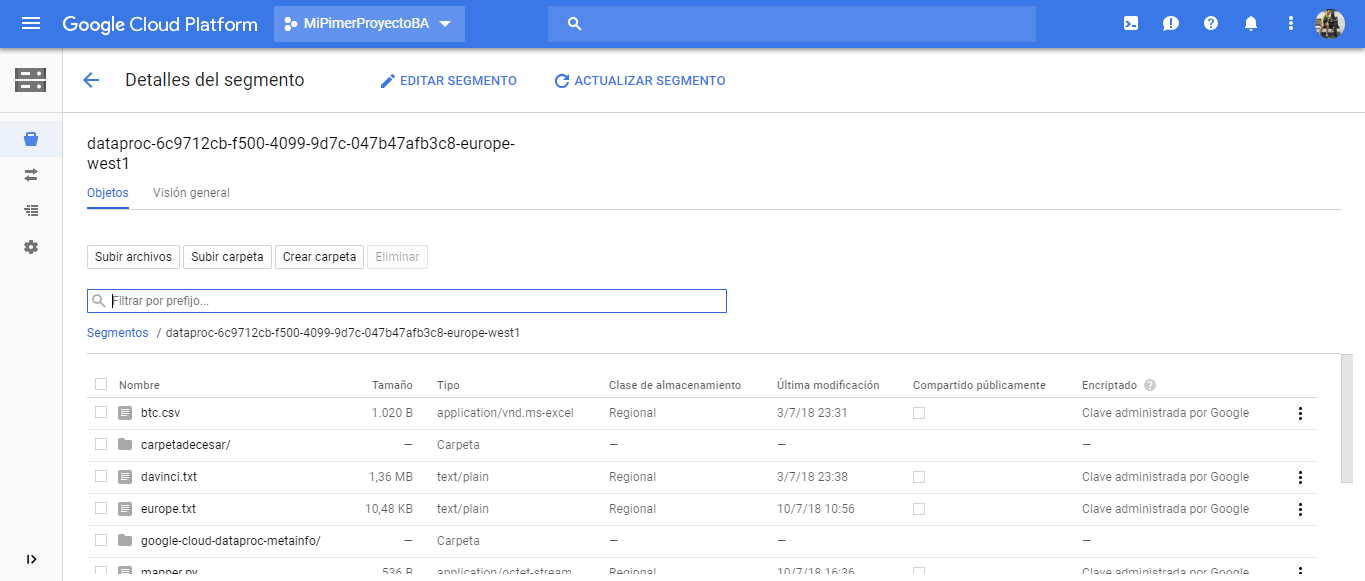
A continuación, nos vamos desde el menú y seguimos la siguiente ruta: Storage > Navegador.



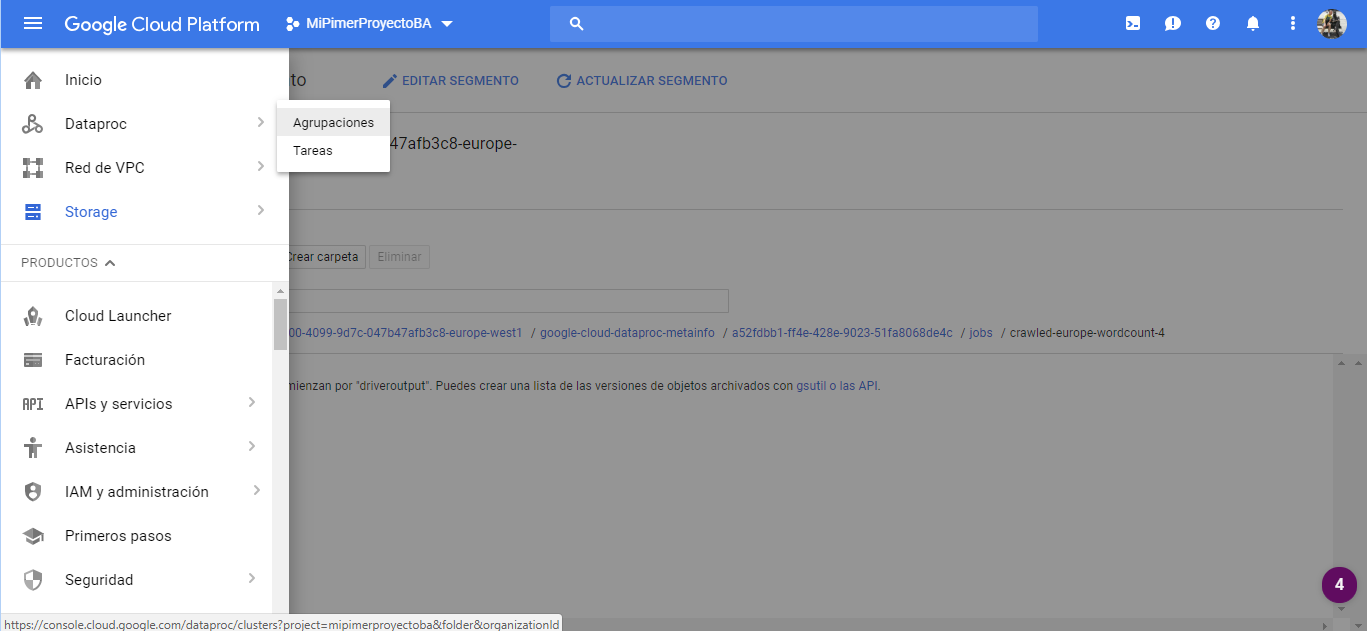
Aquí en el Navegador pulsamos en el nombre del segmento:



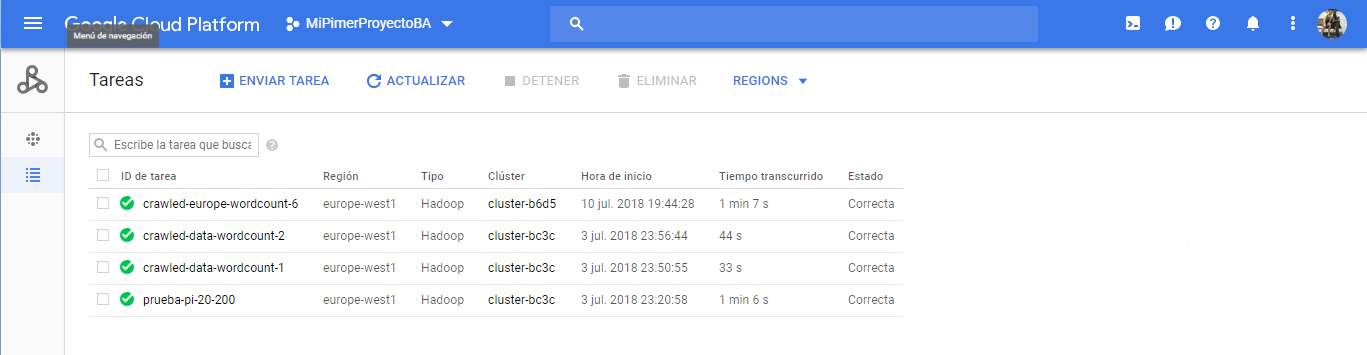
A continuación, subimos el archivo que hemos descargado de hacer el crawler y el scrapy con nombre: europe.txt



Ahora nos vamos a: Menú > Dataproc > Tareas



Creamos una nueva Tarea. Para ello pulsamos en el botón ENVIAR TAREA:



En la nueva tarea definimos: ID de tarea, Región, Clase principal o .jar, Argumentos. En la clase principal o .jar cogemos hadoop-mapreduce-examples.jar ya que tiene un ejemplo para hacer wordcount.

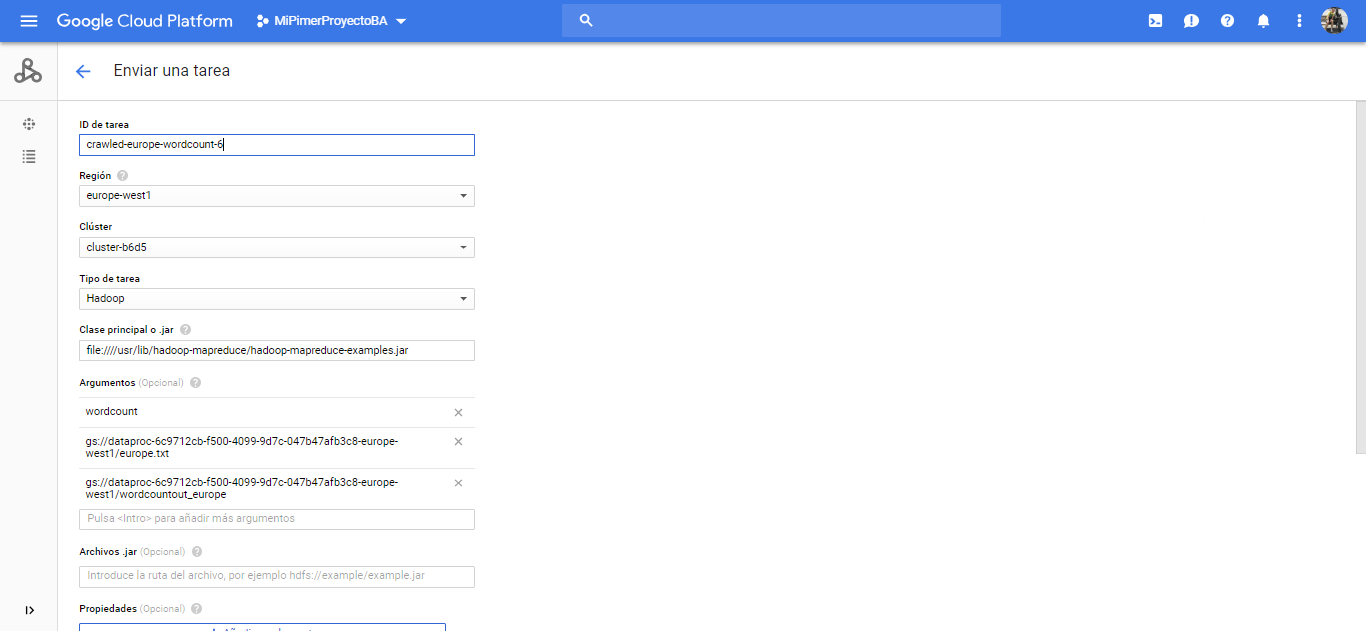
En Argumentos proporcionamos los siguientes argumentos:

1. wordcount (es un ejemplo incluido en hadoop-mapreduce-examples.jar que sirve para contar las palabras)
2. input:

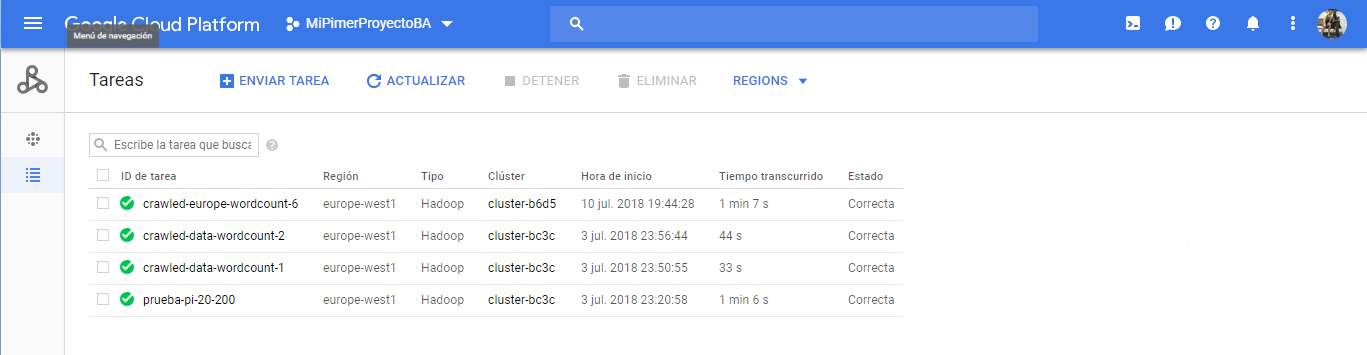
gs://dataproc-6c9712cb-f500-4099-9d7c-047b47afb3c8-europe-west1/europe.txt

1. output:

gs://dataproc-6c9712cb-f500-4099-9d7c-047b47afb3c8-europe-west1/wordcountout\_europe uw



Comprobamos que la tarea ha finalizado con éxito:



A continuación, propocionamos los logs del proceso de wordcount del fichero europe.txt:

18/07/10 17:44:30 INFO gcs.GoogleHadoopFileSystemBase: GHFS version: 1.6.7-hadoop2

18/07/10 17:44:31 INFO client.RMProxy: Connecting to ResourceManager at cluster-b6d5-m/10.132.0.3:8032

18/07/10 17:44:33 INFO input.FileInputFormat: Total input files to process : 1

18/07/10 17:44:33 INFO mapreduce.JobSubmitter: number of splits:1

18/07/10 17:44:33 INFO mapreduce.JobSubmitter: Submitting tokens for job: job\_1531244245731\_0053

18/07/10 17:44:34 INFO impl.YarnClientImpl: Submitted application application\_1531244245731\_0053

18/07/10 17:44:34 INFO mapreduce.Job: The url to track the job: http://cluster-b6d5-m:8088/proxy/application\_1531244245731\_0053/

18/07/10 17:44:34 INFO mapreduce.Job: Running job: job\_1531244245731\_0053

18/07/10 17:44:50 INFO mapreduce.Job: Job job\_1531244245731\_0053 running in uber mode : false

18/07/10 17:44:50 INFO mapreduce.Job: map 0% reduce 0%

18/07/10 17:45:03 INFO mapreduce.Job: map 100% reduce 0%

18/07/10 17:45:20 INFO mapreduce.Job: map 100% reduce 14%

18/07/10 17:45:21 INFO mapreduce.Job: map 100% reduce 29%

18/07/10 17:45:22 INFO mapreduce.Job: map 100% reduce 43%

18/07/10 17:45:28 INFO mapreduce.Job: map 100% reduce 71%

18/07/10 17:45:29 INFO mapreduce.Job: map 100% reduce 100%

18/07/10 17:45:31 INFO mapreduce.Job: Job job\_1531244245731\_0053 completed successfully

18/07/10 17:45:32 INFO mapreduce.Job: Counters: 55

File System Counters

FILE: Number of bytes read=7524

FILE: Number of bytes written=1329567

FILE: Number of read operations=0

FILE: Number of large read operations=0

FILE: Number of write operations=0

GS: Number of bytes read=10732

GS: Number of bytes written=5326

GS: Number of read operations=0

GS: Number of large read operations=0

GS: Number of write operations=0

HDFS: Number of bytes read=139

HDFS: Number of bytes written=0

HDFS: Number of read operations=1

HDFS: Number of large read operations=0

HDFS: Number of write operations=0

Job Counters

Killed reduce tasks=1

Launched map tasks=1

Launched reduce tasks=8

Rack-local map tasks=1

Total time spent by all maps in occupied slots (ms)=26892

Total time spent by all reduces in occupied slots (ms)=403449

Total time spent by all map tasks (ms)=8964

Total time spent by all reduce tasks (ms)=134483

Total vcore-milliseconds taken by all map tasks=8964

Total vcore-milliseconds taken by all reduce tasks=134483

Total megabyte-milliseconds taken by all map tasks=27537408

Total megabyte-milliseconds taken by all reduce tasks=413131776

Map-Reduce Framework

Map input records=248

Map output records=1690

Map output bytes=17366

Map output materialized bytes=7524

Input split bytes=139

Combine input records=1690

Combine output records=545

Reduce input groups=545

Reduce shuffle bytes=7524

Reduce input records=545

Reduce output records=545

Spilled Records=1090

Shuffled Maps =7

Failed Shuffles=0

Merged Map outputs=7

GC time elapsed (ms)=1943

CPU time spent (ms)=13430

Physical memory (bytes) snapshot=2501394432

Virtual memory (bytes) snapshot=35207237632

Total committed heap usage (bytes)=1939865600

Shuffle Errors

BAD\_ID=0

CONNECTION=0

IO\_ERROR=0

WRONG\_LENGTH=0

WRONG\_MAP=0

WRONG\_REDUCE=0

File Input Format Counters

Bytes Read=10732

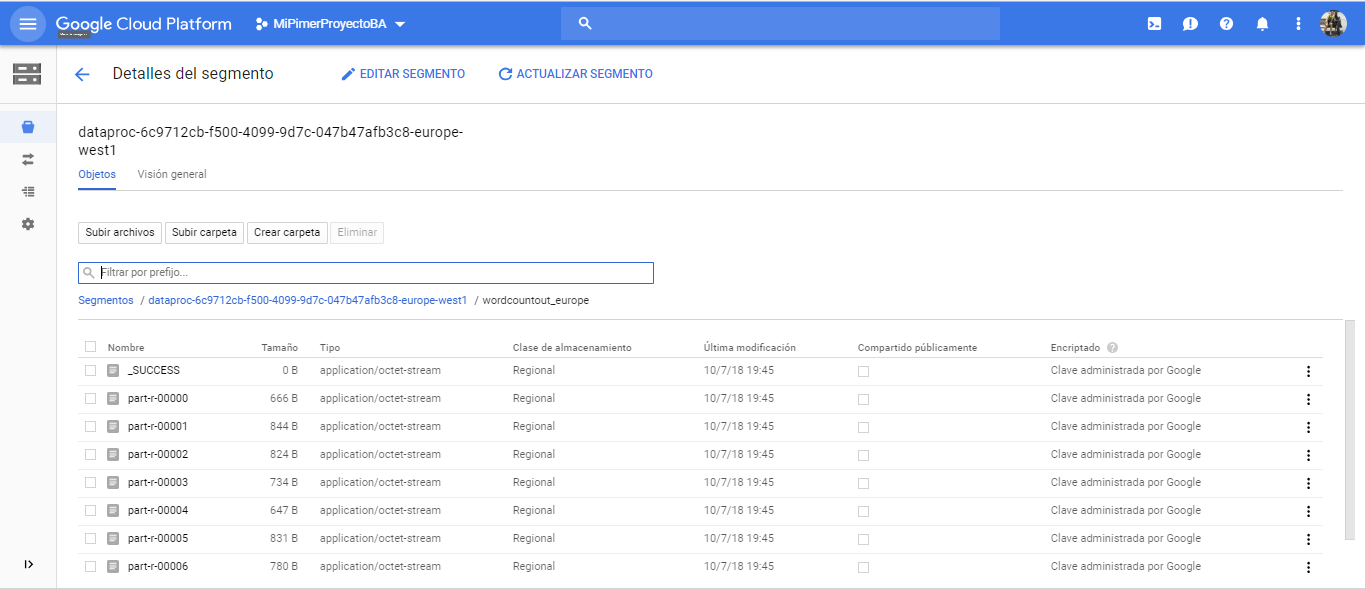
File Output Format Counters

Bytes Written=5326

Salida de tarea completa

Nos metemos en la carpeta que se ha generado para ver los ficheros que se han generado con el wordcount. Para ello seguimos la siguiente ruta:

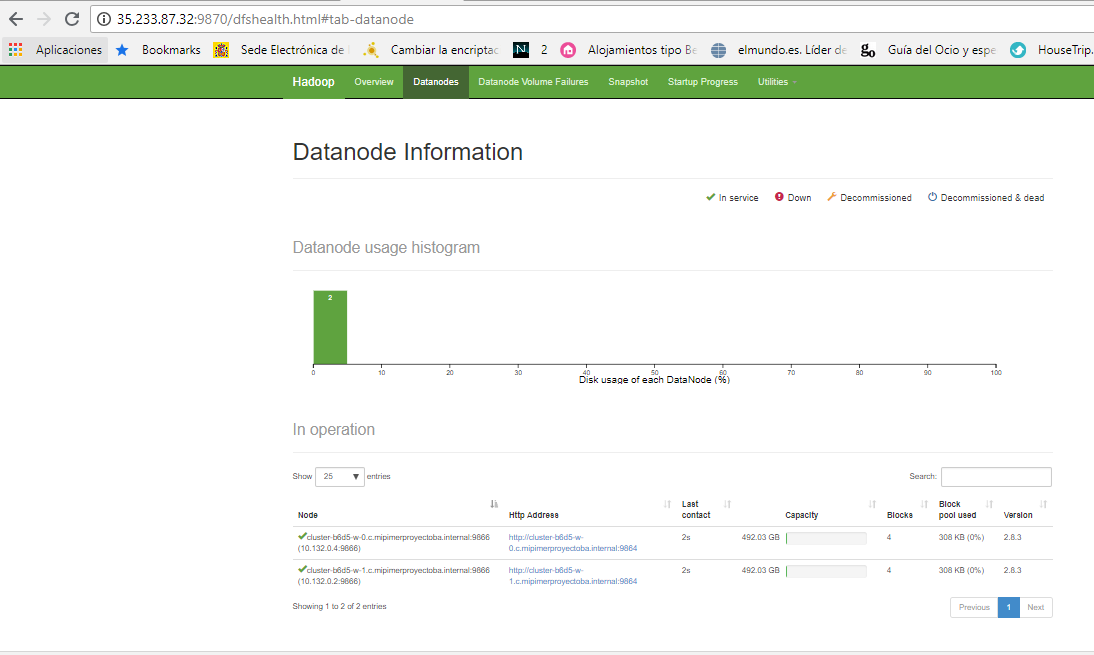
Storage > Navegador > dataproc-6c9712cb-f500-4099-9d7c-047b47afb3c8-europe-west1 > wordcount\_europe



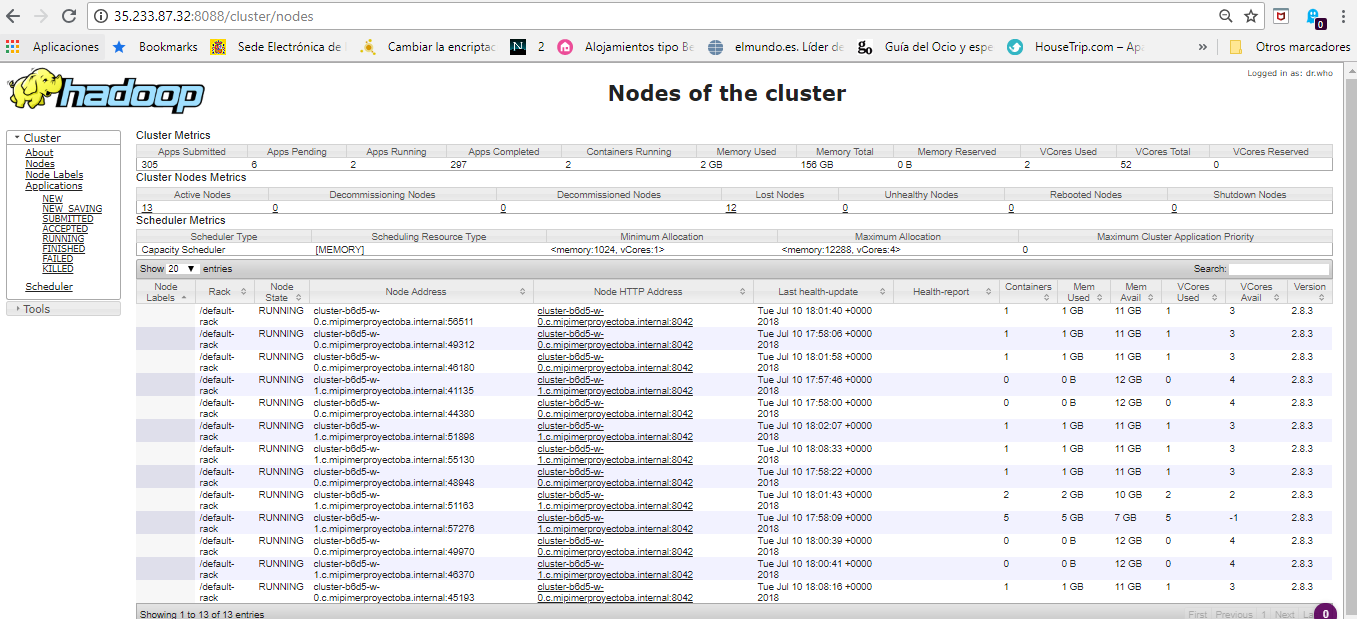
En el siguiente link se encuentran los ficheros que se han generado al hacer el wordcount:

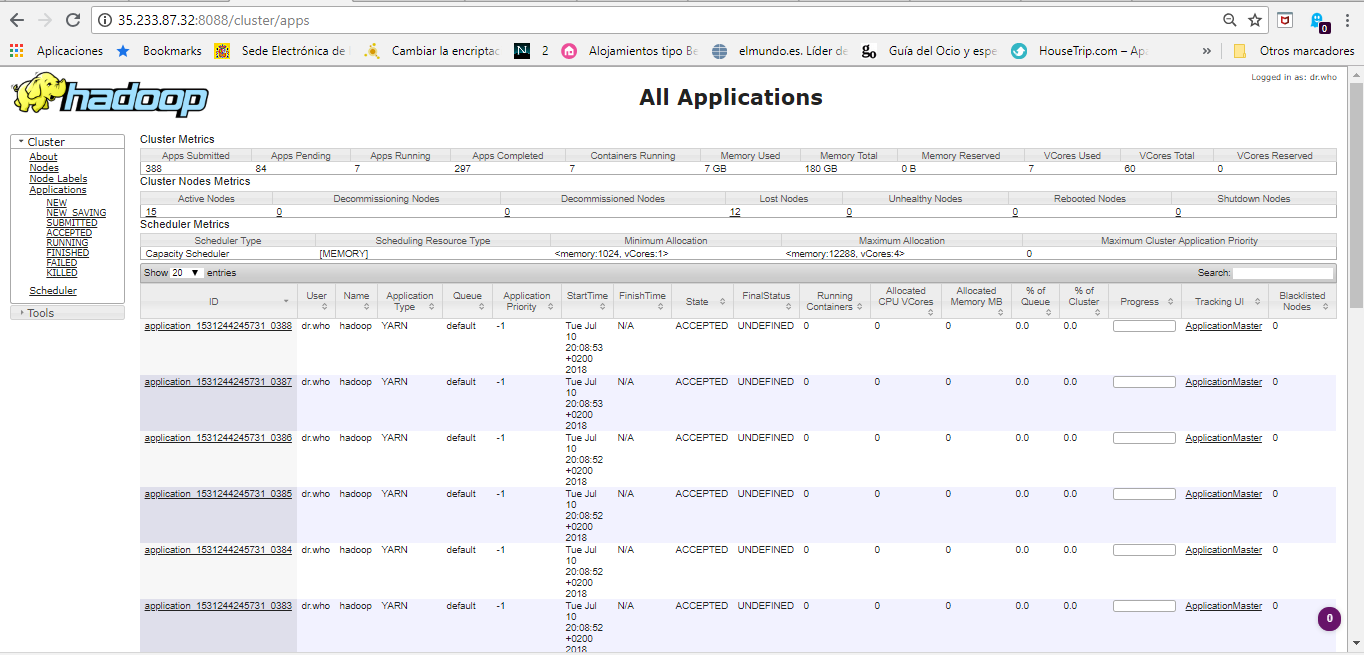
<https://drive.google.com/open?id=1Kp5PgmKTkbE-y63w36y_DSQ51ATydGUm>

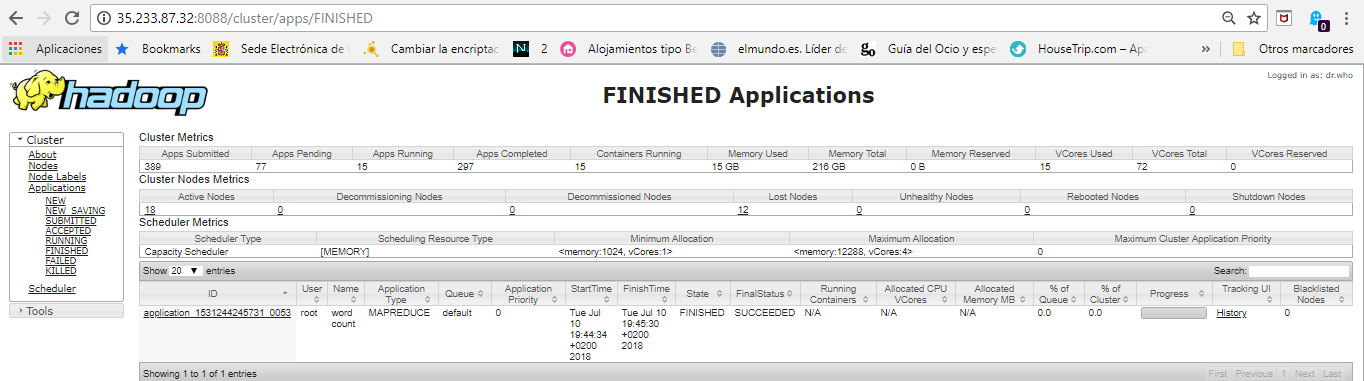
Comprobamos que se han generado los Datanodes en el YARN:

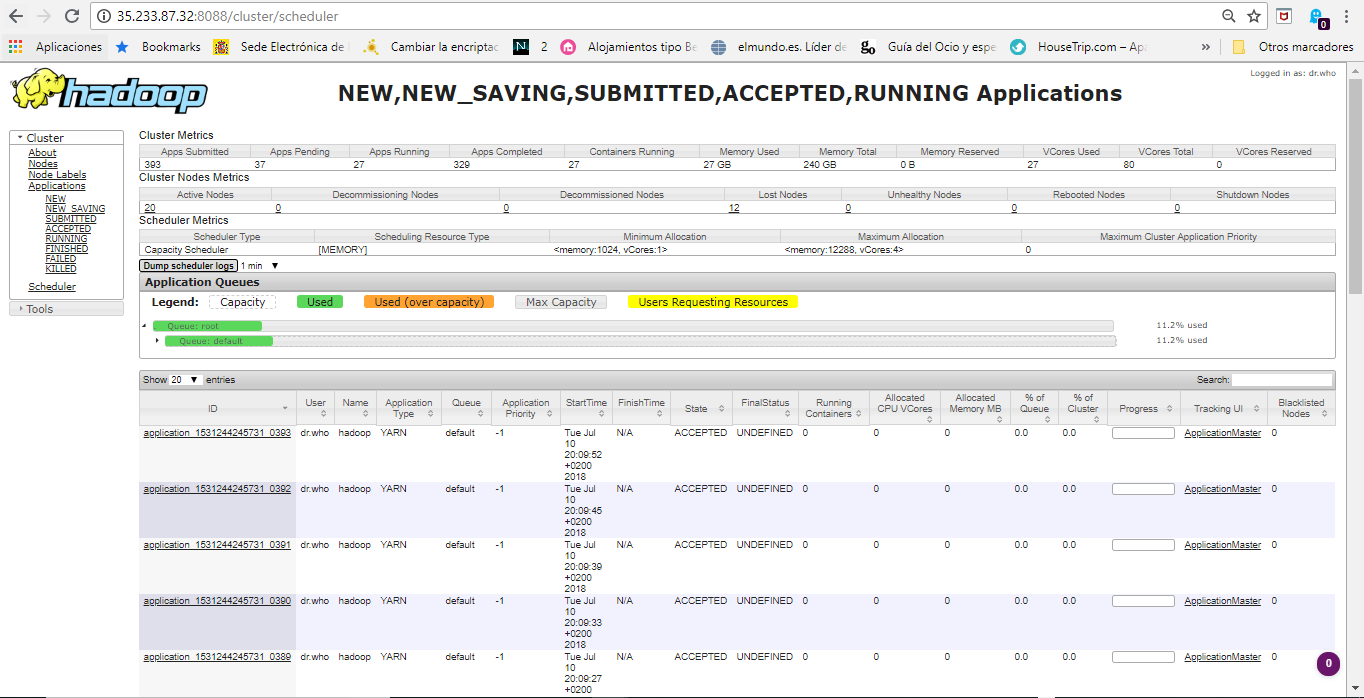


También, en la Hadoop observamos que se han generado los Data Nodes





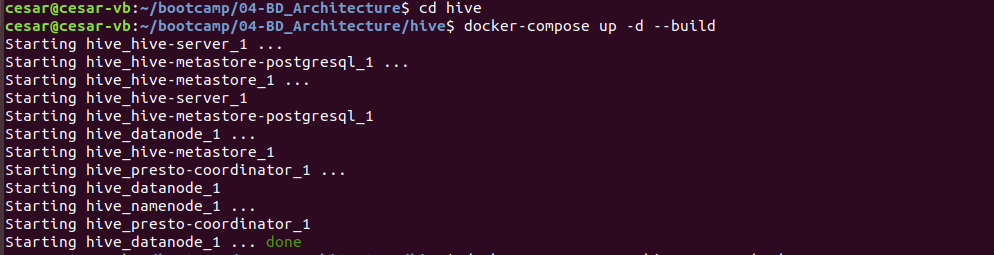




## Sprint 5

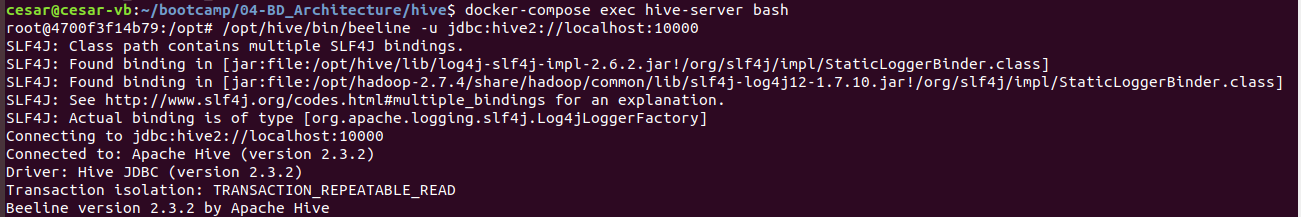
* Utilizar HIVE para categorizar los datos y hacer un par de queries con el command line para extraer datos a un fichero.

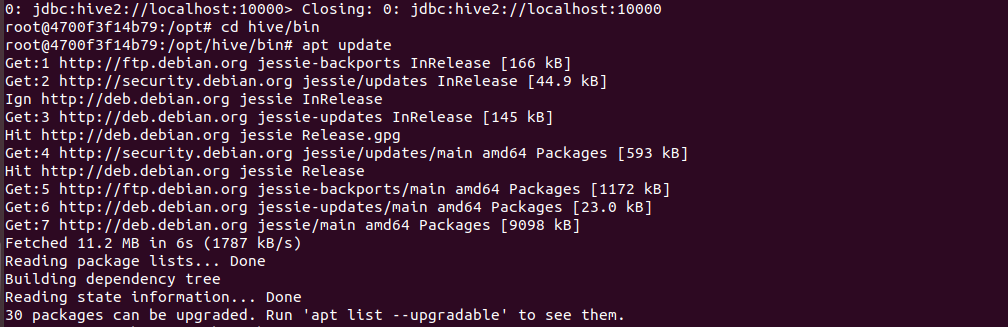
Cambiamos a la carpeta hive y levantamos un docker-compose:



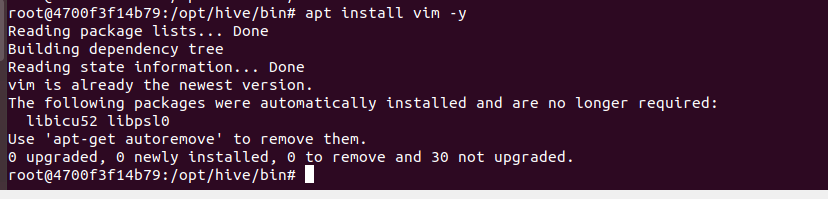
Entramos en la bash de hive a través del comando: docker-compose exec hive-server bash.

A continuación, entramos en el cliente llamado Beeline (/opt/hive/bin/beeline -u jdbc:hive2://localhost:10000) para conectarse con Hive a través del puerto 10000, que normalmente se encuentra instalado en el servidor Hive. Ya dentro de aquí, se pueden lanzar queries como si estuviéramos en MySQL.

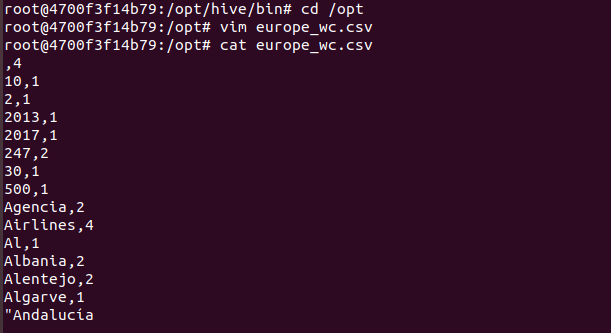


Salimos del Beeline. Como esta máquina no tiene Vim, tenemos que hacer un: apt update. 

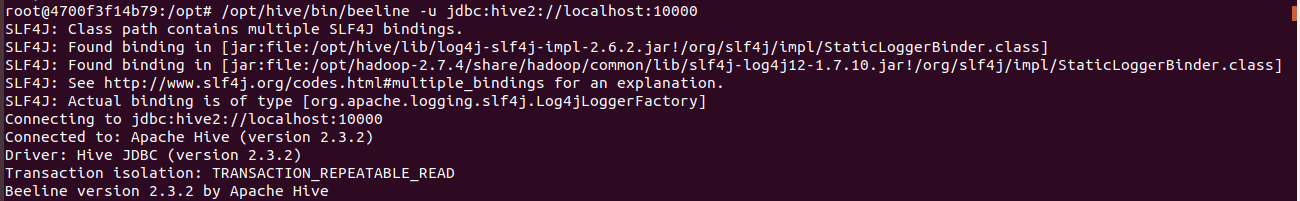
A continuación instalamos VIM:



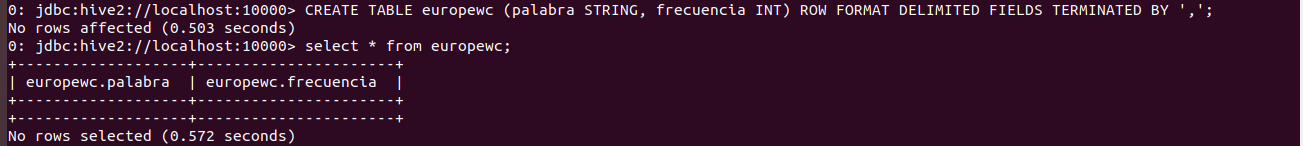
Ahora creamos un archivo *europe\_wc.csv* dentro del servidor, que lo vamos a llenar con los datos del word count que hemos generado previamente:



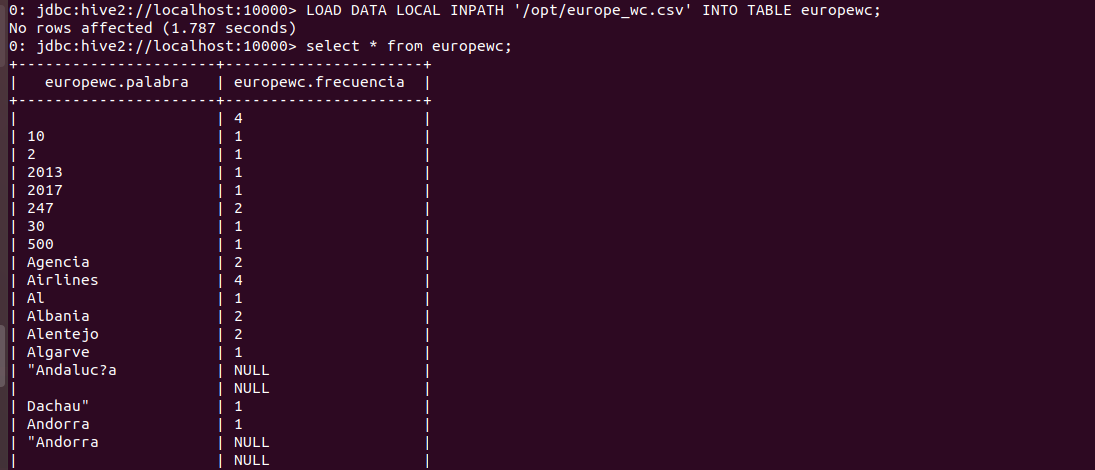
Nos volvemos a conectar al cliente de Hive de nombre Beeline:



Creamos la tabla, en HIVE tenemos que indicarle cual es el delimitador que va a ser la coma:



Ahora hacemos un *LOAD DATA LOCAL INPATH ‘/opt/europe\_wc.csv ’ INTO TABLE* *europewc*. Aquí, estamos indicando con *LOAD DATA LOCAL INPATH* el lugar donde están los datos (subrayado en amarillo) le indicamos en que tabla lo queremos meter, en este caso *europewc* (subrayado en verde):

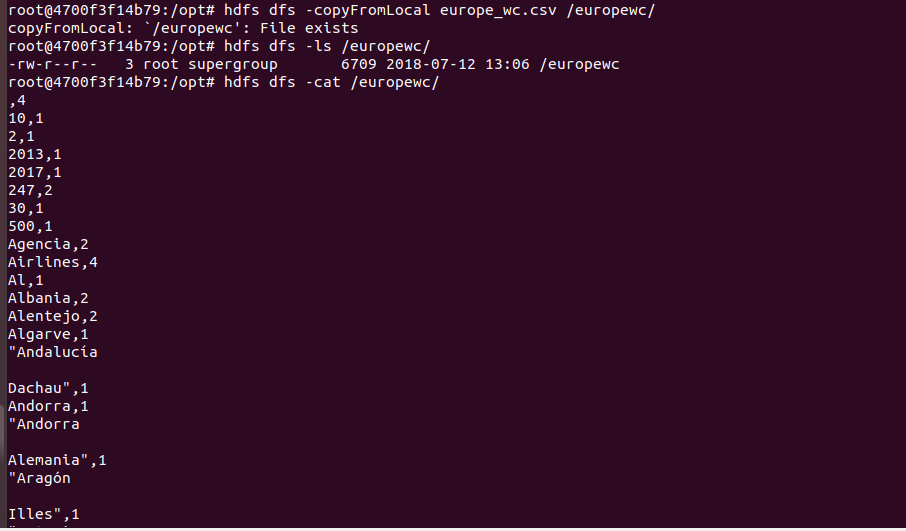


Ahora lo que vamos a hacer una ingesta desde *HDFS.* Para ello nos salimos de Beeline y escribimos el siguiente comando:

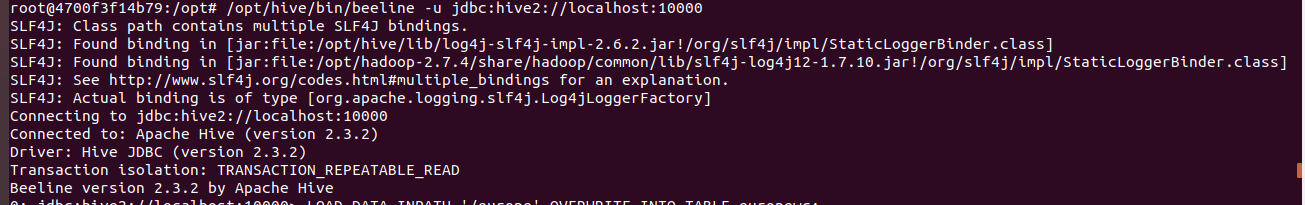


Aquí, lo que está haciendo es haciendo una copia del archivo europe\_wc.csv a la carpeta distribuida de hdfs europewc.

Vemos en Hadoop si se ha creado la carpeta *europewc* y observamos el contenido de dicha carpeta:



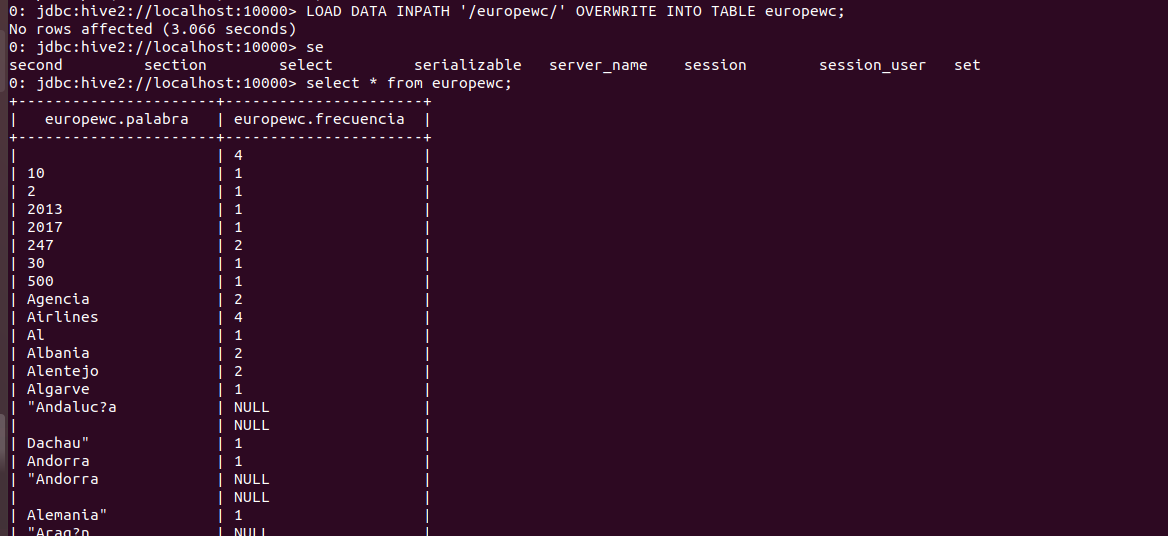
Volvemos al Beeline:



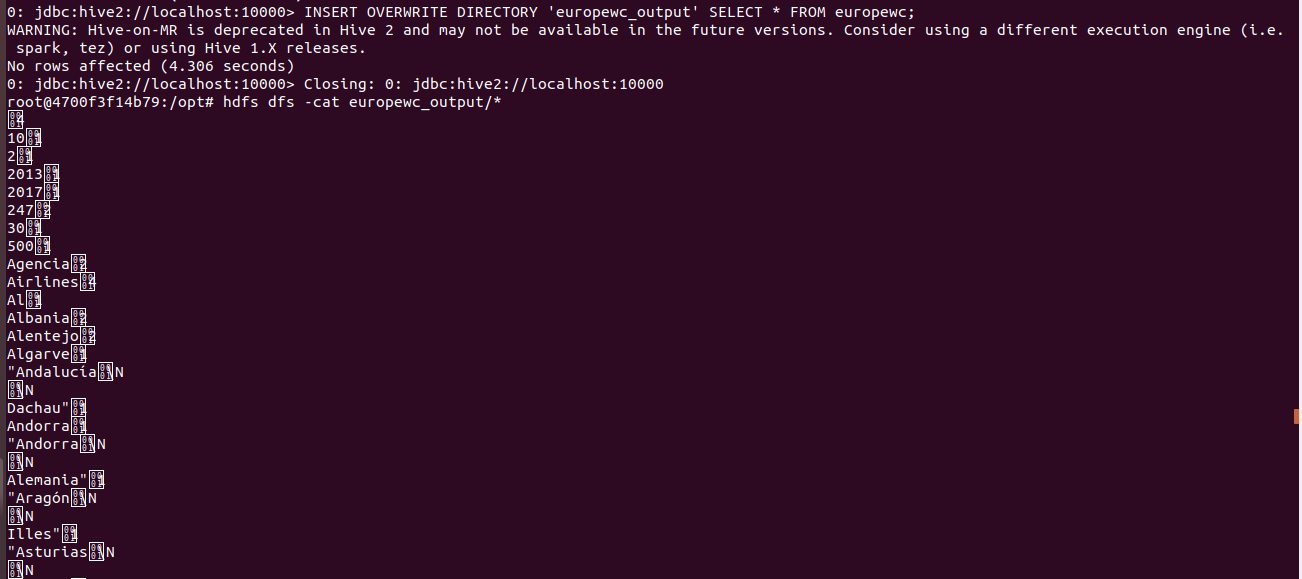
A continuación lanzamos el siguiente comando en el cual decimos donde están los datos, en este caso no es local y especificamos el path (subrayado en color amarillo):

LOAD DATA INPATH '/europewc/' OVERWRITE INTO TABLE alumnos;

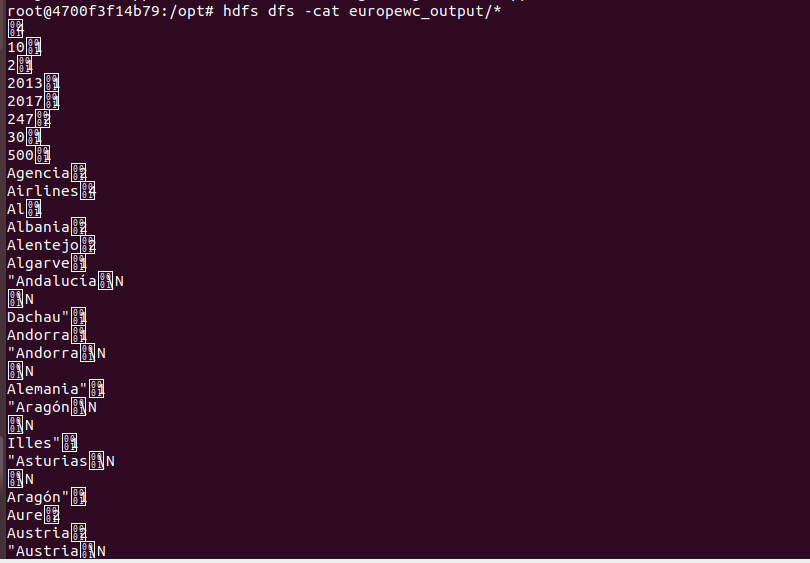
Al no tener el LOCAL, la manera que le estamos diciendo indirectamente que busque los datos desde el HDFS:



También desde el Beeline, podemos hacer un INSERT extrayendo los datos de un SELECT a un fichero del HDFS. Aquí lo que estamos haciendo es insertar un directorio nuevo del HDFS ‘*europewc-output’* y estamos metiendo dentro de este directorio, el resultado de la query *SELECT\* FROM europewc:*



Ahora nos salimos y hacemos un: *hdfs dfs –cat europewc\_output/\*:*

**

Observamos que me ha quitado el delimitador y usa el que él quiere.

Ahora lo vamos a hacer, es volver a Beeline y lanzamos el mismo pero cuando vaya a escribir las filas los va a separar por comas en lugar del separador que emplea HIVE por defecto. Nos salimos de Beeline y observamos que los datos están separados por comas:

