**PRÁCTICA FINAL**

**BIG DATA ARCHITECTURE**

# **Índice de Contenidos**

1. Índice de Contenidos · - ·· - · - ·· - · - ·· - · - ·· - · - ·· - 1
2. Datos de la Práctica · - ·· - · - ·· - · - ·· - · - ·· - · - ·· - 2
3. Introducción y Objetivo · - ·· - · - ·· - · - ·· - · - ·· - · - ·· - 2
4. Origen Principal & Tratamiento Inicial · - ·· - · - ·· - · - ·· - · - ·· - · - ·· - 3
5. Enriquecimiento del Origen Principal · - ·· - · - ·· - · - ·· - · - ·· - · - ·· - 6
6. Diagrama de la Arquitectura del Sistema · - ·· - · - ·· - · - ·· - · - ·· - · - ·· - 7
7. Descripción de la Arquitectura · - ·· - · - ·· - · - ·· - · - ·· - · - ·· - 8
8. Despliegue en Cloud · - ·· - · - ·· - · - ·· - · - ·· - · - ·· - 9
   1. Creación del Cluster de Trabajo y del Segmento de Almacenamiento - ·· - · - ·· - · - ·· - · - ·· - · - ·· - · - ·· - · - 9
   2. Creación de Reglas de Cortafuegos - ·· - · - ·· - · - ·· - · - ·· - · - ·· - · - ·· - · - 10
   3. Operación de Staging - ·· - · - ·· - · - ·· - · - ·· - · - ·· - · - ·· - · - 11
   4. Creación de Tablas HIVE - ·· - · - ·· - · - ·· - · - ·· - · - ·· - · - ·· - · - 11
   5. Carga de Datos en el Modelo - ·· - · - ·· - · - ·· - · - ·· - · - ·· - · - ·· - · - 13
   6. Explotación y Productivización - ·· - · - ·· - · - ·· - · - ·· - · - ·· - · - ·· - · - 13
9. Agradecimientos · - ·· - · - ·· - · - ·· - · - ·· - · - ·· - 14

# **Datos de la Práctica**

|  |  |
| --- | --- |
| **Bootcamp IV Big Data & Machine Learning - KeepCoding** | |
| **Asignatura** | Big Data Architecture |
| **Profesor** | Ricardo Vegas |
| **Alumno** | Francisco Javier Gonzálvez Chico |
| **Título** | Los Mejores Alquileres en Madrid Capital |

# **Introducción y Objetivo**

El objetivo de este trabajo es diseñar, arquitecturizar y definir, dentro de mis posibilidades, un sistema para recibir, almacenar y procesar un dataset de datos.

A grandes rasgos el sistema recibirá los datos indicados en el enunciado de la práctica como origen, se enriquecerá con una agregación proveniente de un proceso de crawling, se cargará en un cluster de Hadoop y se productivizará mediante consultas, conteos e informes y cuadros de mando.

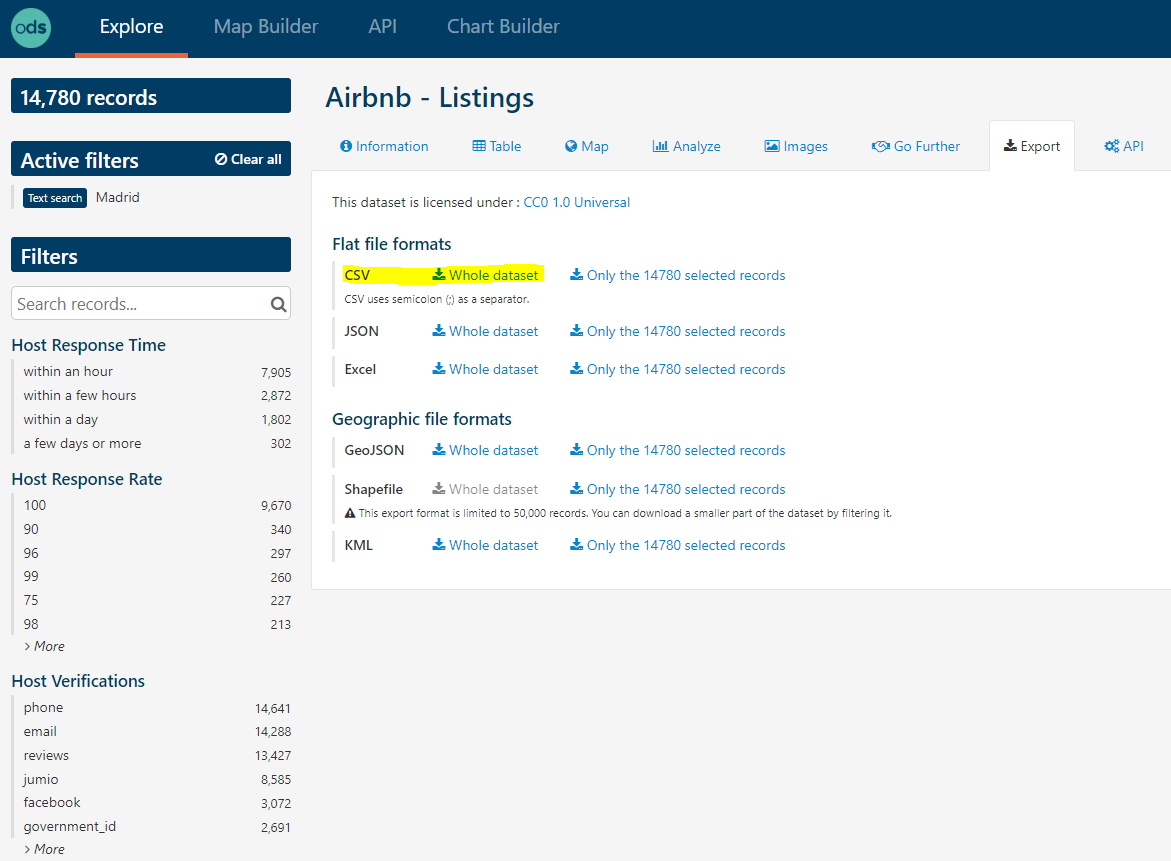
Me he tomado algunas pequeñas licencias, a la hora de confeccionar esta práctica final, con el deseo y la intención de que su presentación fuera lo más adecuada posible: por ejemplo, desordenar la estructura predefinida para los apartados de este trabajo; también he querido utilizar ***Microsoft*** ***Word*** como herramienta para documentar y presentar mi labor al profesor; y por último, he optado por no construir y/o desplegar algunas partes de la arquitectura por falta de tiempo (y también de conocimientos suficientes, todo sea dicho) entendiendo que será suficiente realizar de ellas una descripción ya que se marcaron como ejercicios opcionales.

# **Origen Principal & Tratamiento Inicial**

Utilizaremos un fichero procedente de **AIRBNB**, con datos de los inmuebles disponibles, que descargaremos manualmente de este enlace:

<https://public.opendatasoft.com/explore/dataset/airbnb-listings/export/?disjunctive.host_verifications&disjunctive.amenities&disjunctive.features&q=Madrid&dataChart=%3D&location=16,41.38377,2.15774&basemap=jawg.streets>

La descarga a elegir será la del enlace resaltado en la imagen de la página.



Lo primero que haremos será filtrar y limpiar el archivo descargado que contiene 494.954 (casi medio millón de) filas.

Para esta tarea utilizaremos ***Trifacta Wrangler***, una herramienta de *Data Cleansing* para corregir colecciones de datos y aumentar su calidad de cara a mejor aprovechamiento y productivización. ***Trifacta*** permite guardar la “receta” de limpieza, corrección y saneo de los datos para poder ser utilizada a posteriori en un proceso productivo planificado.

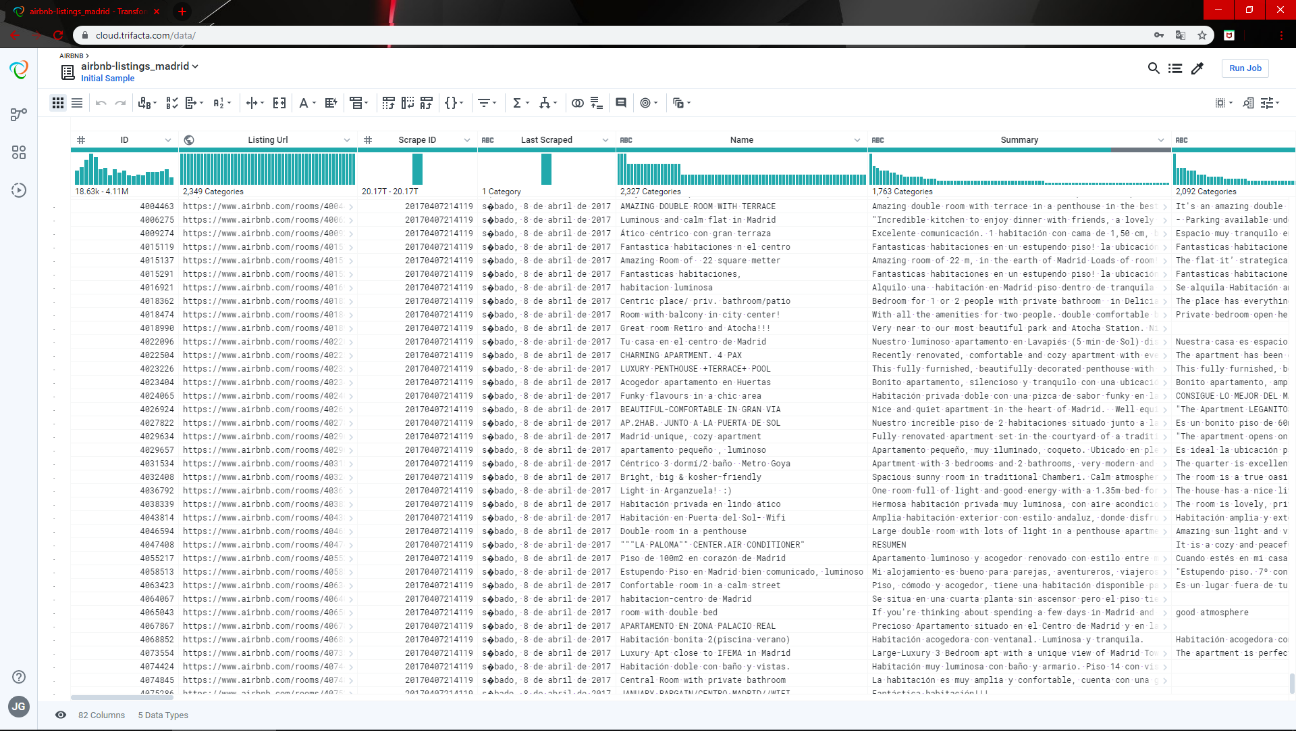
Cargaremos el archivo total de **airbnb** y realizaremos el siguiente filtrado y selección de filas:

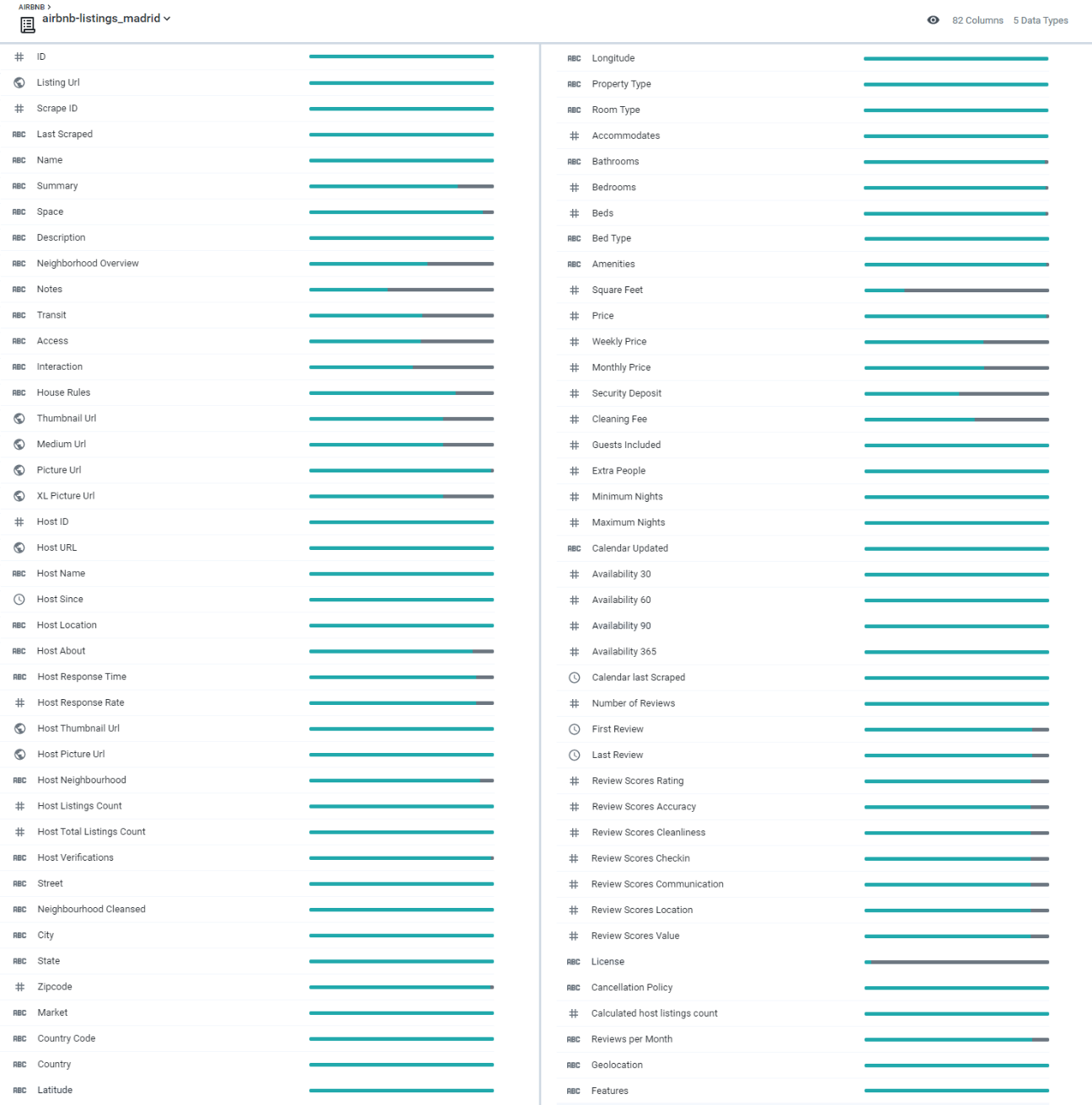
1. Una primera selección sobre la columna **City = “*Madrid*”**, ya que para nuestro diseño vamos a utilizar únicamente el subset de datos de Madrid Capital (descartando incluso las filas con inmuebles en pueblos cercanos, como “***Aravaca***” o “***Las Matas***”). Tras este primer filtrado nos quedan 13.243 filas.
2. En la columna **Neighbourhood Cleansed** observamos filas con más valores que no nos parecen correctos para nuestro subset de Madrid Capital: [“***Aravaca***”, “***Casco Histórico de Barajas***”, “***El Plantío***” y “***El Goloso***”]. Decidimos descartar también las 50 filas que detectamos con esta casuística, obteniendo 13.193 válidas hasta ahora.
3. Detectamos una fila con el valor “***28850***” en la columna **Zipcode**, siendo este código postal el de “***Torrejón de Ardoz***”, decidimos prescindir de dicha fila. El dataset nos queda con 13.192 filas válidas.
4. Suprimimos seis filas más porque no proporcionan valores en ninguna de las tres columnas que marcan los precios de alquiler del inmueble: **Price**, **Weekly Price** y **Monthly Price**. Que exista alguno de estos tres datos nos parece imprescindible para un sistema de análisis y explotación de datos de alquileres de inmuebles. El dataset queda con 13.186 filas válidas.

Procederemos a continuación con el siguiente tratamiento de limpieza:

1. Eliminamos cuatro columnas que en nuestro subset de datos están vacías: **Experiences** **Offered**, **Host** **Acceptance** **Rate**, **Has** **Availability** y **Jurisdiction Names**.
2. Suprimimos dos columnas más porque no proporcionan información útil para nuestro propósito al duplicar otra, que consideramos, más válida: descartamos **Neighbourhood** y **Neighbourhood** **Group** **Cleansed** porque **Neighbourhood Cleansed** tiene la misma información con más calidad.
3. Normalizamos todos los valores de las columnas **City** y **State** a “***Madrid***” y “***Comunidad de Madrid***” respectivamente.
4. Eliminamos otra columna que tampoco proporciona información útil para nuestro fin al duplicar otra con mejor información: prescindimos de **Smart Location** que duplica **State.**
5. Vamos a realizar un saneo de datos sobre los valores de la columna **Zipcode**, reparando los valores incorrectos, siempre que sea posible, o vaciándolos:
   1. Sustituimos los valores “***-***” y “***28***” por “”.
   2. Sustituimos los valores “***28002/n28002***” por “***28002***”.
   3. Sustituimos los valores “***2804***”, “***27004***” y “***Madrid 28004***” por “***28004***”.
   4. Sustituimos los valores “***2805***” y “***28105***” por “***28005***”.
   5. Sustituimos los valores “***25008***” por “***28008***”.
   6. Sustituimos los valores “***2802/n28012***” por “***28012***”.
   7. Sustituimos los valores “***20013***”, “***27013***” y “***280013***” por “***28013***”.
   8. Sustituimos los valores “***2015***” y “***2815***” por “***28015***”.
   9. Sustituimos los valores “***28126***” por “***28026***”.
   10. Sustituimos los valores “***28051/n28051***” por “***28051***”.

El dataset saneado, limpio y definitivo constará de 82 columnas (con sus histogramas sin errores) como se puede apreciar en las imágenes adjuntas.



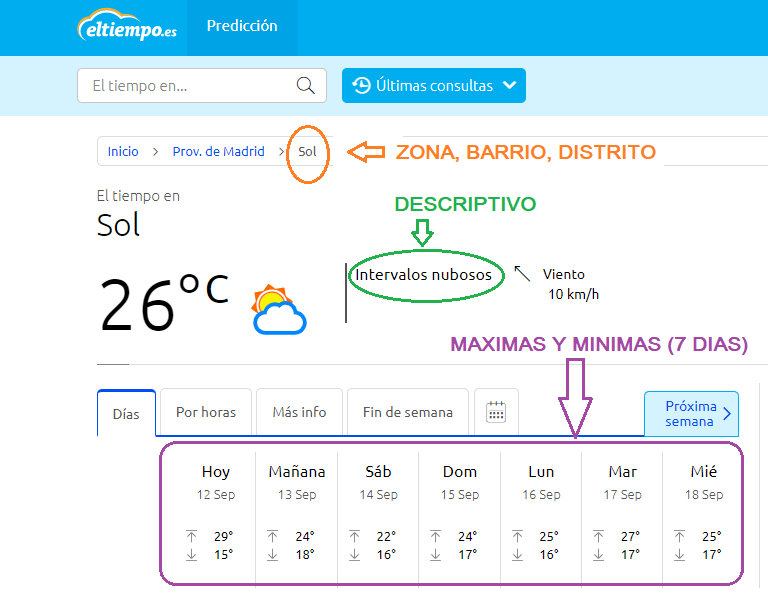


# **Enriquecimiento del Origen Principal**

Se enriquecerá el fichero de datos principal realizando un crawling con scrappy (implementado en Python) sobre la web de datos meteorológicos [https://www.eltiempo.es](https://www.eltiempo.es/)

De esta web se recuperarían, para todas las zonas de Madrid Capital, los siguientes datos para la fecha de la extracción y seis días posteriores:

* La zona, barrio o distrito.
* Las temperaturas máxima y mínima.
* El descriptivo “Soleado”, “Despejado”, “Poco Nuboso”, etc.

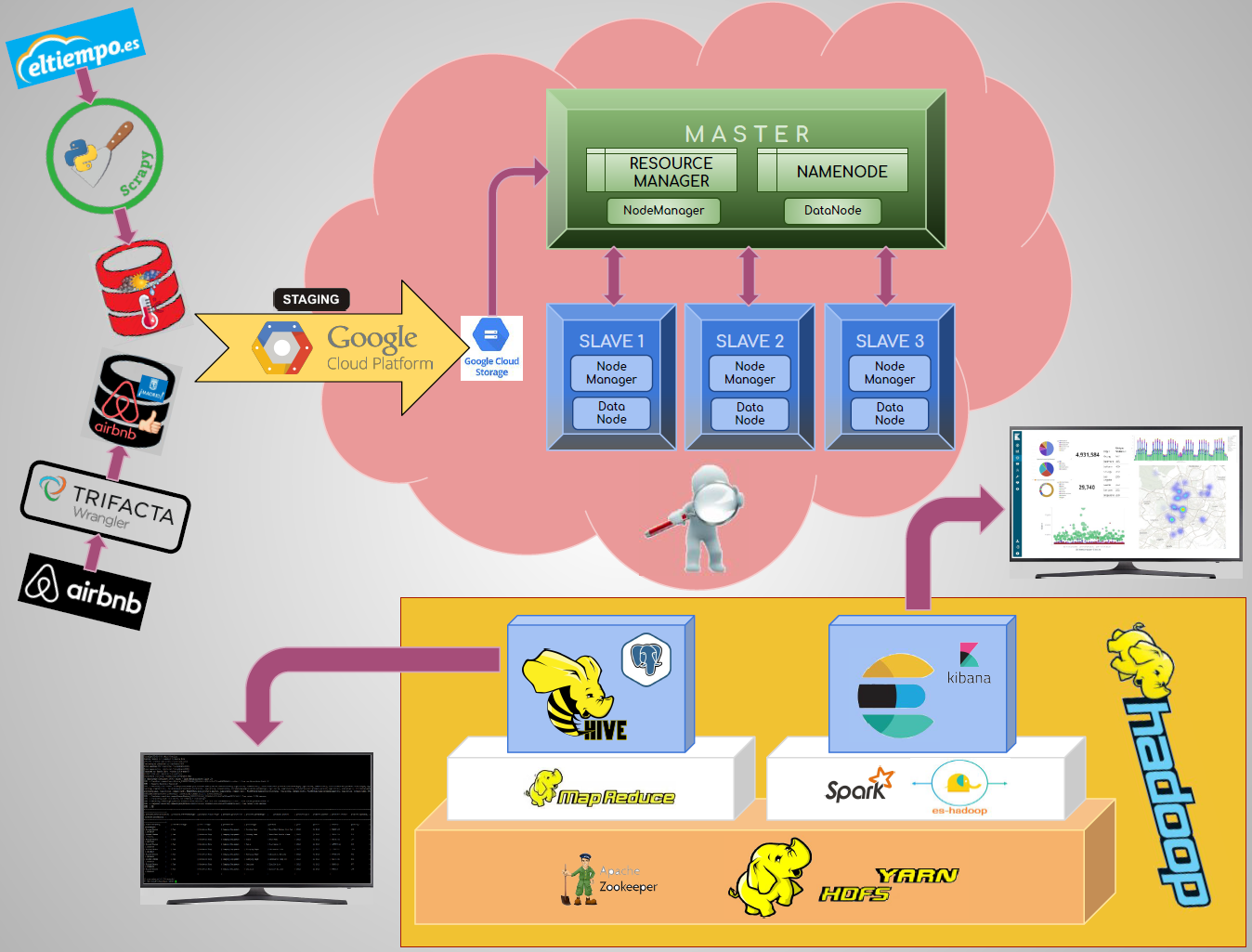


Consideramos estos datos muy interesantes para el target de clientes interesados en alquilar un inmueble a corto plazo en Madrid y se podrían mostrar como agregación en los resultados de las búsquedas que estos usuarios realizaran sobre el subset de datos.

***NOTA*** *El montaje de proceso de crawling no se realiza (por ser opcional).*

# **Diagrama de la Arquitectura del Sistema**

A continuación, se muestra un gráfico de la arquitectura diseñada para el sistema.



Este diagrama se ha confeccionado con la herramienta ***Google Drawings*** y se adjunta también, a continuación, en formato PDF.



# **Descripción de la Arquitectura**

Tal y como se ha explicado en los apartados anteriores, el origen principal del sistema será un archivo de inmuebles en alquiler procedente de **AIRBNB** al que se le aplicará un tratamiento de Data Cleansing y un filtrado, mediante la herramienta ***Trifacta Wrangler***, con objeto de obtener datos perfectamente válidos únicamente del área metropolitana de Madrid.

Por otro lado, utilizando un proceso de crawling implementado con ***Scrappy***|***Python***, sobre la página web <https://www.eltiempo.es>, se obtendrá un archivo con datos meteorológicos de las diferentes zonas, barrios o distritos de Madrid Capital a diferentes fechas, con el objetivo de enriquecer la información del fichero principal.

Estos dos ficheros serán depositados en un segmento de ***Google Storage***, realizando la tarea de **Staging** mediante la utilidad de carga de datos que provee ***Google Cloud Platform***.

En ***Google Cloud Platform*** se realizará el despliegue de un cluster de cuatro máquinas: un maestro y tres esclavos que se encargarán de realizar las tareas de procesamiento de los datos ingestados en la nube.

En dicho cluster se instalará una arquitectura ***Apache Hadoop*** con un **HDFS** –*Hadoop Distributed File System*– como base y sistema de almacenamiento para los datos; integrados y sobre él irán un **YARN** –*Yet Another Resource Negociator*– como administrador de procesos y gestor de recursos y un **ZOOKEEPER** realizando las labores de coordinación en el cluster.

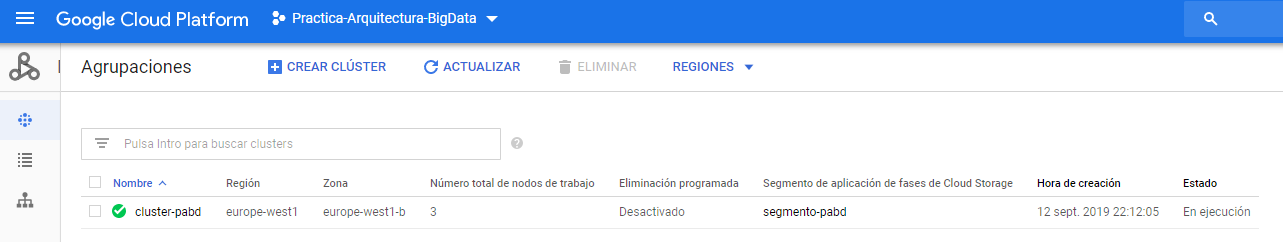
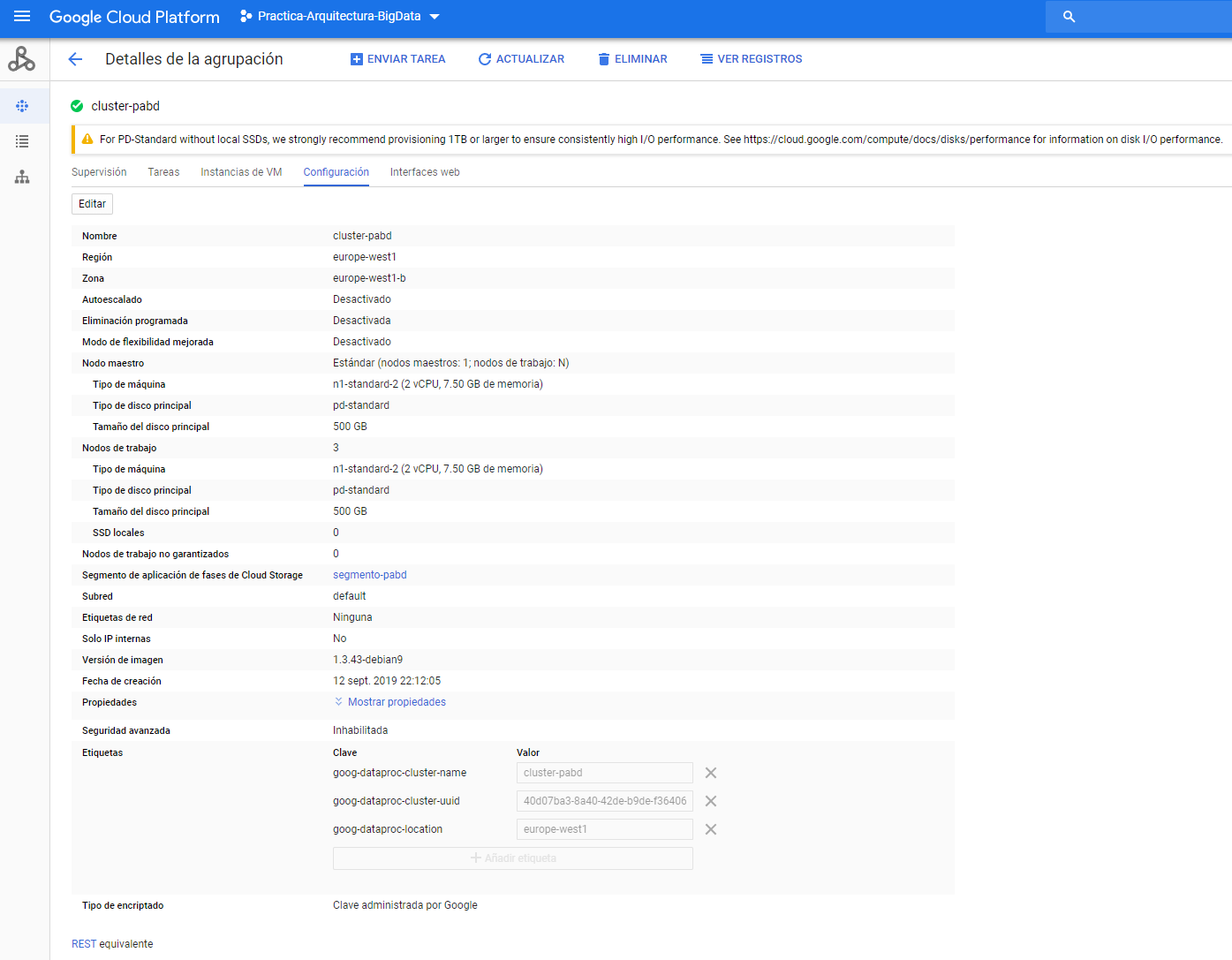
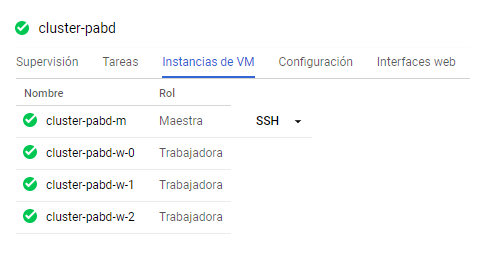
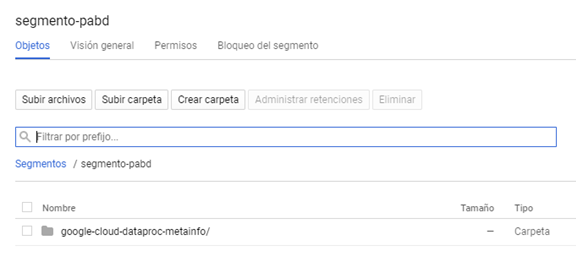
Sobre esta base se montará un modelo de cómputo **MAP-REDUCE** y un framework **SPARK** para poder integrar aplicaciones distribuidas, al cual se le instalará un módulo **elasticsearch-hadoop (es-hadoop)** para enlazar de forma nativa la capa de presentación superior.

Por último, sobrevolando la infraestructura anterior se situará una aplicación **HIVE**, provista de un metastore **POSTGRESQL**, para poder realizar consultas sobre los datos y un motor **ELASTICSEARCH** con **KIBANA** para poder implementar cuadros de mando y visualizaciones de informes.

# **Despliegue en Cloud**

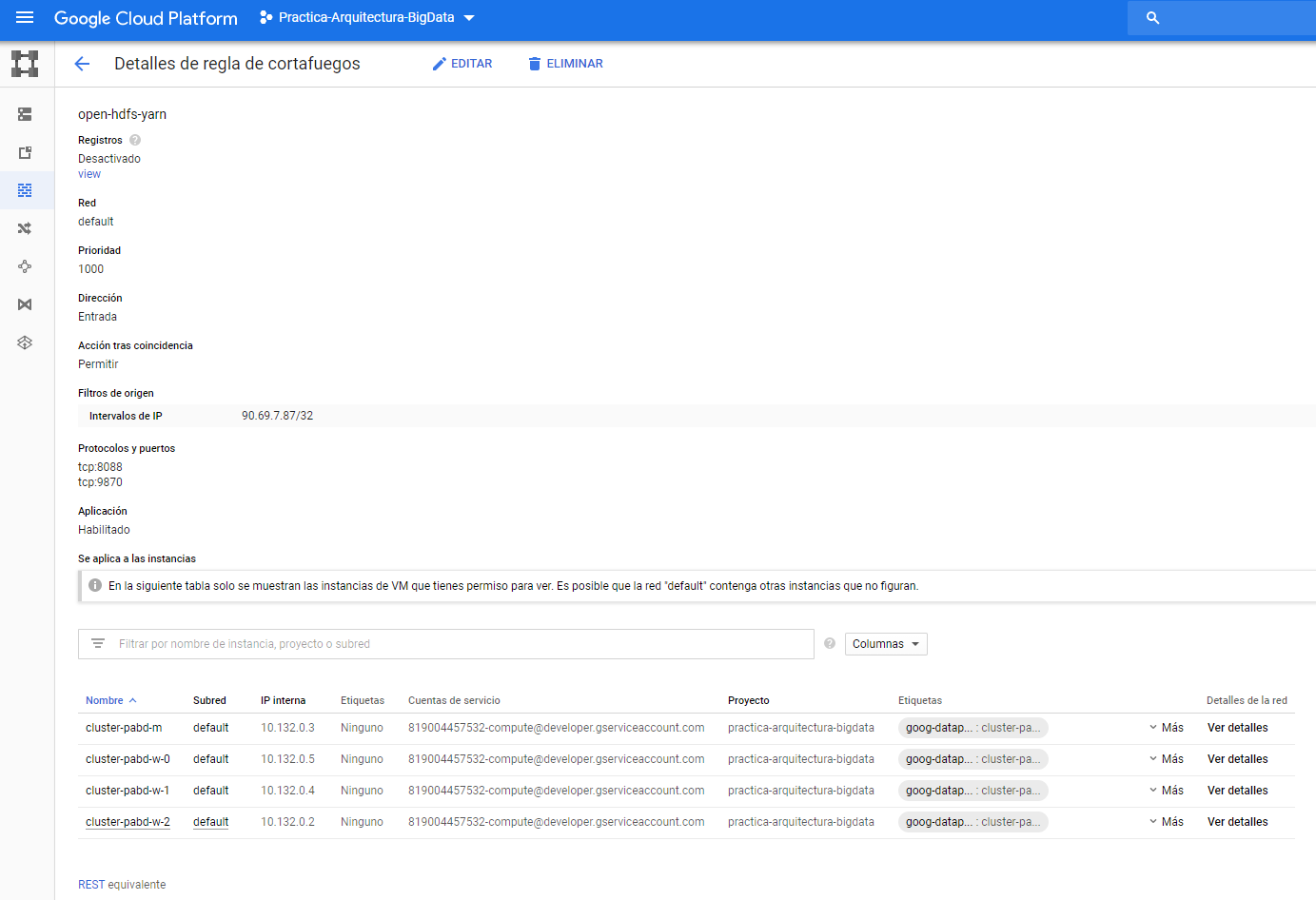
## **Creación del Cluster de Trabajo y del Segmento de Almacenamiento**

Se despliega un cluster en cloud (sobre ***Google Cloud Platform***) con un nodo maestro y 3 nodos de trabajo (esclavos). Se le asocia un segmento de ***Google Storage*** para el staging de los datos.

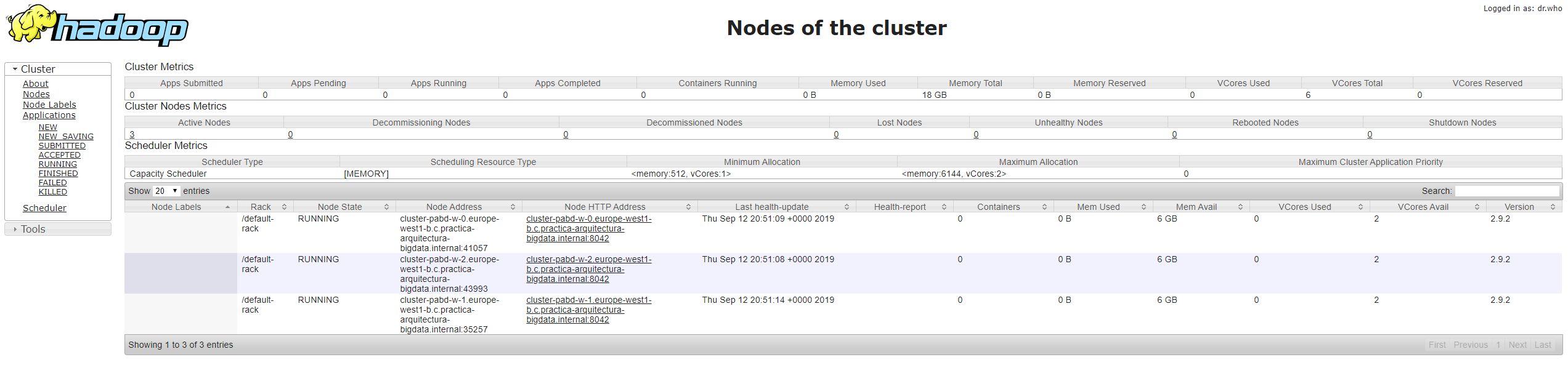


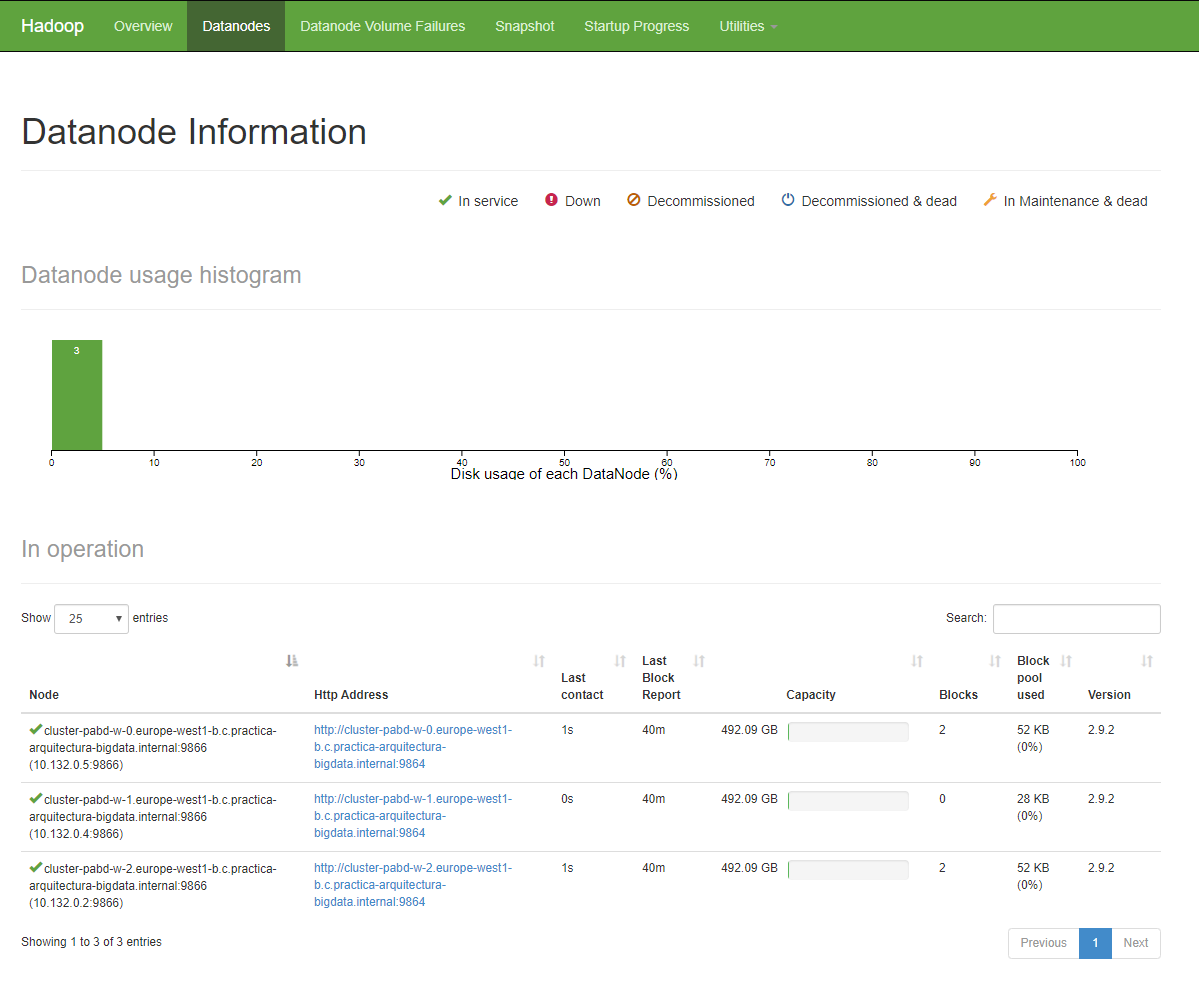
## **Creación de Reglas de Cortafuegos**

Acto seguido creamos una regla de cortafuegos para poder acceder a **HDFS** y **YARN** desde nuestro navegador:



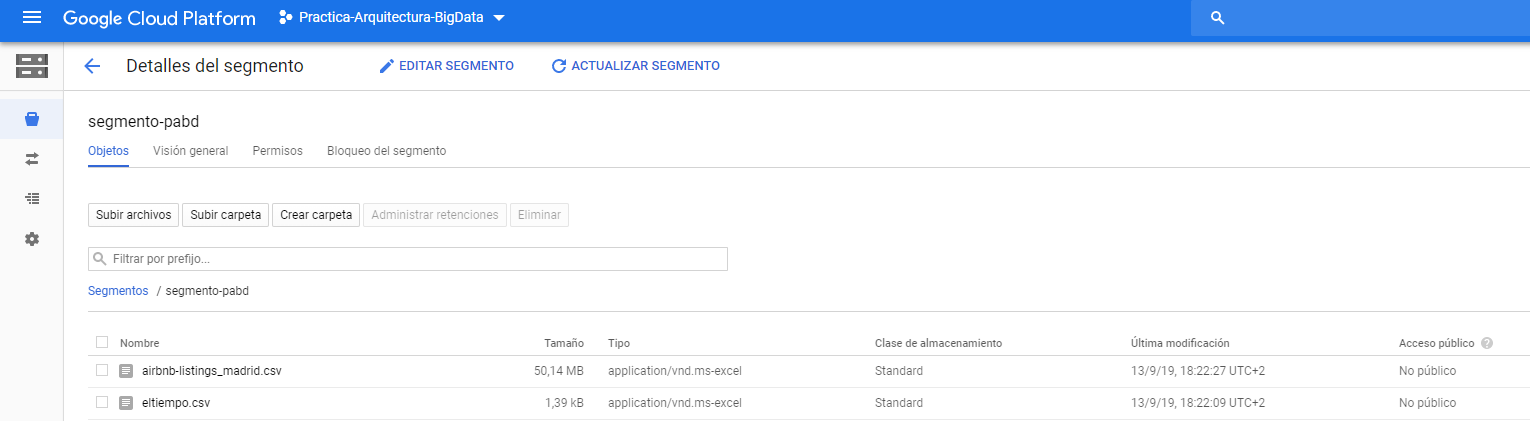
Esta regla aplica, como se puede observar en la imagen anterior, sobre todas las instancias de nuestro cluster. Y ya podemos acceder desde nuestro navegador, tanto a **HDFS** como a **YARN**.





## **Operación de Staging**

Realizamos el Staging de los datos hacia el segmento de almacenamiento definido en ***Google Storage***, utilizando la utilidad “***Subir Archivos***” que provee esta plataforma en su consola.



Subimos al segmento nuestros dos archivos: ***airbnb-listings\_madrid.csv*** y ***eltiempo.csv***.

Estos ficheros se podrían trasladar al **HDFS** o utilizarse para cargar las tablas **HIVE** directamente desde el segmento.

## **Creación de Tablas HIVE**

Abrimos una consola de la máquina maestra y entramos al cliente **BEELINE** con el comando:

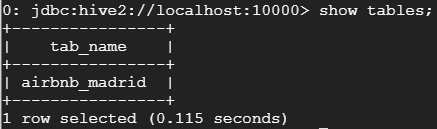
beeline -u jdbc:hive2://localhost:10000

Procedemos a crear la tabla principal del sistema **airbnb\_madrid** mediante la sentencia SQL:

CREATE TABLE airbnb\_madrid (ID INT, Listing\_Url STRING, Scrape\_ID STRING, Last\_Scraped STRING, Name STRING, Summary STRING, Space STRING, Description STRING, Neighborhood\_Overview STRING, Notes STRING, Transit STRING, Access STRING, Interaction STRING, House\_Rules STRING, Thumbnail\_Url STRING, Medium\_Url STRING, Picture\_Url STRING, XL\_Picture\_Url STRING, Host\_ID INT, Host\_URL STRING, Host\_Name STRING, Host\_Since STRING, Host\_Location STRING, Host\_About STRING, Host\_Response\_Time STRING, Host\_Response\_Rate INT, Host\_Thumbnail\_Url STRING, Host\_Picture\_Url STRING, Host\_Neighbourhood STRING, Host\_Listings\_Count INT, Host\_Total\_Listings\_Count INT, Host\_Verifications STRING, Street STRING, Neighbourhood\_Cleansed STRING, City STRING, State STRING, Zipcode INT, Market STRING, Country\_Code STRING, Country STRING, Latitude STRING, Longitude STRING, Property\_Type STRING, Room\_Type STRING, Accommodates INT, Bathrooms STRING, Bedrooms INT, Beds INT, Bed\_Type STRING, Amenities STRING, Square\_Feet INT, Price INT, Weekly\_Price INT, Monthly\_Price INT, Security\_Deposit INT, Cleaning\_Fee INT, Guests\_Included INT, Extra\_People INT, Minimum\_Nights INT, Maximum\_Nights INT, Calendar\_Updated STRING, Availability\_30 INT, Availability\_60 INT, Availability\_90 INT, Availability\_365 INT, Calendar\_last\_Scraped STRING, Number\_of\_Reviews INT, First\_Review STRING, Last\_Review STRING, Review\_Scores\_Rating INT, Review\_Scores\_Accuracy INT, Review\_Scores\_Cleanliness INT, Review\_Scores\_Checkin INT, Review\_Scores\_Communication INT, Review\_Scores\_Location INT, Review\_Scores\_Value INT, License STRING, Cancellation\_Policy STRING, Calculated\_host\_listings\_count INT, Reviews\_per\_Month STRING, Geolocation STRING, Features STRING) ROW FORMAT DELIMITED FIELDS TERMINATED BY ';';

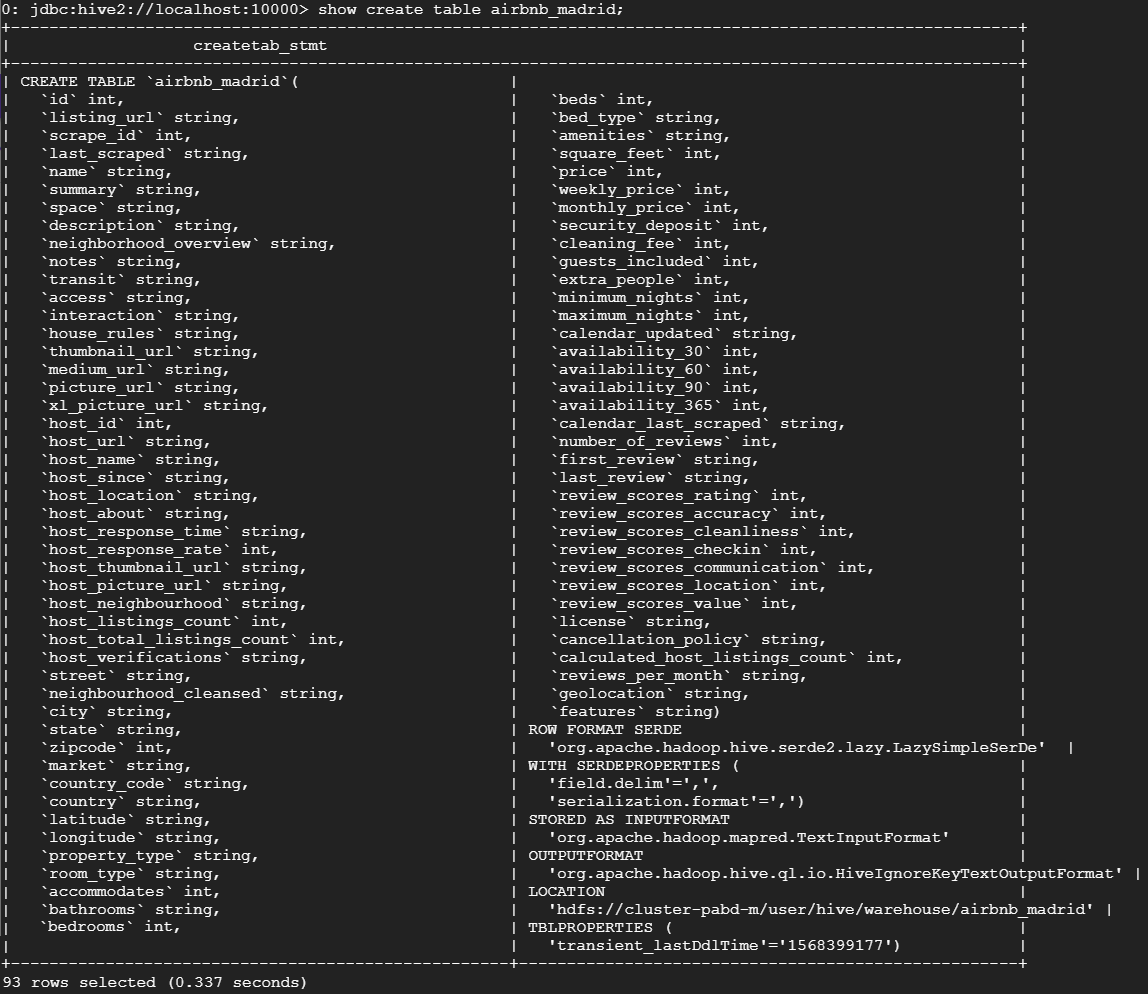
Una vez realizado podemos ver la tabla creada en nuestro sistema con el comando:

show tables;



O también podemos ver todos los detalles de su creación con el comando:

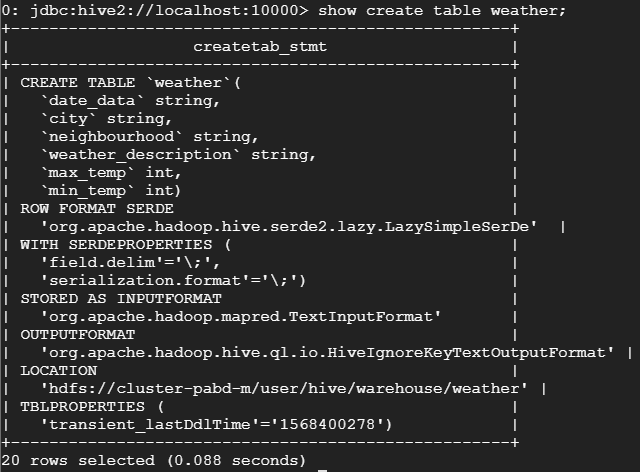
show create table airbnb\_madrid;



Acto seguido crearemos la tabla con los datos meteorológicos.

CREATE TABLE weather (Date\_Data STRING, City STRING, Neighbourhood STRING, Weather\_Description STRING, Max\_temp INT, Min\_temp INT) ROW FORMAT DELIMITED FIELDS TERMINATED BY ';';

show create table weather;



## **Carga de Datos en el Modelo**

Una vez creadas las tablas procedemos a aprovisionarlas desde el segmento de ***Google Storage***.

LOAD DATA INPATH 'gs://segmento-pabd/airbnb-listings\_madrid.csv' INTO TABLE airbnb\_madrid;

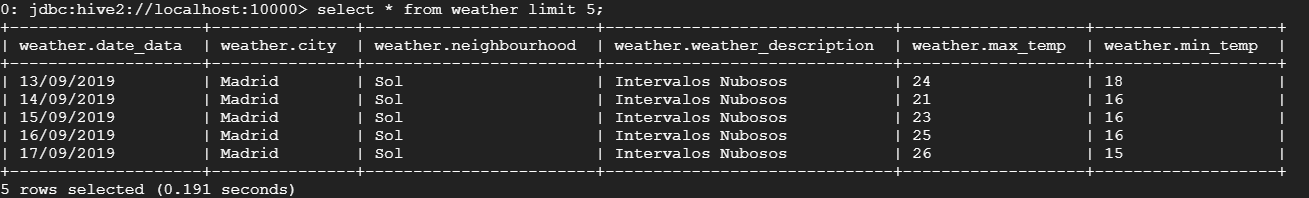
LOAD DATA INPATH 'gs://segmento-pabd/eltiempo.csv' INTO TABLE weather;

Y una vez cargadas, verificamos que tienen datos:

select \* from airbnb\_madrid limit 1;

select \* from weather limit 5;





## **Explotación y Productivización**

Una vez se han aprovisionado las tablas y se ha verificado la corrección de sus datos sólo queda comenzar a diseñar procesos que exploten esa información para luego productivizarlos de forma planificada.

Se pueden definir y desarrollar muchas tareas que pueden resultar útiles, como por ejemplo, informes o cuadros visuales en **KIBANA** que proporcionen datos relevantes de los pisos más caros o más baratos por zona con su información meteorológica semanal agregada, aplicaciones de búsqueda portalizadas, desarrolladas en **PySpark** atacando **HIVE**, de los alquileres por precio (diario, semanal, mensual), por tipo de alquiler (casa entera, habitación), por disponibilidad, parametrizadas por la información meteorológica en la zona, etc.

# **Agradecimientos**

Como nota final de este trabajo me gustaría dar las gracias al profesor **Ricardo Vegas**, por su gran atención corrigiendo nuestros errores, despejando nuestras dudas (en las clases y en Slack) y por su generosidad preparando toda la documentación teórica y práctica para las clases.

Pienso que esta asignatura de Arquitectura Big Data es muy extensa, demasiado diría yo, para ocho sesiones. Me ha dejado algo insatisfecho porque creo firmemente que quizá hubiera venido bien otra semana de clases aprendiendo más cosas, como desplegar OnPremise y en Cloud la capa final de aplicaciones sobre un cluster, formas de ingesta (staging) que las que no hemos podido ni hablar como **sqoop** o bases de datos como **Cassandra**.

Seguro que con otra semana más de clases hubiera ganado yo el Kahoot (es broma… jajaja… el compañero FJAsensi es invencible).

Solo indicar que este trabajo termina con el aprovisionado de los datos en las tablas HIVE, aprovechando que los montajes se definen en el enunciado de la práctica como opcionales.

Sin más, mi más sincero agradecimiento por todas las cosas que he aprendido.

**F. Javier Gonzálvez**