GUÍA : LEVANTAR CLUSTER HADOOP UTILIZANDO GOOGLE CLOUD PLATFORM (GCP)

1. **Introducción:**

   En esta guía, aprenderás cómo crear un clúster en Google Cloud Platform (GCP) para ejecutar Hadoop y Apache Spark, dos de las herramientas más importantes para el procesamiento de grandes volúmenes de datos. Veremos cómo usar Google Dataproc para gestionar clústeres en la nube y cómo realizar una simulación de procesamiento en tiempo real con Kafka y Spark Streaming.

   Al final de esta guía, podrás:

   - Entender cómo usar GCP para crear clústeres de procesamiento de datos.
   - Utilizar Google Cloud Storage como almacenamiento.
   - Levantar un clúster Hadoop y Spark en GCP.
   - Usar Kafka y Spark para realizar simulaciones de procesamiento de datos en tiempo real.

2. **Conceptos previos:**

   a. **¿Qué es Google Cloud Platform (GCP)?**

      Es un conjunto de servicios en la nube proporcionados por Google. Permite a los usuarios realizar tareas como el almacenamiento, procesamiento y análisis de datos, así como desplegar aplicaciones a gran escala. Entre sus servicios más destacados para Big Data están Google Dataproc, BigQuery, Cloud Storage, entre otros.

   b. **¿Qué es Google Dataproc?**

      Servicio de GCP que facilita el despliegue y gestión de clústeres de Hadoop, Spark, y otras herramientas para procesar grandes cantidades de datos. A diferencia de otras soluciones, Dataproc está optimizado para la nube, lo que permite que los clústeres sean creados y administrados rápidamente.

   c. **¿Qué es Google Cloud Storage (GCS)?**

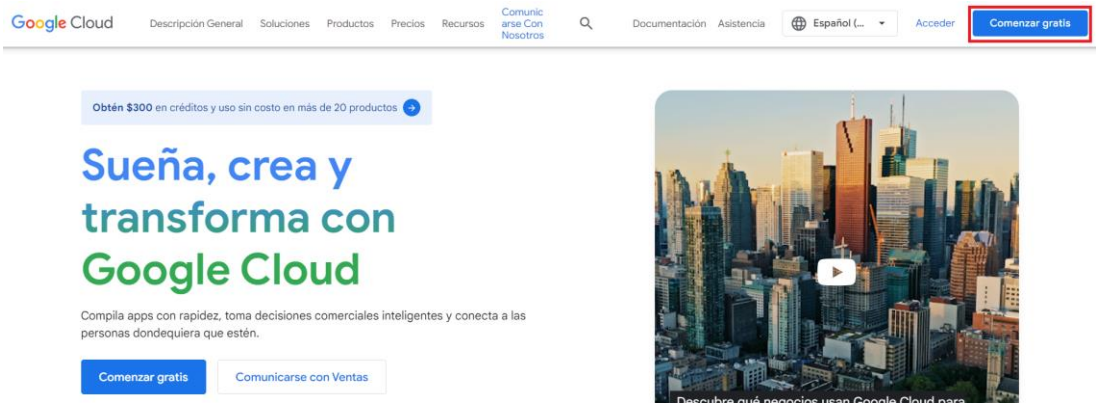
      Es un servicio de almacenamiento de objetos escalable que permite almacenar datos en forma de archivos y carpetas (en buckets). Este almacenamiento es ideal para interactuar con clústeres de Dataproc, ya que puedes subir archivos de datos (como archivos CSV o JSON) para que los procesos de Hadoop o Spark los lean directamente desde GCS.

3. **Herramientas que usaremos:**

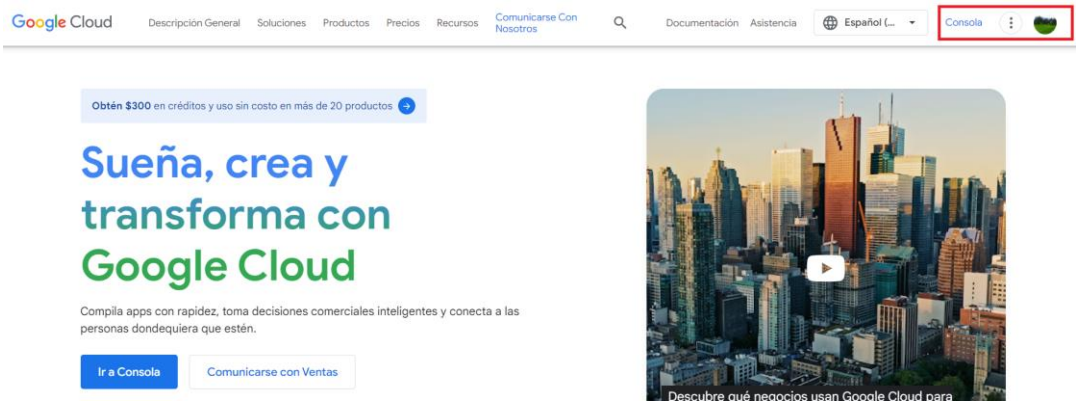
   - **Google Cloud Console**: Interfaz gráfica desde la cual gestionaremos nuestros recursos en GCP.
   - **Google Dataproc**: Servicio para gestionar clústeres de procesamiento de datos (Hadoop, Spark, Hive).

- **Google Cloud Storage**: Servicio de almacenamiento que usaremos para guardar datos que procesaremos con Hadoop y Spark.
- **Apache Hadoop**: Framework para el almacenamiento distribuido y procesamiento de grandes conjuntos de datos.
- **Apache Spark**: Motor de procesamiento en memoria que acelera trabajos analíticos en grandes volúmenes de datos.
- **Apache Kafka**: Plataforma distribuida de transmisión de datos en tiempo real.

4. **Configuración Inicial en GCP**
   a. **Crear una Cuenta en GCP:**
      Si aún no tienes una cuenta de GCP, debes crear una en https://cloud.google.com/. Google ofrece un crédito gratuito para nuevos usuarios, que te permitirá probar muchos servicios sin costo durante un periodo limitado.



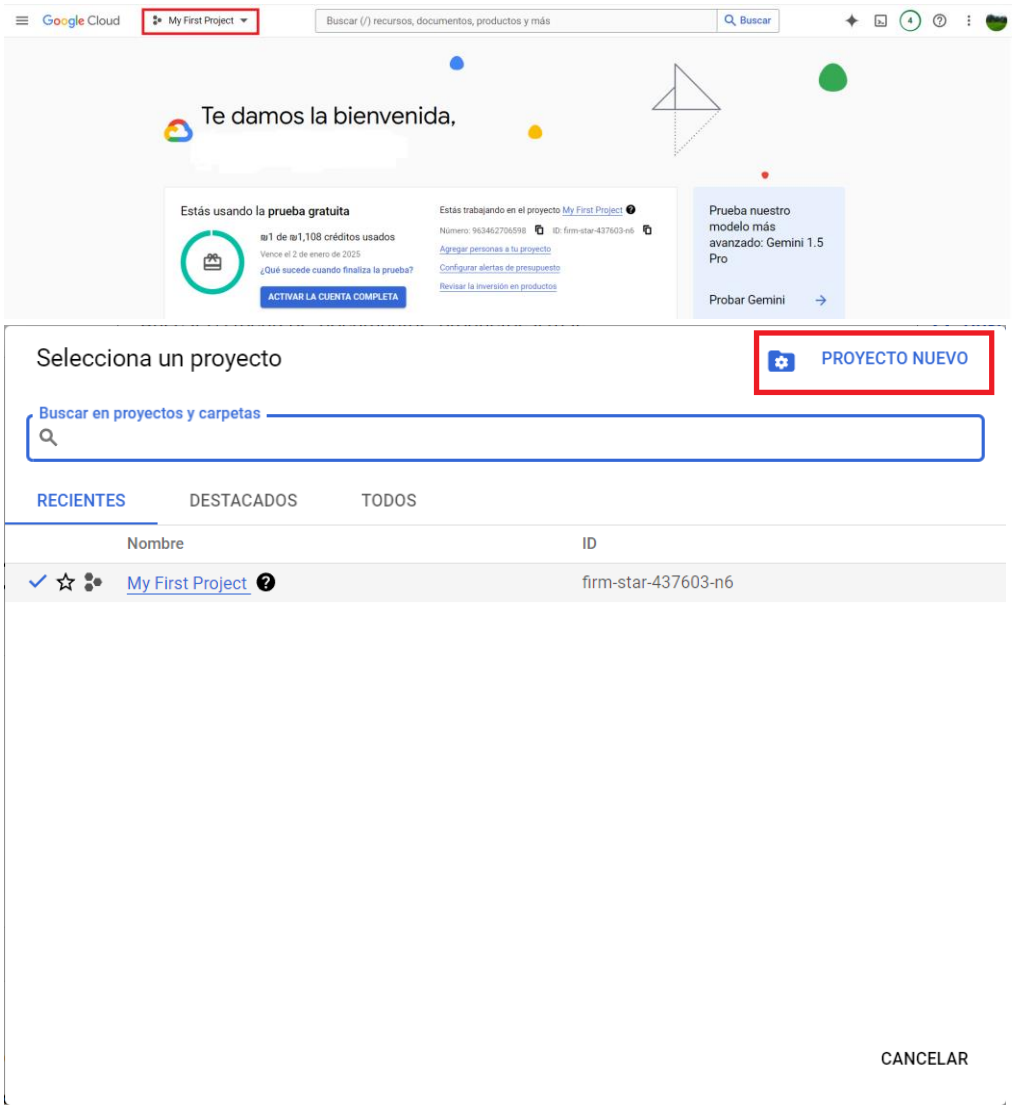
Después de darle click a “Comenzar gratis” seguirás una serie de pasos para activar tu cuenta de versión gratuita. Sigue los pasos de Google y podrás crear tu cuenta sin problemas. Al final del registro debería quedar así:



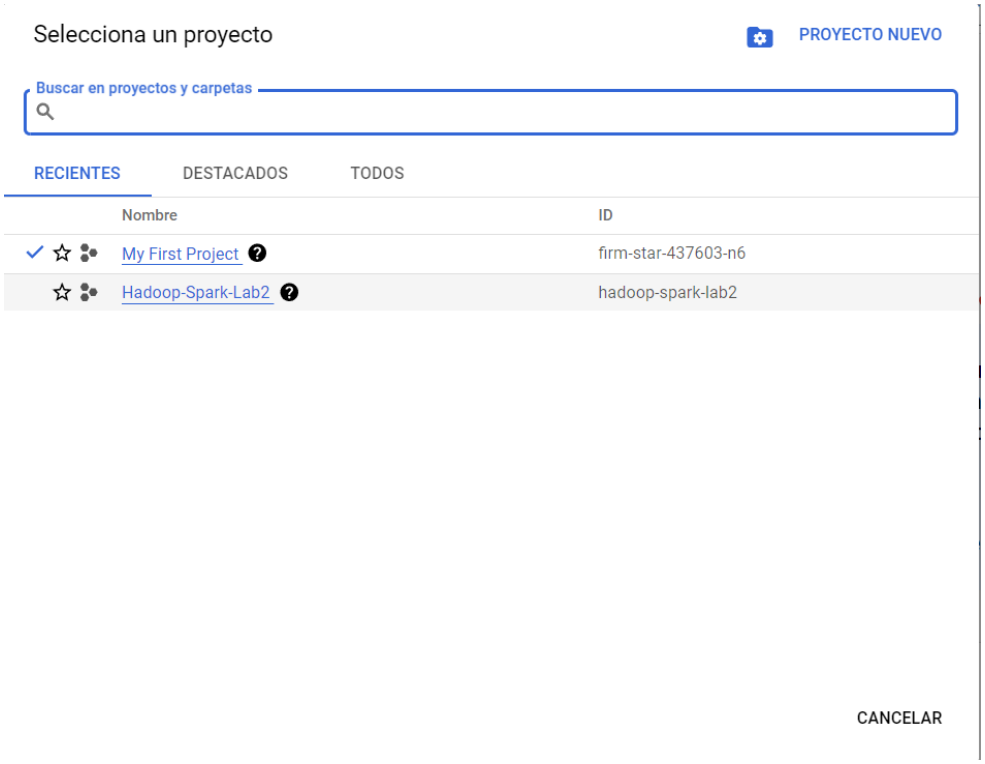
Puedes acceder a la consola haciendo click en el botón “Consola” como se muestra en la imagen adjunta arriba.

b. Crear un proyecto:

i. Ve a la consola de GCP y selecciona "Proyectos".



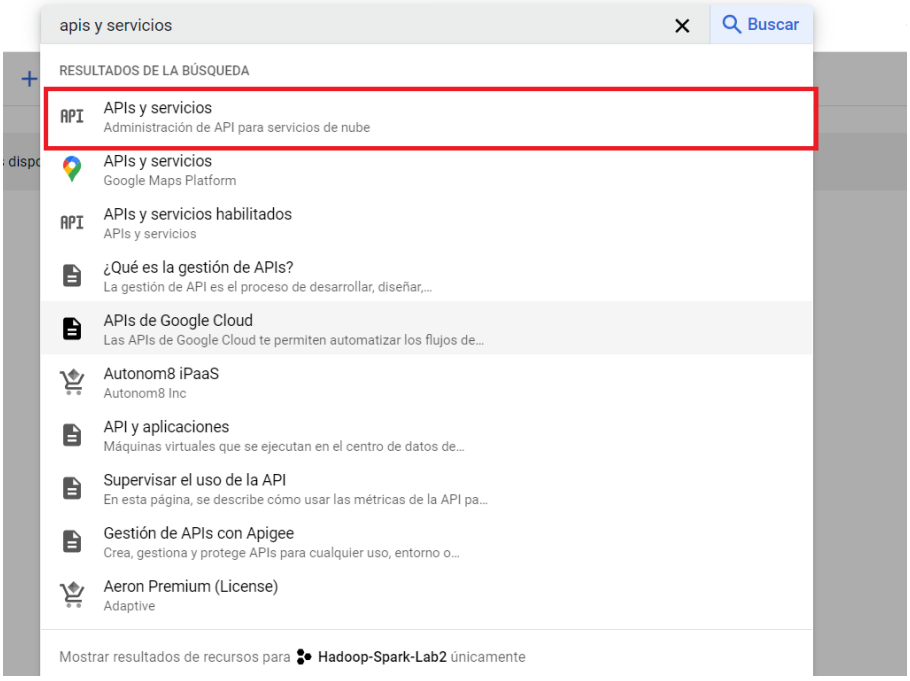
ii. Crea un nuevo proyecto con un nombre descriptivo (En mi caso lo llame Hadoop-Spark-Lab2). Guárdalo y accede a él.



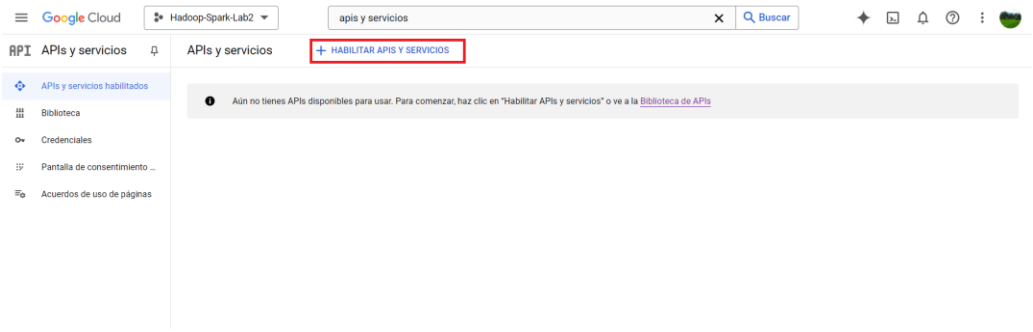
iii. Recuerda el ID del proyecto. Ahora, dale click al nombre de tu proyecto para empezar a crear los clústeres en él. Ahora debemos activar las APIs correspondientes.

c. Habilitar APIs:

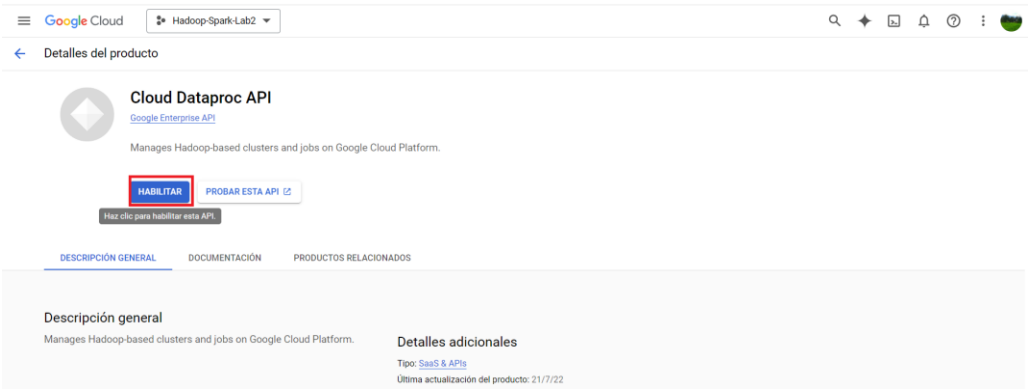
i. En el buscador de la consola de GCP, digita APIs & Servicios

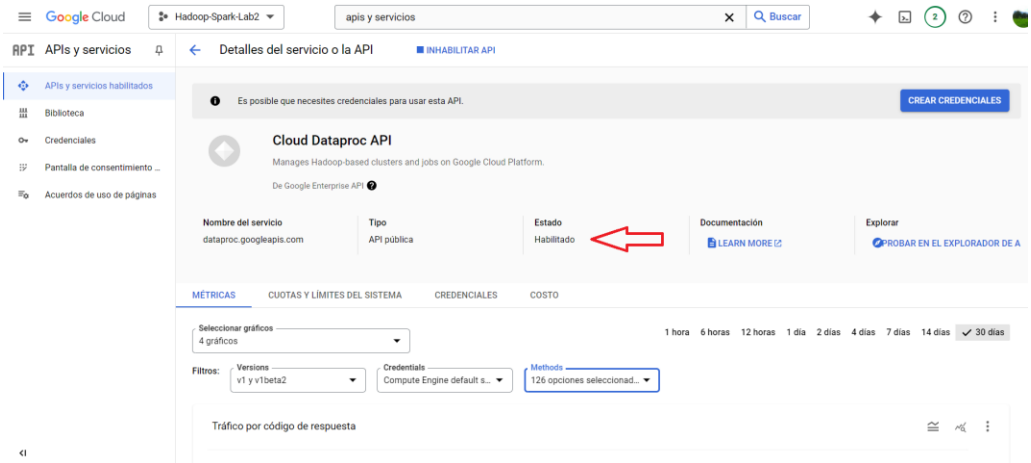


ii. Para habilitar las APIs necesarias, dale click al siguiente boton:



iii. Busca "Dataproc" y habilita la Google Cloud Dataproc API. Puede demorar unos minutos.





d. Configurar IAM (Gestión de roles):

Para asegurarse de tener los permisos adecuados, es útil usar IAM (Identity and Access Management).

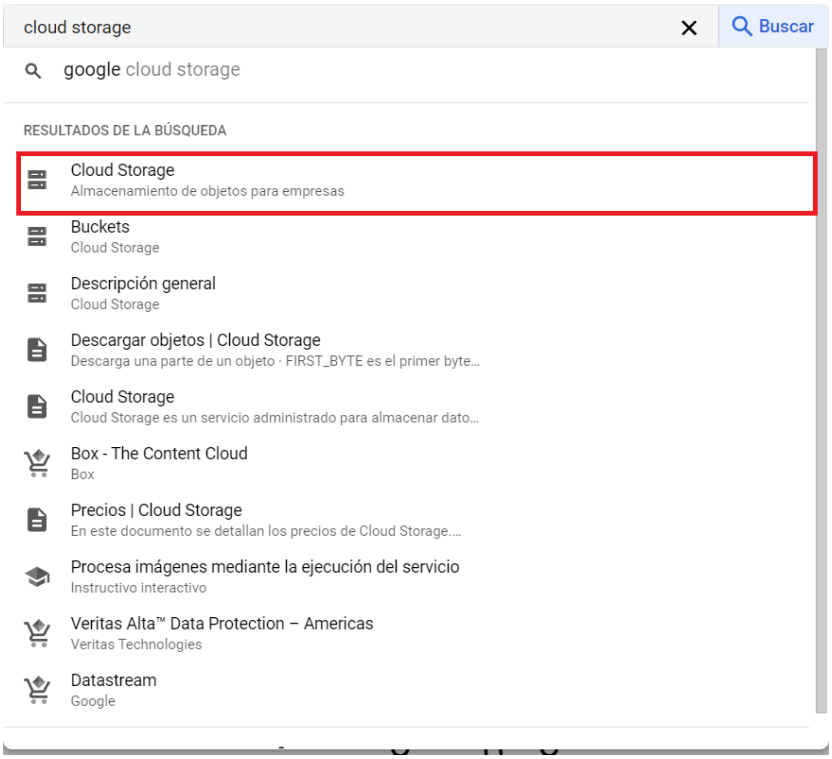
i. Ve a IAM & Admin en el menú de GCP.

ii. Verifica que tengas el rol de "Editor" o "DataProc Editor" para que puedas interactuar con Dataproc y otros servicios.

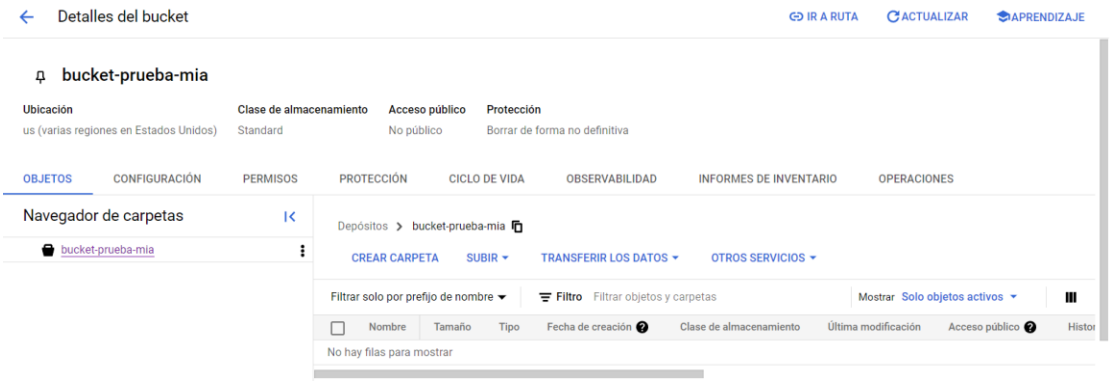
5. Configuración Google Cloud Storage (GCS)

a. Crear un Bucket en Google Cloud Storage

i. En el buscador de la consola de GCP, digita “Cloud Storage”.



ii. Selecciona Cloud Storage
iii. Dale a click a “Crear” para crear un bucket.
iv. Asigna un nombre único para el bucket. Utiliza minúsculas y guiones para que te permita establecer un nombre.
v. Elige la región donde deseas que esté almacenado el bucket.
vi. Configura el nivel de almacenamiento según el uso.
vii. Haz clic en “Crear”.
viii. Darle a aceptar si te pide confirmar que tu bucket se mantendrán en privado.



b. Subir archivos a GCS:

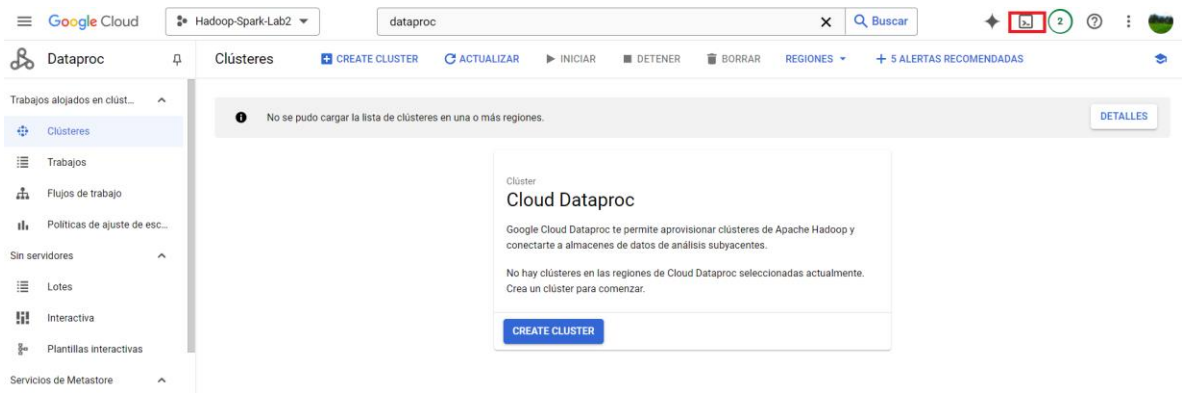
i. Dentro del bucket que creaste, selecciona la opción **Subir**.
ii. Elige un archivo de datos.
iii. Haz clic en Abrir para subirlo a GCS.

6. Crear el Clúster en Google Dataproc

Gcloud es la herramienta de línea de comandos de Google Cloud. La vamos a utilizar para crear nuestro clúster.

a. Utilizar la terminal con gcloud

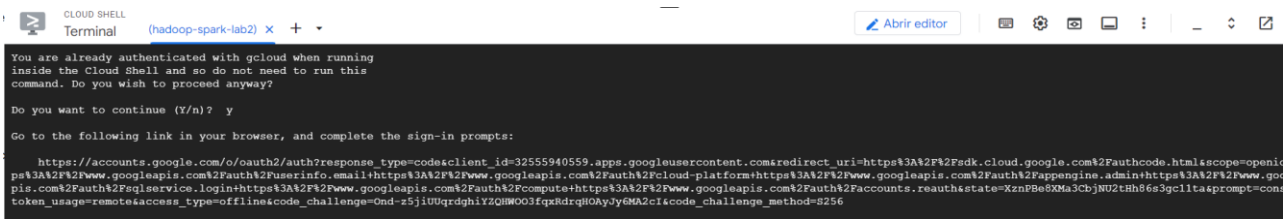
i. Ir a Dataproc
ii. Activar el shell



b. Configurar el entorno

i. Autenticarse en Google Cloud. Digita lo siguiente en la terminal: *gcloud auth login*.
Si le sale este mensaje es porque ya está logeado correctamente.
Además puede proceder de todas formas para logearse utilzando el

link que proporcionan. Aunque no debería haber problema, si le sale el mensaje usted ya está logeado.



ii. Después de autenticarse, selecciona el proyecto de Google Cloud donde crearás el clúster:

gcloud config set project [ID_DEL_PROYECTO]

c. Crear el clúster en Dataproc.

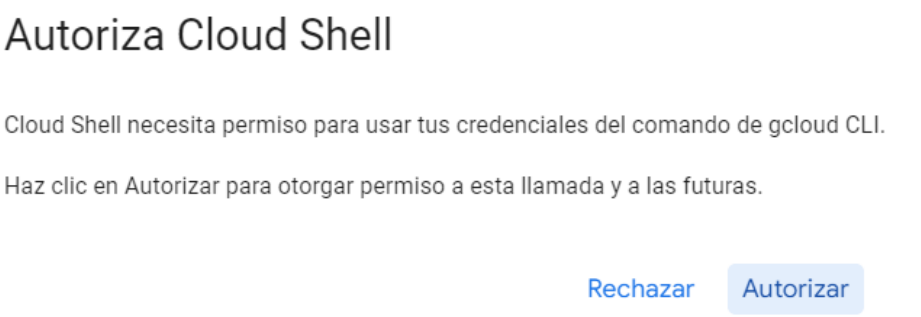
i. Utiliza el siguiente comando

```
gcloud beta dataproc clusters create micluster \
 --enable-component-gateway \
 --bucket bucket-prueba-mia \
 --region us-east1 \
 --zone us-east1-c \
 --master-machine-type n1-standard-2 \
 --master-boot-disk-size 500 \
 --num-workers 2 \
 --worker-machine-type n1-standard-2 \
 --worker-boot-disk-size 500 \
 --image-version 2.1-debian11 \
 --properties spark:spark.jars.packages=org.apache.spark:spark-sql-kafka-0-10_2.12:3.1.3 \
 --optional-components JUPYTER,ZOOKEEPER \
 --max-age 14400s \
 --initialization-actions 'gs://goog-dataproc-initialization-actions-europe-west1/kafka/kafka.sh' \
 --project hadoop-spark-lab2
```

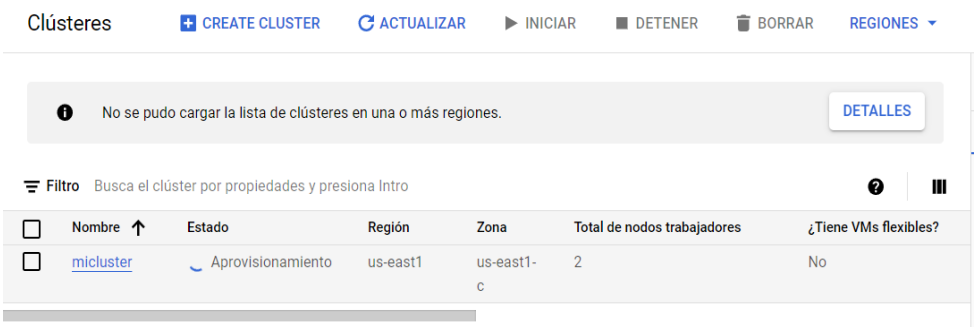
Tener en cuenta que:

- Si le sale error al momento de pegarlo en la consola, debe colocar el comando línea por línea para que se lea correctamente.
- En la siguiente línea, *--bucket bucket-prueba-mia* \
  Debes colocar el nombre de tu cluster especifico:
  --bucket [NOMBRE_DEL_BUCKET] \
- En la siguiente línea, *--project hadoop-spark-lab2*
  Debes colocar el nombre de tu proyecto específico:
  --project [ID_DEL_PROYECTO]

ii. Aceptar permisos. Dale a “Autorizar”



iii. Si te salió el error, vuelve a intentar el comando. Veras que se creó el cluster. El aprovisionamiento puede tardar varios minutos.

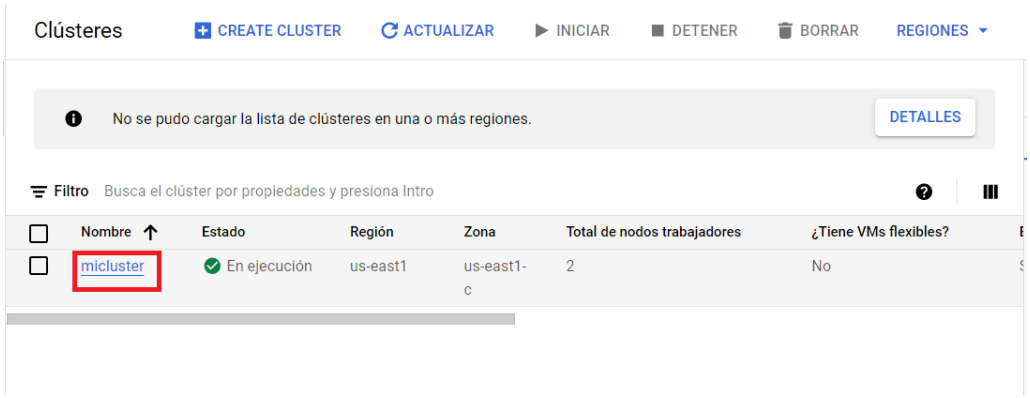


Si quieres que significa cada línea de código, acá te la resumimos:

- **`--enable-component-gateway`**: Habilita el acceso a las interfaces web como Jupyter, HDFS, YARN.
- **`--bucket [NOMBRE_DEL_BUCKET]`**: Asocia el bucket de Google Cloud Storage para almacenar archivos de trabajo.
- **`--region`**: Define la región geográfica (en este caso, **us-east1**).
- **`--zone`**: Zona dentro de la región, **us-east1-c** en este ejemplo.

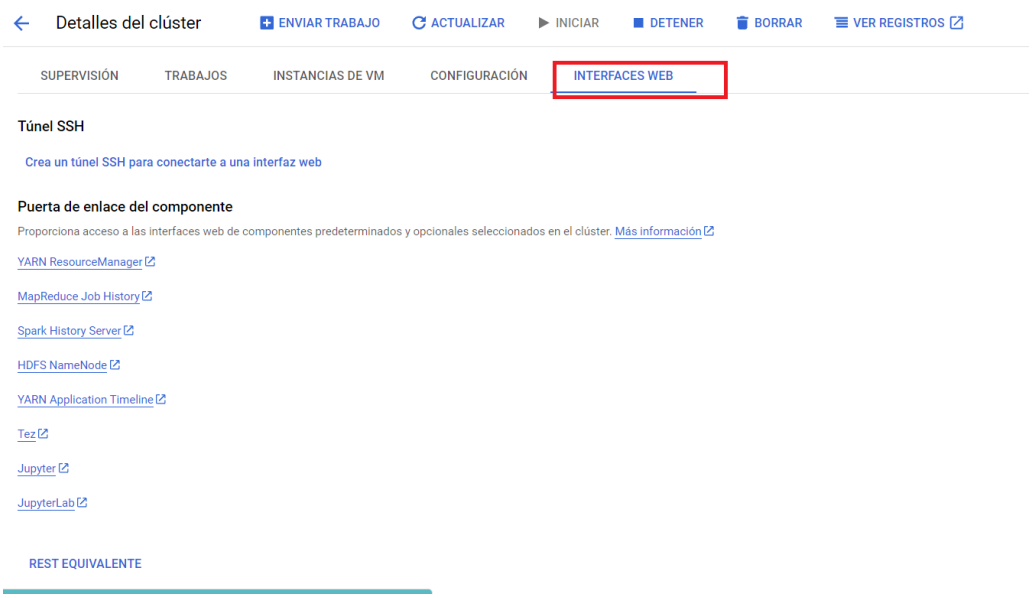
- **`--master-machine-type`** y **`--worker-machine-type`**: Define el tipo de máquinas a utilizar (n1-standard-2, 2 vCPUs y 7.5GB de RAM).
- **`--master-boot-disk-size`** y **`--worker-boot-disk-size`**: Define el tamaño del disco de arranque.
- **`--image-version`**: Especifica la versión de la imagen Dataproc (en este caso, **2.1-debian11**).
- **`--properties spark:spark.jars.packages=org.apache.spark:spark-sql-kafka-0-10_2.12:3.1.3`**: Incluye las dependencias para que **Spark** pueda interactuar con **Kafka**.
- **`--optional-components JUPYTER, ZOOKEEPER`**: Añade componentes opcionales como Jupyter y Zookeeper.
- **`--max-age 14400s`**: Tiempo de vida del clúster en segundos (4 horas en este caso).
- **`--initialization-actions`**: Ejecuta un script de inicialización que instala y configura **Kafka**.

iv. Ya creado el cluster podemos acceder a él.



v. Cierre la terminal

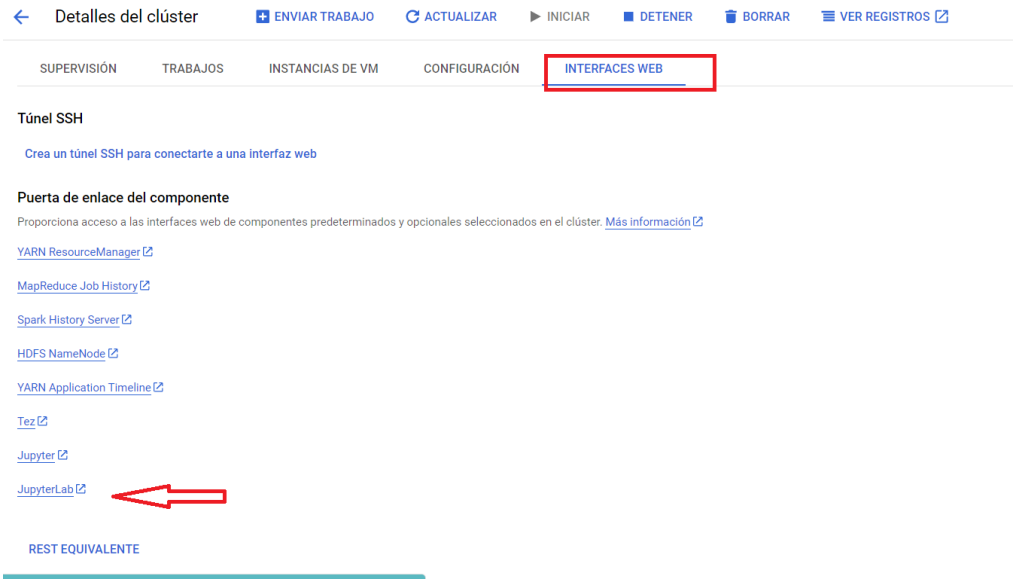
vi. Dirígete a “Interfaces Web”. Y selecciona Jupiterlab



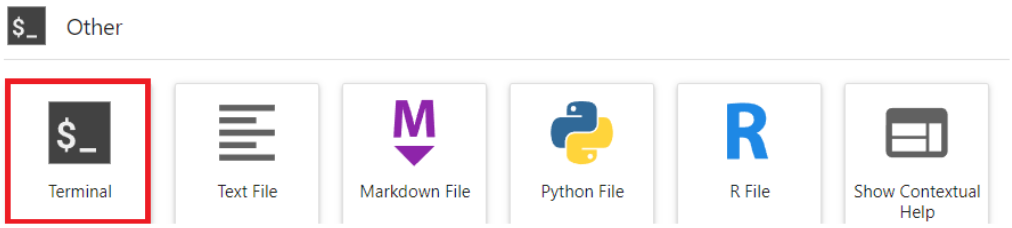
7. Acceder al Cluster

a. Usar la terminal para almacenar en HDFS

i. Ve a JupiterLab

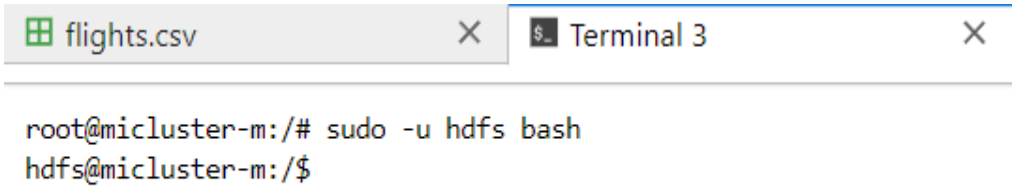


ii. Accede a la terminal

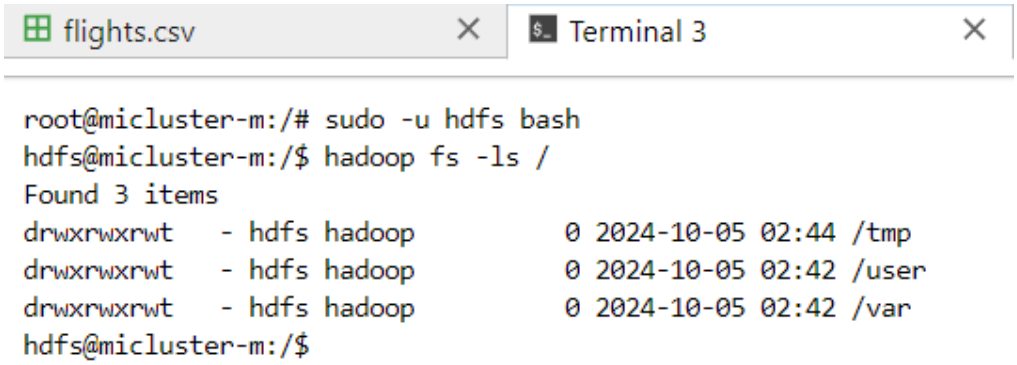


iii. Se abrirá una terminal donde podrás ejecutar comandos del sistema.

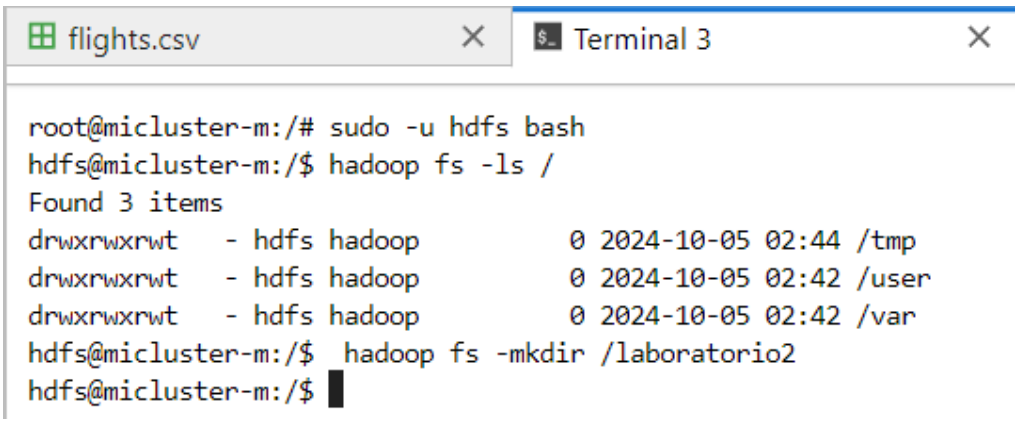
iv. Lo primero que debemos hacer es obtener los permisos suficientes para administrar nuestros archivos utilizando el usuario “hadoop”. Para ello coloca el siguiente comando en el terminal:
sudo -u hdfs bash



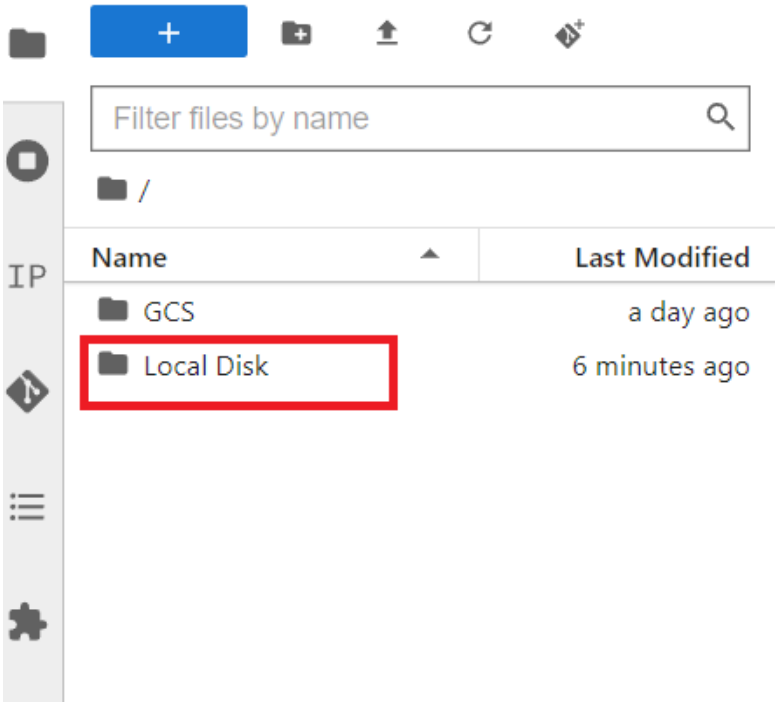
v. Coloca *hadoop fs -ls / para* obtener todos los directorios que se encuentran en tu HDFS. Por defecto son 3:



vi. Ahora creamos un nuevo directorio (lo llamaremos laboratorio2) para almacenar nuestros archivos utilizando este comando: *hadoop fs -mkdir /laboratorio2*



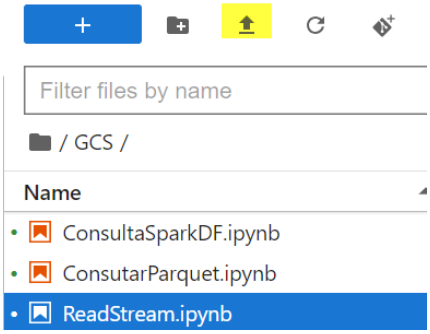
vii. Utilizaremos el archivo flights.csv como modelo de prueba de ingesta de data. Ahora debemos colocarlo en la carpeta local (Local Disk):



viii. Ahora para copiar archivos desde el sistema de archivos local hacia el sistema de archivos distribuido de Hadoop (HDFS) utiliza:

hadoop fs -copyFromLocal /flights.csv /laboratorio2

ix. Subir los notebooks al ConsultaSparkDF_lab.ipynb y ConsutarParquet_lab.ipynb y ReadStream_lab.ipynb a la carpeta GCS



Abrir el archivo ConsultaSparkDF y validar cargar el cvs al un dataframe de Spark

```
ruta_hdfs = "/laboratorio2/flights.csv"

flightsDF = spark.read\
            .option("header", "true")\
            .option("inferSchema", "true")\
            .csv(ruta_hdfs)
```

```
flightsDF.show(5)
```

b. Usar la terminal para usar Apache Kafka

i. Crear un Topic llamado "retrasos". Para ello en la terminal: /usr/lib/kafka/bin/kafka-topics.sh --bootstrap-server micluster-w-0:9092 --create --replication-factor 1 --partitions 1 --topic retrasos

ii. Luego se puede verificar que se ha creado el Topic usando: /usr/lib/kafka/bin/kafka-topics.sh --bootstrap-server micluster-w-0:9092 –list

```
root@micluster-m:/# sudo -u hdfs bash
hdfs@micluster-m:/$ hadoop fs -mkdir /laboratorio2
hdfs@micluster-m:/$ hadoop fs -copyFromLocal /flights.csv /laboratorio2
hdfs@micluster-m:/$ /usr/lib/kafka/bin/kafka-topics.sh --bootstrap-server micluster-w-0:9092 --create --
Created topic retrasos.
hdfs@micluster-m:/$ /usr/lib/kafka/bin/kafka-topics.sh --bootstrap-server micluster-w-0:9092 --list
retrasos
hdfs@micluster-m:/$
```

iii. Iniciamos el productor para las simulaciones usando: /usr/lib/kafka/bin/kafka-console-producer.sh --broker-list micluster-w-0:9092 --topic retrasos

```
hdfs@micluster-m:/$ /usr/lib/kafka/bin/kafka-console-producer.sh --broker-list micluster-w-0:9092 --topic retrasos
>
```

Cuando inicias el productor de Kafka, la consola de tu terminal se cambiará a una interfaz donde puedes introducir datos, espera hasta que suscribas a un consumidor

iv. Abrir el archivo ReadStream_lab.ipynb y ejecutar cada celda, puede que salga warnings pero no te preocupes

v. En el terminal ejecuta, vamos a simular el envío de información de JSON, ejecuta el primero y verifica en el notebook ReadStream_lab el dato que llegó, debe salirte el mensaje

{"dest": "GRX", "arr_delay": 2.6}

Puedes seguir probando uno por uno y verificando de la misma manera
{"dest": "MAD", "arr_delay": 5.4}
{"dest": "GRX", "arr_delay": 1.5}
{"dest": "MAD", "arr_delay": 20.0}

Al finalizar puedes validar que se guardó los archivos en el parquet con el archivo ConsutarParquet_lab.ipynb.

vi. Para asegurarte de que los mensajes se han enviado correctamente, puedes usar un consumidor de Kafka para leer los mensajes del topic retrasos. Ejecuta:
/usr/lib/kafka/bin/kafka-console-consumer.sh --bootstrap-server micluster-w-0:9092 --topic retrasos --from-beginning

También, puedes usar *control + c* para que te muestre cuantos mensajes se han mandado y detener el consumidor.