Pipeline de Dataflow. CSVs en Cloud Storage a tablas de BigQuery

Índice de contenidos

Introducción

En este documento se describen los pasos necesarios para automatizar la importación de ficheros CSVs que se suben a un *bucket* de Cloud Storage a tablas de BigQuery. Incluye una Cloud Function encargada de localizar el archivo que se ha actualizado, leer su esquema e iniciar la plantilla de Dataflow, que almacena los datos en el dataset de BigQuery que indiquemos.

Preparación

Para completar este flujo de datos, se debe disponer de un 'bucket' de Google Cloud Storage con la siguiente estructura de carpetas:

- **/csv** o /**data**: directorio donde se almacenarán los archivos CSV que se desean importar a BigQuery.
- **/schemas**: directorio que aloja los archivos JSON que describe los campos de cada CSV. Estos JSON se generarán automáticamente cuando se despliegue la Cloud Function y se suban los archivos CSV.
- **/templates**: contendrá la plantilla de Dataflow que se genera en los apartados siguientes.

También, se debe haber subido a la carpeta '*templates*' del storage el archivo 'Provincia.csv' que se adjunta a esta memoria. Esto se debe a que, a la hora de crear la plantilla de Dataflow, esta debe poder acceder a un archivo por defecto, se selecciona 'Provincia.csv' por su sencillez.

1. Creación de la plantilla de Dataflow

Junto a esta memoria se adjunta el archivo '*data_ingest.py*' con el código para crear la plantilla de Dataflow, que se almacenará en el Storage y se llamará desde la Cloud Function para llevar los datos a tablas de BigQuery, pero se debe modificar mínimamente para adaptarlo al proyecto que se esté desarrollando.

Abriendo el archivo con un editor hay que especificar en las líneas 14 y 15 el nombre del *bucket* de Cloud Storage que contiene los archivos CSV y el carácter utilizado para delimitar cada campo en los ficheros.

```
13
14  STORAGE_BUCKET = 'preving-bajas'
15  CSV_FIELD_DELIMITER = ';'
16
```

Con este archivo listo, se puede subir al editor de Cloud Shell y crear la plantilla. Para esto es necesario preparar el entorno e instalar las dependencias necesarias:

1. Instalar y ejecutar virtualenv

En la consola de Cloud Shell ejecutar:

- pip install virtualenv
- virtualenv -p python3 venv
- . venv/bin/activate (incluir el punto del principio)

2. Instalar las dependencias del código

- pip install apache-beam[gcp]
- pip install google-cloud-storage

Cuando terminen estos procesos, el entorno está preparado para ejecutar el archivo '*data_ingest.py',* para hacerlo, desde la consola, se tiene que ejecutar con los siguientes parámetros:

python3 [ruta_hasta_el_archivo/]data_ingest.py

--project=[ID_del_proyecto]

--region=europe-west1

--temp_location=gs://[nombre_del_bucket]/tmp

--staging_location=gs://[nombre_del_bucket]/tmp

--runner=DataflowRunner

--experiments=use_beam_bq_sink

--template_location=gs://[nombre_del_bucket]/templates/extract-csv-template

Esto creará la plantilla con nombre *'extract-csv-template'*, en la carpeta '*templates'* del bucket que se haya indicado, ahora, se pueden hacer llamadas HTTP a esta plantilla para ejecutarla, cosa que se hará desde la Cloud Funtion del apartado siguiente.

2. Creación de la Cloud Function para activar la plantilla de Dataflow

La función que se va a crear es la encargada de ejecutar la pipeline de Dataflow que se ha creado en el apartado anterior para cada archivo CSV que se sube al bucket que indiquemos.

Para empezar, debemos crear la función en Google Cloud Functions, dándole un nombre único y asignándole la región 'europe-west1'. Dentro del apartado 'Activador' se debe especificar:

- Tipo de activador: Cloud Storage
- Event type: Finalizar/Crear
- Segmento: seleccionar bucket donde se almacenan los CSV

Tras esto, se pasa al siguiente paso, que será modificar el código que se adjunta a esta memoria y subirlo.

Editar la función de python

Desde un editor de texto, se puede abrir el archivo *main.py* y configurar las constantes de las primeras líneas para adaptarlo al proyecto en desarrollo:

```
7   PROJECT_ID        = 'preving-bajas-301512'
8   SCHEMAS_LOCATION  = 'Bajas/schemas/'
9   TEMPLATE_LOCATION = 'gs://preving-bajas/templates/extract-csv-template'
10  OUTPUT_DATASET    = 'BAJAS_BQ_STG.'
11  CSV_SEPARATOR     = ';'
```

- PROJECT_ID: id del proyecto
- SCHEMAS_LOCATION: directorio del Cloud Storage donde se almacenan los archivos JSON con los esquemas de las tablas. Cada vez que se suba un archivo CSV, del que no se conoce su esquema, la función lo generará automáticamente en esta localización.
- TEMPLATE_LOCATION: ruta y nombre de la plantilla de Dataflow que se ha creado anteriormente, se corresponde con el parámetro –*template_location* que se ha indicado al crearla
- OUTPUT_DATASET: dataset de BigQuery donde se almacenarán las tablas resultantes de la operación.
- CSV_SEPARATOR: separador de columnas que se ha utilizado en los archivos CSV.

Subir el código a Cloud Functions

Volviendo a la pantalla de Cloud Functions, se debe seleccionar :

- Entorno de ejecución: Python3.8
- Punto de entrada: csv_to_template - (es el nombre de la función de entrada)

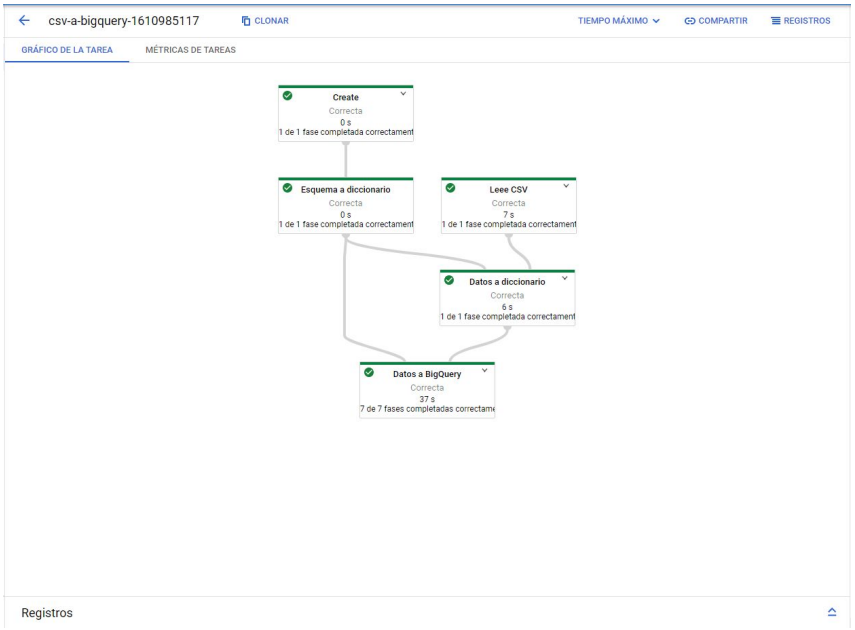
Para subir el código se puede hacer o bien copiando el contenido de *main.py* y *requirements.txt* en los archivos con el mismo nombre en la interfaz de Cloud Functions o comprimiendo ambos en un *.zip* y subiéndolo.

Con esto hecho, podremos desplegar la función y comprobar que, al subir un CSV nuevo, creará, cuando finalice la tarea, una tabla con el mismo nombre que este fichero en el dataset indicado.

3. Depuración y visualización de registros

3.1 Depuración en Dataflow

Cuando se sube un archivo al bucket indicado, se crea una tarea con el nombre *csv-a-bigquery-[marcca de tiempo]*, se puede acceder a este trabajo accediendo a la herramienta de Dataflow y haciendo click sobre él, se mostrarán todas las fases de la pipeline, y, si se ha producido algún error en alguna de estas se puede hacer click sobre ella y abrir el panel de registros situado en la parte inferior de la pantalla.



3.2 Depuración en Cloud Functions

Si comprobamos que no se crea la tarea de Dataflow correspondiente a una ejecución de la Cloud Function, o se sospecha que un determinado error puede tener origen en esta, se pueden visualizar los registros generados desde los detalles de la función.

Importante: tanto para visualizar los registros de Dataflow como los de Cloud Functions, es necesario tener el rol de **visor de registros** en la cuenta donde se está trabajando.

4. Automatización del ETL

Tras la subida de los datos a BigQuery en un dataset de Staging, es común necesitar un proceso de ETL para transformar estos datos y adaptarlos a las necesidades de la aplicación final. Para ello se utilizarán los registros que se crean durante el proceso anterior para la ejecución de una nueva Cloud Function encargada de controlar la ejecución, en orden, de las vistas necesarias para este proceso.

4.1 Declaración de las dependencias entre vistas

Es importante tener claro las dependencias que cada vista que se desea ejecutar tiene sobre las tablas del proyecto, es decir, qué tablas deben estar actualizadas antes de lanzarse, y, para asegurar que la vista solo se ejecuta cuando todas sus dependencias están actualizadas se va a utilizar una hoja de cálculo de Google con tres hojas: Grupo, Dependencias y Vistas.

- **Grupo**: Las vistas se van a agrupar en grupos que no tengan dependencias entre sí y, por lo tanto, se pueden ejecutar paralelamente. Se debe seguir este formato:

	A	B
1	Grupo	Estado
2	1	ESPERA
3	2	ESPERA
4	3	ESPERA
5	4	ESPERA
6	5	ESPERA
7	6	ESPERA
8		

- **Vistas**: Define que vistas perteneces a cada grupo, se debe especificar, el nombre de la vista, el grupo al que pertenece, la tabla y el dataset de destino:

	A	B	C	D
1	Vista_ETL	Grupo	Tabla_Destino	Dataset_Destino
2	dim_Ofertas_Bajas	1	dim_Ofertas_Bajas	BAJAS_BQ_DWH
3	dim_Usuarios	1	dim_Usuarios	BAJAS_BQ_DWH
4	dim_Provincias	1	dim_Provincias	BAJAS_BQ_DWH
5	dim_Clientes	1	dim_Clientes	BAJAS_BQ_DWH
6	dim_Usuarios_Provincias	1	dim_Usuarios_Provincias	BAJAS_BQ_DWH
7	Ofertas_Vivos	2	Ofertas_Vivos	BAJAS_BQ_DWH
8	Hechos_Ofertas	3	Hechos_Ofertas	BAJAS_BQ_DWH
9	Hechos_Objetivo	3	Hechos_Objetivo	BAJAS_BQ_DWH
10	Hechos_Bajas	4	Hechos_Bajas	BAJAS_BQ_DWH
11	Tabla_Hechos	5	Tabla_Hechos	BAJAS_BQ_HECHOS
12	Tabla_Hechos_FActualizacion	6	Tabla_Hechos_FActualizacion	BAJAS_BQ_HECHOS
13				

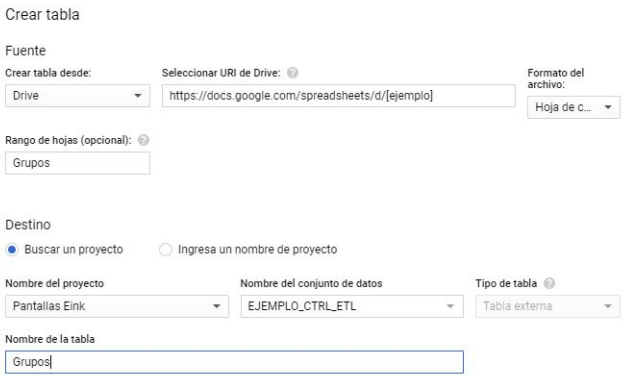
- **Dependencias:** Especifica que tablas deben ser actualizadas antes de ejecutar las vistas de cada grupo:

	A	B	C
1	Grupo	Tabla	Estado
2	1	VISTA_CLIENTES	ESPERA
3	1	qs_provincias	ESPERA
4	1	QS_USUARIOS_SISTEL	ESPERA
5	1	QS_OFERTAS_BIGQUER	ESPERA
6	2	dim_Ofertas_Bajas	ESPERA
7	2	dim_Clientes	ESPERA
8	3	Ofertas_Vivos	ESPERA
9	4	bq_solicitudes_baja	ESPERA
10	4	dim_Ofertas_Bajas	ESPERA

4.2 Construcción del dataset de configuración del ETL

Una vez se tiene este documento terminado y almacenado en un repositorio de Google Drive, se debe crear en BigQuery, un dataset que contenga las tablas que beben de estas tres hojas que se acaban de describir y con el mismo nombre, además de una vista para cada tabla que facilite la recarga de estas tablas en caso de que se necesite cambiar.

Por ejemplo, para tabla ‘Grupos’ dentro del dataset ‘EJEMPLO_CONFIG_ETL’, se puede crear de la siguiente manera:



Finalmente, la tabla final que consumirá la Cloud Function se debe colocar en un dataset denominado 'EJEMPLO_CTRL_ETL', que se tiene que construir a partir de la tabla externa que se acaba de crear. Se puede hacer a partir de una vista que almacene una consulta del estilo:

```
SELECT * from `proyecto_ejemplo.EJEMPLO_CONFIG_ETL.Grupos`
```

Por lo tanto, la tabla final debe ser 'proyecto_ejemplo.EJEMPLO_CTRL_ETL.Grupos' y se deben repetir estos pasos para las tablas de 'Vistas' y de 'Dependencias'.

Una vez se tienen las tres tablas anteriores en el dataset de control del ETL que utilizará la Cloud Function, se van a canalizar los registros creados cada vez que se actualiza una tabla del proyecto para activar esta función.

4.3 Canalización de los registros a un tema de Pub/Sub

Desde la aplicación en Google Cloud de 'Logging', se puede acceder, desde el menú lateral de la izquierda de la pantalla a la herramienta de 'Enrutamiento de registros'.

Desde aquí, se puede crear un nuevo enrutador de la siguiente manera:

1. **Detalles del receptor.**
   a. Nombre del receptor: Nos servirá para identificar esta configuración.
2. **Destino del receptor.**
   a. Se selecciona 'Tema de Cloud Pub/Sub' como servicio del receptor.
   b. Se crea un tema nuevo de 'Pub/Sub' con un nombre fácil de identificar y relacionado con su finalidad.
3. **Elige registros para incluirlos en el receptor.**
   a. En esta sección se especifican los registros que servirán para activar la Cloud Function, se deben introducir los siguientes filtros:

```
resource.type="bigquery_resource" AND

protoPayload.methodName="jobservice.jobcompleted" AND

protoPayload.authenticationInfo.principalEmail : "gserviceaccount.com" AND

(

protoPayload.serviceData.jobCompletedEvent.job.jobConfiguration.query.destina
tionTable.datasetId=("BAJAS_BQ_DWH" OR "BAJAS_BQ_HECHOS") OR

protoPayload.serviceData.jobCompletedEvent.job.jobConfiguration.load.destinat
ionTable.datasetId=("BAJAS_BQ_STG")

)
```

   Cambiando los nombres de los proyectos y de los datasets para adecuarlos al proyecto que se está desarrollando.

   Estos filtros obtienen los trabajos de tipo 'bigquery.resource' que se hayan completado, lanzados por una cuenta de servicio y que hayan operado sobre los datasets 'BAJAS_BQ_DWH', 'BAJAS_BQ_HECHOS' y 'BAJAS_BQ_STG'.

El cuarto y último paso se puede saltar y crear el sumidero.

4.4 Creación de la Cloud Function para la ejecución del ETL

Para terminar, se debe crear la Cloud Function encargada de ejecutar las vistas que conforman el proceso de ETL que se han definido en el dataset de control de este proceso. Para ello, desde el menú para crear una nueva función se deben definir los siguientes valores:

1. **Nombre de la función.**
2. **Región**: Debe coincidir con la ubicación de la base de datos que se debe consultar.
3. **Activador**:
    a. **Tipo de activador**: Cloud Pub/Sub.
    b. Se seleccionará el tema creado en el apartado anterior, donde se enrutan los registros.

Finalmente en la sección siguiente se debe subir el código adjunto a esta memoria, modificando las líneas 8, 9 y 10 para adecuarlas al proyecto:

```
8   PROJECT_ID = "proyecto_ejemplo"
9   ETL_DATASET = "EJEMPLO_BQ_ETL"
10  CTRL_DATASET = "EJEMPLO_CTRL_ETL"
```

Además, se seleccionan los siguientes parámetros de configuración:

1. **Entorno de ejecución:** Python 3.8
2. **Punto de entrada:** run_etl

Con todo esto, se puede hacer clic en 'Implementar' y esperar a que se despliegue la función. All terminar este proceso, el ETL comenzará cada vez que se suban los archivos que activan la plantilla de Dataflow, que a su vez, modifica las tablas dentro de los datasets que enrutan sus registros a esta última Cloud Function, que se ejecutará tantas veces como sea necesario hasta ejecutar todas las vistas definidas anteriormente.