Propuesta de esquema de trabajo SIGMA

Responsable	Fecha de modificación	Razon

Este documento se propone explicar cómo funciona y como trabajar con el esquema de trabajo desarrollo en GCP. Este esquema se propone realizar un proceso end-to-end del procesamiento de un archivo. Desde su extracción en Cloud Storage hasta su explotación en Bigquery.

El documento se ira actualizando a medida que surjan mejoras o alternativas a lo propuesto. La idea es usarse como referencia para nuevos y futuros proyectos.

Cada vez que se realice un cambio "brusco", esto es, nuevas tecnologías. Se debe añadir a la documentación, indicar el responsable y la fecha en la que se modificó.

El proceso contiene los siguientes recursos:

1. Cloud Storage
    a. company_trigger_bucket: en este bucket se carga los archivos a procesar "encriptados" con fernet y "comprimidos" por gzip. Esta carga de archivos se hace desde on-premise o desde la fuente de origen correspondiente. Se llama "trigger" puesto que en la carga de un archivo se "gatilla" un workflow que inicia todo el proceso.
    b. company_output_bucket: en este bucket se almacenan los archivos del bucket "company_trigger_bucket" **desencriptados.** La desencriptación la realiza una cloud function llamada por el workflow "gatillado". Notar que solo se almacena el archivo "desencriptado" pero sigue estando "comprimido".
2. Workflow: el workflow es un archivo .yaml o .json que describe una serie de pasos a realizar. En este caso particular, el workflow:
    a. Se ejecuta cuando se sube un archivo al bucket company_trigger_bucket
    b. Ejecuta la cloud function con un http.post enviandole la data del bucket y el archivo como payload. Ejemplo de payload:

{"bucket": "company_trigger_bucket", "name": "data/yyyymmdd/lkp_fechas.gz.encrypted"}

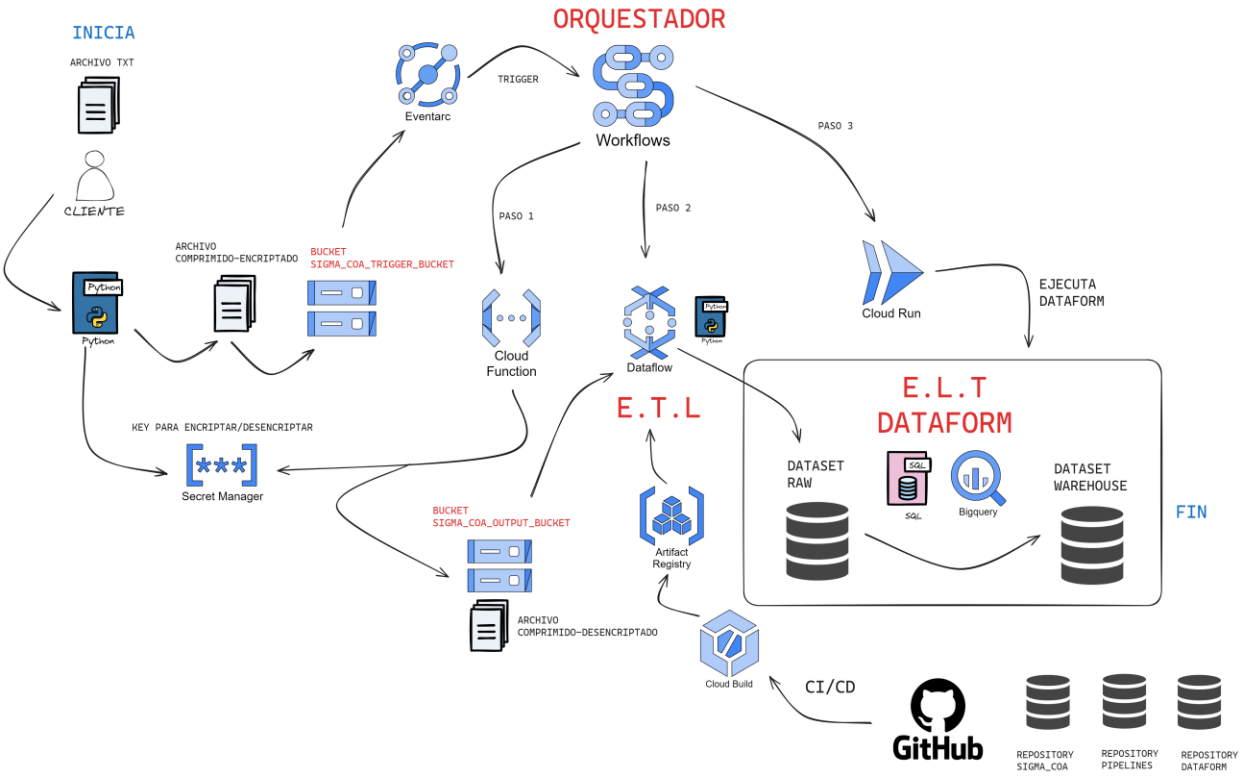
    c. Luego de que la cloud function termine, la misma envia un response con la informacion del bucket en el que almaceno el archivo, y el path al archivo "desencriptado". Ejemplo del response:

{"bucket": "company_output_bucket", "name": "data/yyyymmdd/lkp_fechas.gz"}

    d. El workflow procede a ejecutar un job en dataflow el cual es un flex-template que acepta como parametro principal el uri del archivo comprimido. Siguiendo con el ejemplo, tiene la forma:
    "gs://company_output_bucket/data/yyyymmdd/lkp_fechas.gz"
    e. Terminado el dataflow, por último "invoca" un workflow en dataform.
3. Artifact Registry: este es un repositorio **privado** para almacenar artefactos. Los artefactos pueden ser imágenes Docker, como también paquetes de Python (wheels, tar.gz, etc.) y de otros lenguajes como Java. En él se almacena por un lado la librería "company" y las imágenes docker de los flex-template o de los pipelines de dataflow.

    a. Repo “sigma-pipelines”: aquí se almacena los docker images de los flex-template.
    b. Repo “sigma-docker-build": se almacenan los docker image de las cloud function gen2 y las cloudruns.
4. Repo "sigma-pkg": Aquí es donde se almacenan los paquetes python. En especial, “company” que es el nombre que se le asigno al paquete python que contiene las funciones típicas que se emplean en la creación de pipelines o en cualquier script de python. En ella se almacenan también los SCHEMAS de las fuentes de datos y de las tablas en bigquery. Estos schemas se definen con la librería “pydantic”. Importante: cada vez que se va a cargar una nueva interfaz o archivo, previamente se tiene que haber agregado el modelo a la libreria, caso contrario el pipeline ejecutado por dataflow no lo va a encontrar, y generara un error.
5. Flex-Template: cada pipeline es un conjunto de archivos python que realizan una extracción, transformación, y su posterior load o almacenamiento. Dataflow ES la herramienta de ETL en GCP. Los flex-template son necesarios puesto que son la manera de ejecutar job de dataflow por medio de la REST API u operadores de airflow. Caso contrario, habría que ejecutar un bash script asegurando el mismo enviroment siempre. Estos flex-template se definen con 2 archivos: la imagen Docker que contiene el script python del pipeline, junto con sus dependencias, y un flex-template.json que apunta a la imagen Docker almacenada en Artifact Registry y que indica a su vez los “inputs” que acepta.
6. Bigquery: momentáneamente se definen los siguientes dataset, se planea en el futuro ampliar según las necesidades del negocio.
    a. Dataset raw: este es el dataset en que se almacena la data extraída de cloud storage y transformada por los dataflow.
    b. Dataset warehouse: este es el dataset en el que se almacena las tablas producto de las tablas en el dataset raw. Es el resultado de un proceso de ELT generado por la herramienta Dataform.
7. Dataform es la herramienta de ELT de GCP. Su hermana gemela es la famosa herramienta DBT. Ambas realizan ELT, es decir, el procesamiento ocurre en el mismo lugar en el que se almacena la data, por lo que se usa, en este caso, SQL. Dataform trabaja con archivos. SQLX que son archivos SQL con el agregado de un bloque de configuración “config” en el que se le indica el tipo de tabla u output a generar (tabla, vista, vista materializada, etc.), metadata respecto del output (descripción de las columnas, particiones, etc.), dependencias, y además se le puede agregar assertions que son testeos sobre el output. Con esto último tenemos la posibilidad de manejar la calidad del dato desde Dataform. Además, de SQL se puede usar JavaScript para realizar operaciones. Por último, dataform está conectado con un repositorio GIT para tener control de versiones. En él, cada desarrollador es dueño de su propio workspace en el que realiza las pruebas necesarias antes de guardar “pushear” los cambios a la rama principal que es la que se usara en producción. Dataform permite las buenas prácticas del código modernas como lo es el control de versiones, testeo, isolation de desarrollo, con SQL.
8. Cloud Run: por último, se utiliza una cloud run a la cual se le pasa el archivo cargado o que “triggereo” al Workflow, por ejemplo, “lkp_fechas”. Esta cloud run lo que hacees invocar la ejecucion de un workflow de dataform que incluya todas las tablas, operaciones, assertions, que dependan de manera directa e indirecta de “lpk_fechas”. Esto es, si tabla X se construye a partir de la tabla “lkp_fechas” y la tabla Y a partir de la X, entonces se ejecuta tabla X y tabla Y. Este es el último step del workflow.

De esta manera, la arquitectura final es la siguiente:


INICIA
ARCHIVO TXT
CLIENTE
ORQUESTADOR
Eventarc
TRIGGER
Workflows
PASO 3
PASO 1
PASO 2
ARCHIVO
COMPRIMIDO-ENCRIPTADO
BUCKET
SIGMA_COA_TRIGGER_BUCKET
Python
Cloud Function
Dataflow
Cloud Run
EJECUTA
DATAFORM
E.L.T
DATAFORM
E.T.L
DATASET
RAW
SQL
Bigquery
DATASET
WAREHOUSE
FIN
KEY PARA ENCRIPTAR/DESENCRIPTAR
Secret Manager
BUCKET
SIGMA_COA_OUTPUT_BUCKET
Artifact
Registry
ARCHIVO
COMPRIMIDO-DESENCRIPTADO
Cloud Build
CI/CD
GitHub
REPOSITORY
SIGMA_COA
REPOSITORY
PIPELINES
REPOSITORY
DATAFORM