Documento del Proyecto de Data Warehouse en GCP

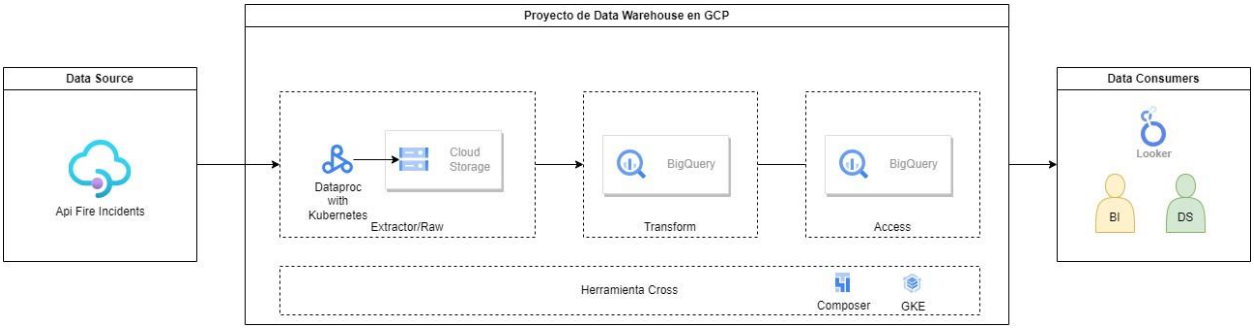
Introducción

Este documento describe el desarrollo de un sistema de Business Intelligence (BI) para el departamento de gestión de emergencias de San Francisco, centrado en el análisis de datos de incidentes de incendios. El objetivo es automatizar la ingesta, transformación y almacenamiento de datos para facilitar análisis eficientes y toma de decisiones informada.

Objetivo del Proyecto

- Automatizar la Recolección de Datos con Enfoque en Escalabilidad y Microservicios: Desarrollar un sistema que no solo automatice la ingesta diaria de datos de una fuente pública, sino que también esté diseñado para escalar dinámicamente y se base en una arquitectura de microservicios.
- Procesamiento y Transformación de Datos: Transformar los datos crudos para análisis.
- Almacenamiento Eficaz: Utilizar BigQuery para almacenar datos procesados.
- Facilitar el Análisis BI: Proveer datos estructurados para el análisis por el equipo de BI.

Arquitectura de Datos



1. Capa de Extracción

- **Objetivo**: Extraer datos de la fuente pública y almacenarlos en Cloud Storage.
- **Componentes**: Dataproc desplegado con Kubernetes, con tareas programadas en Cloud Composer para ejecutar scripts de Python que realicen la extracción de datos.
- **Archivo Python**: 1_extract_api_fire.py

2. Capa de Transformación

- **Objetivo**: Transformar y limpiar los datos para su análisis.
- **Componentes**: Dataproc desplegado con Kubernetes, con tareas programadas en Cloud Composer para ejecutar scripts de Python que realicen la transformación de datos. Se procesarán los datos desde Cloud Storage y se cargarán en BigQuery.
- **Archivo Python**: 2_transform_fire.py

3. Capa de Acceso

- **Objetivo**: Proveer datos estructurados para el análisis de BI.
- **Componentes**: Big query, con tareas programadas en Cloud Composer para ejecutar scripts de Python para disponibilizar para BI.
- **Archivo Python**: 3_access_fire.py

4. Orquestamiendo (Cloud Composer)

- **Objetivo**: Orquesta el flujo de datos de la API hasta la capa de acceso.
- **Componentes**: Cloud Composer para disparar procesos en el Dataproc – Kubernetes para los scrips de Python de datos.
- **Archivo Python**: 4_orchestrate.py

Estructura de Datos

Tabla de Hechos: fact_incidents

- Contiene información detallada sobre cada incidente, como número del incidente, dirección, fecha y hora, y otros campos relevantes.

Tablas de Dimensiones

- **dim_time**: Almacena detalles de tiempo (fecha, año, mes, día).
- **dim_district:** Basada en zipcode y neighborhood_district.
- **dim_battalion:** Contiene información sobre los batallones involucrados.

Conclusión

- **Automatizar la Recolección de Datos con Enfoque en Escalabilidad y Microservicios:**
    - Se ha plantea la ingesta de datos con Kubernetes y Dataproc, coordinados mediante Cloud Composer para la ejecución de tareas. Esta estructura se basa en microservicios, lo que permite que cada parte del sistema opere de forma autónoma y autoescalable según demanda de recursos.
- **Procesamiento y Transformación de Datos:**
    - Los datos se procesan y transforman para análisis mediante scripts de Python en Cloud Dataproc, garantizando que estén transformados adecuadamente para su uso en BI.

- **Almacenamiento Eficaz:**
    - Utilizamos BigQuery para almacenar los datos transformados, aprovechando su escalabilidad y eficiencia en el manejo de grandes volúmenes de datos.

- **Facilitación del Análisis BI:**
    - Los datos se estructuran en tablas de hechos y dimensiones en BigQuery, optimizando las consultas para el equipo de BI y permitiendo análisis complejos y detallados.