## Diseño de la Arquitectura

### **Paso 1 de 6 - Contexto del Proyecto**

**Introducción**

En respuesta al rápido crecimiento y expansión de la organización, el equipo directivo ha identificado una necesidad crítica: unificar y aprovechar el creciente volumen de datos generados a través de nuestras diversas operaciones. Actualmente, la información clave, proveniente de fuentes como transacciones comerciales, interacciones con clientes y registros de sensores, se encuentra dispersa, lo que limita nuestra capacidad para realizar análisis integrales y tomar decisiones estratégicas basadas en datos de manera ágil y precisa.

**El Desafío**

La falta de una infraestructura de datos centralizada y escalable nos impide responder a preguntas de negocio complejas, optimizar operaciones y descubrir nuevas oportunidades de mercado. Para superar este desafío, se ha encomendado al equipo de ingeniería de datos el diseño de una solución robusta que sirva como la columna vertebral de nuestra estrategia de datos.

**Misión y Rol**

Como responsable técnico de esta iniciativa, mi rol es diseñar la arquitectura de un pipeline de datos de tipo **ELT (Extract, Load, Transform)** y un **Data Warehouse (DW)** escalable, desplegado sobre infraestructura en la nube. Este diseño sentará las bases para:

1. **Integrar eficientemente** datos de múltiples fuentes, comenzando con el análisis de datos de mercado de Airbnb en Nueva York como prueba de concepto.
2. **Seleccionar un stack tecnológico** moderno que garantice escalabilidad, rendimiento y facilidad de integración, preparándonos para futuros aumentos en el volumen y la variedad de los datos.
3. **Definir un flujo de datos estructurado** a través de distintas capas dentro del Data Warehouse (Raw, Staging, Core y Gold), asegurando la calidad, gobernanza y accesibilidad de los datos.

Este documento servirá como el mapa maestro que guiará la implementación técnica del proyecto, asegurando que la solución final esté alineada con los objetivos estratégicos de la empresa y proporcione una ventaja competitiva sostenible.

### **Paso 2 de 6 - Descripción General del Pipeline ELT**

**Objetivo del Pipeline**

El objetivo principal de este pipeline es **automatizar el proceso de ingesta, almacenamiento y transformación de datos** provenientes de diversas fuentes. Buscamos crear un flujo de trabajo robusto, escalable y mantenible que convierta datos en crudo en activos de información listos para ser consumidos por analistas de negocio, científicos de datos y herramientas de visualización, permitiendo así responder a las preguntas estratégicas de la organización de manera eficiente.

**Contexto y Necesidades del Proyecto**

Este proyecto responde a la necesidad de pasar de un ecosistema de datos fragmentado a una **fuente única de verdad (Single Source of Truth)**. La arquitectura ELT sobre Google Cloud Platform (GCP) se elige específicamente por su capacidad para manejar grandes volúmenes de datos y su flexibilidad. A diferencia del enfoque tradicional ETL, donde las transformaciones ocurren antes de la carga, el modelo ELT aprovecha la inmensa capacidad de cómputo de los Data Warehouses modernos en la nube, como **Google BigQuery**, para realizar las transformaciones *in-situ*, lo que resulta en una mayor velocidad de ingesta y una mayor agilidad para adaptar las transformaciones a nuevas necesidades de negocio.

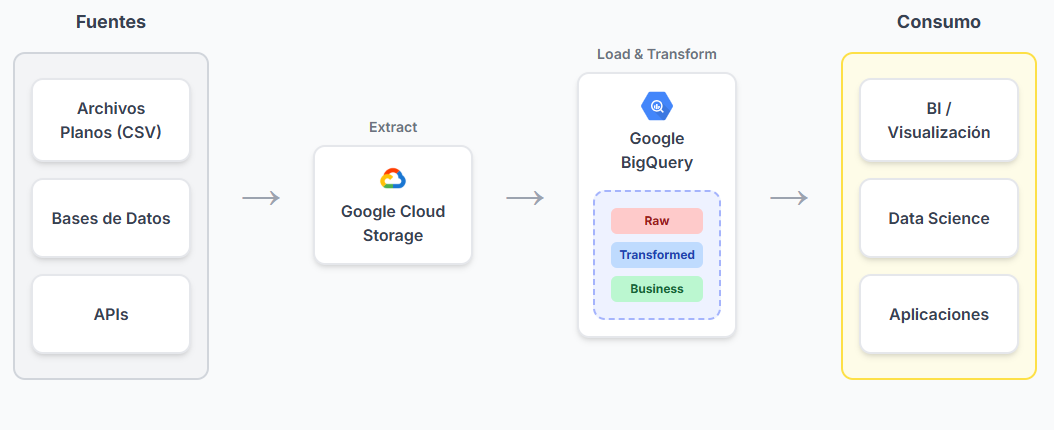
**Descripción de las Etapas ELT**

El pipeline se compone de tres etapas fundamentales, orquestadas por **Google Cloud Composer (Apache Airflow)**:

1. **Extract (Extraer):** En esta primera fase, los datos se extraen de sus sistemas de origen. Para nuestra prueba de concepto, esto implicará tomar el archivo AB\_NYC.csv y otros archivos planos. En futuras iteraciones, se conectará a bases de datos transaccionales (ej. PostgreSQL, MySQL) y APIs de servicios externos. Los datos extraídos se depositan en su formato original en un área de aterrizaje (landing zone) en **Google Cloud Storage (GCS)**, un servicio de almacenamiento de objetos altamente durable y escalable.
2. **Load (Cargar):** Una vez que los datos crudos se encuentran en GCS, la segunda etapa consiste en cargarlos directamente en nuestro Data Warehouse, **Google BigQuery**. Esta carga se realiza de forma masiva y rápida, sin aplicar transformaciones previas. El resultado es una copia exacta de los datos de origen dentro de una capa "raw" en BigQuery, lo que garantiza que siempre tengamos el dato original disponible para auditoría o re-procesamiento.
3. **Transform (Transformar):** Esta es la etapa final y la más crítica. Utilizando el poder de procesamiento de **Google BigQuery**, se ejecutan scripts de SQL para limpiar, estandarizar, enriquecer y modelar los datos. Estas transformaciones convierten los datos crudos en tablas limpias y estructuradas, y finalmente en modelos de datos agregados (capa "gold") que están optimizados para el análisis y responden directamente a las preguntas de negocio. Este proceso se gestionará de manera eficiente, idealmente a través de herramientas como dbt (Data Build Tool), que se integra con BigQuery y facilita el desarrollo, prueba y versionado de las transformaciones SQL.

### **Paso 3 de 6 - Diagrama de Arquitectura**

A continuación, se presenta la descripción detallada de la arquitectura del pipeline ELT, que servirá de base para la creación del diagrama técnico.



**Componentes de la Arquitectura:**

1. **Fuentes de Datos (Data Sources):**
   * **Archivos Planos (Batch):** AB\_NYC.csv como fuente inicial. Representa datos históricos o cargas masivas.
   * **Bases de Datos Transaccionales (Futuro):** Sistemas como PostgreSQL, MySQL que contienen datos de operaciones en tiempo real.
   * **APIs de Terceros (Futuro):** Fuentes de datos externas para enriquecimiento (ej. datos demográficos, clima, etc.).
2. **Capa de Ingesta (Ingestion Layer):**
   * **Google Cloud Storage (GCS):** Actúa como nuestra *Data Lake* o zona de aterrizaje. Todos los datos extraídos, sin importar su origen, se depositan aquí en su formato crudo. Esto desacopla la extracción de la carga y proporciona un repositorio centralizado y económico.
3. **Orquestación (Orchestration):**
   * **Google Cloud Composer (Apache Airflow):** Es el cerebro del pipeline. Define, programa y monitorea los flujos de trabajo (DAGs). Se encarga de:
     + Activar la extracción de datos de las fuentes.
     + Copiar los datos a GCS.
     + Iniciar los trabajos de carga de GCS a BigQuery.
     + Ejecutar los scripts de transformación SQL en BigQuery en la secuencia correcta.
4. **Data Warehouse:**
   * **Google BigQuery:** Es el núcleo de nuestra arquitectura. Un Data Warehouse serverless, escalable y de alto rendimiento donde los datos se almacenan, procesan y modelan. Se estructura internamente en las siguientes capas:
     + **Capa Raw:** Contiene los datos tal como fueron cargados desde GCS.
     + **Capa Staging/Transformed:** Datos limpios, estandarizados y estructurados.
     + **Capa Gold/Business:** Modelos de datos agregados y listos para el consumo.
5. **Capa de Transformación (Transformation Layer):**
   * **SQL (dentro de BigQuery):** Las transformaciones se ejecutan nativamente en BigQuery para máximo rendimiento.
   * **(Opcional) dbt (Data Build Tool):** Se integra con BigQuery para gestionar las transformaciones SQL de manera más robusta, permitiendo versionado, pruebas y documentación del código.
6. **Capa de Consumo (Consumption Layer):**
   * **Herramientas de BI y Visualización:** Looker, Tableau, Power BI. Se conectan a la capa *Gold* de BigQuery para crear dashboards e informes interactivos.
   * **Notebooks de Data Science:** Jupyter, Google Colab. Los científicos de datos acceden a los datos limpios para análisis exploratorio y entrenamiento de modelos de Machine Learning.
   * **Aplicaciones Analíticas:** Otras aplicaciones que necesiten consumir datos procesados.
7. **CI/CD (Integración y Despliegue Continuo):**
   * **GitHub:** Repositorio para todo el código del proyecto (DAGs de Airflow, scripts SQL, configuraciones de dbt).
   * **GitHub Actions:** Automatiza el proceso de prueba y despliegue de cambios en el código, asegurando la calidad y agilidad en el desarrollo.

**Flujo de Datos (Data Flow):**

El flujo sigue la secuencia ELT:

Fuentes de Datos → (Extract por Airflow) → Google Cloud Storage → (Load por Airflow) → BigQuery (Capa Raw) → (Transform por Airflow/dbt) → BigQuery (Capas Staging y Gold) → Capa de Consumo.

### **Paso 4 de 6 - Definición de Capas del Data Warehouse**

Para garantizar la gobernanza, calidad y mantenibilidad de los datos, nuestro Data Warehouse en **Google BigQuery** se estructurará siguiendo una arquitectura de múltiples capas, a menudo conocida como "Arquitectura Medallion". Cada capa tiene un propósito específico y representa un grado progresivo de refinamiento y agregación de los datos. En BigQuery, esto se implementará a través de *datasets* separados para cada capa.

**1. Capa Raw / Staging (Capa de Bronce)**

* **Propósito:** Esta es la primera y más básica capa del Data Warehouse. Su única función es almacenar una **copia exacta y sin procesar** de los datos tal como provienen de las fuentes de origen. Los datos aquí son inmutables y reflejan el estado de la fuente en el momento de la extracción.
* **Rol en el Proceso:**
  + **Aterrizaje de Datos:** Sirve como el punto de carga inicial desde Google Cloud Storage (GCS).
  + **Auditoría y Trazabilidad:** Proporciona un registro histórico completo que permite verificar la consistencia con la fuente original.
  + **Recuperación y Re-procesamiento:** Si se detectan errores en las lógicas de transformación posteriores, siempre podemos reconstruir las capas superiores a partir de estos datos crudos sin necesidad de volver a conectarnos a los sistemas de origen.
* **Ejemplo Práctico:**
  + **Dataset en BigQuery:** raw\_data
  + **Tabla:** raw\_data.airbnb\_listings
  + **Estructura:** La tabla tendrá exactamente las mismas 16 columnas que el archivo AB\_NYC.csv, con tipos de datos genéricos (mayoritariamente STRING) para asegurar una carga sin fallos.

**2. Capa Intermedia / Transformada (Capa de Plata)**

* **Propósito:** En esta capa, los datos de la capa Raw se someten a un proceso de **limpieza, estandarización, enriquecimiento y conformación**. El objetivo es transformar los datos crudos en un conjunto de tablas fiables, consistentes y listas para ser analizadas.
* **Rol en el Proceso:**
  + **Limpieza de Datos:** Manejo de valores nulos, eliminación de duplicados, corrección de formatos.
  + **Estandarización:** Aplicación de tipos de datos correctos (ej. convertir last\_review de STRING a DATE), estandarización de nombres de columnas (ej. a snake\_case), y normalización de valores categóricos.
  + **Enriquecimiento:** Se pueden añadir nuevas columnas derivadas de las existentes (ej. extraer el año de la última reseña) o cruzar con otras tablas de la misma capa para añadir contexto.
* **Ejemplo Práctico:**
  + **Dataset en BigQuery:** transformed\_data
  + **Tabla:** transformed\_data.listings\_cleaned
  + **Transformaciones Aplicadas:**
    - La columna price se convierte a FLOAT64.
    - La columna last\_review se convierte a DATE.
    - Se crea una nueva columna reviews\_per\_month\_imputed para manejar los nulos de reviews\_per\_month.
    - Se eliminan registros donde el precio es 0, ya que se consideran datos inválidos.

**3. Capa de Consumo / Modelo de Negocio (Capa de Oro)**

* **Propósito:** Esta es la capa final y la que será expuesta a los usuarios finales (analistas, herramientas de BI). Contiene **modelos de datos agregados, específicos para el negocio y optimizados para el rendimiento de las consultas**. Los datos aquí están desnormalizados y organizados para responder a preguntas de negocio específicas.
* **Rol en el Proceso:**
  + **Agregación:** Creación de tablas que resumen la información a un nivel útil para el análisis (ej. métricas diarias, mensuales, por barrio).
  + **Modelado de Negocio:** Construcción de vistas o tablas que representan entidades de negocio claras (ej. dim\_hosts, fact\_listings).
  + **Optimización de Consultas:** Las tablas están diseñadas para que las herramientas de BI puedan consultarlas de manera rápida y eficiente, sin necesidad de realizar complejos JOINs o cálculos en tiempo real.
* **Ejemplo Práctico:**
  + **Dataset en BigQuery:** business\_layer
  + **Tablas/Vistas:**
    - business\_layer.agg\_neighbourhood\_metrics: Una tabla agregada con columnas como neighbourhood, avg\_price, total\_listings, occupancy\_proxy.
    - business\_layer.dim\_hosts: Una tabla dimensional con información única de cada anfitrión.
    - business\_layer.fact\_reviews\_monthly: Una tabla de hechos que resume el número de reseñas por mes y por distrito.

### **Paso 5 de 6 - Justificación de Herramientas y Tecnologías**

La selección del stack tecnológico es una decisión estratégica que impacta directamente en la escalabilidad, el costo y la eficiencia del proyecto. A continuación, se justifica la elección de cada componente principal dentro del ecosistema de Google Cloud Platform (GCP).

**1. Google Cloud Platform (GCP) como Proveedor Cloud**

* **Justificación:** Se elige GCP como la plataforma en la nube subyacente debido a su liderazgo en servicios de datos y análisis, su ecosistema de servicios nativamente integrados y un modelo de precios competitivo y flexible.
  + **Escalabilidad:** GCP ofrece una infraestructura global que escala bajo demanda, ideal para una empresa en expansión.
  + **Facilidad de Integración:** Los servicios como GCS, BigQuery y Composer están diseñados para funcionar de manera sinérgica, reduciendo la complejidad de la integración.
  + **Adecuación:** Sus herramientas de Big Data son especialmente potentes para implementar arquitecturas ELT modernas.

**2. Google Cloud Storage (GCS) para la Capa de Ingesta**

* **Justificación:** GCS es la elección ideal para funcionar como nuestro Data Lake o zona de aterrizaje.
  + **Escalabilidad y Costo:** Ofrece almacenamiento de objetos virtualmente ilimitado a un costo muy bajo, perfecto para almacenar grandes volúmenes de datos crudos.
  + **Rendimiento y Durabilidad:** Proporciona alta disponibilidad y durabilidad de los datos, asegurando que nuestra materia prima esté siempre segura y accesible.
  + **Integración:** Se integra de forma nativa con BigQuery y Cloud Composer, permitiendo flujos de datos de alta velocidad desde el almacenamiento hasta el procesamiento.

**3. Google BigQuery como Data Warehouse**

* **Justificación:** BigQuery es el pilar central de nuestra arquitectura analítica.
  + **Escalabilidad y Rendimiento:** Su arquitectura *serverless* separa el cómputo del almacenamiento, permitiendo escalar ambos de forma independiente y automática. Su motor de procesamiento columnar (Dremel) ejecuta consultas sobre terabytes de datos en segundos.
  + **Adecuación al ELT:** Es la herramienta perfecta para el paradigma ELT. Su poder de cómputo permite realizar transformaciones complejas directamente sobre los datos cargados mediante SQL, simplificando el pipeline.
  + **Facilidad de Uso y Mantenimiento:** Al ser un servicio totalmente gestionado, elimina la necesidad de administrar infraestructura, permitiendo al equipo centrarse en la lógica de negocio y el análisis.

**4. Google Cloud Composer (Apache Airflow) para la Orquestación**

* **Justificación:** Para la orquestación de nuestros flujos de trabajo, se elige Cloud Composer, el servicio gestionado de Apache Airflow.
  + **Estándar de la Industria:** Apache Airflow es la herramienta de código abierto líder para la orquestación de pipelines de datos. Cloud Composer nos da todo su poder sin la carga de la administración.
  + **Flexibilidad:** Los pipelines se definen como código (Python), lo que permite una gran flexibilidad, control de versiones y la implementación de lógicas complejas de dependencias y reintentos.
  + **Integración:** Proporciona operadores pre-construidos para interactuar con todos los servicios de GCP (GCS, BigQuery), lo que simplifica enormemente el desarrollo de los DAGs.

**5. GitHub y GitHub Actions para CI/CD**

* **Justificación:** Para garantizar la calidad, mantenibilidad y agilidad del desarrollo, se implementará un flujo de CI/CD.
  + **Control de Versiones:** GitHub es el estándar de facto para el control de versiones de código, permitiendo la colaboración y el seguimiento de cambios en nuestros scripts SQL y DAGs de Airflow.
  + **Automatización:** GitHub Actions permite automatizar los flujos de trabajo de prueba y despliegue. Podemos configurar acciones para que, ante cada cambio en el código, se ejecuten pruebas de calidad automáticamente y, si son exitosas, se desplieguen los cambios a producción de forma segura.
  + **Adecuación:** Esta práctica de "Infraestructura como Código" es fundamental en proyectos de datos modernos para asegurar la reproducibilidad y fiabilidad del pipeline.

### **Paso 6 de 6 - Identificación y Análisis de Fuentes de Datos**

Este paso final valida que la fuente de datos seleccionada para la fase inicial del proyecto es adecuada para responder a las preguntas de negocio clave. Se establece una relación directa entre cada pregunta y los campos necesarios del dataset AB\_NYC.csv.

**Fuente de Datos Principal:**

* **Nombre:** AB\_NYC.csv
* **Tipo:** Archivo plano (CSV)
* **Descripción:** Contiene datos de listados de Airbnb en la ciudad de Nueva York, incluyendo información sobre el anfitrión, ubicación, precio, tipo de habitación y métricas de reseñas.

**Mapeo de Preguntas de Negocio a Campos de Datos:**

| **Pregunta de Negocio Clave** | **Campos Requeridos del CSV** | **Evaluación de Relevancia** |
| --- | --- | --- |
| **¿Cuál es el precio promedio de los alojamientos por barrio y distrito?** | price, neighbourhood, neighbourhood\_group | **Alta.** El dataset contiene todos los campos necesarios para agrupar y calcular el promedio de precios a los niveles geográficos requeridos. |
| **¿Qué tipo de habitación es el más ofrecido y cuál genera mayor revenue estimado?** | room\_type, price, availability\_365 | **Alta.** Se puede contar la frecuencia de room\_type. El revenue estimado se puede calcular como price \* (365 - availability\_365). |
| **¿Cuáles son los anfitriones con más propiedades listadas y cómo varían sus precios?** | host\_id, host\_name, calculated\_host\_listings\_count, price | **Alta.** El dataset permite agrupar por host\_id para contar propiedades y analizar la distribución de price para los anfitriones principales. |
| **¿Existen diferencias significativas en la disponibilidad anual entre barrios?** | availability\_365, neighbourhood | **Alta.** Es posible agrupar por neighbourhood y analizar las estadísticas descriptivas (promedio, mediana, desviación) de availability\_365. |
| **¿Cómo evoluciona el número de reseñas por mes en los diferentes distritos?** | last\_review, neighbourhood\_group | **Media.** El campo last\_review da una idea del momento de la última actividad, pero no un historial completo. Se puede usar como un proxy para analizar la tendencia más reciente. |
| **¿Qué barrios tienen la mayor concentración de alojamientos activos?** | neighbourhood, number\_of\_reviews | **Alta.** Se puede contar el número de listados por neighbourhood. Se puede usar number\_of\_reviews > 0 como un filtro para definir "activos". |
| **¿Cómo se distribuyen los precios y qué outliers existen?** | price | **Alta.** El campo price permite generar un histograma y un boxplot para analizar su distribución e identificar valores atípicos. |
| **¿Qué relación hay entre disponibilidad anual y cantidad de reseñas?** | availability\_365, number\_of\_reviews | **Alta.** Estos dos campos numéricos permiten crear un scatter plot para visualizar la correlación o falta de ella entre ambas variables. |

**Conclusión del Análisis de Fuentes:**

El archivo AB\_NYC.csv es una fuente de datos **altamente relevante y valiosa** para la fase inicial de este proyecto. Contiene la información necesaria para responder de manera sólida y confiable a la gran mayoría de las preguntas de negocio planteadas. Aunque para algunas preguntas (como la evolución histórica de reseñas) la información es limitada, sirve como un excelente punto de partida y prueba de concepto para la arquitectura ELT diseñada. En futuras fases, este dataset podrá ser enriquecido con fuentes de datos adicionales.