M3: DESARROLLO DE PIPELINE DE DATOS

Diseño de la arquitectura general del pipeline ELT: Airbnb NYC

Este proyecto abarca el desarrollo y la creación de un ***pipeline con enfoque en ELT end to end***. Esta arquitectura nos va a permitir recopilar, organizar y transformar la información proveniente de diferentes fuentes de datos. Como primera etapa inicial se cuenta con un archivo CSV con el histórico de reservaciones de propiedades en la ciudad de Nueva York, este diseño está enfocado en enriquecer la información con nuevas fuentes como APIs externas y Web scraping.

El propósito es sentar bases sólidas de gobernanza, calidad y escalabilidad, asegurando que la analítica futura pueda soportar decisiones estratégicas.

I. Objetivo del Pipeline:

Diseñar e implementar un pipeline con enfoque ELT robusto y escalable el cual permita recopilar, cargar y transformar datos de Airbnb NYC. Se va a generar un **data warehouse** con las siguientes etapas establecidas ***raw***, ***staging. Silver y Gold***. Lo que permitirá resolver las preguntas clave del negocio, así como insights estratégicos.

II. Preguntas clave de negocio

1. ¿Cuál es el precio promedio de los alojamientos por barrio y por distrito?
2. ¿Qué tipo de habitación es el más ofrecido y cual genera mayor revenue estimado?
3. ¿Cuáles son los anfitriones con más propiedades listadas y como varían sus precios?
4. ¿Existen diferencias significativas en la disponibilidad anual entre barrios o tipos de alojamiento?
5. ¿Cómo evoluciona el número de reseñas por mes en los diferentes distritos de la ciudad?
6. ¿Qué barrios tienen la mayor concentración de alojamientos activos?
7. ¿Cómo se distribuyen los precios y que outliers existen?
8. ¿Qué relación hay entre la disponibilidad anual y la cantidad de reseñas como proxy de ocupación?
9. ¿Qué boroughs presentan mayor Oferta per cápita (100k hab.) y mayor Densidad de oferta (listings/km^2)?

AUTOR: GUADALUPE RAMÍREZ PÉREZ

III. **Contexto y necesidades**.

- **Fuente actual:** Se cuenta con un archivo CSV con listado de los alojamientos de Airbnb NYC.
- Requerimientos previos: Limpieza, normalización de datos, cálculos de métricas, detección de outliers.
- **Enriquecimiento externo:**
  - ✓ **API de tipo de cambio:** Indicada para estandarizar precios en otra moneda (Pesos mexicanos) para comparaciones globales.
  - ✓ Scraping wikipedia: Wikipedia aporta **datos demográficos y espaciales** (población, km^2, densidad).

IV. **Descripción de las etapas ELT**

1. **Extract:**

- **Propósito:**
  En esta etapa se van a recopilar los datos desde las fuentes heterogéneas.
- Tipos de archivos:
  a) Archivo CSV: data set de Airbnb NYC.
  b) API tipo de cambio: JSON Extracción diaria de tipo de cambio.
  c) Web scraping: HTML → normalizado a JSON/CSV. Extracción de datos demográficos y espaciales.
- Rol en el procesamiento: Repositorio histórico y punto de recuperación

2. **Load:**

- **Proporsito:**
  Se cargan los datos crudos directamente a la data warehouse, capa RAW.
- Tipos de archivos: Los mismos obtenidos en la etapa Extract, pero almacenados como vistas en el data warehouse.
- **Rol de procesamiento:** Habilitar los datos para la transformación, auditorias y trazabilidad de los datos originales.

3. **Transform:**

- **Propósito:**

  Se realiza dentro del data warehause encargado de limpiar y modelar los datos.

  a) **Capa staging:** limpieza, estandarización, validaciones, tipificación de columnas y preparación de datos.

  b) **Capa Silver:** Modelo y creación de tablas dimensionales y de hechos con la granularidad adecuada.

  c) **Gold:** tablas entregables, listas para análisis de datos y que capaces de resolver las preguntas de negocio establecidas.

  - **Tipos de archivos:** Parquet/ORC dentro del DWH.
  - **Rol en el procesamiento:** convertir datos crudos en información estructurada y analítica, aplicando reglas de negocio y dejándolos listos para responder preguntas estratégicas.

V. Diagrama de arquitectura en AWS

La solución propuesta implementa un pipeline ELT en AWS. Integra contenedores Docker para la ingesta, Amazon S3 como Data Lake, DBT+ Redshift Serveless como Data Warehause, y GitHub Actions como motor CI/CD. El orquestador es Amazon MWAA (Airflow), ejecutando tareas en Amazon ECS Fargate (imágenes en Amazon ECR). Catalogo con AWS Glue QA/exploración con Amazon Athena.

1. Fuentes de datos
   - **CSV plano:** Data set Airbnb almacenado en Amazon S3 (bucket)
     Herramientas: S3(almacenamiento), Glue Crawler (catalogación), Athena (verificación rápida).
   - **APIs externas:** Consultas con API REST diarias/incrementales
     Herramientas: Contenedor Docker (Python + requests/httpx + tenacity), ECS Fargate, Secrets Manager (API keys), MWAA (DAG), CloudWatch (logs).
   - **AWS Scraping:**
     Herramientas: Contenedor Docker, ECS Fargate, MWAA, CloudWatch

Salida de Fuentes: S3 Landing/RAW (Parquet/CSV) particionado por fecha y por fuente.

2. **Ingesta y almacenamiento:**
   - Landing zone: Amazon S3 con buckets versionados para el archivo plano CSV, API y scraping.
   - RAW layer: S3 particionado por fecha AWS Glue Crawler para catalogar en AWS Glue Data Catalog.

3. **Transformación:**
    - **Staging:** Sin lógica del negocio solo higiene y conformidad de esquemas.
      Herramientas: DBT core corriendo en contenedor (ECS Fargate), orquestado por AirFlow y Redshift Serverless como destino.
    - **Silver:** Conformar entidades de negocio y relaciones estables para ser consumidas en capa gold.
      Herramientas: dbt en ECS; destino Redshift
    - **Gold:** Modelo dimensional (hechos y dimensiones) orientado a analítica, se definen, cálculos y agregaciones optimizadas para dashboards.
      Herramientas: dbt en ECS; destino Redshift Serverless

Capa	Herramienta	Uso	Entrada	Salida	Calidad	Justificación
Landing	S3 versionado	Recibir CSV/APIs/scraping	CSV/API/HTML	Archivos crudos		Bajo costo, durabilidad y versionado y permitir reprocesos sin perder el insumo original.
Raw	S3 + Glue Crawler/Catalog + Athena	Persistencia cruda + catálogo	Landing	Parquet/ Tablas en catálogo	QA con Athena	Separación de ingesta y transformación; Parquet + particiones abaratan consultas; Glue evita esquemas hardcode; Athena habilita QA sin clusters.
Staging	dbt en ECS + Redshift (stg)	Tipado/limpieza/ estandarización	Raw	Stg_*	dbt tests básicos	Limpieza de datos antes de responder preguntas de negocio, dbt aporta versionado, tests y linaje
Silver	dbt en ECS + Redshift (int)	Entidades canónicas + SCDs	Stg_*	Int_*	tests relaciones/únicos	Modelo canónico reutilizable; centraliza lógica transversal y SCD
Gold	dbt en ECS + Redshift	Hechos/Dimensiones + KPIs	Int_*	Fact_*	tests + docs/lineage	Rendimiento OLAP con Redshift y modelo dimensional para BI; dbt estandariza KPIs, documenta y prueba; base estable para dashboards.

Orquestación	MWAA (Airflow) + ECS Fargate/ECR	MWAA (Airflow) + ECS Fargate/ECR		Runs trazables	SLAs + alertas	Estándar de industria para dependencias, reintentos y backfills; ECS ejecuta jobs contenedorizados; ECR controla versiones e imágenes seguras.
Observabilidad	CloudWatch + SNS/Slack	Logs/métricas/alertas		Notificaciones	Alarmas por etapa	Visibilidad centralizada (logs/metrics) y alertas proactivas para reducir MTTR; enlaces directos a runs y consultas de diagnóstico.
CI/CD	GitHub Actions + OIDC	Build/Push a ECR + Deploy ECS/MWAA/dbt	Repo	Despliegues versionados	Smoke tests (Athena/Redshift)	Automatización segura sin llaves (OIDC), build reproducible (Docker), deploy trazable y smoke tests para verificar post-despliegue.

VI. Diagrama de Arquitectura en AWS - Pipeline ELT