INGENIERÍA DE DATOS

Proyecto final:

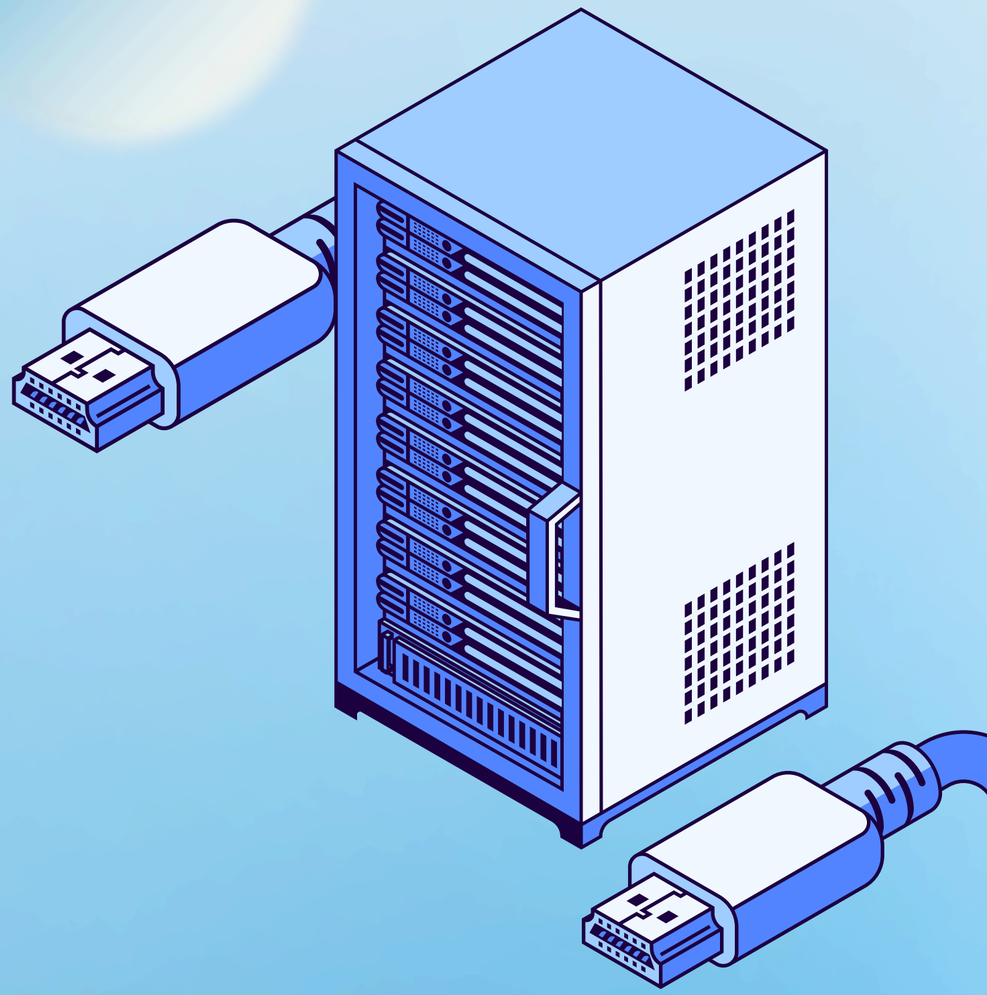
"Pipeline de Datos: Integración Batch y Near Real-Time con Kafka, Logstash y Elasticsearch"

Milene Muñoz

8 de febrero de 2025

1. OBJETIVO

Construir un pipeline híbrido que procese datos en batch y near real-time (NRT) usando herramientas eficientes. Se integrarán datos de distintas fuentes, se procesarán y almacenarán para análisis posterior.



2. HERRAMIENTAS UTILIZADAS

- **Kafka**

  Mensajería y buffer de datos.

- **Logstash**

  Ingesta y transformación de datos.

- **Elasticsearch**

  Almacenamiento y búsqueda.

- **Python**

  Scripts para generación y procesamiento.

- **Kibana**

  Visualización y análisis.

3. ARQUITECTURA DEL PIPELINE

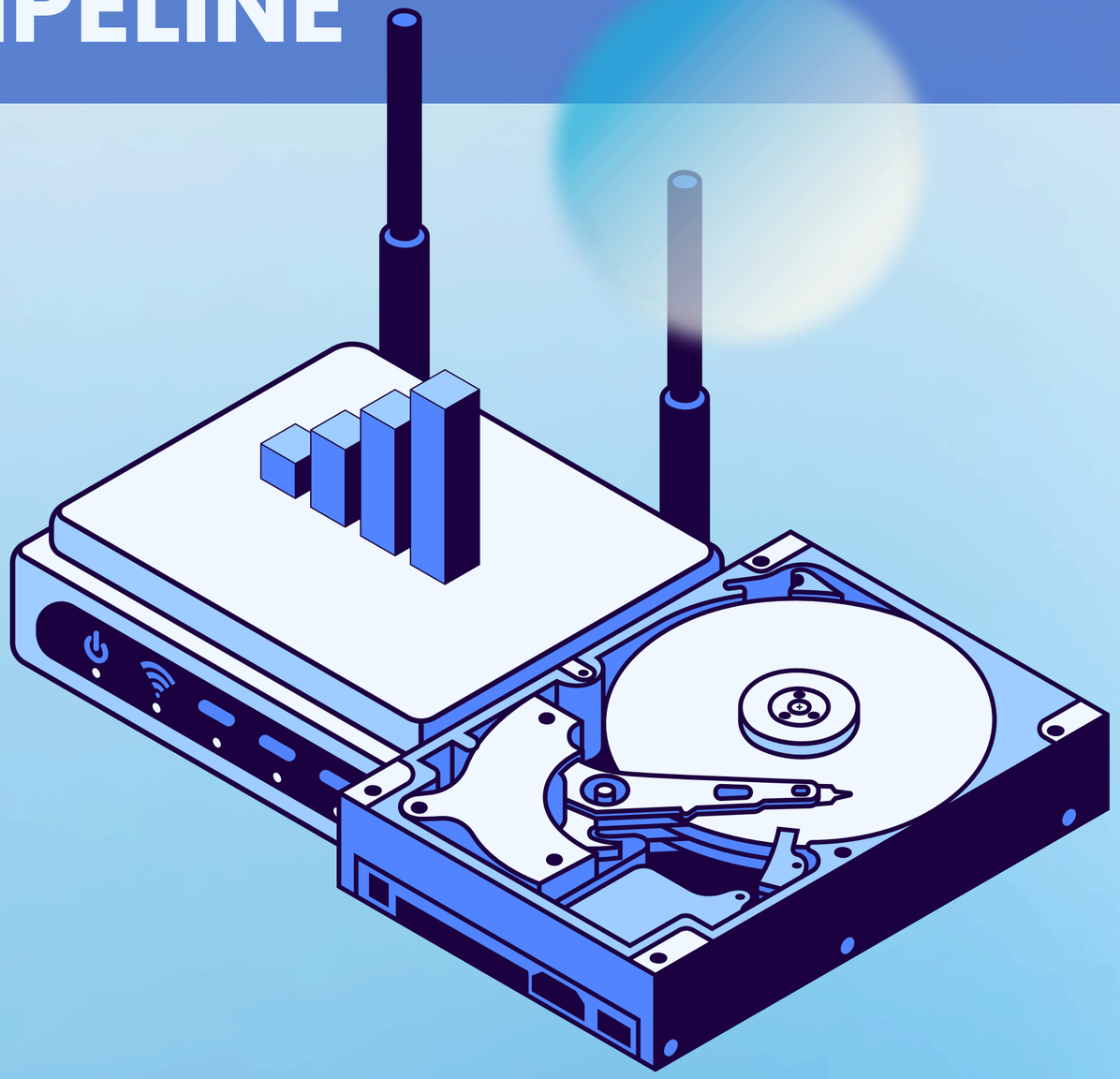
Flujo de datos

a. Flujo Batch (cada 24 horas)

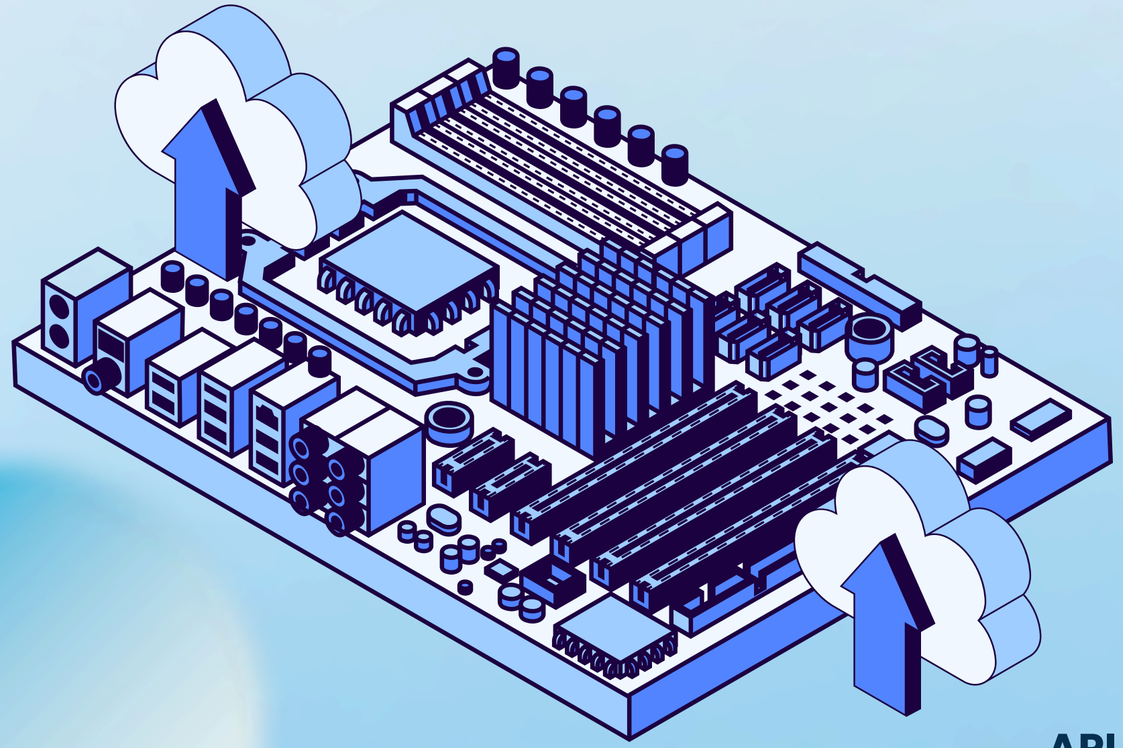
- Se extraen datos desde una API externa de datos financieros.
- Logstash transforma y almacena los datos en Elasticsearch.

b. Flujo Near Real-Time

- Logs de transacciones se envían a Kafka.
- Logstash los consume, transforma y los almacena en Elasticsearch.
- Elasticsearch indexa los datos y permite consultas rápidas.



4. FLUJO BATCH



Origen:

API financiera (Alpha Vantage).
Se utiliza un script para obtener datos de la API cada 24 horas.

Proceso:

- Script Python descarga datos diarios y genera un CSV.
- Logstash ingiere el CSV y lo envía a Elasticsearch.

API Financiera → Script Python (CSV) → Logstash → Elasticsearch

5. FLUJO NRT

Origen:

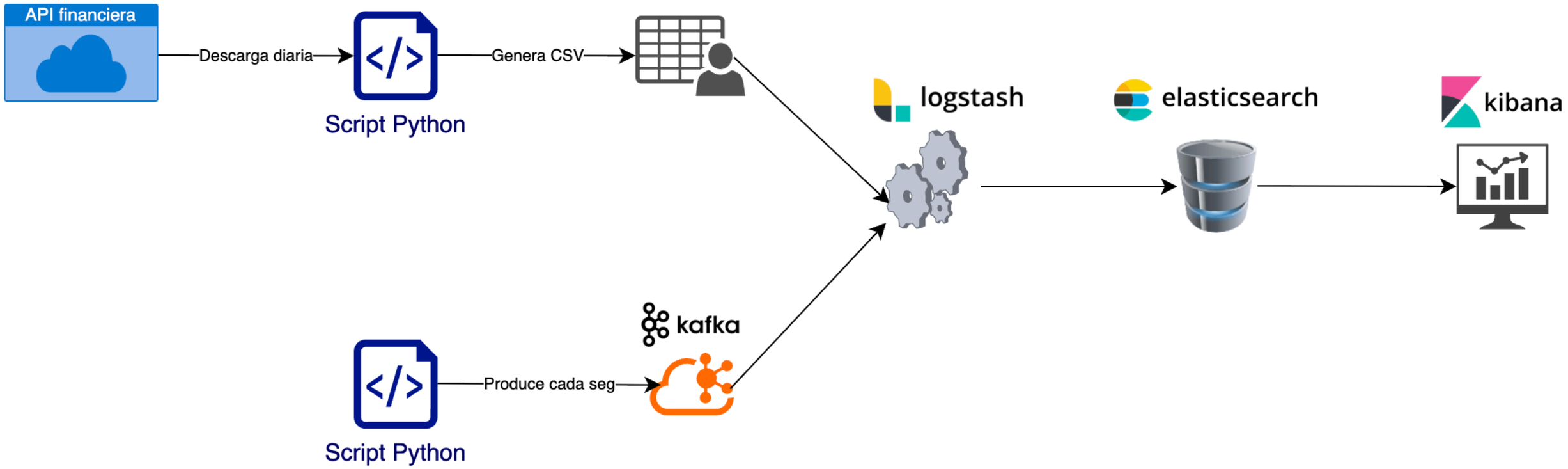
Script Python genera logs simulados de operaciones financieras (transacciones de compra/venta, errores de conexión, etc.) cada segundo.

Proceso:

- Datos enviados a Kafka (buffer de mensajes).
- Logstash consume datos de Kafka y aplica transformaciones: campos calculados (precio_promedio), categorización (categoria_monto) y enriquecimiento de datos (descripcion_estado).
- Se envía los datos a Elasticsearch.

Script Generador → Kafka → Logstash → Elasticsearch

6. DIAGRAMA


API financiera
Descarga diaria
Script Python
Genera CSV
logstash
elasticsearch
kibana
kafka
Produce cada seg
Script Python

7. CONCLUSION

Este proyecto logró integrar exitosamente dos flujos de datos (batch y near real-time). Mediante el uso de herramientas modernas se demostró cómo las mismas pueden resolver problemas complejos de manera eficiente. Cada herramienta cumplió un rol específico y complementario, demostrando su utilidad en un contexto práctico:

- Kafka aseguró la entrega confiable de datos.
- Logstash procesó y transformó datos en tiempo real.
- Elasticsearch almacenó y permitió búsquedas rápidas.
- Kibana facilitó la visualización.
- Python automatizó tareas clave.
- Docker simplificó la gestión del entorno.

¡GRACIAS!