



BASES DE DATOS

Estefi Batallas

Maestría en ciencia de Datos

Introducción

- ✓ **Objetivo:** Automatizar la limpieza y procesamiento de **cuatro bases de datos** en un entorno de producción petrolera.
- ✓ **Problema:** La limpieza manual en **Jupyter Notebook** es engorrosa y poco eficiente, especialmente con datos que se actualizan diariamente.
- ✓ **Solución:** Implementación de un **pipeline automatizado** utilizando **Apache Airflow, Knime** y **Elasticsearch**.



Descripción de la data

- 📍 Coordenadas de Pozos: Ubicación geográfica y tipo de arena de cada pozo. 963 registros con 4 variables.
- 📅 Histórico de Actividades: Registro de intervenciones realizadas en los pozos con fechas, tipos y resultados. Contiene 248 registros con 17 variables. Desde las fechas 1977 hasta 2022
- 📊 Histórico de Producción: Datos de caudal de fluido, petróleo, agua, gas y calidad del crudo. contiene 3145 registros con 8 variables.
- 📊 Histórico de Presiones: Datos de presión del reservorio obtenidos con distintas pruebas de medición. contiene 70935 registros con 13 variables



Arquitectura del Pipeline de Datos



E

- Fuente de datos:Archivos CSV /EXCEL /MySQL
- Airflow extrae los datos desde archivos historicos sin procesar



L

- Los datos sin transformar se cargan directamente en Elasticsearch
- Se almacenan en Kibana



T

- Elasticsearch estructura los datos mediante indexacion
- Kibana aplica los filtros y dashboards para la visualizacion



DAG corriendo en AIRFLOW para generación de datos de producción

DAG: pruebas_real_time Genera datos sintéticos de pruebas de producción cada minuto y los envía a Elasticsearch

Schedule: * * * * * Next Run ID: 2024-02-07, 10:11:00 UTC

08/02/2025 06:59:10 All Run Types All Run States Clear Filters

Auto-refresh 25

Press shift + / for Shortcuts

DAG pruebas_real_time / Run 2025-02-08, 06:22:44 UTC Task generar_y_enviar_datos

Task Duration

More Details List All Instances

Task Instance Notes

Add Note

Status success

Task ID generar_y_enviar_datos

Run ID manual__2025-02-08T06:22:44.162284+00:00

Map Index -1

Operator PythonOperator

Trigger Rule all_success

Duration 00:00:02

Started 2025-02-08, 06:22:47 UTC

Ended 2025-02-08, 06:22:49 UTC

Process ID (PID) 121

Hostname eaa403ceb9cc

Pool default_pool

Pool Slots 1

Executor <default>

Executor Config {}

Unix Name airflow

Max Tries 1

Clear task Mark state as... Filter DAG by task



DAG corriendo en AIRFLOW para generación de datos de producción

Elasticsearch dashboard titled "Editing production_dashboard".

Search bar: Search Elastic

Toolbar: Options, Share, Save as, Switch to view mode, Save, Refresh.

Filters: Search, Add filter, Create visualization, All types, Add from library.

Visualizations:

- Well Test Per Well:** A stacked area chart showing the Average of BPPD over time for various wells. The Y-axis ranges from 0 to 350,000. The X-axis shows dates from 2025-02-08 to 2025-02-09. The legend lists well names: ACAH-273HUI, CLBC-048HUI, CLBB-012HUI, CHSA-006HS, ACAB-059T, ACAR-264HTS, ACAC-167TS, ACAD-244HS2UI, ACA-015UJ, ACAD-100TII, ACAL-233T, ACAP-103TS, YLBD-003UI, CNOC-036US, ACA-020TS, ACAD-099S1T, ACAI-106R1T, ACA-009UI, ACAC-063UI, ACAD-100TII.
- ACAD-095T:** A table showing the last 20 entries for well ACAD-095T, ordered by timestamp. Columns: Date, Last BPPD, Last BFPD, Last Date keyword.
- Last Well Test:** A table showing the last 20 entries for all wells, ordered by Wellbore.keyword. Columns: Wellbore.keyword, Last BOPD, Last BFPD, Last Date keyword.



Challenge and Milestones



1 Manejo y Limpieza de Datos

Datos desorganizados y limpieza compleja:

- La limpieza de cuatro bases de datos en Jupyter Notebook era lenta y poco eficiente.
- Se encontraron problemas con formatos de fecha inconsistentes y valores nulos (? en Knime)

Solución:

- Uso de Knime para automatizar la limpieza y normalización de datos.
- Aplicación de nodos como Missing Value, String to Date, y Joiners para unificación de fuentes.



Challenge and Milestones



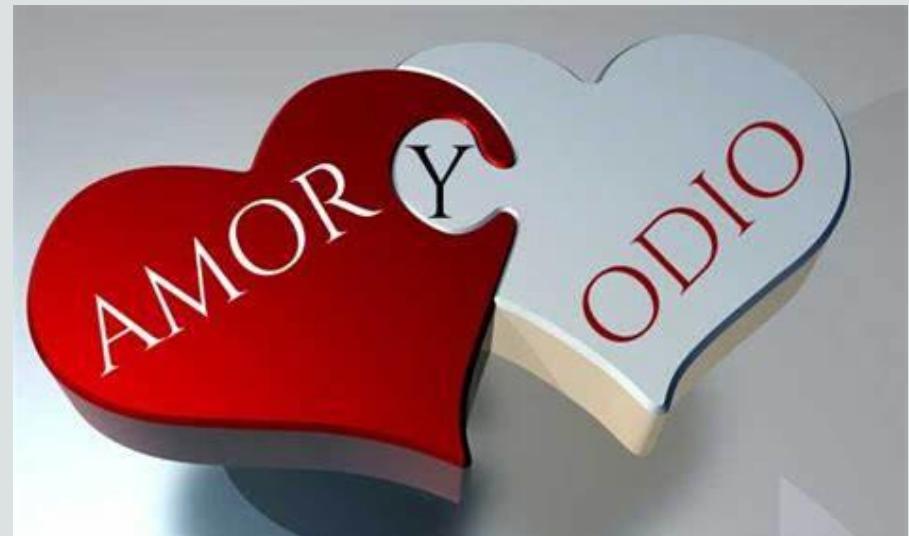
2 Integración de Tecnologías

Dificultades en la comunicación entre contenedores Docker

- Airflow y Elasticsearch estaban en redes de Docker separadas, lo que impedía el envío de datos.
- MySQL y Knime requerían configuraciones adicionales para acceso desde contenedores externos.

Solución:

- Configuración de Docker Networks para conectar Airflow, MySQL, Elasticsearch y Kibana.
- Uso de POST Request en Knime para enviar datos a Elasticsearch.



Challenge and Milestones

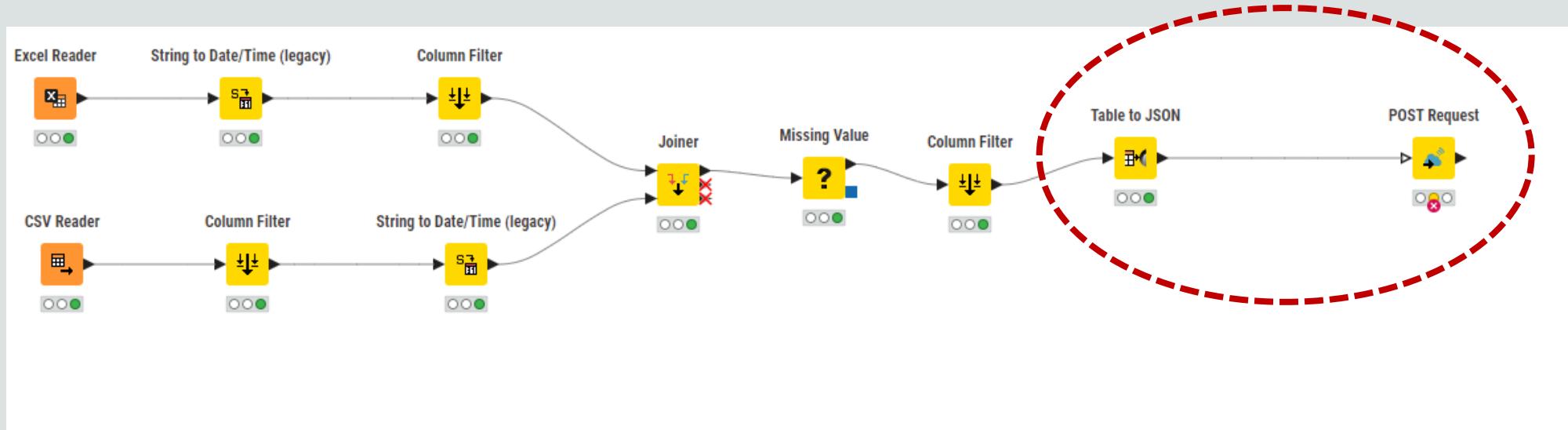
③ Envío de Datos a Elasticsearch

✓ Errores de formato JSON en Elasticsearch

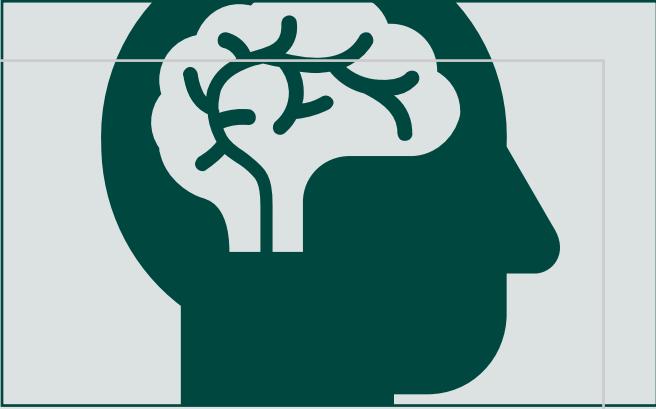
- Knime generaba datos en JSONBlobCell, lo que causaba fallos en el POST Request.
- Elasticsearch requería un formato específico para la carga masiva de datos (_bulk).

✓ Solución:

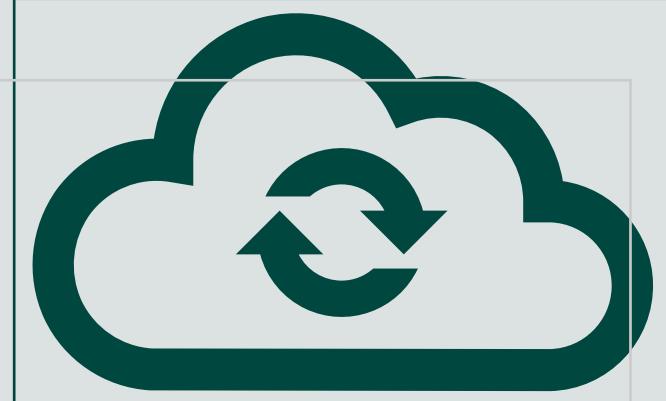
- Conversión de JSONBlobCell a String antes de enviarlo a Elasticsearch.
- Validación de datos con curl antes de la ingestión final.



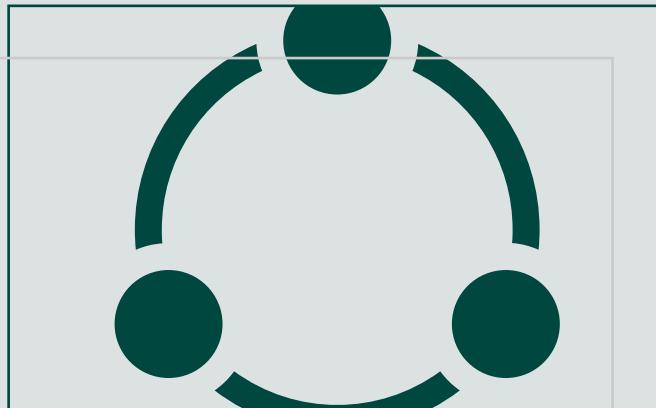
CONCLUSIONES



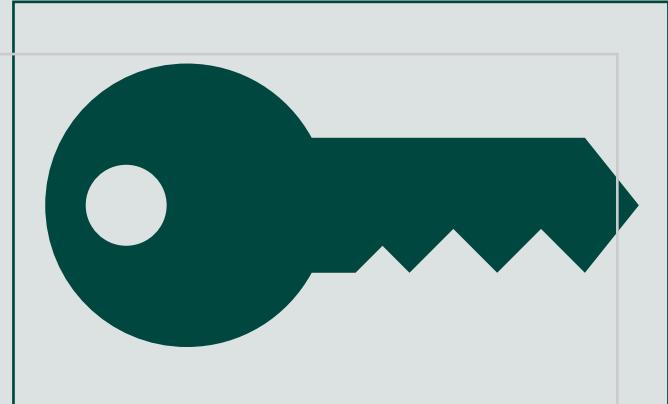
📌 Integración de Modelos de Machine Learning



📌 Optimización del Pipeline en la Nube



📌 Implementación de un Flujo Streaming Completo



📌 Mejorar la Seguridad y Gobernanza de los Datos



Thank You!



Production Engineering
& Execution Team

