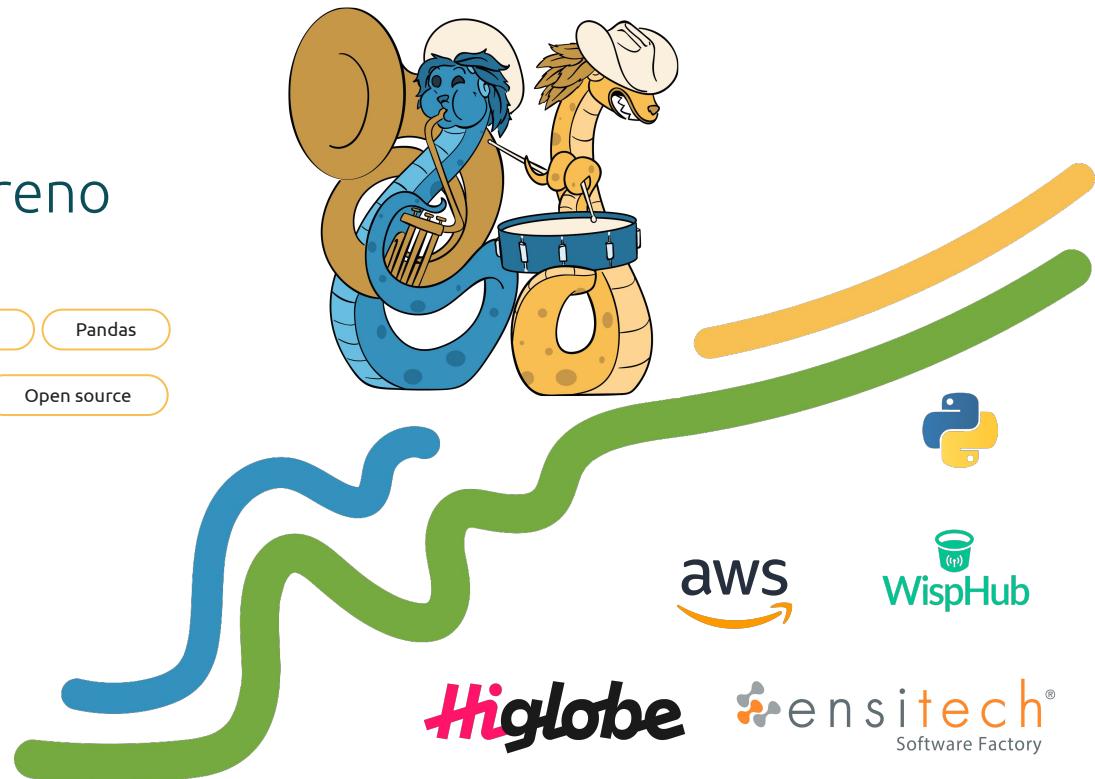


# Python y los esquemas ETLs

Hugo Alejandro Ramírez Moreno  
PyCon Latam Sep. 2024



# Temario

- |  |  |   |
|--|--|---|
| <p><b>01</b> Acerca de mi.</p> <hr/>     | <p><b>02</b> Cómo percibimos los datos</p> <hr/> | <p><b>03</b> Python y los esquemas ETL.</p> <hr/> |
| <p><b>04</b> Caso de uso real.</p> <hr/> | <p><b>05</b> DevOps.</p> <hr/>                   | <p><b>06</b> Demo.</p> <hr/>                      |
| <p><b>07</b> Preguntas.</p>              |  |   |

# 1

Acerca de mi.

# Data Scientist

“Los casos especiales no son lo suficientemente especiales como para romper las reglas”. Zen de python

*Ni más ni menos:*

*Estudié ciencias físicas e informática, busqué siempre involucrarme en clases de filosofía, me apasionan temas relacionados al pasado, presente y futuro de la humanidad, la ciberseguridad, la ciencia y el software libre. Forme parte del desarrollo de la TDA (Televisión Digital Abierta), Fui voluntario de la FSF (Free Software Foundation) de la comunidad CANAIMA en Venezuela con el programa de alfabetización tecnológica a niños y personas mayores, forme parte de la PyVe, he participado en la PyAr y actualmente formo parte de la Python CDMX.*

Linkedin:

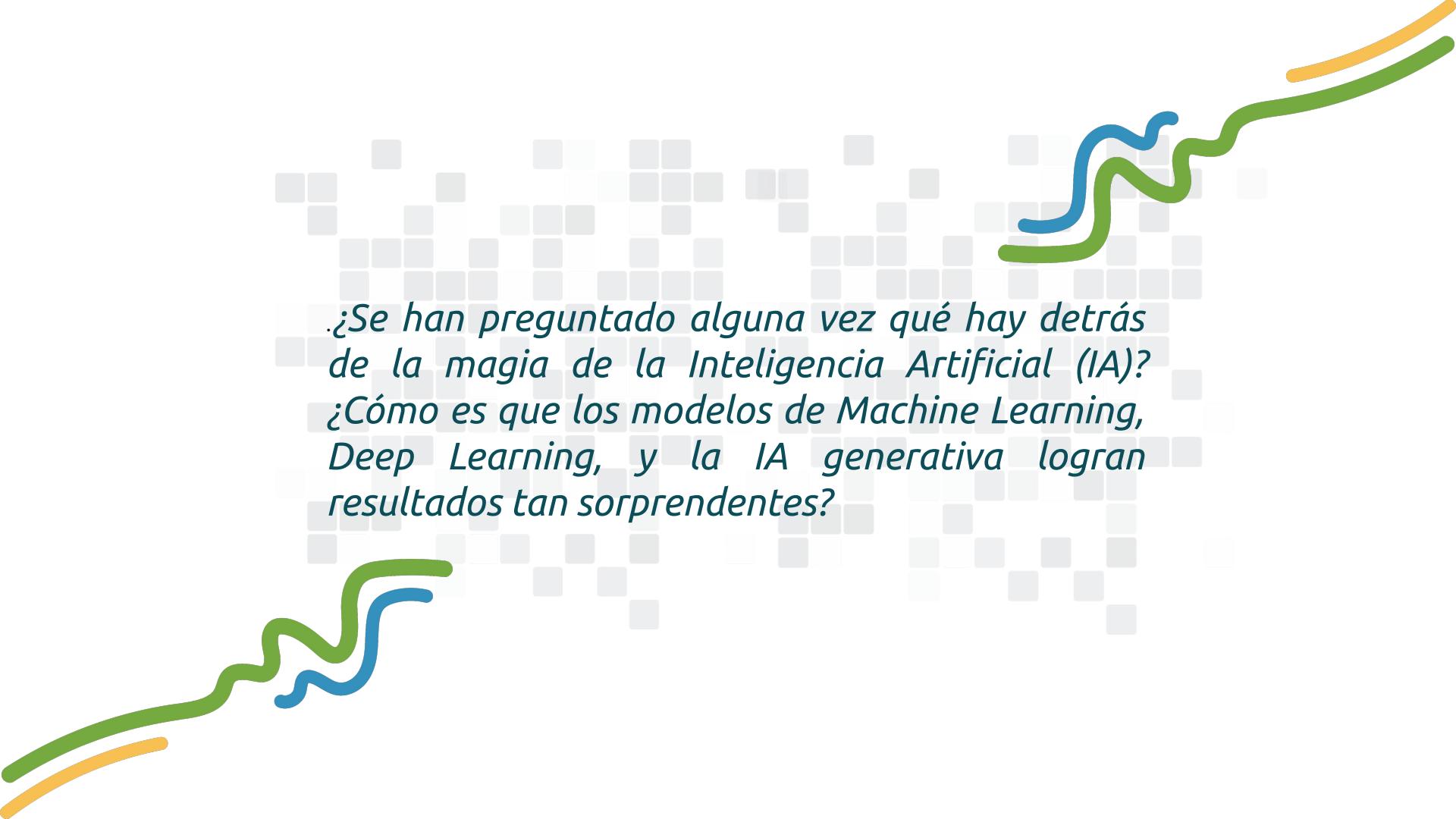


@hughpythoneer



Hugo Alejandro Ramírez Moreno

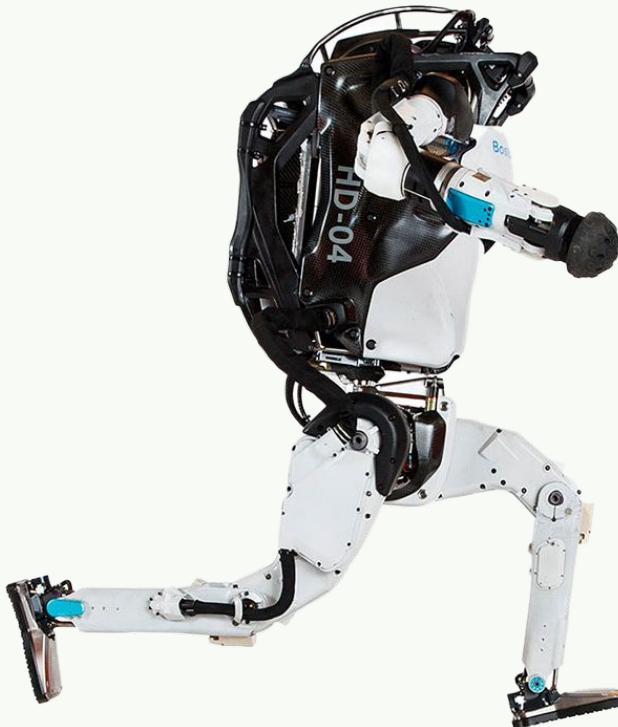




*.¿Se han preguntado alguna vez qué hay detrás de la magia de la Inteligencia Artificial (IA)?  
¿Cómo es que los modelos de Machine Learning, Deep Learning, y la IA generativa logran resultados tan sorprendentes?*

## Boston Dynamics

Boston Dynamics, fundada en 1992 como un spin-off del MIT, es una empresa de robótica e ingeniería que desarrolla robots avanzados como Spot, un robot cuadrúpedo, y Atlas, un humanoide. Sus robots se utilizan en seguridad, inspección y logística, destacándose por su movilidad y agilidad combinadas con tecnología avanzada.





*.¿Nace una nueva forma de  
percibir los datos?*



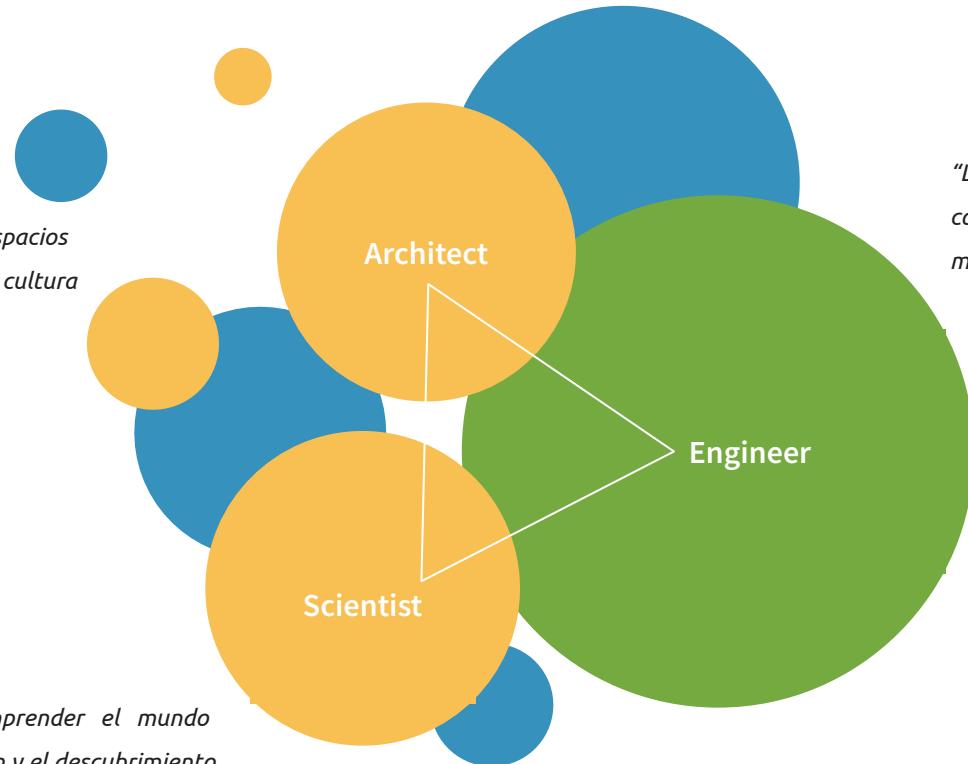
Más  
Precisos

Más  
inteligentes

Más  
Personalizados

## La triada perfecta.

"La **arquitectura** busca crear espacios funcionales y estéticos que reflejan la cultura y necesidades humanas."



"La **ciencia** busca comprender el mundo mediante la investigación y el descubrimiento de verdades fundamentales."

"La **ingeniería** busca innovar y aplicar conocimientos para resolver problemas y mejorar la vida humana."



*.¿Qué nos hemos propuesto los  
aficionados?*



*.¿Qué estamos haciendo?*

# 2

**Exploración:** Industria de los  
datos, equipos, perfiles,  
comunidad.

# Microsoft

## Equipo de IA y Arquitectura

### El objetivo

Cómo integrar modelos avanzados de inteligencia artificial, como GPT y DALL-E, en aplicaciones empresariales usando Azure.

### Hallazgos:

- Baja calidad de los datos, uso inteligente de la herramientas, claridad en los prompts de negocios, costos, infra descentralizada.



Adriana Gomes <adrianagomes@microsoft.com>  
para mí ▾

Traducir al español X

mar  
**20**  
mié

Caso de Uso, y proyecto de ETL en GEP...  
[Miralo en Google Calendar](#)

Cuándo mié 20 de marzo de 2024 12:00 – 12:30 (GMT-6)  
Ubicación Microsoft Teams Meeting  
Participantes Adrián Hernandez Contreras, César Martínez Morales, Víctor Del Ángel, Adriana Gomes\*

Sí ▾  Quizás  No  Más opciones



# MongoDB

MongoDB.local Ciudad de México

## El objetivo

Ir a las sesiones de MongoDB.local Ciudad de México para aprender sobre tecnologías, herramientas y mejores prácticas que facilitan la creación de aplicaciones basadas en datos sin distracciones.

## Hallazgos:

- Resistencia al cambio, capacidad técnica, falta de claridad en los esquemas NoSQL, recientes integraciones en los principales clouds de la industria.



# Amazon

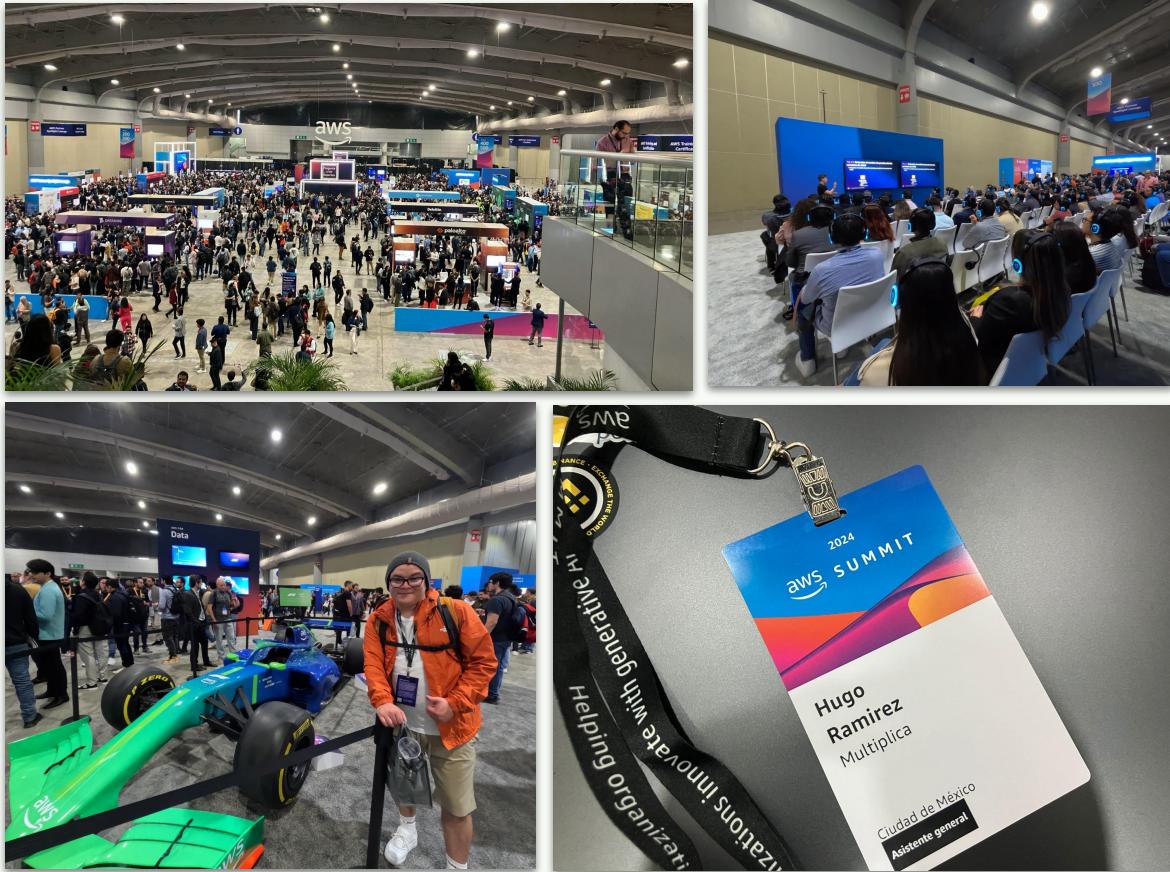
## AWS Summit Ciudad de México 2024

### El objetivo

Disfrutar de un día lleno de acción en el AWS Summit Ciudad de México 2024, donde lo último en innovación en la nube cobra vida. Descubrir cómo las tecnologías de vanguardia, desde la IA generativa hasta la analítica de datos, están revolucionando los diferentes sectores e impulsando a las organizaciones para que lideren en la era digital.

### Hallazgos:

- Todas las consultoras ofrecen lo mismo, poca innovación, altos costos, nada orientado a la calidad de los datos, costos muy elevados



# Python CDMX

## Charla para la comunidad Python

### El objetivo

Compartir y explorar cómo el proceso ETL es vital para la Inteligencia Artificial, desglosando cada una de sus fases con un ejemplo real.

### Hallazgos:

- Impresión por el tema, compartimos el mismo dolor, poca calidad de los datos en los proyectos de IA, Python como aliado, grandes retos en la industria y mucho por hacer.



# GitTogether CDMX

GitHub Open Source CDMX & Notion Campus Leaders

## El objetivo

Participar e integrarme a la GitHub Presente CDMX que es un evento mensual creado por desarrolladores para desarrolladores. Donde se aprender acerca de los más recientes flujos de trabajo de GitHub, nuevas características y cómo contribuir a código abierto. En esta ocasión el tema fue Notion y su integración con github para administrar tus proyectos

## Hallazgos:

- Comparaciones entre Jira y Notion, Confusiones en el uso de los tableros, Poco control y seguimiento exhaustivo de las tareas, Preocupación por la seguridad y los costos de las herramientas de gestión.



# Meeting AWS GLUE / SageMaker Studio Lab



AWS Glue



AWS Sagemaker



AWS Bedrock

Meeting - Amazon Web Services  
Viernes, 23 agosto · 12:00 – 12:30

https://chime.aws/5689722665  
1 invitado  
1 sés virtualmente

Hugo Ramirez Casa - Editar  
¡Gracias por programar una reunión con Prelude!

Invitados: Hugo Ramírez, Lucas Schildt

Has recibido una invitación a una reunión en línea con tecnología de Amazon Chime.

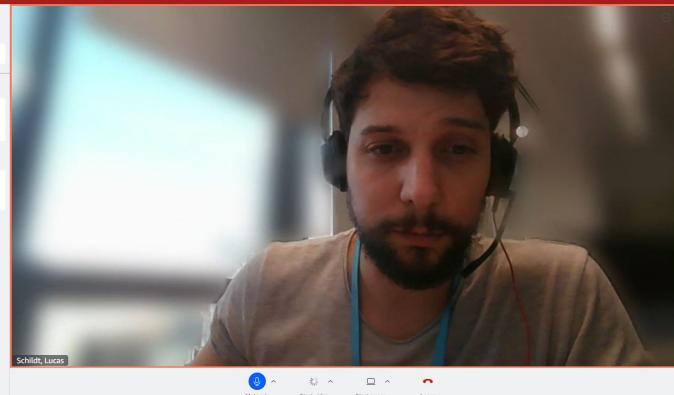
Haz clic para unirte a la reunión:<https://chime.aws/5689722665>  
ID de reunión: 5689 72 2665  
Se recomienda usar auriculares o utilizar el micrófono y los altavoces del ordenador.

Llama con tu teléfono:  
Número gratuito de Estados Unidos: 1 800 446 4463  
ID de reunión: 5689 72 2665  
Conexión telefónica móvil con un sello de calidad de servicio.  
Estados Unidos (1): +1 206-462-5569,,5689722665#  
Estados Unidos (1): +1 206-462-5569,,5689722665#

¿Asistirás?

Attendees

- Speaker: Schildt, Lucas
- Present: Hugo Ramirez, Lucas Schildt
- Invited: prelude@prelude.amazon.com, prelude+mid=61af61d8-a026...



# IASalón

Mexico City IBM

## El objetivo:

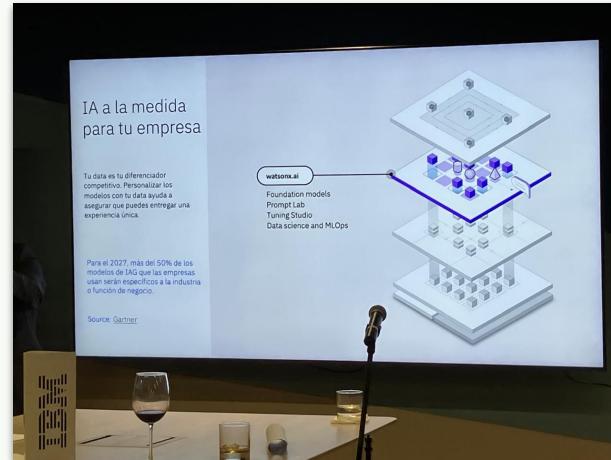
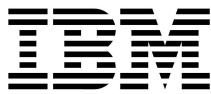
Conocer las últimas innovaciones de IBM en inteligencia artificial con Watson. Entender cómo IBM aborda la resiliencia y seguridad en el almacenamiento de datos.

Explorar soluciones en la nube específicas para el sector financiero. Descubrir oportunidades de aceleración para Fintech. Hacer networking con profesionales y potenciales socios de la industria.



## Hallazgos:

- IBM presentó Watson como su solución de IA, abordando también la resiliencia y seguridad del almacenamiento de datos. Se destacó el IBM Cloud para el sector financiero y se introdujo un programa de aceleración para Fintech. El evento concluyó con un cóctel y networking para los asistentes.



# 3

Python y los esquemas ETL.



*.¿Están los científicos de datos  
haciendo ciencia de datos?*

# Lo que esperaba



vertex.ai

AWS Bedrock

**Información general**

**Modelos funcionionales**

**Destacar**

**ANTHROPIC**

La familia de modelos Claude 3 de Anthropic (Davinci, Sonnet y Claude) permite a los clientes elegir el compromiso entre precisión, integridad, velocidad y costo que se adapte a sus necesidades y objetivos. Todos los modelos pueden procesar imágenes y devolver resultados en texto dentro de un rango de 200 000 tokens de respuesta.

**Áreas de juego**

**Ejemplo de casos de uso**

Amazon Bedrock admite muchas



AWS Bedrock

Google Cloud

**Vertex AI**

**Comienza a usar Vertex AI**

Con Vertex AI, los desarrolladores de aprendizaje automático y los científicos de datos pueden llenar sus proyectos de la base de generación de ideas y la ejecución de trabajo de investigación y rendibilidad. [Vista de información sobre Vertex AI](#)

**Instructivos**

Puedes interactuar interactivamente para aprender a entrenar, evaluar y sancionar tu modelo. Puedes usar un modelo de entorno de desarrollo de Vertex AI.

**MOSTRAR LISTA DE APIs**



Microsoft Azure

**Azure AI services | Azure OpenAI**

Mostrando de 0 a 0 de 0 registros.

No hay Azure OpenAI para mostrar.  
Rellena una entrada vacía en el campo de búsqueda para obtener más información.



IBM

**watsonx.ai**

Welcome back, Louis

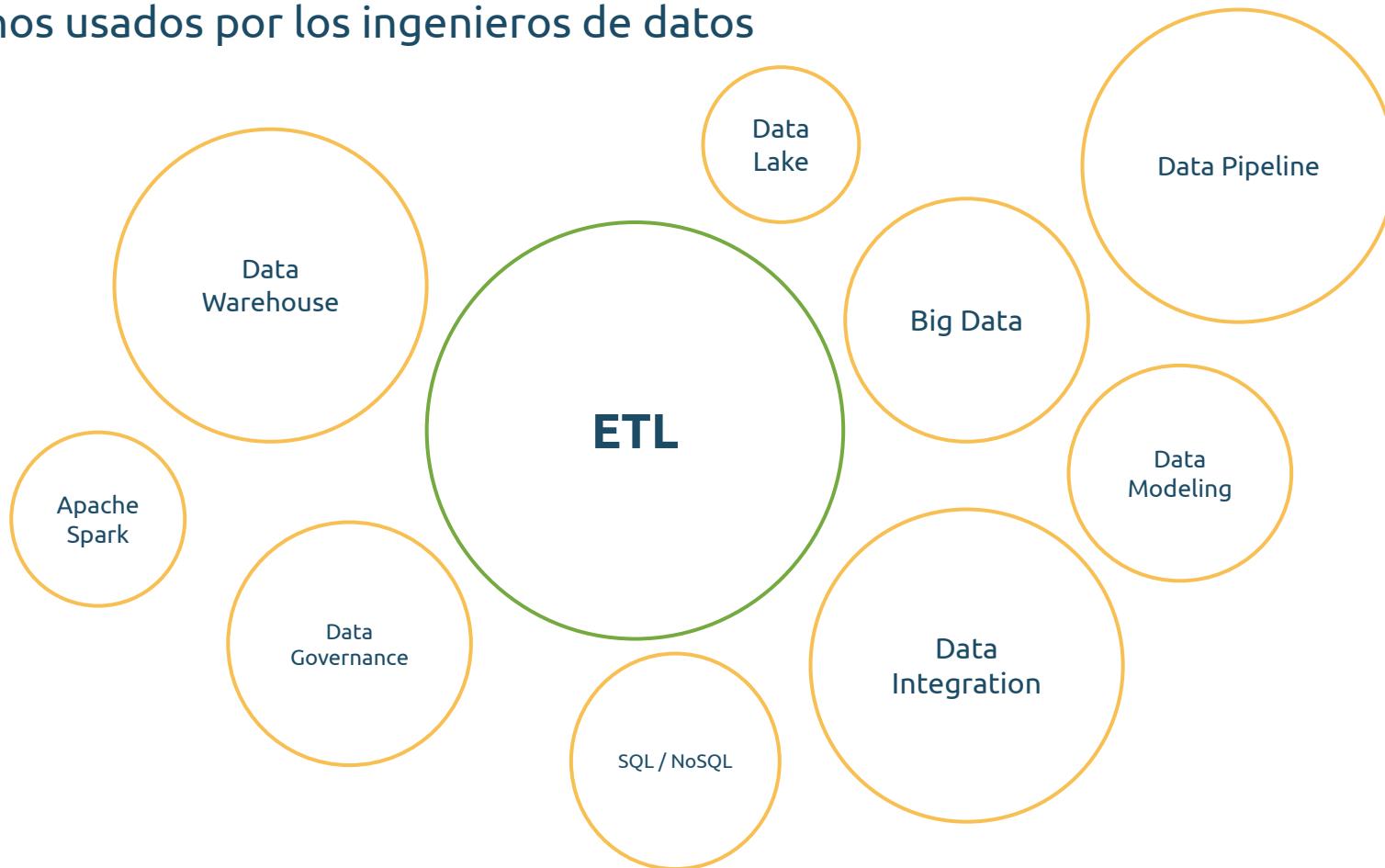
Start your free trial → Explore the Interactive Demo →

WatsonX is a next-generation enterprise studio for AI builders to train, validate, tune and deploy AI models.

Demo: Generative AI and machine learning with IBM watsonx.ai (\$30)

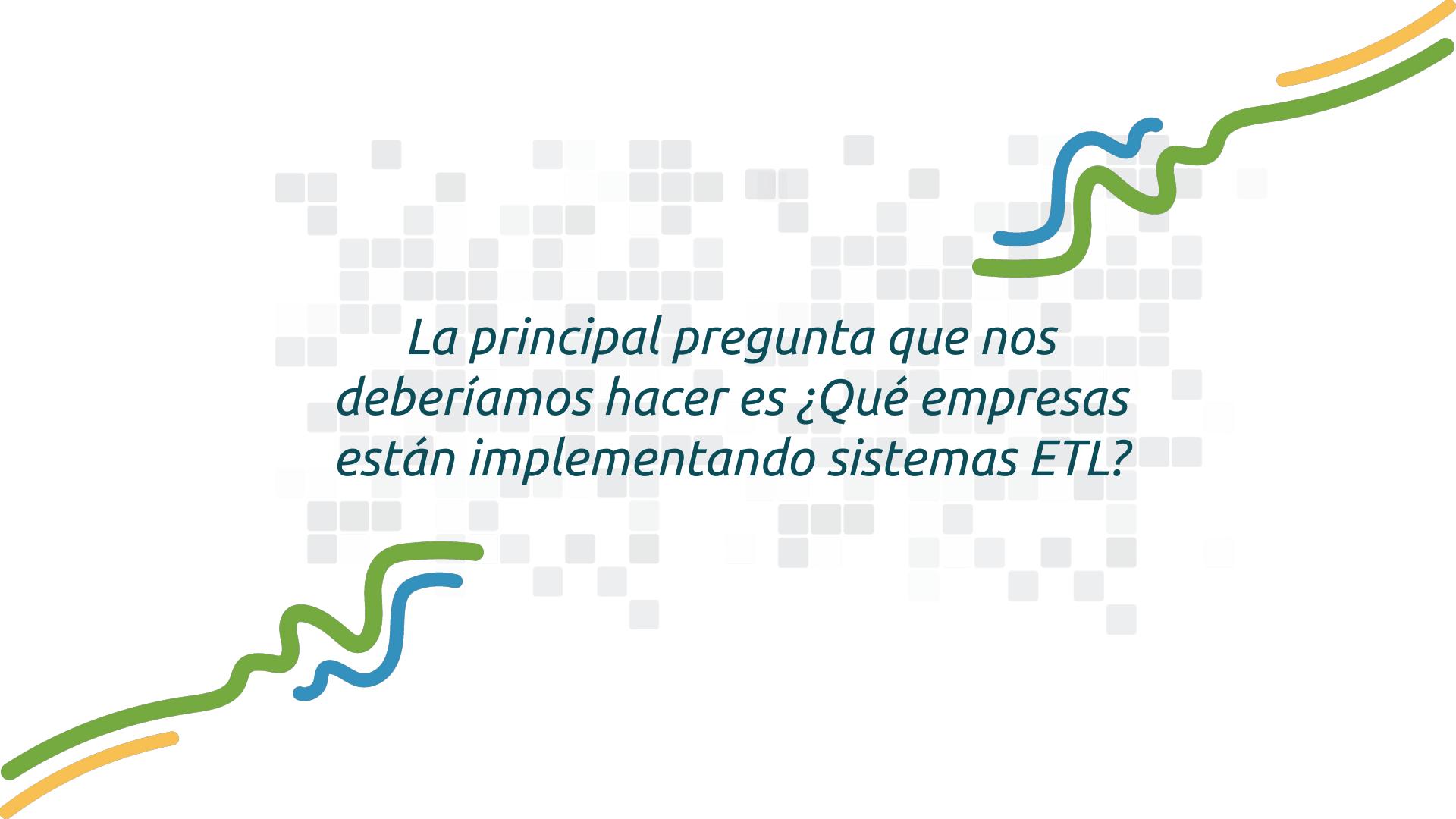
Hello! How can we help you?

## Términos usados por los ingenieros de datos





*¿Cuáles son las principales razones por las que los científicos de datos enfrentan desafíos en el proceso de extracción, transformación y carga (ETL) de datos?*



*La principal pregunta que nos  
deberíamos hacer es ¿Qué empresas  
están implementando sistemas ETL?*



## Amazon

- Amazon Web Services (AWS): Utiliza y ofrece servicios ETL como AWS Glue, que facilita la preparación y carga de datos para análisis y machine learning.

## Google

- Google Cloud Platform (GCP): Ofrece servicios ETL como Google Cloud Dataflow y Google Cloud Dataprep para la integración y preparación de datos.

## Microsoft

- Azure: Proporciona herramientas como Azure Data Factory y SQL Server Integration Services (SSIS) para procesos ETL en la nube y on-premises.



ORACLE



## IBM

- IBM DataStage: Una herramienta ETL robusta que es parte de la plataforma IBM InfoSphere para la integración de datos.

## Oracle

- Oracle Data Integrator (ODI): Utilizada para la integración de datos en entornos empresariales, soportando procesos ETL complejos.

## Salesforce

- MuleSoft: Parte de Salesforce, proporciona capacidades de integración de datos y servicios ETL para conectar aplicaciones, datos y dispositivos.

  
NETFLIX

## Facebook

- Utiliza procesos ETL extensivos para gestionar datos masivos y alimentar su plataforma de análisis interno, así como servicios de terceros como Apache Hadoop y Apache Hive.

## Netflix

- Implementa ETL con herramientas como Apache Spark y Apache Kafka para procesar grandes volúmenes de datos de usuarios y contenidos en tiempo real.

## Spotify

- Utiliza ETL para procesar y analizar datos de uso y preferencias de los usuarios, empleando herramientas como Apache Airflow para la orquestación de flujos de datos.



## Retail y Comercio Electrónico

- Walmart, Alibaba: Utilizan ETL para integrar y analizar datos de ventas, inventarios y clientes.

# Alibaba

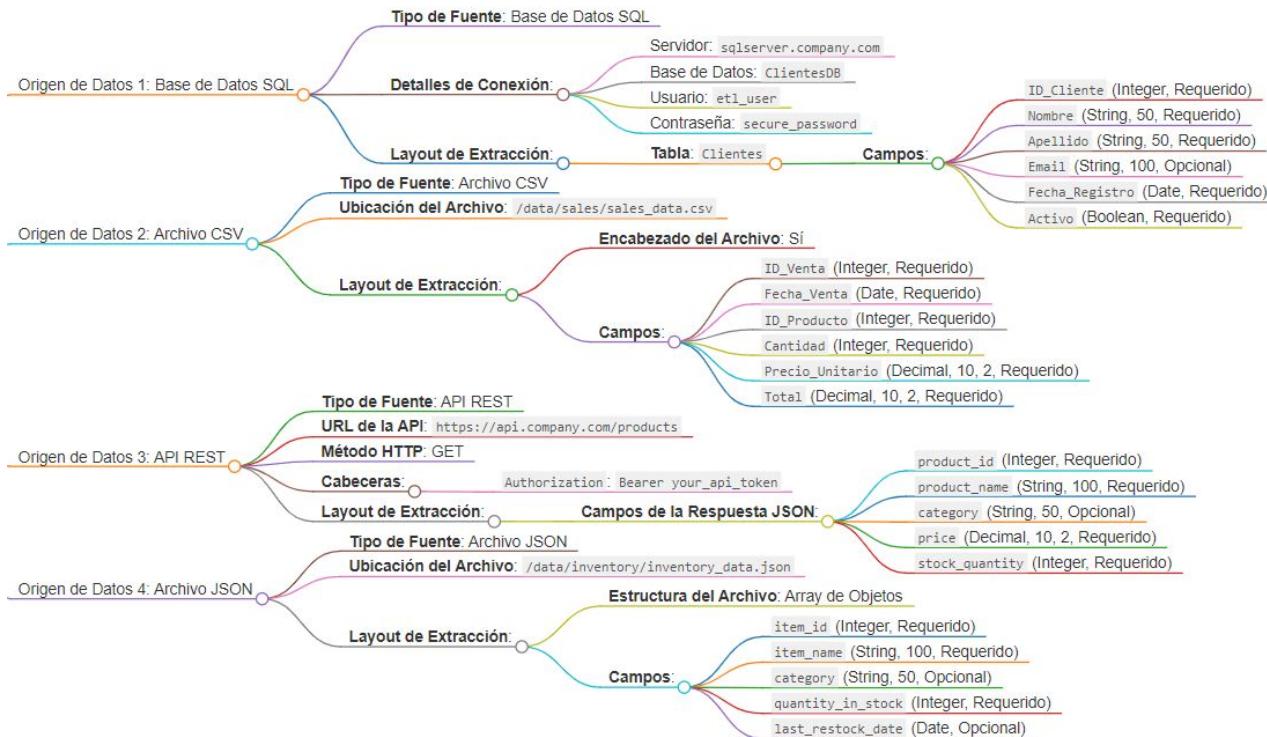


*Reflexión*

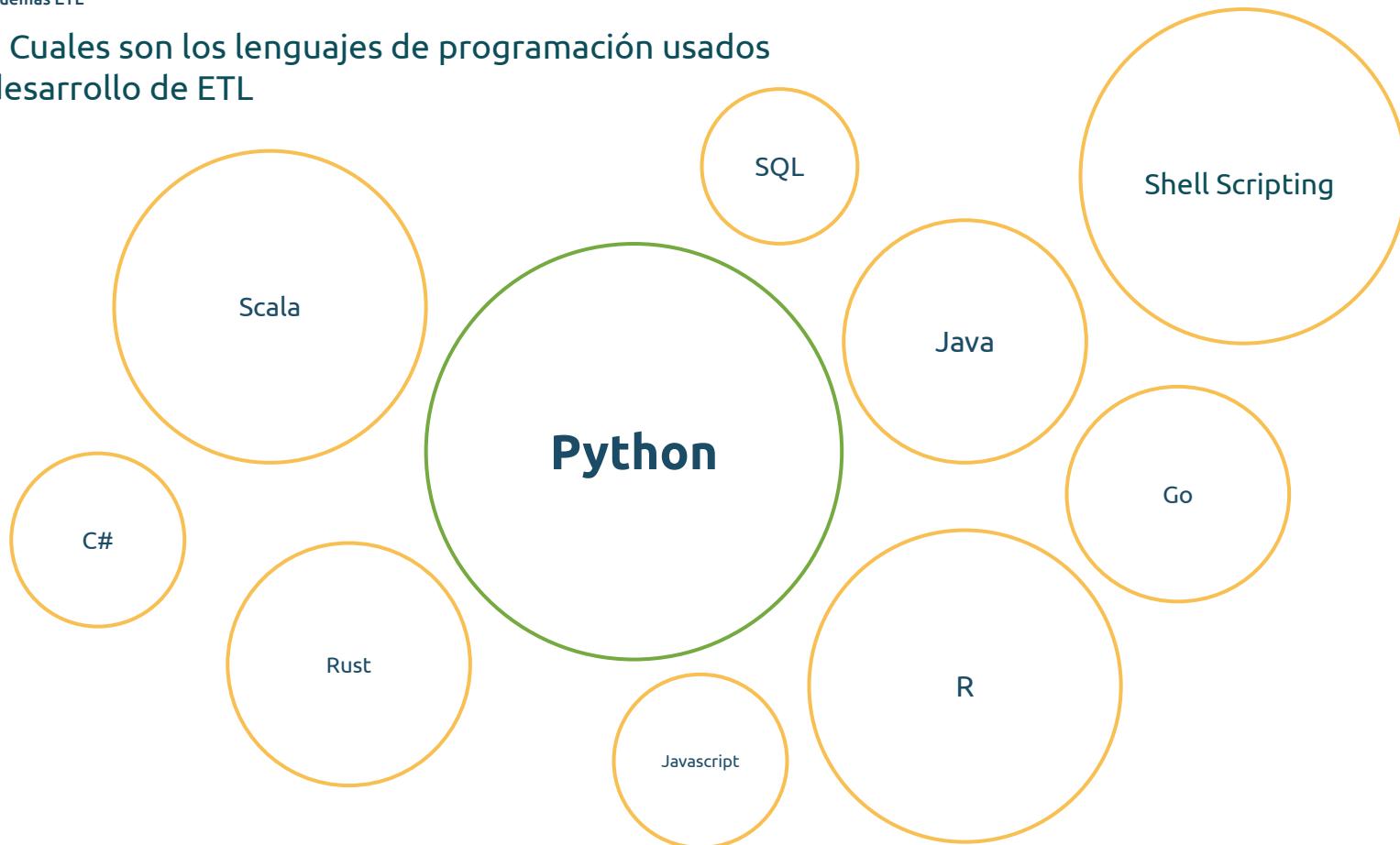
# 4

Caso de uso real.

# Tipos de fuentes de datos



## Tipos de Cuales son los lenguajes de programación usados para el desarrollo de ETL



# GEPP

Embotellador Oficial PepsiCo en México

## El objetivo

Desarrollar e implementar los procesos de ETL en las distintas arquitecturas de la nube para los core de datos de los principales servicios de GEPP el embotellador Oficial de PEPSICO en México.

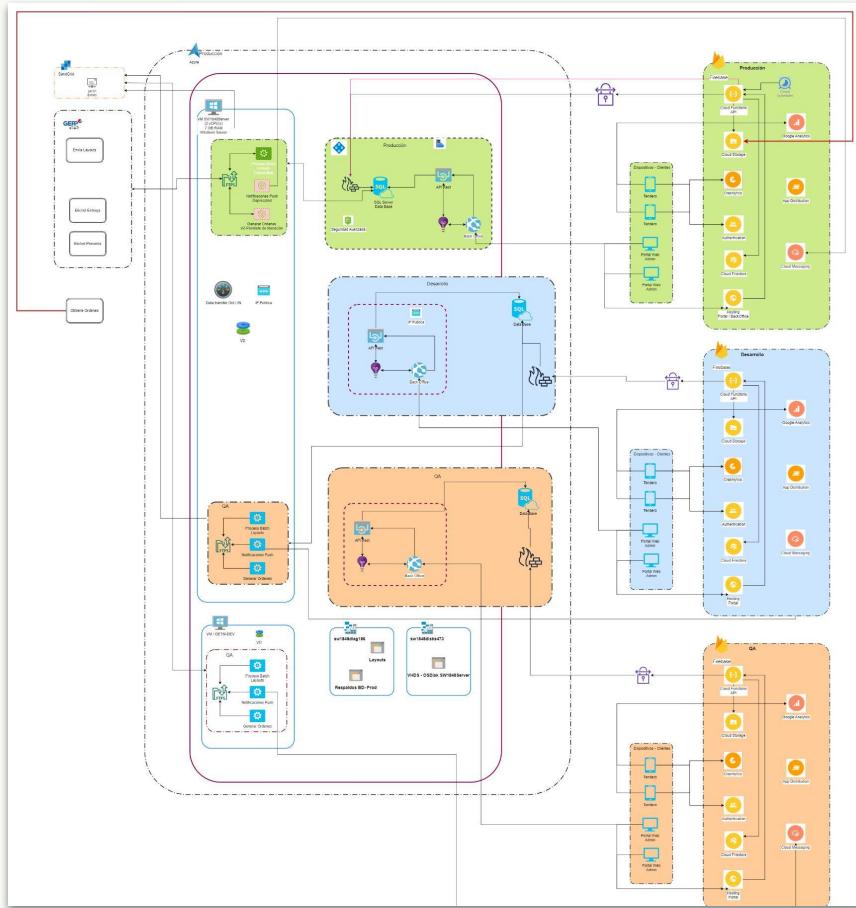
- Mi pedido e-pura.
- Rutas ausentes TCOM.
- Gep en tus Manos.

A screenshot of a Microsoft Azure Databricks workspace. The left sidebar shows a file tree with a folder named "layouts-batch-etl" containing subfolders "config", "log", "utils", and files "main.py", "README.md", and "requirements.txt". The main area displays a Python notebook titled "layouts-batch-etl" with several lines of code. The code includes imports for "pyspark.sql", "sparkSession", and "databricksutils", along with some utility functions and configurations for a batch ETL process.

## Arquitectura base de un ETL

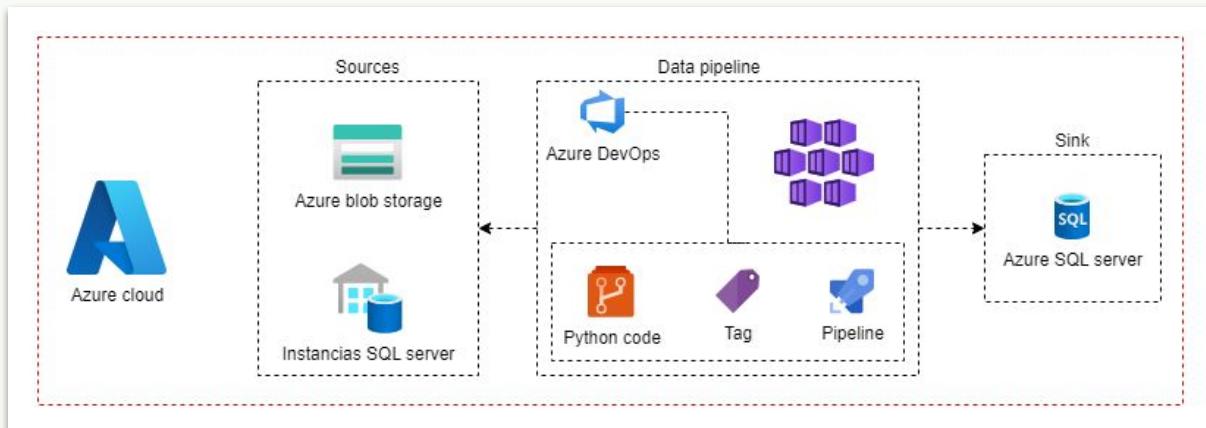


Arquitectura en el:  
**Pasado**



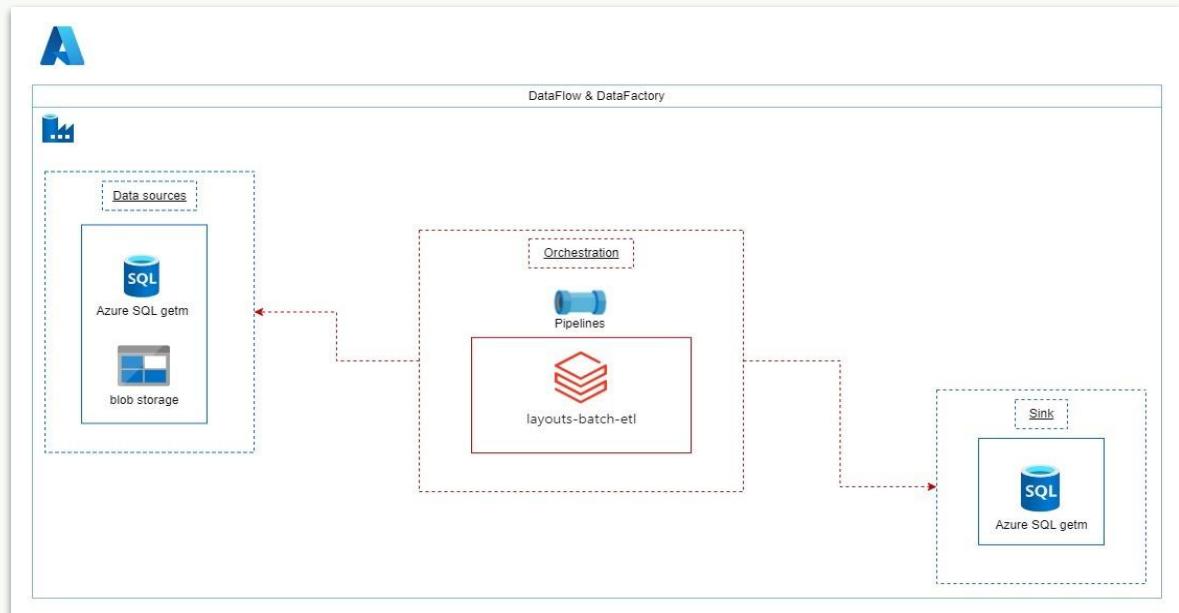
Arquitectura:

# Versión 1.0



Arquitectura:

# Versión 2.0





*Soluciones técnicas*

## Paralelismo en la lectura

El paralelismo en la lectura se refiere a la capacidad de realizar múltiples operaciones de lectura de datos de manera simultánea, en lugar de secuencialmente. Esto es especialmente útil en procesos ETL donde grandes volúmenes de datos necesitan ser procesados eficientemente.

### Ventajas del Paralelismo en la Lectura

Mayor Velocidad:

- Permite reducir significativamente el tiempo necesario para leer grandes volúmenes de datos.

Eficiencia de Recursos:

- Aprovecha al máximo los recursos del sistema, como múltiples núcleos de CPU y memoria distribuida.

Escalabilidad:

- Facilita el manejo de grandes volúmenes de datos a medida que la cantidad de datos y la demanda de procesamiento crecen.

```
# Se crea una instancia de la clase Layouts con los archivos especificados en `files`.
layout = LayoutsBase(files)

# Se define un diccionario `tasks` que mapea nombres de tareas a métodos específicos de la clase.
# Cada clave del diccionario es un nombre descriptivo de la tarea, y el valor es el método correspondiente
# de la clase `Layouts` que se debe ejecutar para completar esa tarea.
tasks = {
    'layout_clients': layout.layout_clients,
    'layout_clients_cdg': layout.layout_clients_cdg,
    'result_layout_program_sales': layout.layout_clients_sales,
    'result_layout_program_portfolio': layout.layout_clients_portfolio,
    'result_layout_promotions': layout.layout_promotions,
    'result_layout_customer_promotions': layout.layout_customer_promotions,
}

# Se utiliza un gestor de contexto para crear un ThreadPoolExecutor, que permite ejecutar las tareas de forma paralela.
with concurrent.futures.ThreadPoolExecutor() as executor:
    # Se crea un diccionario `futures` donde cada clave es el nombre de la tarea y el valor
    # que representa la tarea asincrónica. `executor.submit` se utiliza para iniciar la ejecución de la tarea.
    futures = {name: executor.submit(task) for name, task in tasks.items()}
    # Se espera a que todas las tareas se completen y se recopilan los resultados en el diccionario `results`.
    # `future.result()` bloquea hasta que la tarea asociada se completa y devuelve el resultado.
    results = {name: future.result() for name, future in futures.items()}

    # Se extraen los resultados de las tareas específicas del diccionario `results` y se asignan a variables.
    layout_grocers_value = results['layout_clients']
    layout_grocers_cdg_value = results['layout_clients_cdg']
    layout_program_sales_value = results['result_layout_program_sales']
    layout_program_portfolio_value = results['result_layout_program_portfolio']
    layout_promotions_value = results['result_layout_promotions']
    layout_customer_promotions_value = results['result_layout_customer_promotions']

print('parallelism layout base successfully ✅')
```

## Python y los esquemas ETL

### Partitionado de Datos

El particionado de volúmenes grandes de datos es una técnica crucial en el manejo y procesamiento eficiente de grandes cantidades de datos, especialmente en entornos de Big Data. Consiste en dividir un conjunto de datos grande en partes más pequeñas y manejables, llamadas particiones, que pueden ser procesadas en paralelo para mejorar el rendimiento y la escalabilidad.

### Ventajas del Partitionado de datos

- Mejora del Rendimiento: Permite la lectura y escritura de datos en paralelo, reduciendo el tiempo total de procesamiento.
- Escalabilidad: Facilita el manejo de crecientes volúmenes de datos al distribuir la carga de trabajo.
- Mantenimiento Simplificado: Hacer operaciones de mantenimiento, como copias de seguridad y restauración, es más manejable en particiones más pequeñas.
- Optimización de Consultas: Las consultas pueden ser más rápidas porque solo se escanean las particiones relevantes.

### Desafíos del Partitionado

- Equilibrio de Particiones: Asegurarse de que las particiones tengan tamaños equilibrados para evitar cuellos de botella.
- Administración de Metadatos: Mantener información precisa sobre las particiones puede ser complejo.
- Consistencia de Datos: Garantizar que las particiones sean consistentes y que las operaciones de particionamiento no introduzcan errores.

### Herramientas y Tecnologías Comunes

- Apache Hadoop: HDFS permite particionar datos y distribuirlos entre nodos.
- Apache Spark: Ofrece particionamiento nativo en RDDs y DataFrames para procesamiento paralelo.
- Apache Hive: Soporta particionamiento de tablas para optimizar consultas.
- SQL Databases: Bases de datos como PostgreSQL, MySQL, y Oracle soportan particionamiento de tablas.
- NoSQL Databases: Bases de datos como Cassandra y MongoDB utilizan particionamiento para distribuir datos.

El particionado de volúmenes grandes de datos es esencial para mantener el rendimiento y la escalabilidad en sistemas que manejan grandes cantidades de información. La elección de la estrategia de particionado adecuada depende de la naturaleza de los datos y los patrones de acceso. Implementar particionado de manera efectiva puede resultar en una mejora significativa en la eficiencia de los procesos ETL y en la capacidad de manejar datos a gran escala.

```
# Se crea una instancia de DataAccessLayer.
instance = DataAccessLayer()

# Se crea una instancia de BuildDataFrame.
builder = BuildDataFrame()

# Se procesa el layout 'grocers_to_dispute' utilizando el builder y la instancia creada,
# pasando el dataframe 'layout_grocers_value' y especificando el método '_grocers_' junto con las columnas deseadas.
grocers_to_dispute = process_layout(builder, instance, layout_grocers_value, '_grocers_', ['column name', 'column value'])

# Define una función para calcular el número de fragmentos en que se debe dividir un dataframe basado en un tamaño.
def calculate_fragments(dataframe) -> int:
    max = 10000 # Tamaño máximo de filas por fragmento.
    size_dataframe = len(dataframe) # Total de filas en el dataframe.
    num_fragments = size_dataframe // max # Número básico de fragmentos.
    if size_dataframe % max != 0: # Si hay un residuo, se necesita un fragmento adicional.
        num_fragments += 1
    return num_fragments

# Define una función para dividir un dataframe en 'count' número de partes.
def dataframe_partitions(dataset, count):
    partitions = np.array_split(dataset, count) # Utiliza numpy para dividir el dataframe.
    return partitions

# Procesa un layout aplicando un tipo de dato (si se especifica), calculando fragmentos, y aplicando una función:
def process_layout(builder, instance, dataframe, method, columns, type_cast=None):
    if type_cast:
        dataframe = dataframe.astype(type_cast) # Cambia el tipo de dato de las columnas si se especifica.
    num = calculate_fragments(dataframe) if len(dataframe) > 10000 else 2 # Calcula el número de fragmentos necesarios.
    partitions = dataframe_partitions(dataframe, num) # Divide el dataframe en fragmentos.
    builder._attrib_(partitions, instance, method, columns) # Asigna atributos al builder.
    return builder._builder_() # Construye y devuelve el resultado final.

# Clase para construir un dataframe procesado a partir de fragmentos.
class BuildDataFrame:

    # Asigna atributos necesarios para el procesamiento.
    def __init__(self, partitions, instance, method, columns):
        self.partitions = partitions # Fragmentos del dataframe.
        self.is_instance = instance # Instancia de DataAccessLayer.
        self.is_method = method # Método a aplicar a cada fragmento.
        self.columns = columns # Columnas a incluir en el procesamiento.

    # Construye el dataframe procesado.
    def __builder__(self):
        self.method_to_dispute = getattr(self.is_instance, self.is_method) # Obtiene el método de la instancia.
        self.processed_dataframes = [] # Lista para almacenar dataframes procesados.
        for i, partition_dataset in enumerate(self.partitions): # Itera sobre cada fragmento.
            dataset = partition_dataset[self.columns] # Selecciona las columnas especificadas.
            response = self.method_to_dispute(dataset) # Aplica el método.
            self.processed_dataframes.append(response) # Añade el resultado a la lista.
        self.response = reduce(DataFrame.union, self.processed_dataframes) # Combina todos los dataframes procesados.
        return self.response # Devuelve el dataframe combinado.
```

# Comandos Básicos de BCP

```
bcp {table | view | "query"} {in | out | queryout | format} data_file  
[-m max_errors] [-f format_file] [-e err_file] [-F first_row] [-L last_row]  
[-b batch_size] [-n native_type] [-c character_type] [-w wide_character_type]  
[-N keep_nulls] [-r row_term] [-t field_term] [-T trusted_connection]  
[-S server_name[\instance_name]] [-U login_id] [-P password]
```

## Parámetros Comunes

- Importación y Exportación de Datos: Permite copiar datos entre una tabla de SQL Server y un archivo de datos en formato de texto (TXT, CSV, etc.).
- Soporte para Grandes Volúmenes de Datos: Diseñado para manejar grandes cantidades de datos de manera eficiente.
- Flexibilidad en el Formato de Datos: Admite varios formatos de datos y permite especificar formatos personalizados.
- Integración con Otros Procesos: Puede ser utilizado en scripts y procesos automatizados para la manipulación de datos.

- **-S server\_name[\instance\_name]:** Nombre del servidor y la instancia.
- **-U login\_id:** ID de inicio de sesión.
- **-P password:** Contraseña del usuario.
- **-T:** Utiliza la autenticación de Windows.
- **-c:** Utiliza el tipo de datos de carácter.
- **-t field\_term:** Especifica el delimitador de campo.
- **-r row\_term:** Especifica el delimitador de fila.
- **-b batch\_size:** Número de filas por lote de transacción.
- **-F first\_row:** Primera fila a copiar.
- **-L last\_row:** Última fila a copiar.
- **-m max\_errors:** Número máximo de errores permitidos.

## Mejores Prácticas (Best Practices) para el Uso de BCP

### Utilizar Autenticación Segura:

- Preferir la autenticación de Windows (-T) sobre la autenticación SQL (-U y -P), cuando sea posible.

### Control de Errores:

- Utilizar el parámetro -e para especificar un archivo de errores y -m para establecer el número máximo de errores permitidos.

### Optimización de Lotes:

- Ajustar el tamaño del lote (-b) para equilibrar la velocidad de carga y la utilización de recursos.

### Formato de Datos:

- Usar archivos de formato (-f) para manejar datos con formatos complejos o personalizados.

### Validación de Datos:

- Validar y limpiar los datos antes de la importación para minimizar errores y problemas de integridad.

### Paralelización:

- Dividir grandes conjuntos de datos en fragmentos más pequeños y procesarlos en paralelo para mejorar el rendimiento.

```
# Define una función para ejecutar un comando bcp para importar datos desde un archivo CSV
def run_bcp_command(path_csv, table_name):
    # Construye el comando bcp utilizando variables para la tabla, el archivo CSV, y las creaciones
    bcp_command = f"/opt/mssql-tools/bin/bcp {table_name} in {path_csv} -c -t, -S {DATABASE_NAME}"
    try:
        # Ejecuta el comando bcp utilizando subprocess.run. `shell=True` permite ejecutar el comando
        # `check=True` hace que se lance una excepción si el comando falla.
        subprocess.run(bcp_command, shell=True, check=True)
    except subprocess.CalledProcessError as e:
        # Captura cualquier error que ocurra durante la ejecución del comando bcp y lo imprime
        print(f"Error executing bcp command: {e}")
```

## Python y los esquemas ETL

The screenshot shows the Microsoft Azure Databricks interface. On the left, the sidebar includes options like Workspace, Recientes, Catalogo, Flujos de trabajo, Computo, Implementación de datos, Ejecuciones, Machine Learning, Zona de pruebas, Experimentos, Características, Modelos, and Servicio. The main area displays a Python notebook titled 'main'. The code within the notebook is as follows:

```
#!/usr/bin/python
# File: main.py
# Author: hugo ramirez (hugoramirez@multiplica.com)
# (Multiplica) node - Mexico

# from pyspark.sql import SparkSession
from databricks.sdk.runtime import dbutils

import sys
sys.path.insert(0, '/workspace/Users/hugoramirez@multiplica.com/layouts-batch-etl/src/')

import logging
import os
from modules.extract_module import *
from modules.transform_module import *
from modules.load_module import *
from utilities.utils import *

spark.conf.set("spark.sql.legacy.timeParserPolicy", "LEGACY")
table_name = "tmp_to_update_layout_012"
df.write.jdbc(url=JDBC_URL, table=table_name, mode="append", properties=connectionProperties)
```

The screenshot shows a Databricks job run history. It indicates a successful execution at 19:45 (47 m). The job consists of one Spark task, which contains one job named 'Job 32'. This job has one stage named 'Stage 35', which completed successfully. The code for the job is identical to the one shown in the first screenshot.



The screenshot shows a Python script named '\_grocers\_.py'. The code is as follows:

```
def _grocers_(self, layout_grocers_value) -> pd.DataFrame:
    try:
        self.layout_grocers_value = layout_grocers_value
        self.id_bodega = self.layout_grocers_value['ID_Bodega'].to_list()
        self.nud = self.layout_grocers_value['NUD'].to_list()
        self.rows = self.TEMPLATE_COL_001
        remove = ['Region', 'Territorio', 'Bodega', 'Programa']
        self.rows = [item for item in self.rows if item not in remove]
        self.rows = ', '.join(self.rows)
        id_bodega_str = format_tuple_for_sql(tuple(self.id_bodega))
        nud_str = format_tuple_for_sql(tuple(self.nud))
        query = f"(SELECT {self.rows}, ID_Region, ID_Territorio FROM Tenderos WHERE ID_Bodega IN {id_bodega_str} AND NUD IN {nud_str}) AS t"
        ID_Territorio
        self.response = spark.read.jdbc(url=self.jdbc_url, table=query, properties=connectionProperties)
        return self.response
    except Exception as e:
        raise Exception("operation error", e)
```

# Performance

## Integración a los recursos de Azure.

**Microsoft Azure | databricks**

Flujos de trabajo > Ejecuciones >  
Ejecución de ADF\_getm-factory-dev\_DataPipelineETL\_Main\_d2769e68-286c-47bb-901d-10646358b1ac

**Salida**

17.99 minutos

```

✓ 17.99 minutos
1
File set_promociones_to_update.parquet successfully uploaded to Azure Blob Storage. ✅
File set_promociones_clientes_to_insert.parquet successfully uploaded to Azure Blob Storage. ✅
Update DataFrame Layout012 is empty, skipping update. ⚠️
The parallelism load layouts base module took 23.710075 seconds to execute ✅
File set_maestro_ruta_cliente_lista_precios_to_insert.parquet successfully uploaded to Azure Blob Storage. ✅
File set_maestro_ruta_cliente_lista_precios_to_update.parquet successfully uploaded to Azure Blob Storage. ✅
File set_maestro_precios_to_insert.parquet successfully uploaded to Azure Blob Storage. ✅
File set_maestro_precios_to_update.parquet successfully uploaded to Azure Blob Storage. ✅
File set_maestro_productos_to_insert.parquet successfully uploaded to Azure Blob Storage. ✅
File set_maestro_productos_to_update.parquet successfully uploaded to Azure Blob Storage. ✅
File set_promociones_configuracion_to_insert.parquet successfully uploaded to Azure Blob Storage. ✅
File set_promociones_configuracion_to_update.parquet successfully uploaded to Azure Blob Storage. ✅
File set_productos_tipo_mercado_to_insert.parquet successfully uploaded to Azure Blob Storage. ✅
File set_maestro_rutas_to_insert.parquet successfully uploaded to Azure Blob Storage. ✅
The parallelism load layouts master module took 1.554 seconds to execute ✅
End load module ✅
Files processed 1197 ✅
The work of cleaning the workspace OFF ⚠️
Done.
[Finished data pipeline]
The pipeline 'main' took 1076.0233056545258 seconds to execute. 🎉

```

✓ 0.09 segundos

3

**Detalles de ejecución de la tarea**

- ID trabajo: 463163633914804
- ID ejecución tarea: 109861526958153
- Ejecutar como: Hugo Ramirez
- Inicio: 22/08/2024, 12:57:41
- Fin: 22/08/2024, 13:15:51
- Duración: 18m 9s
- Duración de la c... -
- Estado: Satisfactorio

[Ver los eventos de ejecución](#) [Ver bibliotecas de ejecuciones](#)

**Cuaderno**

/Users/hugoramirez@multiplica.com/layouts-batch-etl/main

**Cómputo**

- standard-batch-cluster

Nodo único: Standard\_DS3\_v2 - DBR: 13.3 LTS (includes Apache Spark 3.4.1, Scala 2.12)

[Detalles](#) [IU de Spark](#) [Logs](#) [Métricas](#)

**Utilización de la CPU**

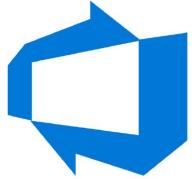
**Uso de intercambio de memoria**

**Recibido a través de la red**

5

DevOps.

## Python y los esquemas ETL



Despliegue basado en:

# Azure DevOps

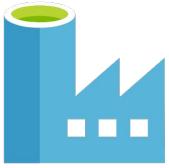
The screenshot shows the Azure DevOps Pipelines dashboard for the project "Gep en tus Manos". It lists four recent pipeline runs:

- #20240306.2 • updated -version (Run ID: c3641bf0, 4m 9s ago)
- #20240306.1 • updated -version (Run ID: c3641bf0, 3m 17s ago)
- #20240222.1 • updated -version (Run ID: 349d1956, 3m 16s ago)
- #20240125.2 • updated -version (Run ID: 9a26939b, 3m 27s ago)

The screenshot shows the detailed view of the pipeline run #20240306.2. It includes the following sections:

- Deployment:** Shows the build step, which took 3m 59s. Sub-tasks include: Initialize job (2s), Checkout gep-pealad-... (2s), Checkout gep-getm-... (1s), Set version TAG (less than 1s), Build Image (2m 47s), Push Image (1m 0s), Generate Yaml Files (less than 1s), Publish Artifacts (less than 1s), Push to Remote (less than 1s), Post-job: Checkout ge... (less than 1s), Post-job: Checkout ge... (less than 1s), Finalize Job (less than 1s), and Report build status (less than 1s).
- Push Image:** A log entry showing the Docker image push command: `/usr/bin/docker push ***/layouts-batch-etl:dev-0.50`. The log also lists various Docker images pushed, including node, alpine, and debian variants, along with their tags and creation times.

## Python y los esquemas ETL



Despliegue basado en:

# Azure Data Factory

The screenshot displays two Azure Data Factory (ADF) environments: 'getm-factory-dev' and 'getm-databricks-dev'.  
**getm-factory-dev:** This window shows a complex pipeline named 'ExecDataPipeline' consisting of seven parallel execution steps. Each step is labeled 'Ejecutar la canalización' and contains sub-tasks like 'DataPipelineETL', 'PiplineTendencias', 'PiplineProgramas', 'PiplinePromociones', and 'PiplineMaestros'. The pipeline is currently in a 'Validar todo' (Validate all) state.  
**getm-databricks-dev:** This window provides detailed execution logs for a specific run of the pipeline. It includes a log viewer showing tasks such as file uploads to Azure Blob Storage, parallelism load layouts, and end load module. The log concludes with '[Finished data pipeline]' and '[The pipeline 'main' took 1076.0233056545258 seconds to execute.]'.  
**Bottom Navigation:** Buttons for 'Detalles', 'IU de Spark', 'Logs', and 'Métricas' are visible at the bottom right of the Databricks execution details.

# Escalabilidad

**Escalabilidad Vertical (Scaling Up):**  
Aumentar la capacidad del cluster existente añadiendo más recursos, como CPU, memoria RAM y almacenamiento.

**Escalabilidad Horizontal (Scaling Out):**  
Añadir más servidores para distribuir la carga de trabajo. Esto puede involucrar particionamiento de datos y distribución del procesamiento entre múltiples máquinas.

**Uso de Tecnologías de Big Data:**  
Procesar grandes volúmenes de datos en un clúster de máquinas. Con herramientas como Apache Hadoop o Spark.

## Optimización de Consultas y Procesos:

- Indexación: Utilizar índices para acelerar el acceso a los datos.
- Particionamiento: Dividir grandes tablas en partes más pequeñas para mejorar la gestión y el acceso a los datos.
- Optimización de Consultas: Refinar las consultas SQL para reducir el tiempo de ejecución y el uso de recursos.

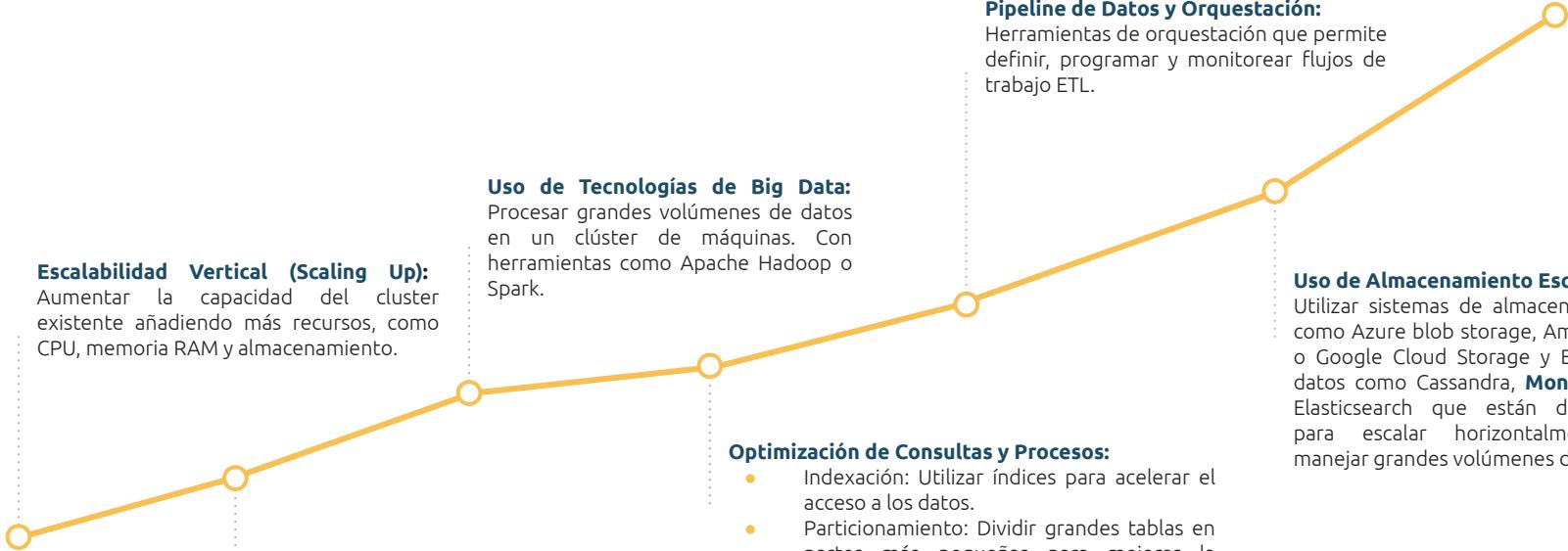
## Microservicios y Contenedores:

Descomponer el proceso ETL en pequeños servicios independientes que pueden escalarse individualmente.

## Pipeline de Datos y Orquestación:

Herramientas de orquestación que permite definir, programar y monitorear flujos de trabajo ETL.

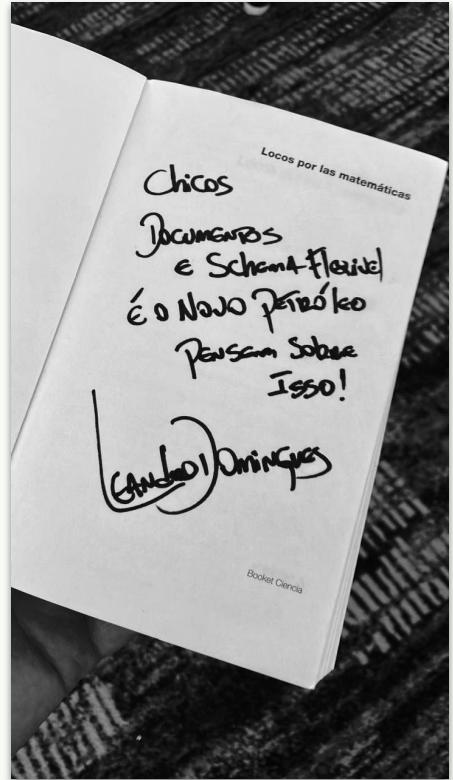
**Uso de Almacenamiento Escalable:**  
Utilizar sistemas de almacenamiento como Azure blob storage, Amazon S3 o Google Cloud Storage y Bases de datos como Cassandra, **MongoDB** o Elasticsearch que están diseñadas para escalar horizontalmente y manejar grandes volúmenes de datos.





# Consejos: Leandro Domingues

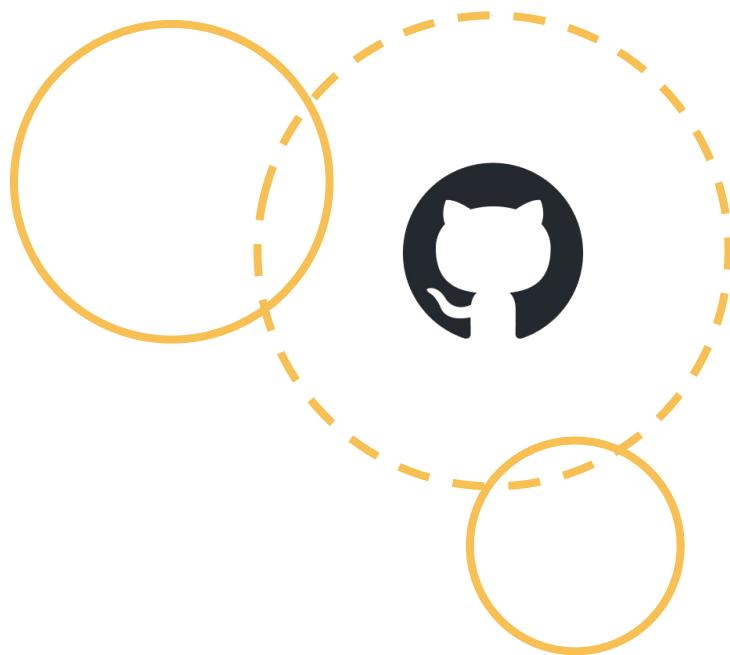
Microsoft Data Platform MVP, MongoDB Certified Developer



# 6

## Demo.

# Demo

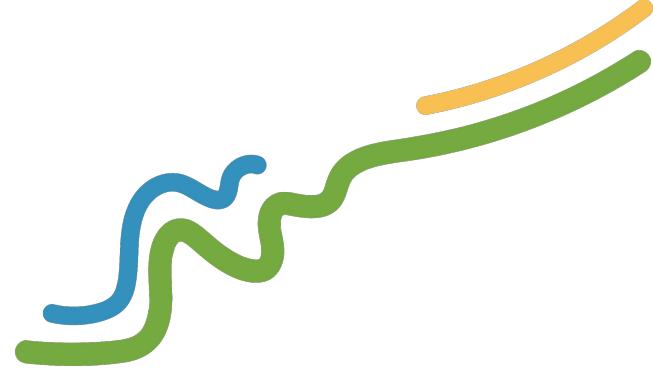




Thanks you...

7

# Preguntas.



to be continued...

