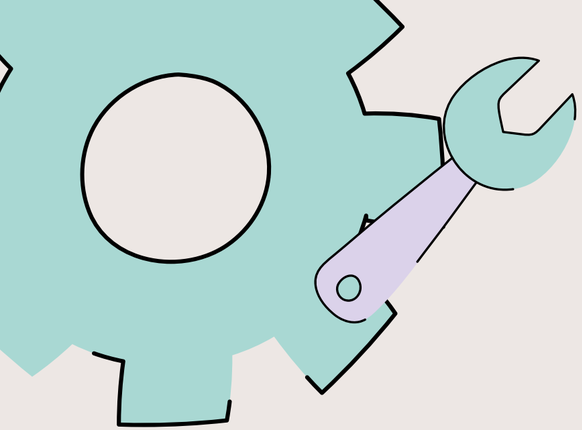
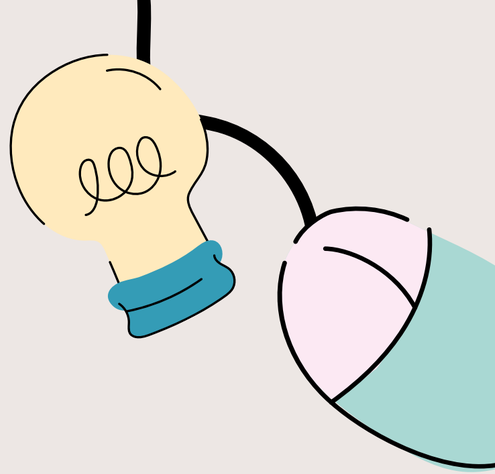
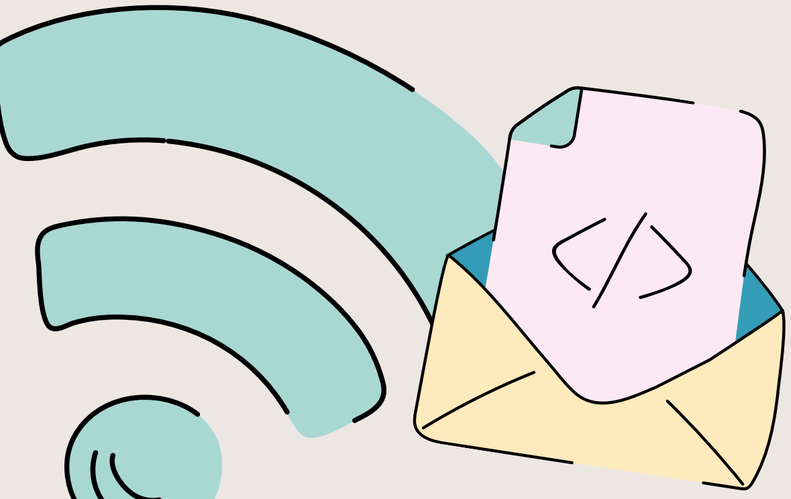


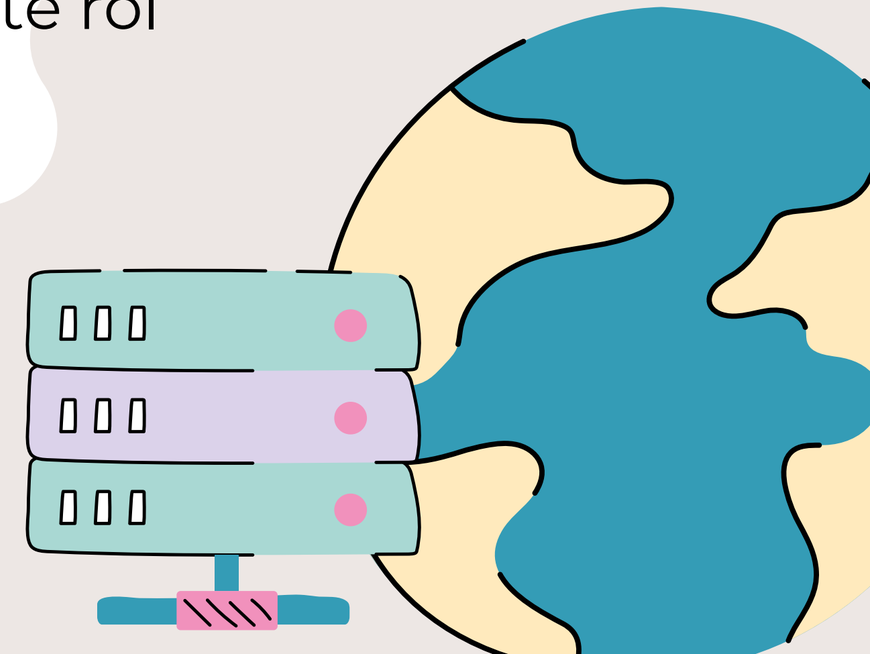


ROADMAP DATA ENGINEER

Una guía clara y sencilla para este rol

Por Juan Duran





Introducción

El camino para convertirte en **Data Engineer** puede parecer desafiante, pero con la información adecuada y una guía clara, es totalmente alcanzable.

Esta presentación está pensada para quienes quieren comenzar en el mundo de la **ingeniería** de **datos**, pero no saben por dónde empezar.

Te mostraré un **recorrido** estructurado, con **conceptos** clave, **herramientas** y **habilidades** que necesitas dominar, explicados de manera sencilla y accesible.

No es una lista rígida ni complicada, sino un **mapa flexible** para que tomes decisiones informadas y sigas aprendiendo a tu propio ritmo.

Empezaaamooos!!

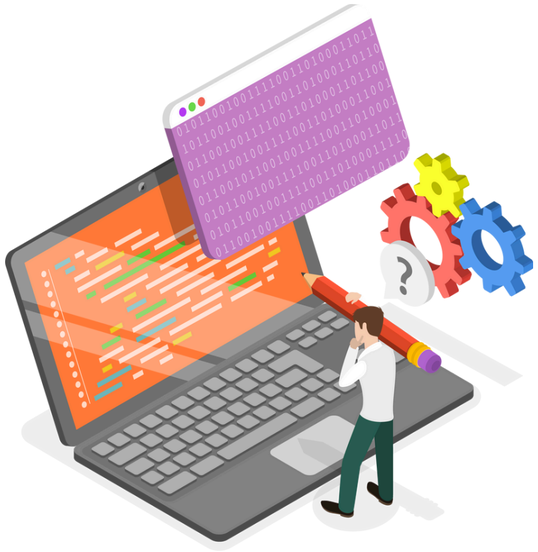
¿Por qué convertirte en Data Engineer?

Como **Data Engineer**, eres quien **construye** la base sobre la cual se desarrollan los análisis, las predicciones y las aplicaciones inteligentes. Sin una buena **arquitectura** de **datos**, incluso el mejor modelo de Machine Learning no servirá de mucho.

Además, la **demanda** de Data Engineers está creciendo constantemente. Cada vez más empresas están invirtiendo en sus capacidades de datos, y los Data Engineers son esenciales para **construir** y **mantener** esas **infraestructuras**.

¿Por qué es una buena opción?

- **Alta demanda**: Las empresas necesitan Data Engineers para garantizar que sus sistemas de datos sean robustos, escalables y eficientes.
- **Buenos salarios**: Es una de las áreas más rentables
- **Constante crecimiento**: La tecnología y las herramientas evolucionan, lo que significa que siempre tendrás algo nuevo que aprender y aplicar.



Aprende a programar

✅ Enfócate en **Python**: Es el **lenguaje más utilizado** en el mundo de los datos y la ingeniería, por su simplicidad y potencia.

✅ Aprende **estructuras básicas**: Asegúrate de dominar lo esencial, como **listas**, **diccionarios**, **funciones** y **bucles**.

✅ **Organiza** tu **código**: Aprende a estructurar tu código de forma clara y organizada.

✅ Usa **entornos virtuales** y **buenas prácticas**: Los entornos virtuales te permiten gestionar tus dependencias de manera aislada, lo que es esencial para evitar conflictos entre proyectos. Además, adoptar buenas prácticas desde el principio te ayudará a evitar errores y escribir código más limpio.

👉 No hace falta que seas un crack desde el principio, pero sí es importante que te sientas cómodo trabajando con estos conceptos.

Domina las BBDD

🔹 **Aprende SQL** (súper importante): SQL es el lenguaje estándar para interactuar con bases de datos relacionales. Es fundamental que domines consultas SELECT, filtros, joins y subconsultas.

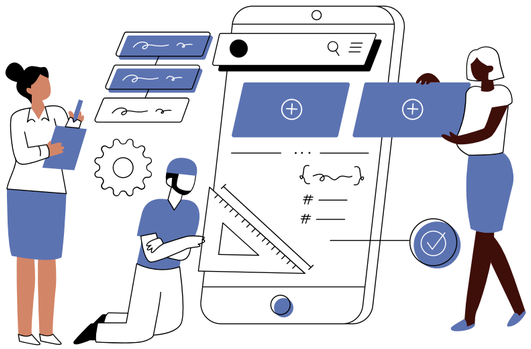
🔹 **Bases de datos relacionales** (**MySQL**, **PostgreSQL**): Las bases de datos relacionales son esenciales para manejar datos estructurados.

🔹 **Bases de datos NoSQL** (MongoDB): **MongoDB**, por ejemplo, es una base de datos NoSQL popular que usa un modelo de documentos, lo que es ideal para manejar grandes volúmenes de datos no estructurados o semiestructurados.

🔹 **Saber cuándo usar cada tipo según el proyecto**: No todas las bases de datos son adecuadas para cualquier tarea.

🔹 **Aprende sobre índices, esquemas y optimización básica**: Los índices mejoran el rendimiento de las consultas, y saber cómo optimizar consultas o diseñar esquemas eficientes es crucial para manejar grandes volúmenes de datos sin sacrificar el rendimiento.

Qué es un pipeline ETL/ELT



💡 Un **pipeline** de datos es un **proceso automatizado** que permite mover los datos desde su origen hasta su destino final de manera eficiente. Este proceso se lleva a cabo en tres etapas principales:

1. **Extraer datos** de una fuente: Los datos pueden provenir de diversas fuentes, como bases de datos, APIs o archivos planos.
2. **Transformarlos** (limpiar, convertir, unir): En esta etapa, los datos extraídos son limpiados (eliminando valores nulos, duplicados, etc.), transformados (modificando su formato o estructura) y unidos si es necesario (combinando datos de diferentes fuentes)
3. **Cargarlos** en su destino final: Una vez transformados, los datos se cargan en el sistema de almacenamiento final.

🔧 **Herramientas** útiles: Para crear estos pipelines, hay herramientas poderosas como **Airflow**, que te ayuda a programar y monitorear flujos de trabajo

⚙️ **Automatizar** estos procesos es **clave** en tu día a día como Data Engineer, ya que te permite mantener los datos actualizados de manera eficiente y sin intervención manual.

Almacenamiento y procesamiento de datos

Data Lakes (ej. S3): Los Data Lakes son repositorios de datos donde se almacenan grandes volúmenes de datos sin procesar o crudos, como logs, archivos de texto o datos de sensores.
Data Warehouses (BigQuery, Snowflake): Los Data Warehouses están diseñados para almacenar datos estructurados y optimizados para realizar análisis. Herramientas como BigQuery y Snowflake permiten consultas rápidas y análisis avanzados

🔄 Entiende cómo se **procesan grandes volúmenes.**

📚 **Arquitecturas útiles**: La arquitectura Lambda, que permite procesar datos en tiempo real y batch, y la arquitectura Medallion, que organiza los datos en capas para garantizar su calidad y facilidad de acceso.

🧠 **Saber elegir** dónde y cómo guardar los datos es una parte esencial del rol de un Data Engineer.

Cloud y DevOps

☁️ Trabaja en la **nube** (**AWS, Azure, GCP**). Con ellas puedes:

- Subir datos y almacenarlos de forma segura.
- Crear bases de datos escalables y listas para producción.
- Lanzar máquinas virtuales.

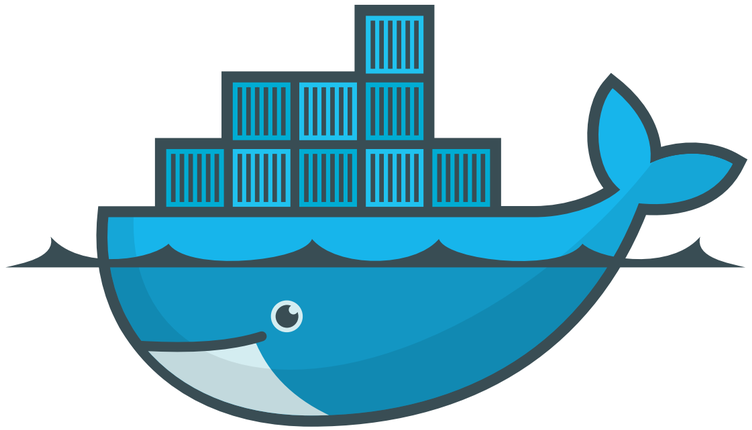
🔄 **Automatiza** tareas con scripts o servicios.

🛠️ Aprende **DevOps** básico:

- **Git**: control de versiones y colaboración.
- **CI/CD**: integración y despliegue continuo.
- **Docker**: para empaquetar tus aplicaciones en contenedores.
- **Terraform**: para definir infraestructura como código.

📌 ¿Por qué importa esto?

Porque te **permite construir proyectos sólidos, replicables**, y que otros puedan mantener.



Python para procesamiento de datos

🔧 **Transforma** datos con **pandas**.

🧩 **Conecta** con bases de datos:

- Usa librerías como **SQLAlchemy**
- Puedes extraer datos directamente a un **DataFrame**

📂 **Maneja** archivos grandes:

- Aprende a trabajar con formatos como CSV, Parquet, o JSON.
- Evita cargar todo en memoria si no es necesario

💼 **Organiza** tu código bien: Sigue buenas prácticas de codificación.

🎯 **Trabaja** con eficiencia:

- No todo cabe en memoria: aprende a optimizar tus scripts
- Evita cargar más de lo necesario.

Herramientas clave del día a día

Apache Spark: Ideal para procesar grandes volúmenes de datos distribuidos en varios nodos.

Apache Kafka: Tecnología clave para trabajar con datos en tiempo real. Te permite recibir, procesar y distribuir eventos (como clics, transacciones, logs) en tiempo real

dbt: Es una herramienta que te permite transformar datos directamente en el warehouse usando solo SQL.

Apache Airflow: Sirve para automatizar y programar flujos de trabajo (pipelines).

Git:Tu mejor amigo para trabajar en equipo.

📚 **Familiarízate** con ellas poco a poco: No hace falta aprender todo de golpe.

Soft skills y buenas prácticas

- **Escribe código limpio y comentado**

- **Documenta tus procesos**:
  - Explica cómo usar tu pipeline, tus funciones o scripts.
  - Documentar bien ahorra tiempo a todo el equipo.

- **Comunica tus ideas** al equipo:
  - Comparte avances y bloqueos.
  - Trabajar en equipo es fundamental: tú no lo haces todo solo.

- **Aprende a pedir ayuda**:
  - No es una señal de debilidad, sino de inteligencia.
  - Tu equipo está para apoyarte (y tú para apoyar también).

🧠 **Recuerda**:
La actitud, las ganas de aprender y la capacidad de colaborar son tan importantes como saber Python o SQL.

Y para concluir...

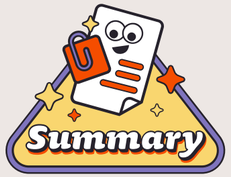
💬 Esto no es una checklist obligatoria... es una **brújula**.

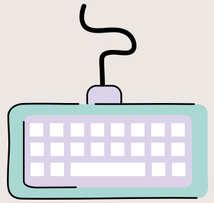
📍 **Cada paso** que des, por pequeño que parezca, te acerca a tu objetivo.

⏳ **No corras. Aprende a tu ritmo**.

💡 Lo más importante no es saberlo todo, sino tener:

- **Curiosidad**: por entender cómo funcionan las cosas.
- **Constancia**: para seguir incluso cuando cueste.
- **Ganas**: de mejorar cada día, un poco más.





Gracias

Por Juan Duran

“Coding, Gaming and Leveling Up”