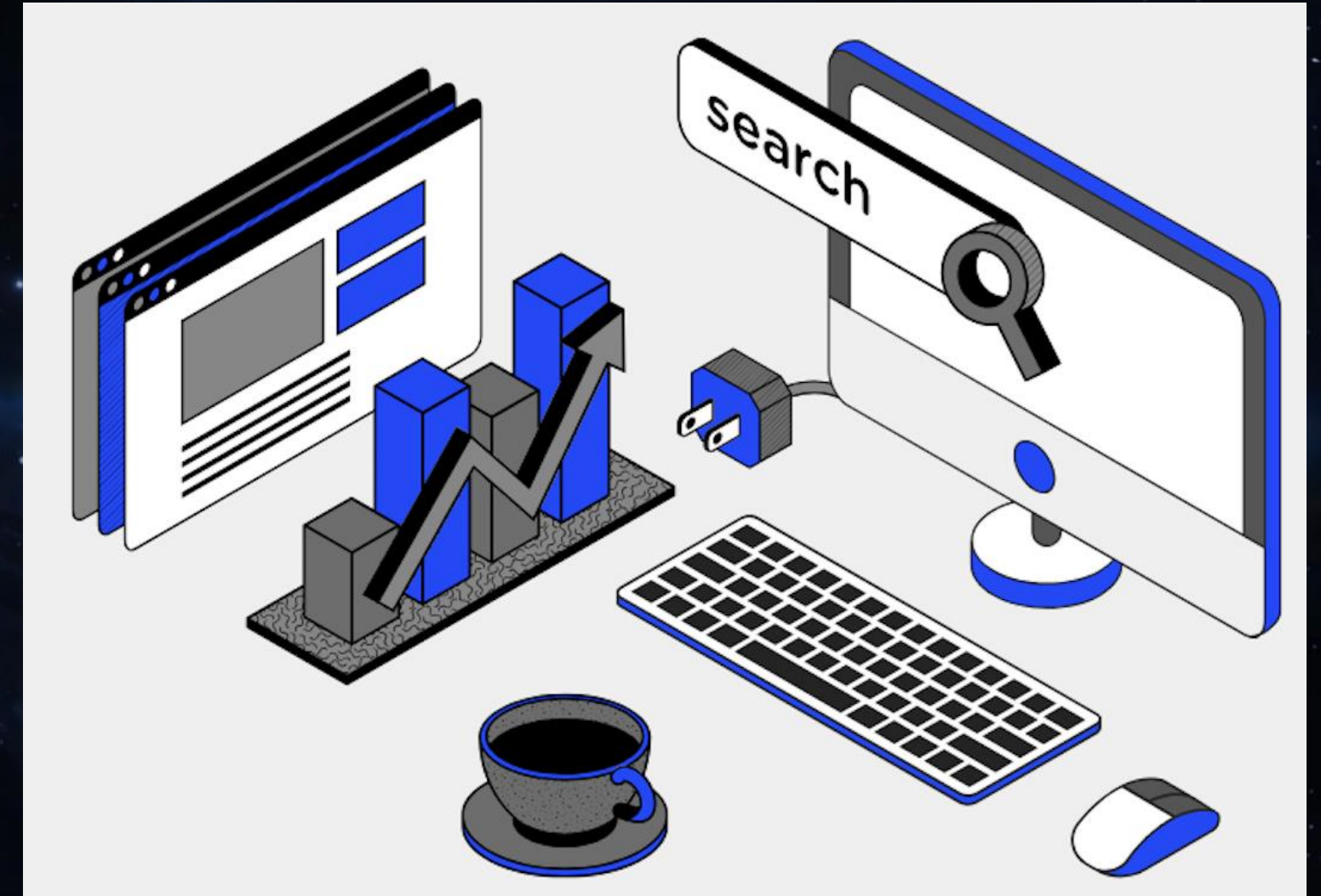


#HazloConDatos



Fundamentos de la Ingeniería de Datos y Python



Kremlin Huaman
Senior Data Engineer



<https://www.linkedin.com/in/khuamans/>

Agenda

1. Bienvenida e introducción al curso

- Objetivos del curso y dinámica de trabajo.
- Materiales, recursos y canales de comunicación.

2. Herramientas del curso

- Python, Pandas, NumPy, SQLAlchemy.
- VS Code / Jupyter Notebooks.
- Git y GitHub para control de versiones.

3. ¿Qué es la Ingeniería de Datos?

- Rol del Data Engineer dentro del ecosistema de datos.
- Arquitecturas modernas: Data Lake, ETL y pipelines.

4. ¿Por qué Python para Ingeniería de Datos?

- Versatilidad y ecosistema de librerías.
- Casos de uso reales en data engineering.

5. Introducción a Git

- Conceptos clave: repositorio, commit, push, branch.
- Conexión con GitHub y flujo de trabajo básico.

Bienvenida e introducción al curso

Objetivo del curso:

Aprender a construir y automatizar pipelines de datos con Python, aplicando las buenas prácticas de la Ingeniería de Datos y utilizando herramientas del ecosistema moderno de datos.

Modalidad de las sesiones:

- Todas las sesiones serán transmitidas en vivo por YouTube.
- Cada sesión incluirá teoría, práctica guiada y ejercicios aplicados.



Carpeta compartida:

- Se habilitará una carpeta para materiales, notebooks y ejercicios del curso.
- Todo el contenido estará disponible luego de cada sesión.



Canales de comunicación:

- Grupo oficial de WhatsApp para resolver dudas, compartir anuncios y materiales.
- Uso colaborativo del espacio para consultas y apoyo entre participantes.



Herramientas del curso

Lenguaje principal

Python: Base del curso. Lenguaje versátil para construir ETL, automatizar tareas y procesar grandes volúmenes de datos.

Procesamiento y transformación

Pandas: Manipulación y limpieza de datos tabulares.

SQLAlchemy: Conexión a bases de datos y ejecución de consultas SQL desde Python.

Orquestación y automatización

Airflow: Creación y programación de flujos ETL.

Permiten definir dependencias, tareas y monitorear ejecuciones automáticas.

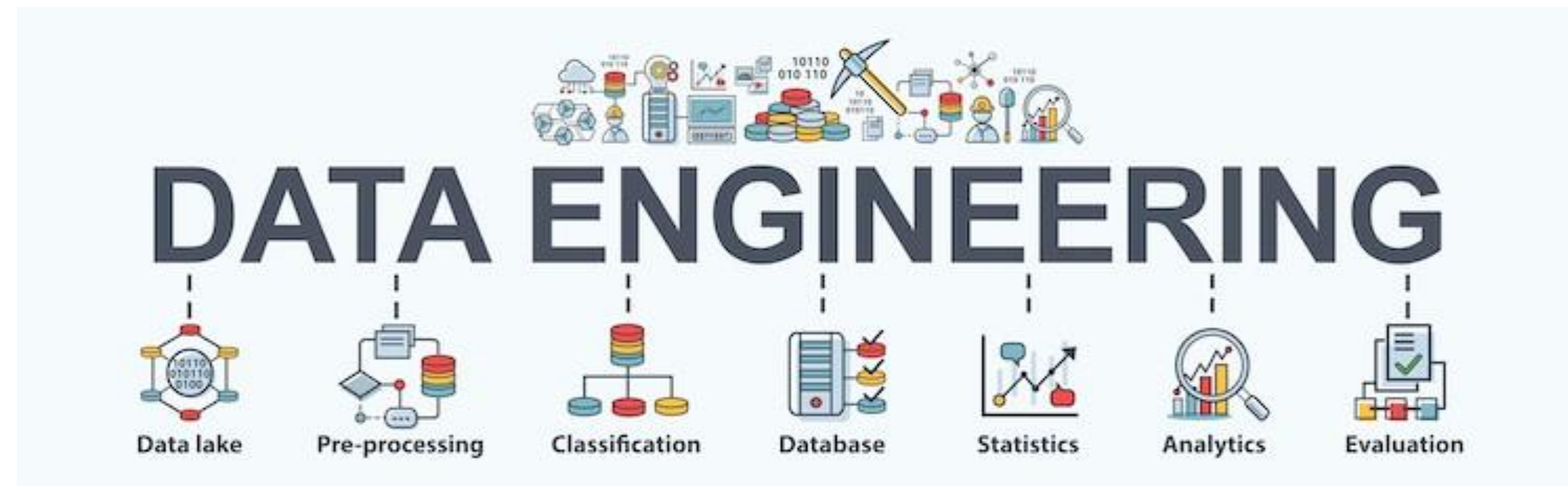
Control de versiones

Git: Seguimiento de cambios y trabajo colaborativo.

GitHub: Repositorio central para almacenar código y compartir proyectos

¿Qué es la Ingeniería de Datos?

La Ingeniería de Datos es la disciplina que se centra en diseñar, construir y mantener sistemas que permiten recopilar, almacenar y analizar grandes volúmenes de datos.



Rol del Ingeniero de datos en Proyectos

Funciones principales:

- Diseñar y mantener pipelines de datos.
- Integrar múltiples fuentes de datos.
- Optimizar el almacenamiento y consulta de datos.
- Colaborar con científicos de datos, analistas y otros roles.

Rol del Ingeniero de datos en Proyectos

Habilidades clave:

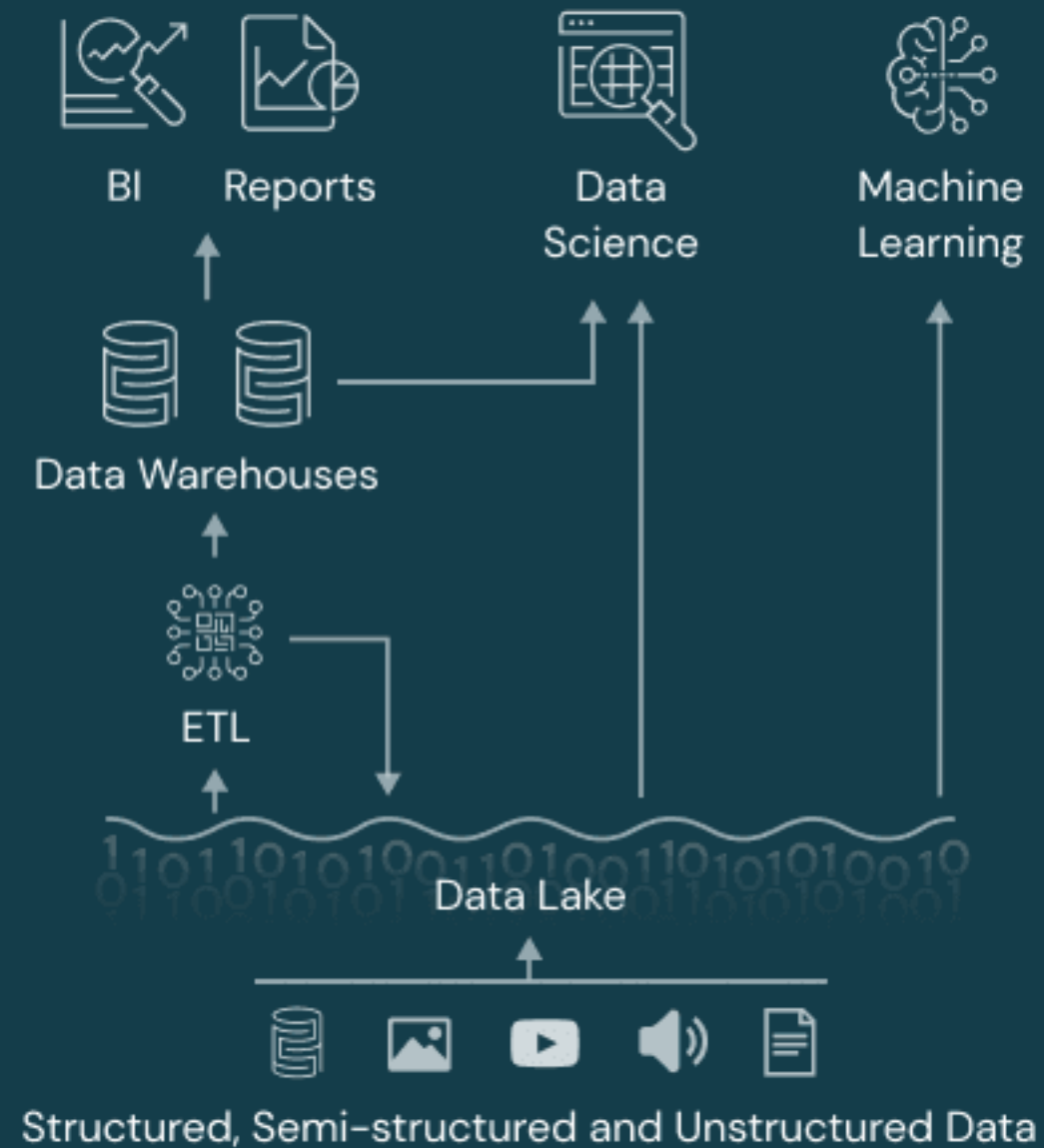
- Conocimientos en lenguajes de programacion (python, Scala, SQL).
- Familiaridad con herramientas como Spark, Databricks y bases de datos

Arquitecturas Modernas

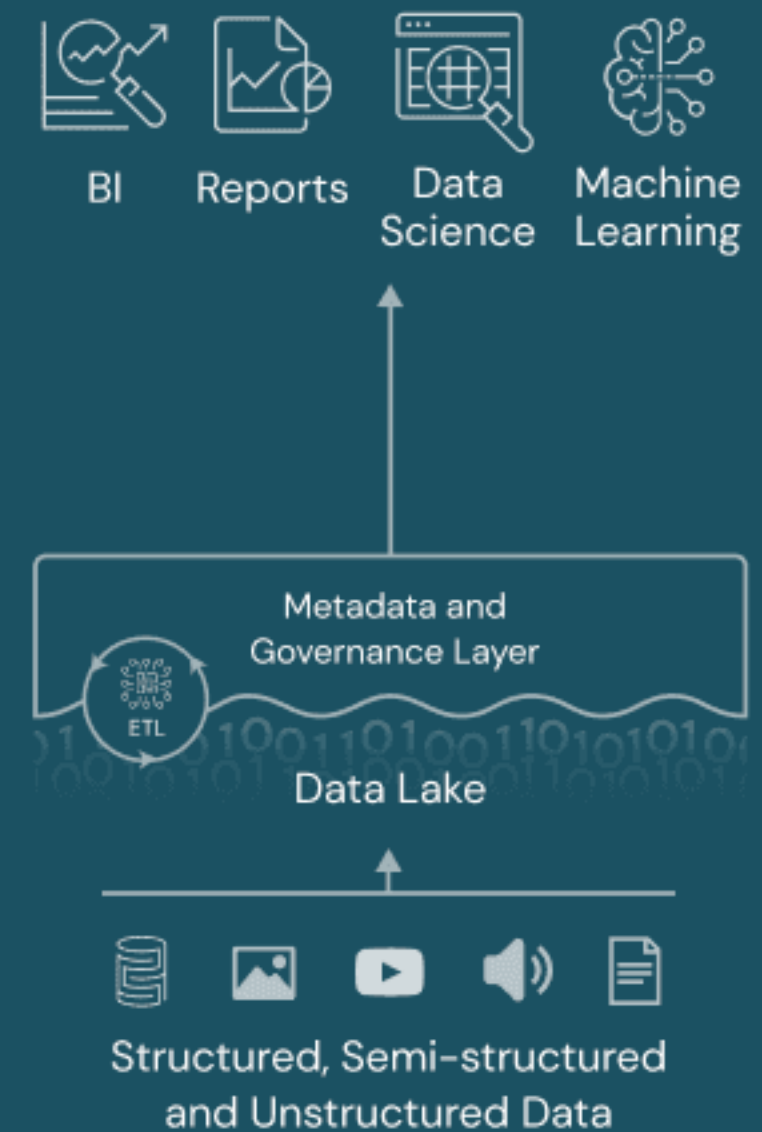
Data Warehouse



Data Lake



Data Lakehouse

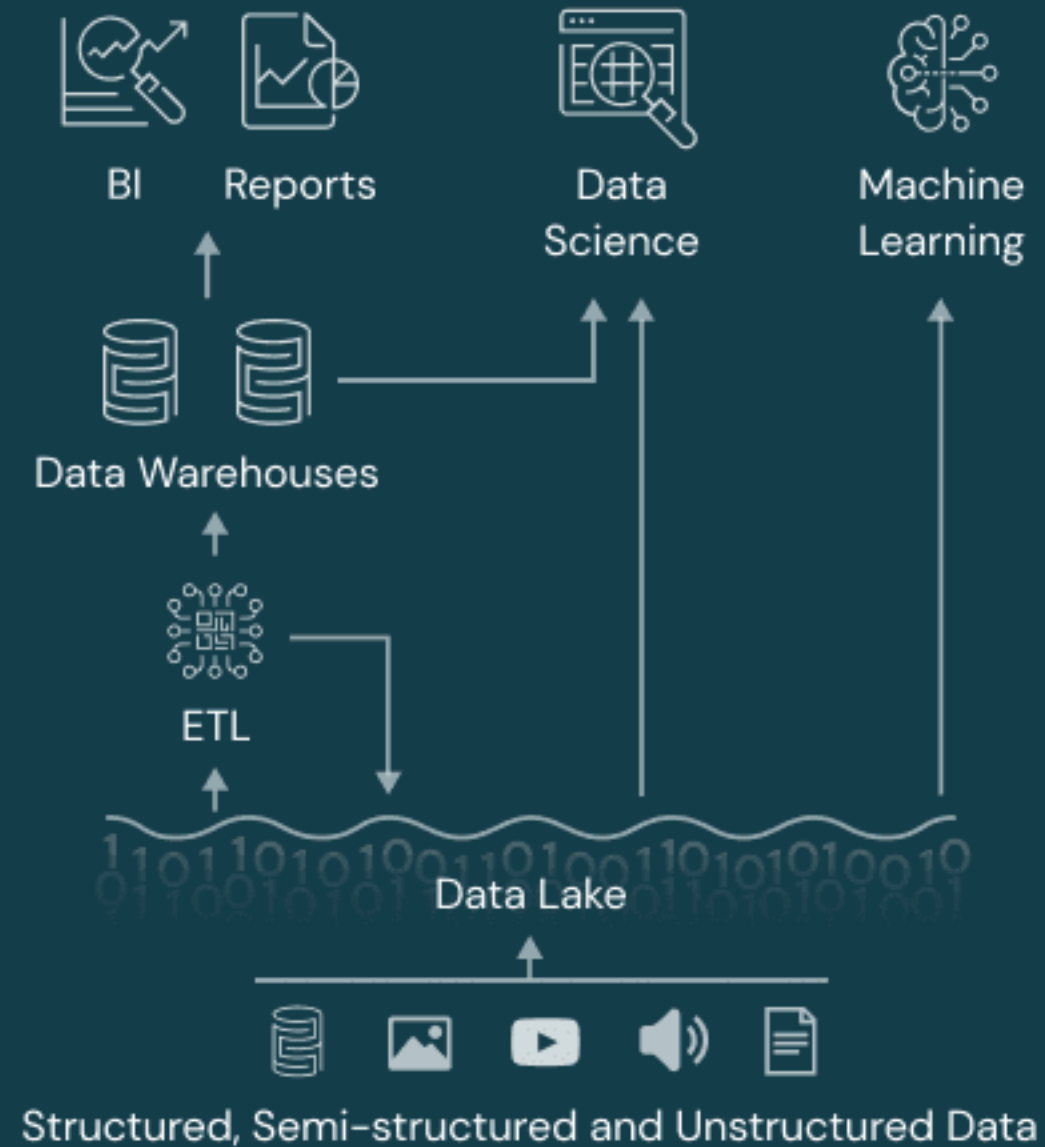


Arquitecturas Modernas

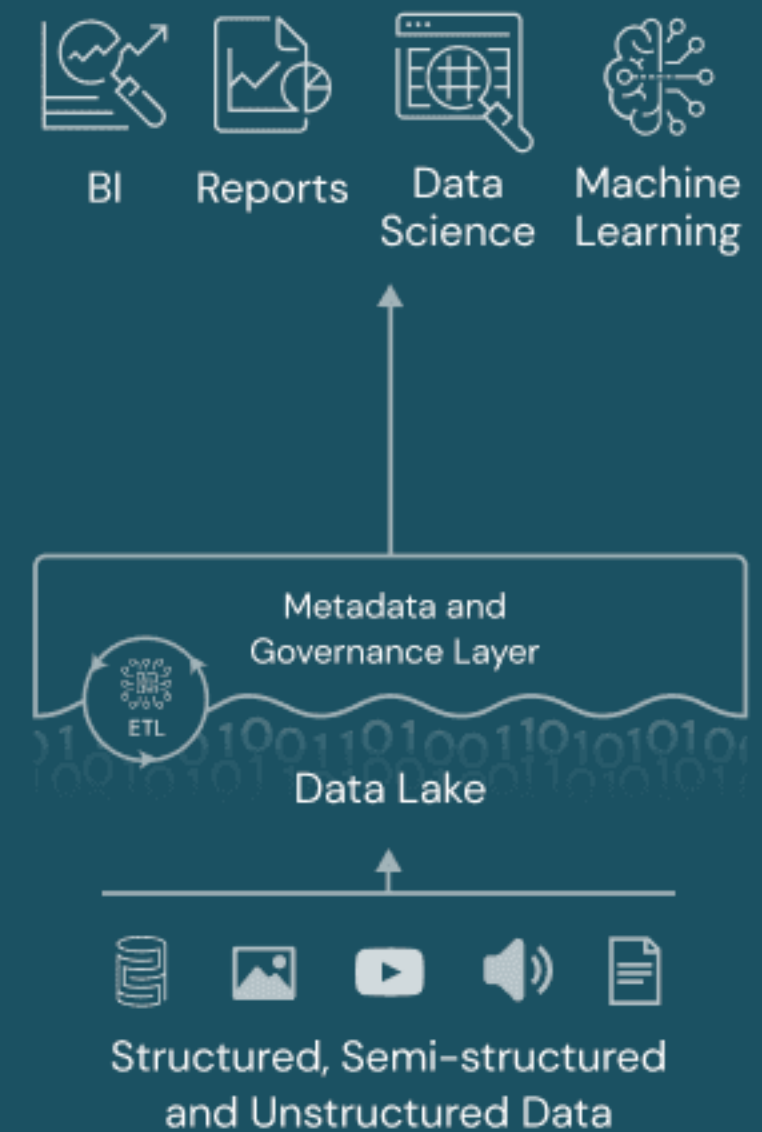
Data Warehouse



Data Lake



Data Lakehouse

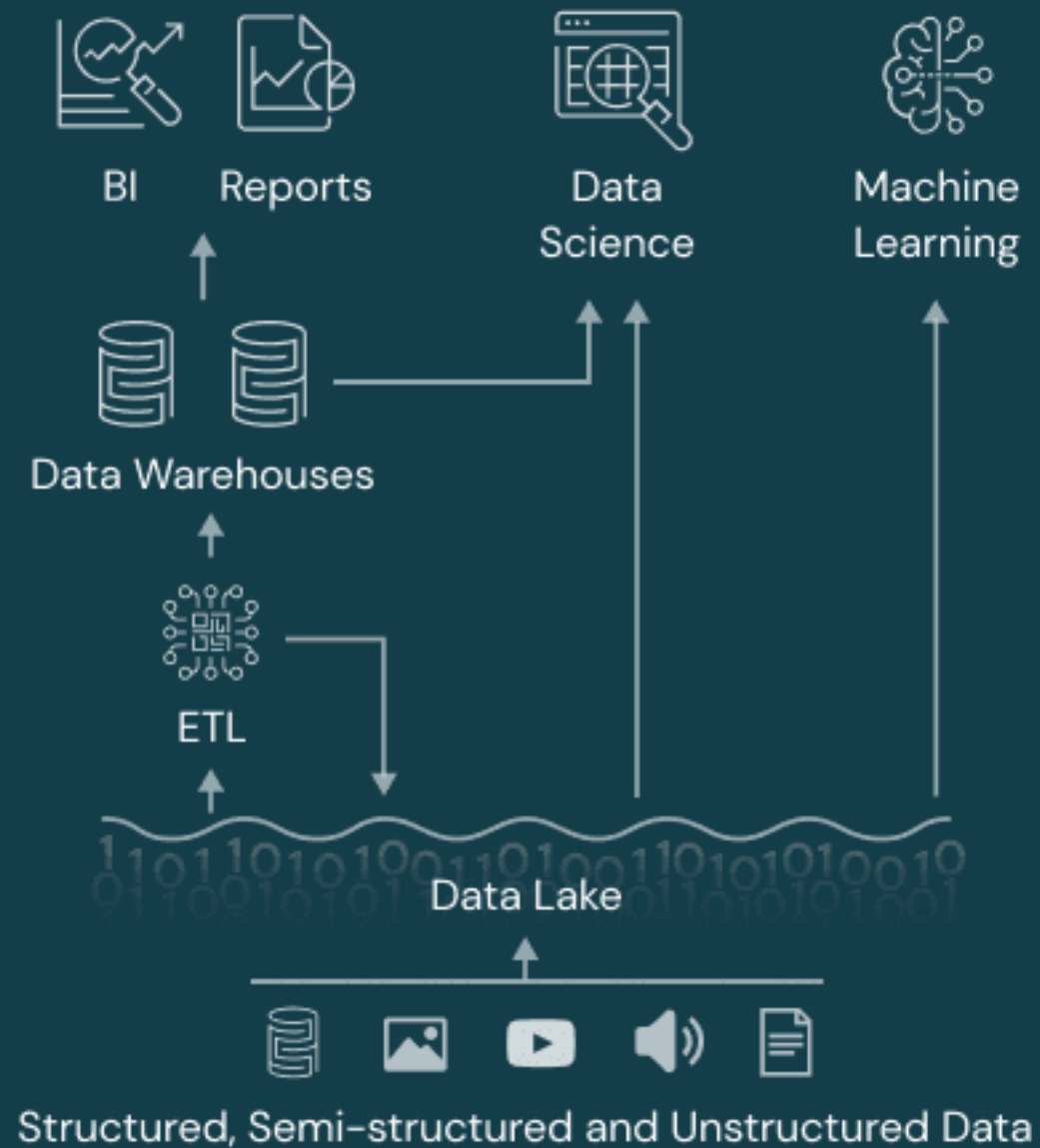


Arquitecturas Modernas

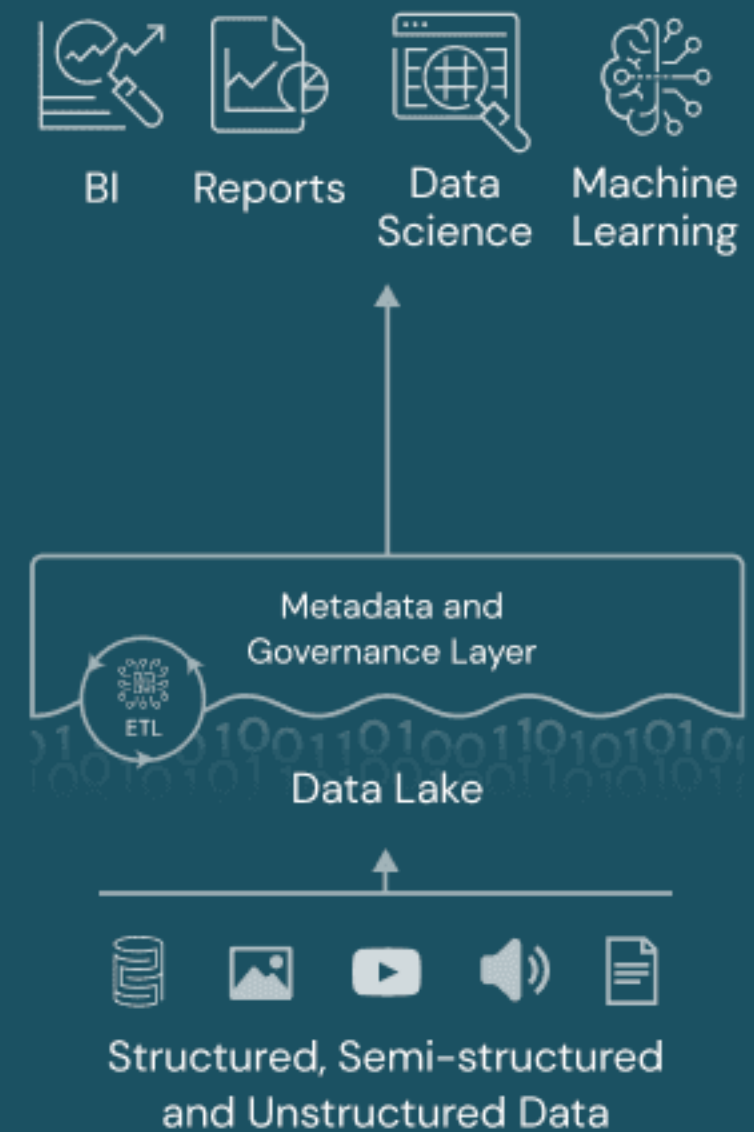
Data Warehouse



Data Lake



Data Lakehouse



¿Por qué Python para Ingeniería de Datos?

Python se ha convertido en el lenguaje más utilizado en el mundo de los datos.

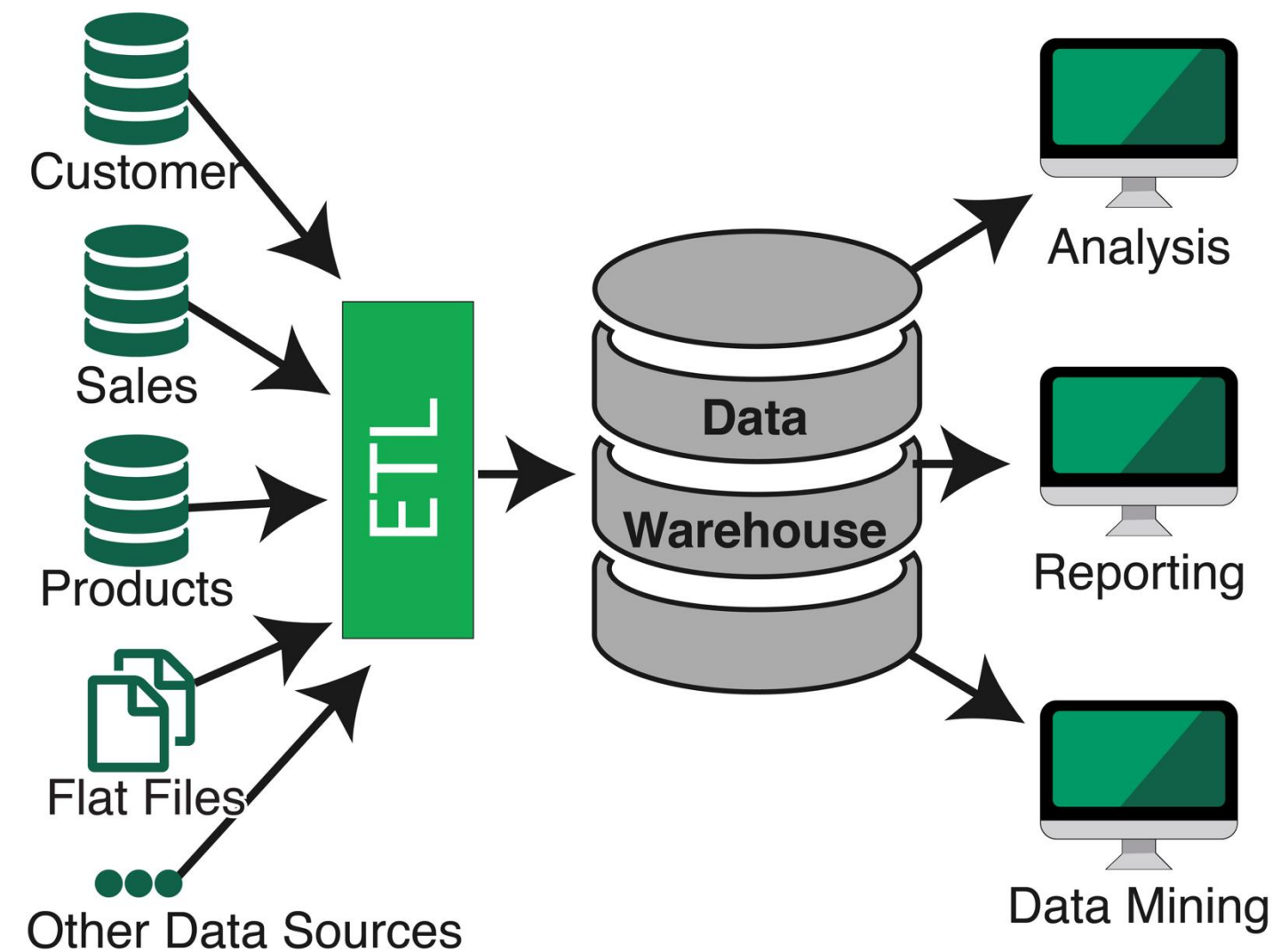
Es potente, versátil y cuenta con una gran comunidad, lo que lo convierte en la herramienta ideal para construir pipelines, integrar sistemas y transformar datos a escala.

Ventajas de python

- Facilidad de aprendizaje y lectura
- Ecosistema de librerías especializado en datos
- Automatización y escalabilidad
- Integración con múltiples fuentes y plataformas
- Comunidad y soporte

Casos de uso

- Construir un ETL automatizado que extrae datos de una API y los carga a una base SQL.
- Procesar millones de registros en Data Lakes usando PySpark.
- Implementar controles de calidad y validación de datos en pipelines.
- Crear scripts de mantenimiento, limpieza o auditoría de tablas.



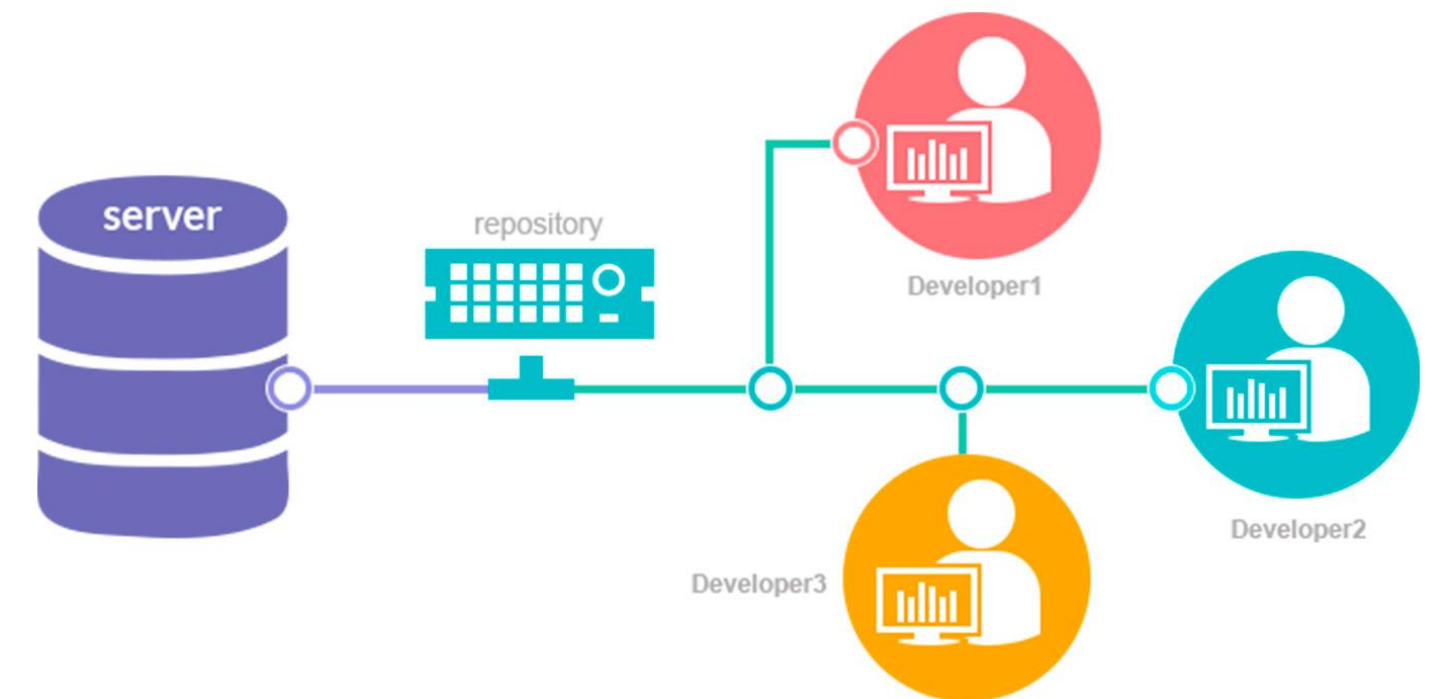
Sistema de control de versiones

Sistema de Control de versiones

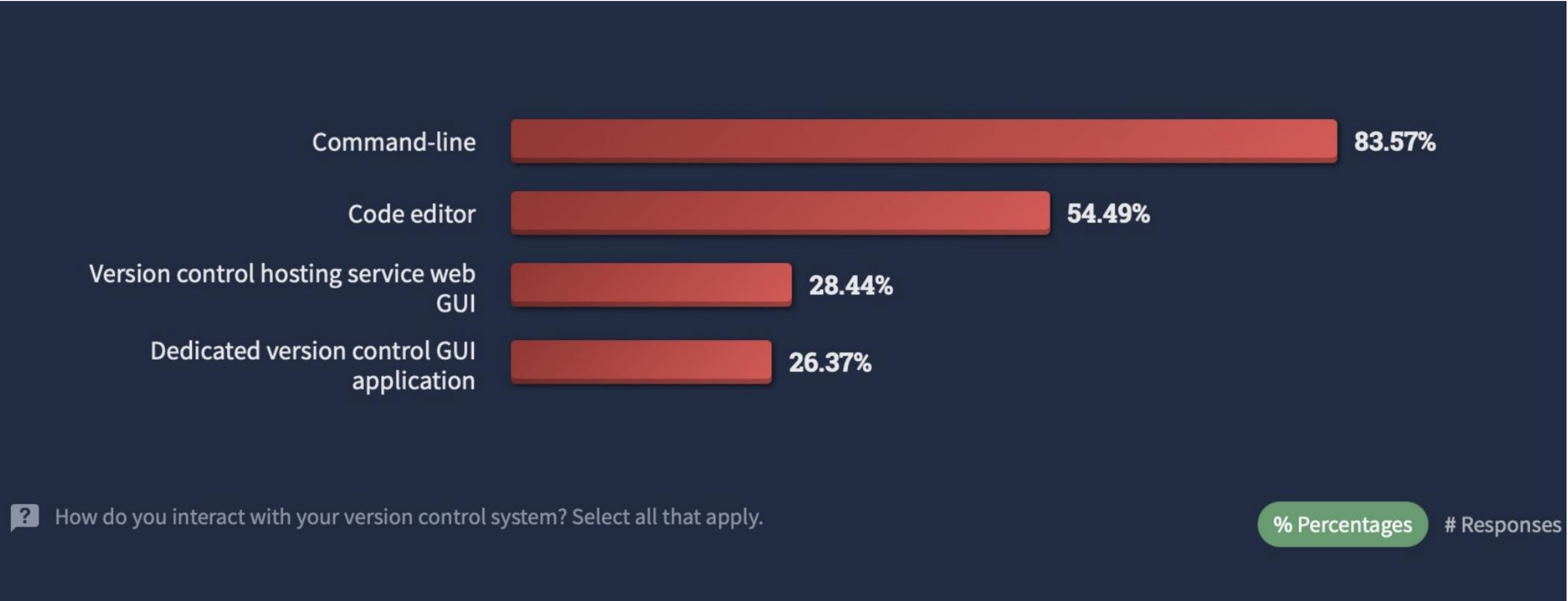
Sistema que registra cambios en un archivo o conjunto de archivos a lo largo del tiempo para que puedas recuperar versiones específicas más adelante.

- Mantener un historial de cambios.
- Facilitar la colaboración entre equipos.
- Permitir la experimentación sin afectar el código base.
- Respaldo y restauración.
- Auditoría y responsabilidad.

Sistema de Control de Versiones



Sistema de Control de versiones





¿Qué es Git y por qué es importante?

1

Descentralizado

Git es un sistema de control de versiones distribuido, lo que significa que cada desarrollador tiene una copia completa del repositorio en su máquina local.

2

Rápido y Eficiente

Git realiza operaciones de manera rápida y eficiente, lo que lo convierte en una herramienta ideal para proyectos grandes y de larga duración.

3

Código Abierto

Git es de código abierto, lo que significa que es gratuito y constantemente mejorado por la comunidad.

Conceptos básicos de Git

Repositorios

Un repositorio de Git es un directorio que contiene todos los archivos de un proyecto y su historial de versiones.

Commits

Un commit es una instantánea del estado actual de tu proyecto. Cada commit tiene un identificador único y un mensaje descriptivo.

Ramas

Las ramas permiten tener múltiples líneas de desarrollo simultáneas. Cada rama tiene su propio historial de commits.

Flujo de trabajo con git

1

Inicializar

Crear un nuevo repositorio de Git en tu proyecto.

2

Agregar

Preparar los archivos que quieres incluir en el próximo commit.

3

Confirmar

Crear un nuevo commit con un mensaje descriptivo.

4

Enviar

Subir tus commits al repositorio remoto.

Trabajar con ramas

Creación de ramas

Crea nuevas ramas para experimentar con características o correcciones sin afectar la rama principal. Mantén un flujo de trabajo ordenado y organizado.

Fusión de ramas

Combina el trabajo de diferentes ramas mediante el proceso de fusión (merge). Resuelve cualquier conflicto que pueda surgir durante este proceso.

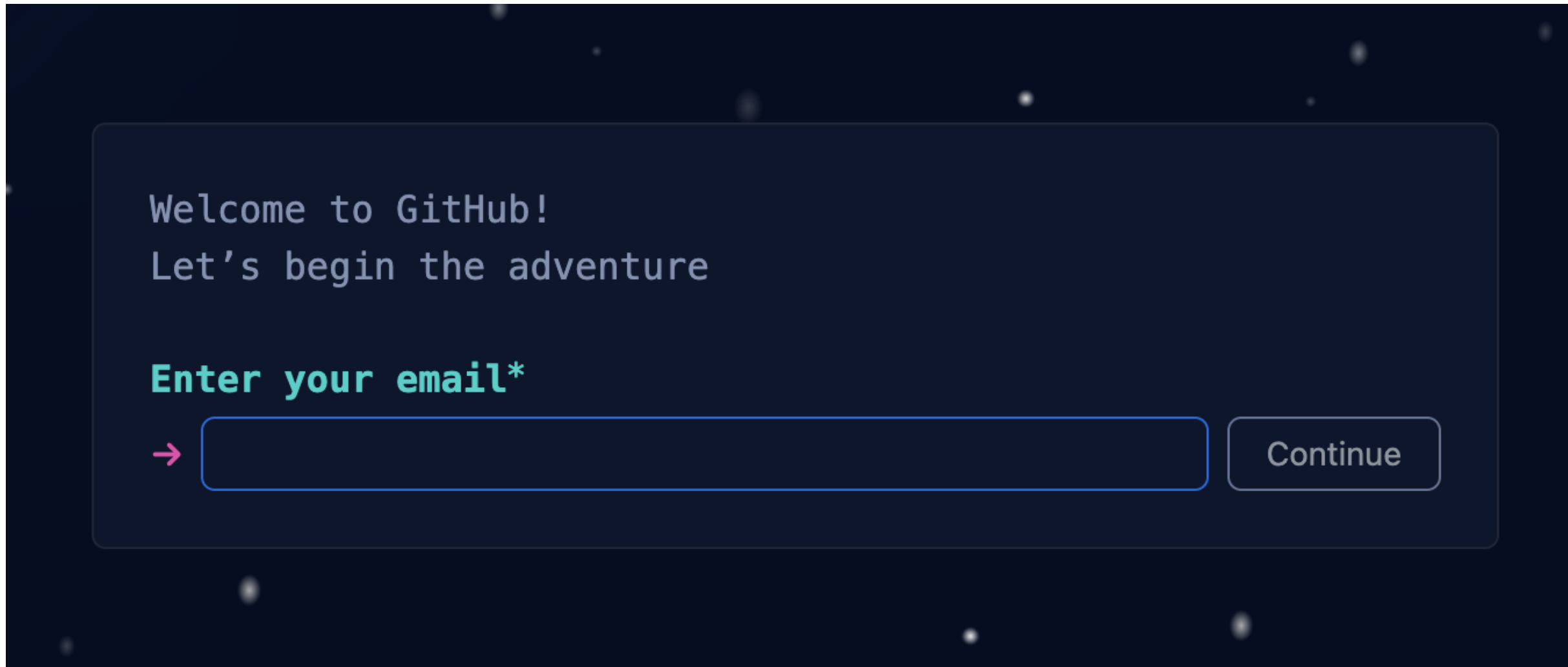
Resolución de conflictos

Cuando Git no puede resolver automáticamente los cambios en conflicto, debes revisarlos manualmente y decidir qué versión mantener. Esto te permite conservar el control sobre tu proyecto.



Crear una cuenta

<https://github.com/>

A screenshot of the GitHub account creation interface. The background is dark blue with a subtle pattern of white dots. A central white rounded rectangle contains the text "Welcome to GitHub!" and "Let's begin the adventure". Below this, the text "Enter your email*" is displayed in a light blue color. To the left of a white email input field is a small red arrow pointing right. To the right of the input field is a white "Continue" button.

Welcome to GitHub!
Let's begin the adventure

Enter your email*

→

Continue

Colaboración en GitHub



Buenas practices y estrategias de Git

Commits Significativos

Realiza commits lógicos y bien documentados que faciliten el seguimiento del historial del proyecto. Evita mezclar cambios dispares en un solo commit.

Ramas Organizadas

Mantén un sistema de ramas claro y coherente, con nombres descriptivos que indiquen el propósito de cada rama. Evita el desorden y la confusión.

Revisión de Código

Implementa revisiones de código sistemáticas para mantener altos estándares de calidad, identificar problemas y compartir conocimientos dentro del equipo.



Muchas Gracias