**PARTE II: Teoria**

1. **¿Qué es Apache Airflow y cuál es su función principal en un entorno de administración de flujos de trabajo? ¿has trabajado con apache Airflow?.**

Apache Airflow es una plataforma de código abierto para programar, gestionar y monitorear flujos de trabajo complejos. Su principal función es orquestar tareas que dependen entre sí, permitiendo automatizar procesos como pipelines de datos o tareas repetitivas.

No he trabajado con apache Airflow.

1. **¿Qué es Apache Spark y en qué se diferencia de Hadoop? ¿has trabajado con apache Airflow?.**

Apache Spark es una plataforma de procesamiento de datos en memoria, diseñada para manejar grandes volúmenes de datos de manera rápida y eficiente. Spark permite realizar análisis de datos distribuidos y operaciones complejas como consultas SQL, machine learning, y procesamiento de streams en tiempo real, todo con un enfoque en la velocidad.

La principal diferencia es que Spark procesa los datos en memoria, mientras que Hadoop los procesa en disco, lo que hace que Hadoop sea más lento.

No he trabajado con Spark.

1. **¿Qué es la ETL/ELT en el contexto de la ingeniería de datos y cuáles son sus componentes principales? ¿Qué herramientas has usado para hacer una ETL/ELT?.**

ETL (Extract, Transform, Load) y ELT (Extract, Load, Transform) son procesos fundamentales para mover y transformar datos entre sistemas.

* Extract (Extracción): Los datos se extraen de diferentes fuentes (bases de datos, APIs, archivos).
* Transform (Transformación): Los datos se limpian, filtran y transforman según los requisitos.
* Load (Carga): Los datos transformados se cargan en el destino (data warehouse, base de datos)

1. **¿Cuál es la diferencia entre Git y GitHub? ¿Cómo utilizarías Git en un proyecto de ingeniería de datos?**

Git es un sistema de control de versiones distribuido. Permite a los desarrolladores realizar un seguimiento de los cambios en su código, gestionar múltiples versiones de un proyecto, y colaborar con otros de manera eficiente.

GitHub es una plataforma en la nube para alojar repositorios Git. Es un servicio que utiliza Git como sistema de control de versiones y ofrece herramientas de colaboración, revisión de código, y gestión de proyectos.

* Los scripts ETL/ELT, notebooks de análisis de datos, y configuraciones de infraestructura pueden cambiar con el tiempo. Git permite realizar un seguimiento de las versiones de estos archivos y retroceder si es necesario.
* Git facilita el trabajo en equipo, permitiendo que varios ingenieros de datos trabajen en diferentes partes del proyecto (por ejemplo, en diferentes ramas) y luego fusionar los cambios de manera controlada.
* Con Git y GitHub, se puede implementar revisiones de código para que otros miembros del equipo revisen los cambios en los scripts de procesamiento de datos antes de fusionarlos en la rama principal.

1. **¿Qué es un pull request en Git y cómo lo utilizarías en un flujo de trabajo colaborativo?**

Un pull request es una solicitud para fusionar cambios de una rama en otra, generalmente en la rama principal, dentro de un repositorio Git. Facilita la revisión de código y asegura que los cambios sean validados antes de integrarlos.

Se puede usar de esta forma:

* Un desarrollador crea una nueva rama y realiza cambios.
* Envía un pull request para que el equipo revise los cambios.
* Otros miembros revisan, comentan y, si todo está bien, aprueban la fusión.
* Una vez aprobado, los cambios se integran en la rama principal.

1. **¿Puedes explicar qué es el branching en Git y cómo implementarías una estrategia de branching en un proyecto de ingeniería de datos?**

Branching en Git es la capacidad de crear ramas (branches) separadas de la línea principal de desarrollo, lo que permite a los desarrolladores trabajar en diferentes funcionalidades, correcciones de errores o experimentos sin afectar el código estable.

Cada rama es como una copia independiente del proyecto, donde se pueden realizar cambios y experimentar sin modificar la rama principal. Una vez que los cambios están listos y probados, las ramas pueden fusionarse en la rama principal.

1. **¿Qué es la integración continua (CI) y cómo la implementarías en un proyecto de ingeniería de datos?**

La Integración Continua (CI) es una práctica de desarrollo que implica integrar regularmente (varias veces al día o semana) los cambios de código en un repositorio compartido. Cada integración se verifica automáticamente mediante un conjunto de pruebas, asegurando que el nuevo código no dañe el funcionamiento existente. El objetivo es detectar errores rápidamente, mejorar la calidad del software y reducir el tiempo de integración de nuevas funcionalidades.

Para implementar Integración Continua (CI) en un proyecto de ingeniería de datos, seguiría estos pasos:

1. **Configurar un repositorio Git**: Asegurarse de que todos los scripts de ETL, pipelines de datos, y configuraciones estén bajo control de versiones en un repositorio como GitHub o GitLab.
2. **Escribir pruebas automatizadas**: Implementar pruebas unitarias y de integración que verifiquen el correcto funcionamiento de los scripts de procesamiento de datos. Por ejemplo:

* Pruebas que validen transformaciones de datos.
* Pruebas que comprueben la calidad de los datos procesados.

1. **Elegir una herramienta CI**: Utilizar una plataforma de CI como **GitHub Actions**, **Jenkins**, **Travis CI**, o **GitLab CI**. Estas herramientas ejecutan automáticamente las pruebas cada vez que se realiza un commit o pull request.
2. **Automatizar el pipeline de CI**:

* Cada vez que se suban cambios al repositorio, el sistema CI debería:
  + 1. Descargar el código.
    2. Instalar las dependencias (como bibliotecas Python).
    3. Ejecutar las pruebas automáticas (unitarias, integración y validación de datos).

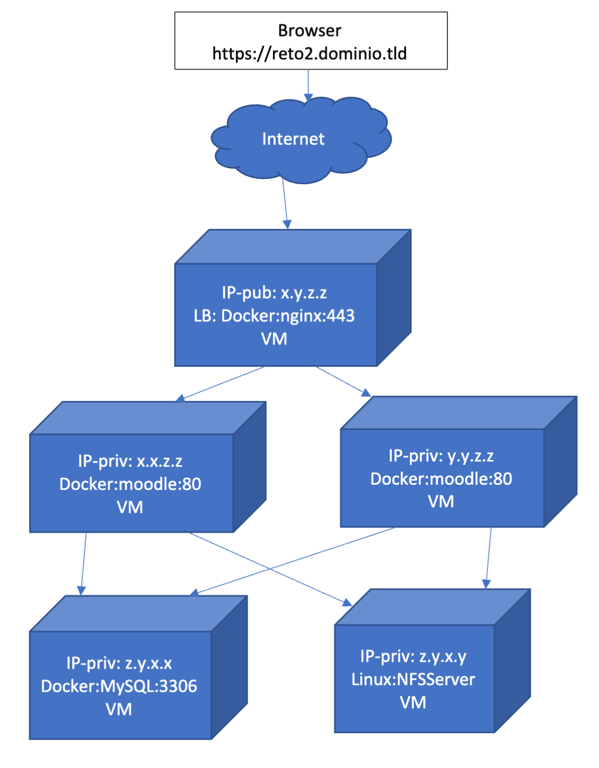
1. **Validar datos**: Además de las pruebas de código, se deben ejecutar scripts que validen la consistencia, calidad y formatos de los datos procesados.
2. **Notificaciones y feedback rápido**: Configura notificaciones para alertar al equipo si fallan las pruebas o si hay errores en los datos.
3. **¿Has trabajado con herramientas de orquestación de contenedores como Docker Compose o Kubernetes? ¿Puedes describir un caso de uso en el que los hayas utilizado?**

He trabajado muy poco con herramientas de orquestación de contenedores como Docker Compose o Kubernetes. Sin embargo, actualmente estoy realizando un proyecto donde necesito crear una página web de Moodle en AWS utilizando Docker.

El proyecto incluye:

* Dos instancias que ejecutan la página de Moodle, cada una dockerizada.
* Estas dos instancias están conectadas a otra instancia que actúa como servidor NFS, para compartir archivos.
* Además, hay otra instancia que contiene la base de datos (MySQL) en Docker.
* La conexión a las instancias principales de Moodle se realiza a través de un Load Balancer (LB) que distribuye el tráfico entre ellas.

Este proyecto me está ayudando a aprender más sobre la orquestación de contenedores en un entorno distribuido.



1. **¿Qué es un data lake y cómo se diferencia de un data warehouse? ¿Has trabajado con alguna tecnología específica para implementar data lakes?**

Un data lake es un repositorio de almacenamiento masivo que puede contener datos en su forma bruta (sin estructurar, semi-estructurados o estructurados) provenientes de diversas fuentes. Los datos se almacenan en su formato original y pueden ser procesados y analizados más adelante según sea necesario.

En cambio un data warehouse es un repositorio más estructurado, diseñado para almacenar datos que ya han sido procesados, limpiados y organizados. Los datos en un data warehouse suelen estar estructurados en tablas y están optimizados para realizar consultas y análisis específicos, generalmente con datos históricos que ayudan a la toma de decisiones.

No he trabajado con tecnologías para implementar data lakes.