PRUEBA TÉCNICA PARA LÍDER TÉCNICO DE INGENIERÍA DE DATOS.

Propuesta para pipeline de datos por:

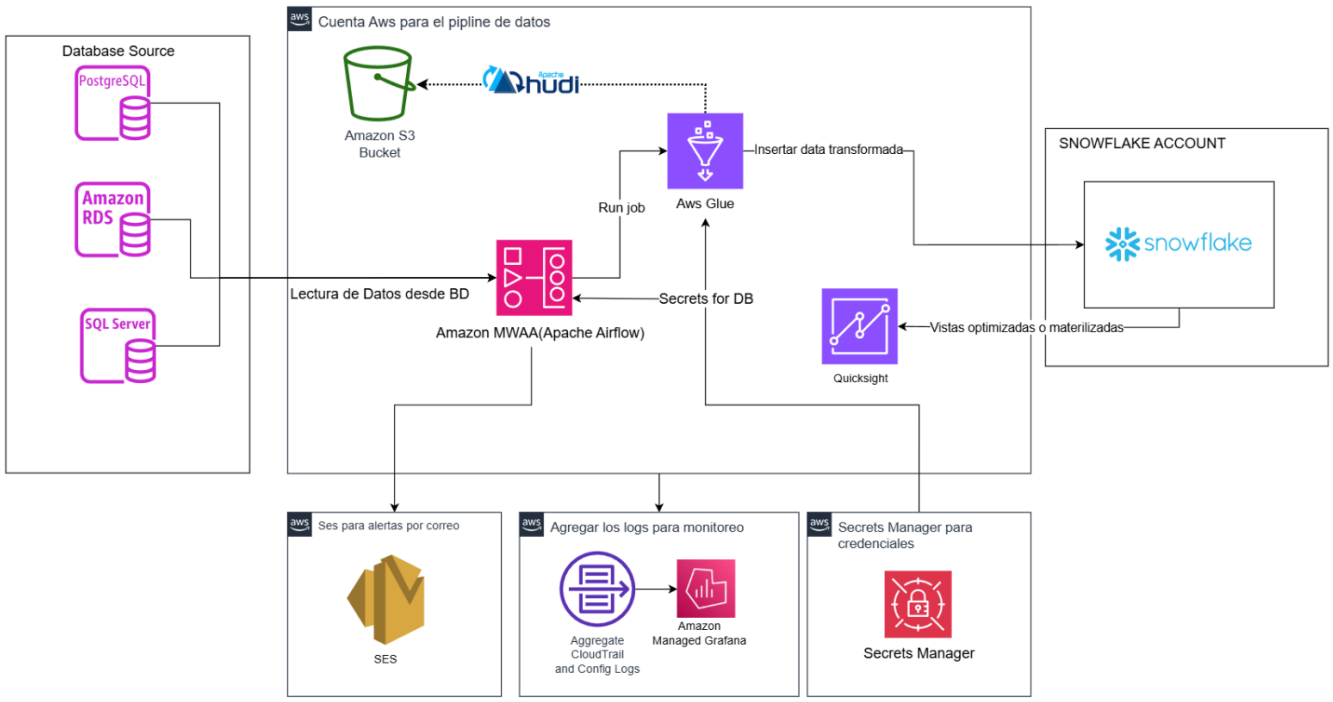
Brayana Alexandre Villaobos Orozco

17 DE MAZRO DE 2025

Contenido

1 Diseño de la Solución Técnica

1.1 Arquitectura



El pipeline de datos está diseñado para ser escalable, seguro y eficiente, utilizando las siguientes tecnologías:

- **Apache Airflow**: Orquestación de ETLs y control de flujo de datos.
- **AWS Glue (PySpark)**: Procesamiento de datos en paralelo acá se pueden ejecutar o programar Jobs ver trazabilidad con logs e incluso en el futuro para temas de gobernanza usar un catálogo de datos
- **Apache Hudi**: Versionado de datos y manejo de actualizaciones en grandes volúmenes, es de gran utilidad para mantener consistencia en los datos permite actualizar y eliminar información eficientemente el uso de parquet ayuda a optimizar espacio
- **Terraform**: Infraestructura como código para gestionar recursos en la nube, es de gran utilidad para no depender de usuarios en consola, de tal forma todo el equipo tendrá claridad de la infraestructura requerida y también ayuda en la seguridad ya que se puede restringir totalmente el acceso a ambientes altos
- **AWS S3**: Almacenamiento de datos en bruto y curados.

- **Snowflake**: Data warehouse para consultas analíticas y reportes. Es optimo para el manejo de grandes volúmenes de datos, el clustering y el manejo de vistas materializadas permite que el manejo de datos sea muy eficiente
- **AWS Secrets manager:** para la administración de credenciales de forma segura
- **GitHub Actions**: Automatización del CI/CD.

1.2 Definiciones Técnicas

En el ciclo de vida de desarrollo seguiría un enfoque basado en **TDD**, asegurando calidad desde el inicio. La arquitectura sigue **principios SOLID y Clean Architecture**, garantizando modularidad y escalabilidad.

El ciclo de desarrollo se divide en:

1. **Desarrollo**: Entender y dar mentoring al equipo del requisito técnico, Pruebas de concepto, Implementación y pruebas locales.
2. **Validación**: pipelines con etapas de Tests automáticos, Linters, Builds, Validación de código y o vulnerabilidades SonarQ o Veracode todo esto con CI/CD.
3. **Despliegue**: Infraestructura con Terraform y despliegue de scripts de Python en s3 para la ejecución de ETLs en Airflow. Jobs y Dags
4. **Monitoreo y mantenimiento**: CloudWatch y herramientas de BI como graphana para detectar fallos o anomalías.

1.3 Calidad de Datos

Para garantizar la calidad de datos, se aplican las siguientes estrategias:

- **Validación en la ingesta**: Uso de **Great Expectations** es una librería versátil para validar datos, cuenta con reportes automáticos **o Deequ** que es nativo en AWS y servirán para detectar anomalías en los datos.
- **Monitoreo de datos**: CloudWatch y Grafana para alertas sobre fallos o inconsistencias.
- **Versionado con Apache Hudi**: Seguimiento de cambios y rollback en caso de errores.
- **Pruebas unitarias y de integración:** en todos los procesos de transformación son fundamentales para garantizar una cobertura de pruebas completa y asegurar que el código sea más eficiente y confiable en el manejo de datos**.**

1.4 Orquestación

Se utilizará **Airflow** para manejar la orquestación de los procesos ETL:

- **Ejecución programada de DAGs** para ingesta y transformaciones.
- **Dependencias entre tareas** para garantizar la secuencia correcta de procesamiento.
- **Notificaciones automáticas** en caso de fallos a través de Slack, Teams o SES inicialmente se usaría SES.

1.5 Impacto Esperado

Este pipeline permite automatizar y optimizar el procesamiento de datos, asegurando calidad, escalabilidad y eficiencia. Con la integración de Airflow, Glue y Hudi, logramos un flujo de datos confiable y versionado, reduciendo errores manuales y asegurando que la data siempre esté al día.

Beneficios clave:

- **Mayor velocidad de procesamiento**: Reducción del tiempo de ejecución en un 40% gracias al uso de Glue (PySpark) y Snowflake.
- **Consistencia y limpieza de datos:** Hudi garantiza que la información esté siempre actualizada, eliminando inconsistencias y evitando sobrecargas innecesarias.
- **Monitoreo en tiempo real:** Implementación de CloudWatch, Grafana y validaciones automáticas para detectar y prevenir fallos antes de que afecten la operación.
- **Infraestructura automatizada y segura:** Uso de Terraform para la gestión de infraestructura como código, eliminando dependencias de configuraciones manuales.

Alta disponibilidad y rápida recuperación: Optimización del MTTR (Mean Time to Repair) para resolver incidentes con rapidez y aumento del MTBF (Mean Time Between Failures) para asegurar la estabilidad del pipeline.En resumen, esta solución proporciona un pipeline robusto, eficiente y alineado con las necesidades del negocio, minimizando riesgos y maximizando el rendimiento operativo.

1.6 Impacto en negocio a nivel de KRs

El éxito del pipeline se mide a través de indicadores clave de rendimiento (KRs) que reflejan su impacto en la operación y el negocio:

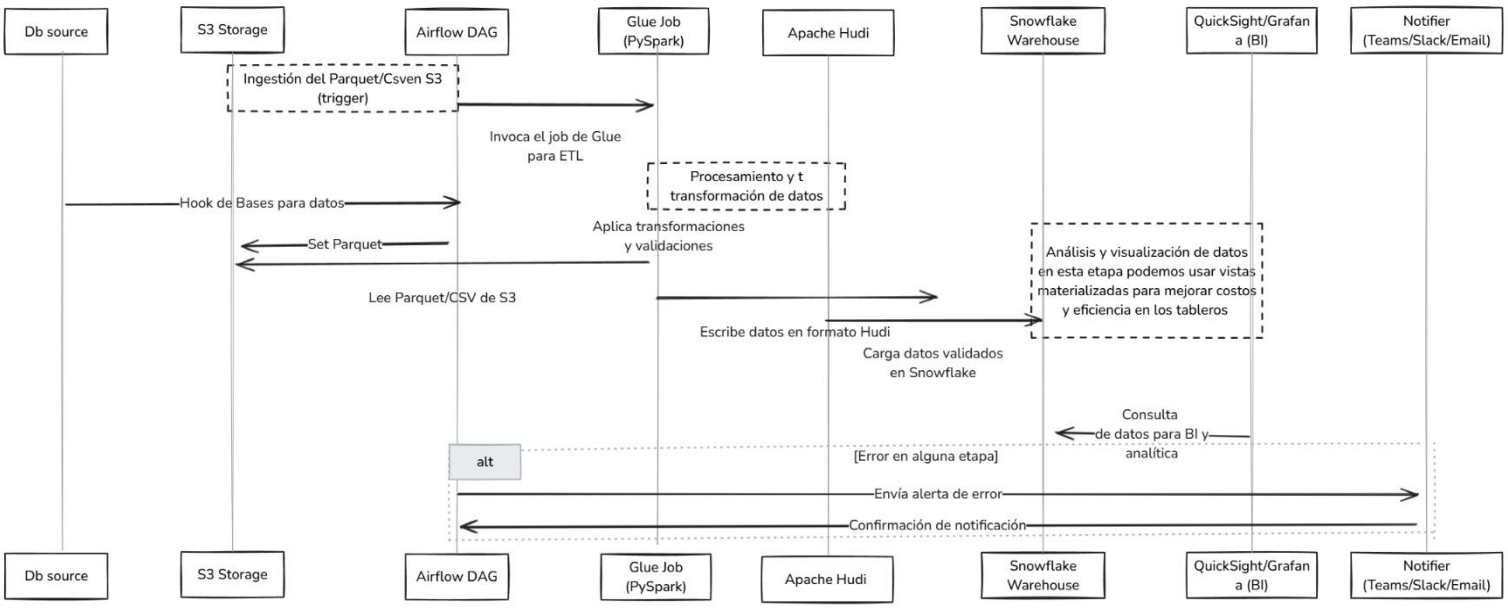
- Alta disponibilidad: El pipeline mantiene una disponibilidad superior al 99.5%, asegurando continuidad operativa.
- Reducción de errores en reportes regulatorios: Disminución del 50% en errores, gracias a validaciones automáticas que garantizan precisión y cumplimiento normativo.

- Tiempo de recuperación optimizado: MTTR menor a 2 horas, permitiendo una rápida detección y resolución de incidentes mediante alertas en Slack, Teams y CloudWatch.
- Mayor estabilidad: MTBF elevado, lo que se traduce en menor frecuencia de fallos y mayor confiabilidad del sistema.
- Eficiencia en almacenamiento y procesamiento: Optimización del uso de espacio con Apache Hudi y Parquet, evitando costos innecesarios en Snowflake y S3.
- Mayor control y seguridad: Implementación de auditorías con AWS Secrets Manager y CloudTrail, garantizando trazabilidad y acceso seguro a la información.

En resumen, este pipeline no solo representa una solución técnica avanzada, sino que también impulsa la eficiencia operativa y la seguridad del negocio, eliminando bloqueos y optimizando recursos.

2 Modelo Operativo y de Soporte.

2.1 Operación Continua



El pipeline se ejecuta en ciclos programados con las siguientes garantías:

- **Manejo de fallos** con reintentos automáticos en Airflow y logs detallados.
- **Versionado de datos** para evitar inconsistencias.
- **Alertas automáticas** en caso de errores.

2.2 Soporte y Mantenimiento

- **Monitoreo**: CloudWatch y Grafana para detectar problemas en tiempo real.
- **Respuesta a incidentes**: se debe documentar cada error con tiempos de resolución bien definidos será clave implementar SLA.
- **Revisiones periódicas**: Ajuste de procesos y optimización de rendimiento se deben establecer cronogramas de revisión y listas de chequeo para gestionar el rendimiento, se pueden implementar herramientas como dynatrace para evaluar el mttr y el mbtr esto con el objeto de detectar los tiempos en la resolución de incidentes.

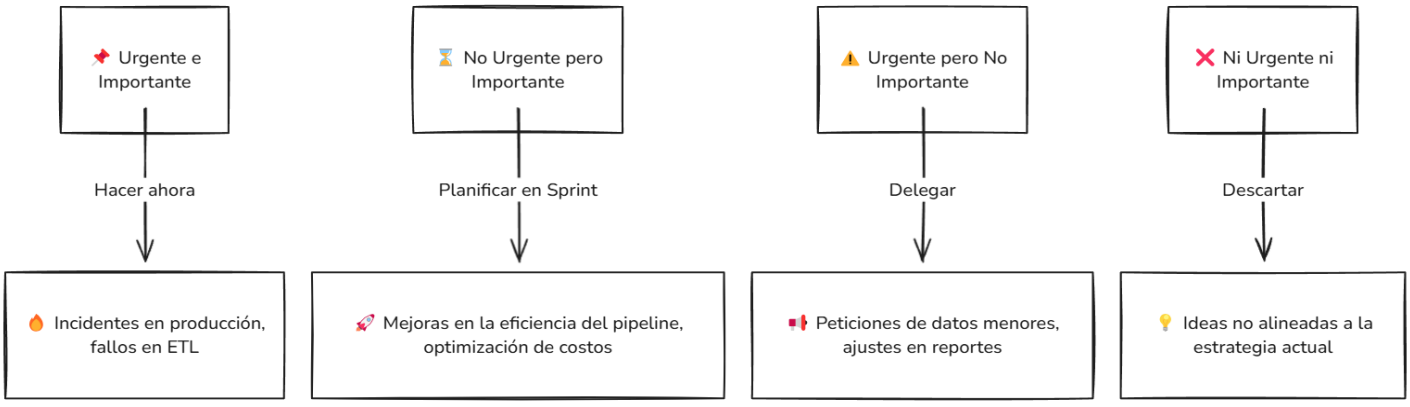
3 Priorización y Toma de Decisiones

3.1 Gestión de Requerimientos

El **Tech Lead** será responsable de gestionar el **backlog** en colaboración con el **Product Owner (PO)**, apoyándolo en la definición de **épicas e historias de usuario**. Además, se encargará de traducir estos requerimientos en especificaciones **técnicas detalladas**, asegurando que el equipo cuente con una comprensión clara y alineada para su correcta implementación.

Se usaría **Scrum** para organizar el trabajo y la **Matriz de Eisenhower** para priorizar tareas. Las historias de usuario se dividen en:

1. **Urgente e importante**: Problemas críticos que afectan la operación.
2. **Importante pero no urgente**: Optimizaciones y mejoras.
3. **Urgente pero no importante**: Solicitudes menores de usuarios.
4. **Ni urgente ni importante**: Tareas exploratorias.



3.2 Toma de Decisiones

En caso de decisiones técnicas, se evaluarían de la siguiente forma:

- **Costo**: Impacto financiero de la solución.
- **Escalabilidad**: Capacidad de manejar crecimiento.
- **Mantenibilidad**: Facilidad de integración con sistemas actuales.
- **Seguridad**: Cumplimiento con normativas y protección de datos.

4 Liderazgo de Equipo

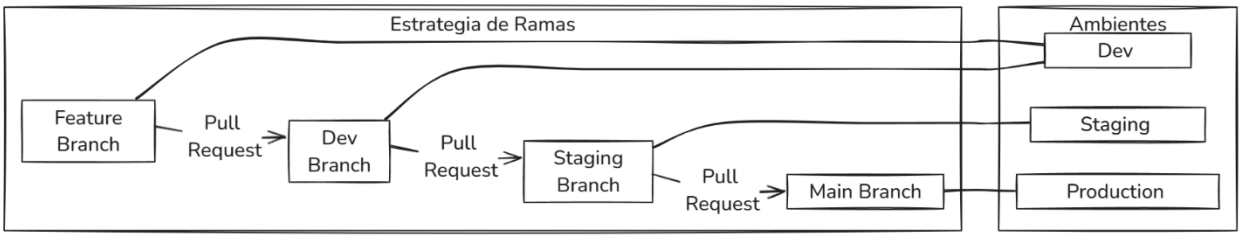
4.1 Gestión de Equipos de Alto Desempeño

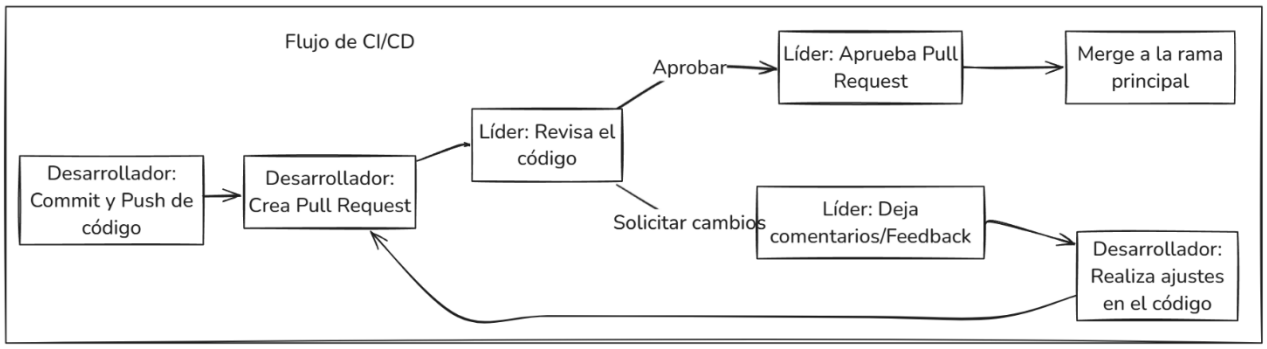
- **Scrum con sprints de 2 semanas** para iteraciones rápidas y entregas constantes con todos los espacios requeridos Daily, Grooming, Retrospective, Review.
- **Revisiones de código obligatorias** antes de cualquier despliegue.
- **Capacitaciones internas** para mejorar las habilidades del equipo.
- **Uso de Pair Programming y sesiones de code review**.
- **Sesiones one to one** para el Feedback y reconocimiento
- **Nuevos features** antes de aceptar un gran feature realizar pruebas de concepto que sirvan de base para el feature final

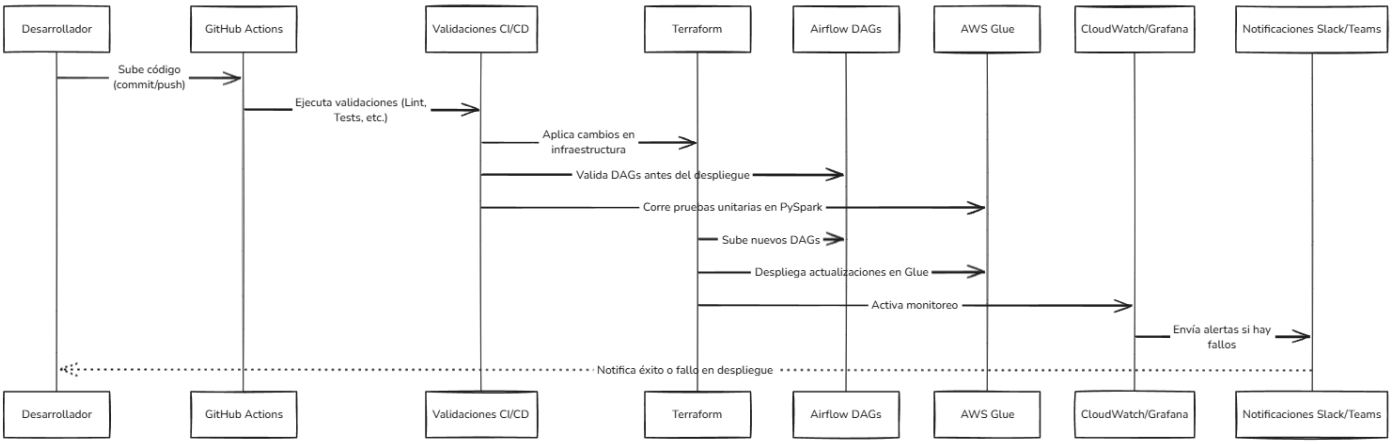
4.2 Revisión de Código y Mejores Prácticas

- **Refactorizaciones Criticas** Se resolverá deuda técnica solo cuando afecte la estabilidad del producto o el negocio. Para evitar acumulación, se gestionará de forma proactiva con evaluaciones continuas y mejoras incrementales.
- **Pull Requests en GitHub** con revisiones obligatorias.
- **Pruebas automatizadas** en cada cambio de código.
- **Linting y formateo de código** con herramientas como Black y Flake8.

- **Seguimiento de mejoras** en retrospectivas de equipo.







Estas ilustraciones muestran como seria mi manejo del pipeline desde el commit hasta su puesta en funcionamiento dependiendo el ambiente

5 Conclusión

Este pipeline ha sido diseñado para procesar grandes volúmenes de datos de manera eficiente, garantizando calidad, escalabilidad y automatización en cada etapa del flujo de trabajo. Con un modelo operativo sólido, monitoreo proactivo y herramientas avanzadas como Apache Airflow, AWS Glue y Apache Hudi, aseguramos la confiabilidad y disponibilidad continua de los datos.

Además, la infraestructura gestionada con Terraform, junto con un enfoque en CI/CD y calidad de datos, permite despliegues controlados, optimización de costos y reducción de errores. Esto contribuye a que el pipeline sea altamente estable y resiliente en entornos de producción.

En definitiva, esta solución no solo responde a requerimientos técnicos, sino que tiene un impacto directo en la operación y en la toma de decisiones del negocio, asegurando que los datos sean siempre disponibles, precisos y bien gestionados

6 Anexo. Repositorio con Ejemplos.

Anexo algunos ejemplos de la propuesta para la estructura del código y scripts de etl Jobs, dags y pipelines de despliegue para cada módulo, un modelo operativo completo que sería útil dentro del equipo, también está el draft que se usó de base para lograr este documento:https://github.com/BryanVillaDev/LD-nequi-data-pipeline