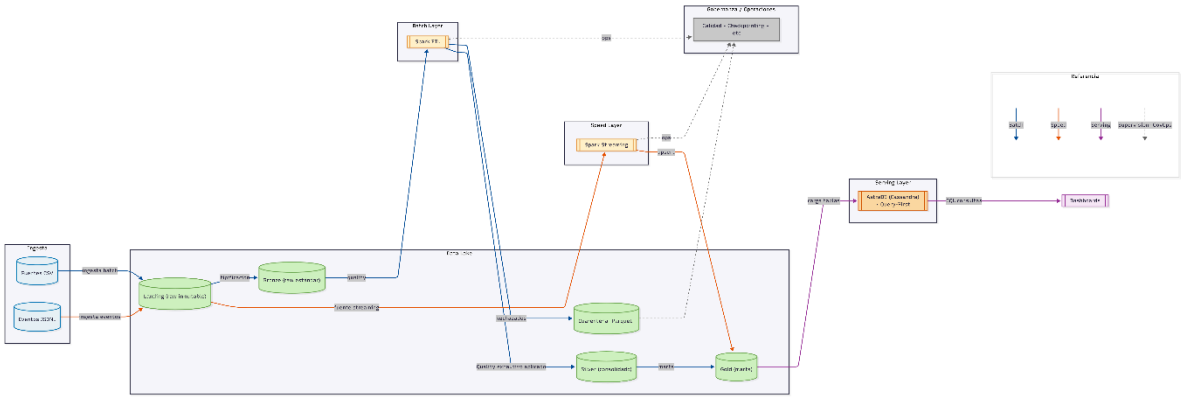
Proyecto Cloud Provider Analytics

- Materia: Minería de datos II
- Alumno: Emilio Gomez Lencina

DOCUMENTACIÓN

DIAGRAMA DEL PROYECTO:



El diagrama fue realizado en Mermaid y dado su tamaño, esta subido acá

ARQUITECTURA:

Se eligió una de tipo Lambda por los siguientes motivos:

• Necesidad de Batch: Se requiere procesar datos maestros, gestión de clientes (CRM) y facturación, los cuales se ingieren de manera periódica (diaria o mensual)

Ademas, el proyecto asume que los datos ingresan con ciertas inconsistencias (valores nulos, tipos ambiguos o datos anómalos), asi que en esta capa es donde se realiza el ETL. Este enfoque asegura que la fuente de verdad histórica (las zonas Silver/Gold) sea confiable

• Necesidad de Streaming: Se requiere el cálculo de métricas operativas, uso de plataforma y costos incrementales en tiempo real, lo que se realiza en la capa de velocidad

• Utilizacion de una Serving Layer: El diseño propuesto consolida la información limpia y procesada para su explotación, lo cual es manejado por la Serving Layer de Lambda, donde convergen los resultados del Batch Layer y los resultados rápidos y del Speed Layer

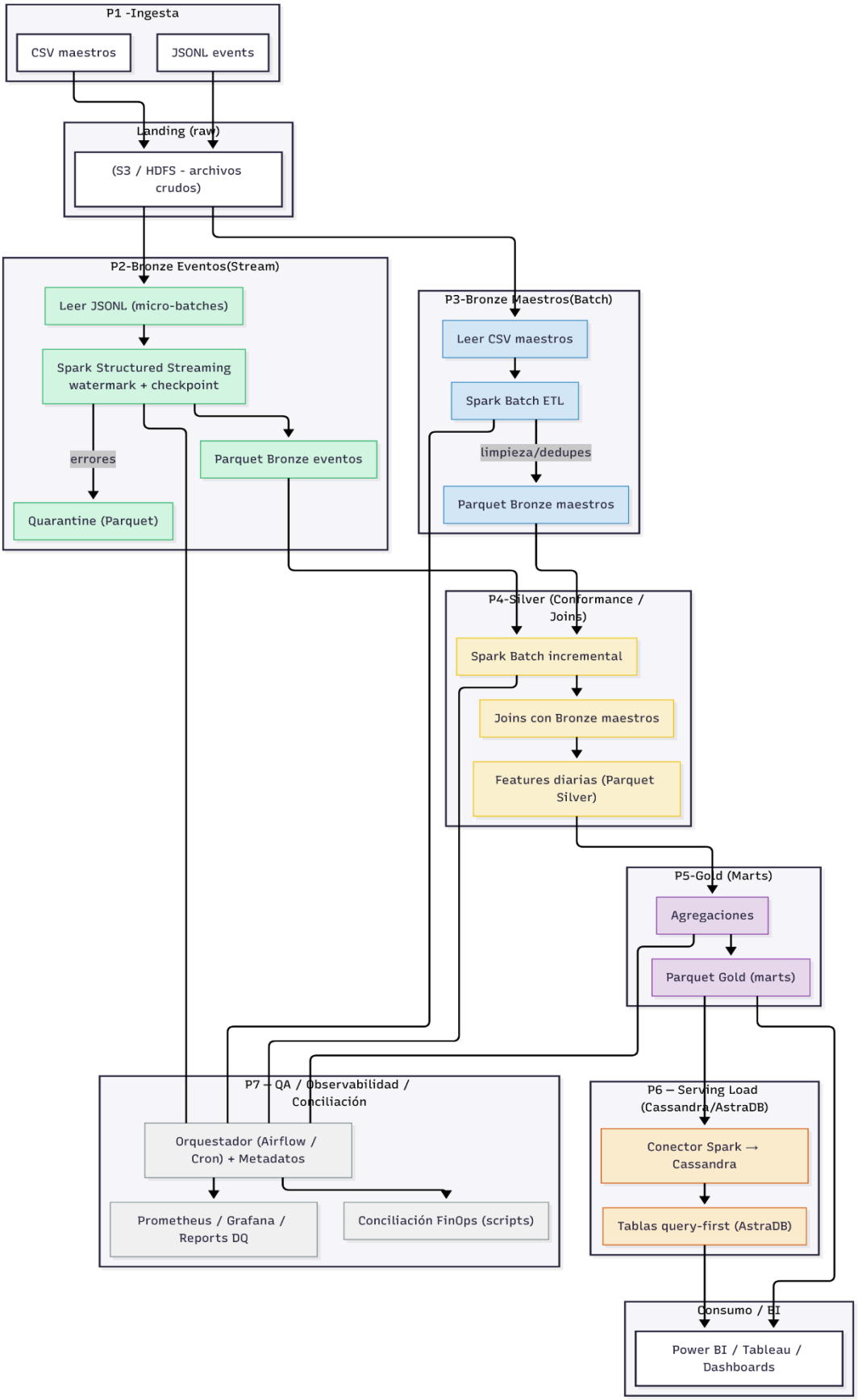
MAPEO DE REQUISITOS A COMPONENTES

Requisito	V de Big Data	Componente / Tecnología	Justificación e Implementación
Procesamiento de métricas operativas en tiempo casi real (Near Real-time)	Velocidad	Speed Layer (Spark Structured Streaming) + JSON	La capa Speed procesa el flujo continuo usage_events_stream/*.JSON mediante micro-batches de Structured Streaming, logrando baja latencia sin perder consistencia.
Procesamiento de datos maestros y de facturación en modo Batch	Volumen y Valor	Batch Layer (Spark ETL) + CSV	Permite procesar grandes volúmenes de datos históricos o maestros (CRM, billing, encuestas) de forma periódica. Spark garantiza paralelismo y escalabilidad; los datos transformados generan valor analítico a partir de agregaciones y uniones con entidades de negocio.
Almacenamiento escalable de datos crudos e inmutables	Variedad	Data Lake - almacenamiento distribuido (HDFS/S3)	La Landing conserva los archivos originales (CSV, JSON) en su formato nativo, asegurando la trazabilidad y reproducibilidad del pipeline. Soporta múltiples estructuras y fuentes, reflejando la variedad de los datos empresariales.
Tipificación y normalización de datos con formato binario optimizado	Volumen y Variedad	Bronze Zone + Parquet particionado	Se aplican reglas de tipificación, deduplicación y trazabilidad. Parquet, formato columnar comprimido, mejora rendimiento y eficiencia de lectura en grandes volúmenes y permite un acceso más rápido desde Spark.
Calidad, conformidad y veracidad de los datos	Veracidad	Silver Zone + gobernanza / operaciones	En la Silver se ejecutan controles de calidad: normalización, detección de nulos, outliers, control de versiones de esquema (schema_version), y envío de fallas a quarantine. Garantiza la veracidad y consistencia del dato antes de la exposición analítica.
Diseño para el consumo analítico rápido y confiable (marts)	Valor y Velocidad	Gold Zone (marts de negocio)	Los datos limpios y enriquecidos se organizan en marts (FinOps, Soporte, Producto/GenAI), optimizados para consultas y visualización. Se crean indicadores clave que generan valor de negocio inmediato.

Consultas rápidas, escalables y tolerantes a fallos	Disponibilidad y Tolerancia a la Partición	Serving Layer (AstraDB / Cassandra)	Cassandra tiene una alta disponibilidad y replicación distribuida, ideal para un modelo query-first. Como tenemos volumenes de datos muy grandes, es ideal por la latencia baja y porquecumple con el principio AP del teorema CAP.
Adaptación a cambios de esquema y reprocesamientos idempotentes	Variabilidad	Lambda Architecture + Checkpointing / Upserts idempotentes	Los cambios de esquema (por ejemplo, aparición de carbon_kg o genai_tokens) se absorben mediante versiones (schema_version) y reprocesos controlados. Checkpointing y upserts garantizan idempotencia y evitan duplicados o inconsistencias al re-ejecutar.

PIPELINES

Diagrama de flujo



En caso de no poder visualizar bien el diagrama, el mismo se puede encontrar acá

Desarrollo de Pipelines

Pipeline 1: Ingesta y Estandarización (Landing y Bronze)

Esta fase se centra en la entrada de datos al Data Lake y su preparación inicial.

- Input: Los datos ingresan en dos formatos principales: CSV (para el procesamiento Batch) y JSON eventos (para el procesamiento Streaming). Estos archivos se almacenan tal cual en la Landing Zone,), que es el repositorio de archivos originales y no modificables

- Output: Los datos pasan por una estandarización mínima, que incluye la tipificación explícita de datos y la deduplicación básica. Los datos se guardan en formato Parquet particionado en la zona Bronze

Pipeline 2: Procesamiento de Eventos (Speed Layer)

- Input: El Spark Streaming lee el flujo de eventos (JSON events) directamente

- Procesamiento: Se utiliza Spark Structured Streaming, que opera bajo un modelo de micro-batches en entornos de producción, y se encarga del procesamiento de datos en tiempo real o casi tiempo real.
- Mecanismos de Control: Se implementan mecanismos como Watermark (para manejar datos tardíos) y Checkpointing que garantiza la idempotencia (reprocesar sin duplicar) y la tolerancia a fallos.

- Errores: Los registros que son rechazados por las validaciones de streaming se desvían a una zona de Cuarentena (Parquet).

- Output: El resultado final del streaming se actualiza o inserta (upsert) directamente en la zona Gold (Marts), asegurando que las métricas más recientes estén disponibles para el consumo.

Pipeline 3 Procesamiento Batch (Batch Layer)

- Input: El proceso comienza leyendo los archivos CSV maestros desde la zona Bronze (Parquet Bronze maestros).

- ETL: Se realiza el trabajo sobre la data, que incluye conformación, joins con datos incrementales, y cálculo de Features diarias (como flags de anomalía o daily_cost_usd).

- Errores: Los registros que no cumplen con el control de calidad riguroso se dirigen a la zona de Cuarentena (Parquet)

- Output a Silver: Los datos limpios y conformados se almacenan en la zona Silver (consolidado)

.

Pipeline 4: Consolidación (Gold/Marts)

- Input: La zona Gold (Marts) es alimentada por la capa Batch (a través de la zona Silver) y por la capa Streaming (a través de su output)
- Esta capa contiene datos curados y optimizados para el análisis, organizados en Data Marts por áreas de negocio (FinOps, Soporte, Producto/Usage)

- Output: Se realizan agregaciones finales para preparar los datos para la consulta.

Pipelinje 5: Serving load- layer

- Input: La carga (Serving Load) se realiza desde la zona Gold hacia la Serving Layer y se se utiliza el Conector Spark para cargar la data en AstraDB (Cassandra).

- Output: Los datos se modelan y almacenan en tablas query-first, diseñadas específicamente para las consultas que realizará el negocio, aprovechando la alta disponibilidad (AP) y tolerancia a la partición de Cassandra

Pipeline 6: Consumo

- Input: Las aplicaciones de negocio, representadas por los Dashboards (Power BI / Tableau), consultan la Serving Layer (AstraDB)

- Las consultas se realizan mediante CQL (Cassandra Query Language), que es el lenguaje de consulta para Cassandra.

Pipeline 7: Gobernanza – Operaciones

Este componente sino que monitorea y soporta la ejecución de los pipelines (P2 y P3)

Orquestador: Controla el proceso y la ejecución programada de los procesos Batch
Se utilizará Cron o Airflow.

- Funciones:
    - Proporciona visibilidad del estado de los procesos.
    - Gestiona el Checkpointing y los datos de Cuarentena (Parquet) para el reporte de Data Quality (DQ)
    - Garantiza la consistencia y la convergencia de los resultados entre la capa Batch (precisa) y la capa Streaming (rápida)

IDENTIFICACION DE RIESGOS:

Asunción Inicial	Riesgo Identificado	Mitigación Propuesta
El volumen inicial de datos de uso y facturación es manejable por un clúster de Spark moderado (por ejemplo, inferior a 1TB/día).	Rendimiento por volumen: Tiempos de procesamiento excesivos en la Batch Layer debido a la escalabilidad exponencial de los datos masivos.	1. Utilizar Parquet particionado en las zonas Bronze y Silver para optimizar el almacenamiento y la lectura secuencial de grandes volúmenes de datos. 2. Emplear optimizaciones en Spark, como CACHE o PERSIST, para reutilizar DataFrames clave en memoria y evitar recalculos
El requisito de Near real-time (NRT) es suficiente para las métricas operativas (uso/costos incrementales).	Latencia: El uso de Micro-batches (el estándar de Spark Structured Streaming) introduce una latencia mínima que podría no cumplir con un requerimiento estricto de tiempo real puro (<1ms).	1. Limitar el tamaño del micro-batch (ej., 1 a 5 segundos) para mantener la latencia baja. 2. Usar withWatermark para la gestión de datos tardíos (late data) en el stream de eventos.
La lógica de negocio será duplicada en el Batch Layer y Speed Layer.	inconsistencia: La duplicación de la lógica en la Arquitectura Lambda podría llevar a resultados diferentes entre la capa Batch y la capa Speed	1.Garantizar que las reglas de calidad y transformación se definan de manera explícita y centralizada para ambos pipelines. 2.Utilizar la Batch Layer para la validación
Los esquemas de los archivos CSV y JSON se manejarán con estructuras definidas.	fallas en ETL: Datos nulos, tipos ambiguos, o la evolución del esquema (carbon_kg, genai_tokens) pueden provocar el fallo total del pipeline.	Implementar la zona de Cuarentena (Quarantine) en Parquet para aislar los registros que no cumplan con las reglas de calidad y evitar que contaminen Silver/Gold.
La Serving layer (Cassandra) priorizará la disponibilidad y tolerancia a la partición (AP).	inconsistencias: el uso de niveles bajos de consistencia para maximizar la disponibilidad podría hacer que haya datos ligeramente desactualizados a los dashboards.	Configurar la consistencia ajustable de Cassandra al nivel de QUORUM para lecturas y escrituras, lo que asegura que la mayoría de los nodosestén sincronizados antes de responder
El uso de Spark mitiga el costo de la duplicación de motores de cómputo.	Sobreconsumo: Los costos pueden incrementarse debido a la complejidad de la arquitectura Lambda y la operación 24/7 del Speed Layer.	1.Aplicar políticas de gobernanza para la gestión de recursos. 2.Configurar la prioridad de los procesos por ejemplo, asignando más recursos a los procesos analíticos críticos.

ROUGH ESTIMATE:

Etapa	Duración estimada	Entregables
1. Análisis y diseño de arquitectura	1 - 2 semanas	Diagrama Lambda, mapa de componentes, DQ Matrix, definición de SLAs
2. Implementación del Data Lake (Landing - Bronze)	Provista	Ingesta inicial CSV/JSON, almacenamiento Parquet, metadatos de ingestión
3. Pipeline de Streaming (Speed Layer)	1 semana	Spark Structured Streaming con checkpoint y watermark; Bronze Eventos
4. Pipelines de Batch (Bronze -Silver - Gold)	1 semana	ETL batch incremental, cálculos FinOps, Soporte, Producto
5. Carga y modelado en Cassandra (Serving)	1-1,5 semanas	Tablas query-first en AstraDB, conexión con BI
6. Implementación de Calidad y Observabilidad (DQ & QA)	1 semana	Reglas DQ, alertas, conciliación FinOps, logs de ejecución
7. Dashboards y BI (Power BI / Superset)	1 semana	Dashboards operativos y financieros (uso, costos, carbono)
8. Validación, pruebas e integración final	1 semana	Validaciones E2E, prueba de idempotencia, cierre técnico

ROLES

En este caso, el equipo es unipersonal así que no existen roles definidos. Si no fuera así, se podría definir roles clásicos como DevOps, Data analyst, data engineer, QA, etc

RECURSOS TÉCNICOS NECESARIOS -EN UN PRINCIPIO-

Recurso	Uso
cluster Spark	Procesamiento batch y streaming
5 TB en S3/HDFS	Landing y Zonas Bronze, Silver, Gold
AstraDB (3 nodos / 2 replicas)	Serving Layer
orquestador simple (Airflow o Cron)	Automatización, observabilidad, alertas
monitoreo (Grafana/Prometheus)	
PowerBi/Tableau	Visualización
GitHub	Versionado y despliegue controlado