Informe Proyecto Final - Implementación de Arquitectura Data Fabric para Análisis de Ventas Retail

Alumno: Angel Teodoro Jaramillo Sulca
Curso: Ingeniería de Datos – Módulo Data en AWS

1. Resumen Ejecutivo

DMC Retail enfrentaba limitaciones para obtener información consolidada y oportuna sobre ventas, rendimiento por tienda y comportamiento de productos debido a que los datos transaccionales residían en un ERP PostgreSQL on-premise. El proyecto implementa una arquitectura Data Fabric en AWS que centraliza la ingesta, catalogación, procesamiento y consumo analítico de los datos de ventas.

Se construyó un Data Lake en Amazon S3 con capas diferenciadas para datos crudos y confiables, se automatizó la ingesta desde la base de datos origen, se catalogaron los activos de datos para facilitar su descubrimiento y se procesaron los datos con tecnologías de procesamiento distribuido para generar métricas de negocio. Los resultados se integraron en un almacén analítico para su consumo por herramientas de BI, habilitando reportes de ingreso por tienda y categoría con calidad y trazabilidad.

El valor de negocio incluye visibilidad operativa, reducción del tiempo de obtención de informes, soporte a decisiones comerciales y una base escalable para futuros casos de uso analítico.

2. Objetivos del Proyecto

Objetivo General
Diseñar e implementar una arquitectura Data Fabric en AWS para habilitar el análisis de ventas retail en DMC Retail.

Objetivos Específicos

- Extraer datos desde la base de datos PostgreSQL on-premise hacia un Data Lake en Amazon S3.
- Procesar y transformar los datos utilizando herramientas de procesamiento distribuido para obtener métricas de negocio.
- Catalogar los datos con un catálogo de metadatos para facilitar descubrimiento y gobernanza.
- Cargar los datos procesados en un almacén analítico y exponerlos mediante dashboards en una herramienta de BI.

3. Alcance y Supuestos

Alcance

El proyecto abarca la extracción, procesamiento y carga de las tablas del esquema `erp`: `venta`, `detalle_venta`, `articulo`, `tienda` y `vendedor`. Los datos se almacenan inicialmente en la capa `/raw` del bucket `dmc-aws-project-atjs1`, se transforman y se escriben en la capa `/trusted` en formato columnar optimizado, y finalmente se cargan en la tabla analítica `reporte_ventas_tienda` en el almacén de datos.

Supuestos

- Se dispone de acceso de solo lectura a la base de datos PostgreSQL con credenciales proporcionadas por el cliente.
- La calidad de los datos permite realizar transformaciones básicas (limpieza de nulos, normalización de formatos); casos complejos de calidad se documentan y se gestionan como excepciones.
- La cuenta AWS tiene permisos para crear y operar recursos necesarios (S3, Glue/EMR, Redshift, IAM, Step Functions).

4. Arquitectura y Flujo de Datos

Componentes principales

- **Fuente transaccional:** PostgreSQL on-premise (ERP) que contiene las tablas de ventas y catálogo.
- **Ingesta:** proceso automatizado que extrae datos desde la base origen y deposita archivos en S3.
- **Data Lake (S3):** bucket `dmc-aws-project-atjs1` con prefijos `/raw`, `/trusted`, `/scripts` y `/logs`.
- **Catálogo de datos:** servicio de catalogación que registra esquemas y metadatos para las tablas ingestas.
- **Procesamiento:** motores de procesamiento distribuido para limpieza, enriquecimiento y agregación de datos.
- **Almacén analítico:** solución de data warehouse para consultas analíticas y cargas periódicas.
- **Visualización:** herramienta de BI para dashboards y análisis por usuarios de negocio.
- **Orquestación:** flujo de trabajo que coordina ingesta, catalogación, procesamiento y carga, con notificaciones y manejo de errores.

Flujo de datos (pasos)

1. Extracción: los datos transaccionales se exportan desde PostgreSQL y se depositan en `s3://dmc-aws-project-atjs1/raw/`.
2. Catalogación: un crawler detecta nuevos archivos y actualiza el catálogo de metadatos.

3. Procesamiento: jobs de procesamiento leen los datos crudos, aplican limpieza, normalización y agregaciones, y escriben resultados en `s3://dmc-aws-project-atjs1/trusted/` en formato optimizado.
4. Carga analítica: los datos procesados se cargan en el almacén analítico para consultas y reporting.
5. Consumo: dashboards y reportes se actualizan con los datos cargados; alertas y monitoreo informan sobre el estado del pipeline.

5. Detalle Técnico por Fase

Fase 1 — Configuración S3
Se creó el bucket `dmc-aws-project-atjs1` con prefijos organizados por capas: `/raw` para datos crudos, `/trusted` para datos procesados, `/scripts` para artefactos de automatización y `/logs` para registros de ejecución. Se aplicaron políticas que bloquean el acceso público, cifrado en reposo y control de versiones opcional. Se definieron reglas de ciclo de vida para mover datos históricos a almacenamiento de menor costo.

Fase 2 — Ingesta
Se definió un proceso de ingesta que conecta a la base de datos origen y exporta las tablas objetivo a la capa `/raw`. El proceso incluye manejo de errores, registro de métricas (conteo de filas, tamaño de archivos) y generación de logs. Se estableció un esquema de nombres y particionado por fecha para facilitar la gestión y el rendimiento.

Fase 3 — Catalogación
Se configuró un catálogo de datos que registra las tablas y sus esquemas. Un crawler programado detecta nuevos archivos en `/raw` y actualiza las entradas del catálogo, permitiendo a los equipos descubrir datasets y sus metadatos. Se definieron políticas de retención de metadatos y etiquetas para gobernanza.

Fase 4 — Procesamiento
Los jobs de procesamiento realizan limpieza de datos (normalización de formatos, manejo de nulos, eliminación de duplicados), enriquecimiento (joins con tablas de referencia) y agregaciones de negocio (ingreso por tienda y categoría). Los resultados se escriben en la capa `/trusted` en formato columnar optimizado y particionado por fecha para mejorar consultas analíticas y reducir costos.

Fase 5 — Data Warehouse y Consumo
Se diseñó la tabla analítica `reporte_ventas_tienda` en el almacén de datos para soportar consultas de negocio. El proceso de carga toma los archivos procesados desde `/trusted` y los incorpora al almacén. Se definieron consultas de validación y vistas para facilitar el consumo por herramientas de BI y usuarios finales.

6. Seguridad, Roles e IAM

Roles y permisos necesarios

- **Roles de procesamiento:** permisos para leer/escribir en S3 y acceder al catálogo de metadatos.
- **Role de carga analítica:** permisos para leer desde S3 y ejecutar operaciones de carga en el almacén analítico.
- **Roles de orquestación y notificación:** permisos limitados para invocar jobs y enviar alertas.

Principios de seguridad aplicados

- Aplicación de **least privilege**: políticas restringidas a prefijos específicos del bucket y a recursos concretos.
- Cifrado en reposo y en tránsito para proteger datos sensibles.
- Auditoría y monitoreo mediante registros de acceso y CloudTrail para trazabilidad.
- Rotación y gestión segura de credenciales; uso de roles en lugar de credenciales embebidas cuando sea posible.

Ejemplo de controles operativos

- Restricción de acceso a datos sensibles mediante etiquetas y políticas de acceso basadas en roles.
- Revisión periódica de permisos y auditorías de cumplimiento.
- Alertas ante accesos inusuales o fallos recurrentes en el pipeline.

7. Automatización con Step Functions

Descripción del workflow

Se diseñó un flujo de orquestación que coordina las etapas del pipeline: ejecución de la ingesta, actualización del catálogo, ejecución de jobs de procesamiento, carga al almacén analítico y refresco de dashboards. El flujo incluye manejo de errores, reintentos configurables y notificaciones al equipo de operaciones.

Mecanismos de control

- Dependencias explícitas entre etapas para garantizar consistencia.
- Reintentos y backoff en tareas susceptibles a fallos transitorios.
- Notificaciones y logs centralizados para facilitar la resolución de incidentes.
- Posibilidad de ejecución on-demand o programada según ventanas de negocio.

8. Pruebas y Validación

Validaciones por fase

- **Ingesta:** verificar presencia de archivos en `/raw`, conteo de filas y checksum para detectar corrupción.
- **Catalogación:** comprobar que el catálogo refleja los esquemas esperados y que las particiones se detectan correctamente.
- **Procesamiento:** revisar logs de ejecución, confirmar estado exitoso de jobs y validar esquema y conteo de filas en `/trusted`.
- **Carga analítica:** comparar conteos y sumas agregadas entre los archivos procesados y la tabla en el almacén; ejecutar consultas de control.
- **Consumo:** validar que los dashboards muestran datos coherentes y que los filtros y métricas clave funcionan según requisitos.

Pruebas adicionales

- Pruebas de regresión ante cambios en esquemas.
- Pruebas de rendimiento para cargas grandes y ventanas de procesamiento.
- Pruebas de seguridad y acceso para verificar políticas IAM.

9. Evidencias Requeridas

Para la entrega académica se deben incluir las siguientes evidencias:

- Captura del bucket S3 mostrando las carpetas `/raw` y `/trusted`.
- Captura del catálogo de datos con las tablas inferidas y sus esquemas.
- Captura de la consola de ejecución de jobs con estado exitoso (SUCCEEDED).
- Captura de la consola del almacén analítico con la query de validación y resultados (Top 5 tiendas).
- Captura del dashboard de BI mostrando las visualizaciones solicitadas.
- Paquete con los archivos de código fuente y scripts utilizados (listados en la sección correspondiente).
- Documento final (informe) con portada, índice y anexos.

10. Código Fuente Entregado

Se entrega un conjunto de artefactos de automatización y procesamiento; a continuación se listan los nombres y una breve descripción de cada uno (sin incluir fragmentos de código):

- **ingesta_postgres.py** — Script responsable de la extracción de datos desde la base de datos origen y la escritura de archivos en la capa `/raw` del Data Lake. Incluye manejo de errores y registro de métricas de ingesta.

- **transformacion_spark.py** — Job de procesamiento que realiza limpieza, normalización, joins y agregaciones para producir los datasets analíticos en `/trusted`. Diseñado para ejecución en un motor distribuido.
- **transformacion_glue_detalle_venta.py** — Job específico para normalizar y validar la tabla de detalle de ventas, asegurando consistencia de columnas y tipos.
- **transformacion_glue_articulo.py** — Job para estandarizar la tabla de artículos, unificar categorías y validar precios y códigos.
- **transformacion_glue_ventas.py** — Job para procesar la tabla de ventas, normalizar fechas y totales, y preparar para el join con detalle de ventas.
- **queries_redshift.sql** — Archivo con las sentencias DDL y las consultas de validación utilizadas en el almacén analítico; contiene instrucciones para crear la tabla analítica y las consultas de control.

11. Plan de Entrega y Checklist

Checklist antes de la entrega final

- [ ] Informe final en formato formal con portada, índice y secciones completas.
- [ ] Capturas de S3 `/raw` y `/trusted`.
- [ ] Captura del catálogo de datos con tablas y esquemas.
- [ ] Captura de ejecución de jobs con estado exitoso.
- [ ] Captura del almacén analítico con la query de validación.
- [ ] Captura del dashboard de BI con las visualizaciones requeridas.
- [ ] Código fuente empaquetado y documentado.
- [ ] Revisión de placeholders (ARNs, account-id, credenciales) y documentación de valores a reemplazar.
- [ ] Compresión de todos los artefactos en un ZIP o empaquetado según instrucciones del aula.

Plan de entrega

1. Reunir evidencias y empaquetar código.
2. Generar documento final y anexos.
3. Subir paquete al aula virtual y notificar al docente con resumen de cambios.
4. Mantener repositorio con versiones y control de cambios para auditoría.

---

12. Conclusiones y Recomendaciones

Conclusiones

La implementación propuesta centraliza y automatiza el flujo de datos de ventas, proporcionando una base confiable para análisis y reporting. La separación de capas en el Data Lake y la catalogación de activos facilitan la gobernanza y el descubrimiento de datos. El pipeline diseñado es escalable y permite incorporar nuevos orígenes y casos de uso analítico.

Recomendaciones

- Implementar particionado por fecha y compresión en la capa `/trusted` para optimizar costos y rendimiento de lectura.
- Habilitar monitoreo y alertas (métricas de latencia, errores y volúmenes) para detectar y resolver incidentes rápidamente.
- Evaluar tecnologías de manejo de cambios (por ejemplo, formatos con soporte de upsert) si los datos requieren actualizaciones frecuentes.
- Revisar periódicamente la configuración de permisos y aplicar políticas de least privilege para reducir riesgos de seguridad.
- Optimizar costos revisando tamaños de clústeres de procesamiento y opciones de almacenamiento en el almacén analítico.

---

13. Anexos

Referencia de la tabla analítica

- **Nombre:** `reporte_ventas_tienda`
- **Propósito:** almacenar métricas agregadas de ingreso por tienda y categoría para consumo analítico.

Descripción de la carga analítica

- El proceso de carga toma los archivos procesados desde la capa `/trusted` y los incorpora al almacén analítico mediante un proceso controlado que valida conteos y sumas agregadas antes y después de la carga.

Notas sobre parámetros y placeholders

- En los artefactos de despliegue y en las instrucciones de carga se incluyen placeholders para ARNs, identificadores de cuenta y roles. Estos valores deben ser reemplazados por los valores reales de la cuenta AWS antes de ejecutar en un entorno de producción.
- Las credenciales de acceso a la base de datos origen deben gestionarse de forma segura y no deben incluirse en artefactos de código sin cifrado o gestión de secretos.