

Escola Tècnica Superior d'Enginyeria Electrònica i Informàtica La Salle

Trabajo Final de Máster

**MÁSTER UNIVERSITARIO EN INGENIERÍA DE DATOS MASIVOS
(BIG DATA)**

Despliegue de infraestructura end to end
para la gestión del dato

Alumno

Profesor Ponente

Mentor Externo

Pol Gràcia Espelt

Joan Navarro

Joan Navarro

ACTA DEL EXAMEN DEL TRABAJO FINAL DE MÁSTER

Reunido el Tribunal calificador en la fecha indicada, el alumno

D. Pol Gràcia

expuso su Trabajo Final de Máster, titulado:

Despliegue de infraestructura end to end para la gestión del dato

Acabada la exposición y contestadas por parte del alumno las objeciones formuladas por los Sres. miembros del tribunal, éste valoró dicho Trabajo con la calificación de

Barcelona,

VOCAL DEL TRIBUNAL

VOCAL DEL TRIBUNAL

PRESIDENTE DEL TRIBUNAL

Abstract

Amb la proliferació de dades i la necessitat de processar, analitzar i consumir informació en temps real, és fonamental comptar amb infraestructures robustes i escalables. No obstant això, el desplegament de sistemes integrals que abasten des de la ingestió fins al consum de dades suposa un desafiament tècnic i operatiu, sobretot davant la vasta diversitat de solucions disponibles al mercat i la complexitat tècnica de la interconnexió entre les capes. Aquest treball aborda el desenvolupament d'un sistema que, de forma automàtica, desplega tots els components necessaris per establir una arquitectura end to end de la dada, cobrint les fases d'ingestió, emmagatzematge i processament. Aquesta proposta busca eliminar la complexitat d'integrar múltiples tecnologies i oferir a les empreses una solució optimitzada, escalable i a punt per a la seva implementació, simplificant així la posada en marxa de projectes de Big Data. Els resultats evidencien la viabilitat de crear una infraestructura que s'adapta a diverses necessitats i que promet facilitar l'adopció de tecnologies de processament i emmagatzematge de dades en empreses emergents i consolidades amb costos reduïts d'implementació i desplegament. En conclusió, aquesta investigació proporciona una via eficaç i àgil per desplegar solucions de Big Data, possibilitant a les organitzacions centrar-se en l'extracció de valor a partir de les seves dades més que en els desafiaments tècnics del desplegament.

Abstract

Con la proliferación de datos y la necesidad de procesar, analizar y consumir información en tiempo real, es fundamental contar con infraestructuras robustas y escalables. Sin embargo, el despliegue de sistemas integrales que abarquen desde la ingesta hasta el consumo de datos supone un desafío técnico y operativo, sobre todo ante la vasta diversidad de soluciones disponibles en el mercado y la complejidad técnica de la interconexión entre las capas. Este trabajo aborda el desarrollo de un sistema que, de forma automática, despliega todos los componentes necesarios para establecer una arquitectura end to end del dato, abarcando las fases de ingesta, almacenamiento y procesamiento. Esta propuesta busca eliminar la complejidad de integrar múltiples tecnologías y ofrecer a las empresas una solución optimizada, escalable y lista para su implementación, simplificando así la puesta en marcha de proyectos de Big Data. Los resultados evidencian la viabilidad de crear una infraestructura que se adapta a variadas necesidades y que promete facilitar la adopción de tecnologías de procesamiento y almacenamiento de datos en empresas emergentes y consolidadas con costes reducidos de implementación y despliegue. En conclusión, esta investigación proporciona una vía eficaz y ágil para desplegar soluciones de Big Data, posibilitando a las organizaciones centrarse en la extracción de valor a partir de sus datos más que en los desafíos técnicos del despliegue.

Abstract

With the proliferation of data and the need to process, analyze, and consume information in real-time, it is essential to have robust and scalable infrastructures. However, the deployment of comprehensive systems that span from data ingestion to consumption poses a technical and operational challenge, especially given the vast diversity of solutions available in the market and the technical complexity of interconnecting layers. This work addresses the development of a system that automatically deploys all the necessary components to establish an end-to-end data architecture, covering the phases of ingestion, storage, and processing. This proposal aims to eliminate the complexity of integrating multiple technologies and provide businesses with an optimized, scalable, and ready-to-implement solution, thereby simplifying the implementation of Big Data projects. The results demonstrate the viability of creating an infrastructure that adapts to various needs and promises to facilitate the adoption of data processing and storage technologies in emerging and established companies at reduced implementation and deployment costs. In conclusion, this research provides an effective and agile way to deploy Big Data solutions, enabling organizations to focus on extracting value from their data rather than on the technical challenges of deployment.

Contenido

Introducción	11
1 Capítulo 1: Marco Teórico.....	16
1.1 Desafíos en la Gestión de Grandes Volúmenes de Datos	16
1.1.1 Complejidad en la Elección de Servicios y Plataformas	16
1.1.2 Interconexión y Cohesión entre Componentes	16
1.1.3 Gestión de la Calidad de los Datos	17
1.1.4 Escalabilidad y Flexibilidad	17
1.1.5 Seguridad y Privacidad	17
1.1.6 Integración con Infraestructuras Legadas	18
1.1.7 Formación y Capacidad del Personal	18
1.2 Ecosistema del Big Data	18
1.2.1 Capa de Gestión y Recopilación de Datos	19
1.2.2 Capa de Infraestructura y Almacenamiento	20
1.2.3 Capa de Procesamiento y Transformación	28
1.2.4 Capa de Orquestación y Automatización	29
1.2.5 Capa de Análisis y Consulta	30
1.2.6 Capa de Presentación y Visualización	32
1.2.7 Capa de Seguridad y Cumplimiento	33
2 Capítulo 2: Propuesta de solución	35
2.1 Arquitectura Delta Lake	36
2.1.1 Características de la arquitectura Delta Lake.....	36
2.1.2 Metadatos y sus funcionalidades en Delta Lake	37
2.2 Elementos de la Capa del Dato	38
2.2.1 Introducción a las capas lógicas del dato: Bronze, Silver y Gold.....	38
2.2.2 Azure Blob Storage como almacenamiento principal	39
2.2.3 Azure Data Factory para la ingesta de datos.....	41
2.2.4 Databricks cómo Motor de Procesamiento de Datos	44
2.2.5 Apache Airflow para la Orquestación	47
2.2.6 Comunicación entre los componentes.....	48
2.3 Interconexión y definición del despliegue automático.....	51
2.3.1 Principios de diseño del ejecutable.....	51

2.3.2	Componentes claves para la interconexión	53
2.3.3	Secuencia de despliegue	55
2.4	Seguridad y Gobernanza de datos	58
2.4.1	Unity Catalog	58
2.4.2	Seguridad en la transmisión y almacenamiento de datos	58
2.4.3	Control de Aceceso Basado en Roles	59
2.5	Coste-Efectividad de la solución	60
2.5.1	Costes de implementación	60
2.5.2	Costes operativos recurrentes	60
2.6	Conclusión del capítulo	62
3	Capítulo 3: Desarrollo e implementación	63
3.1	Metodología	63
3.1.1	Entorno de Desarrollo	63
3.2	Desarrollo	67
3.2.1	Autenticación de la aplicación en Azure	67
3.2.2	Despliegue infraestructura de Almacenamiento	68
3.2.3	Despliegue de componentes de Ingesta de datos	70
3.2.4	Despliegue de componentes de Transformación y Procesado	71
3.2.5	Contenerización de Airflow	74
3.2.6	Configuración de interconexiones entre los componentes	76
3.2.7	Unificación del proceso mediante Docker compose.....	78
3.3	Procedimiento de ejecución del despliegue	81
4	Capítulo 4: Casos de estudio y validación	87
4.1	Caso de estudio: Validación del proceso end-to-end	87
4.1.1	Ingestión de datos	89
4.1.2	Pre-processamiento de datos	90
4.1.3	Transformación y enriquecimiento de datos	92
4.1.4	Análisis de datos.....	93
4.1.5	Orquestación con airflow	94
4.1.6	Validación	95
5	Conclusiones.....	97
5.1	Conclusiones del trabajo realizado	97
5.2	Puntos fuertes y débiles del trabajo realizado.....	98
5.2.1	Puntos fuertes	98

5.2.2	Puntos Débiles.....	98
5.3	Limitaciones del trabajo realizado	100
5.4	Lineas de contexto de trabajo.....	101
5.4.1	Ejecución en Varios Proveedores de Nube	101
5.4.2	Elección de Componentes Personalizados.....	101
5.4.3	Despliegue de Entornos Dev y Pro Interconectados.....	101
5.4.4	Integración con Herramientas de CI/CD como Travis	101
5.4.5	Integración con Herramientas de Control de Versiones como GitHub.....	102
6	Referencias.....	103

Acrónimos

ADB: Azure Databricks

ADBC: Azure Databricks Connector

ADF: Azure Data Factory

ADLSG2: Azure Data Lake Gen 2

AWS: Amazon Web Services

ELT: *Extract Load Transform*

ETL: *Extract Transform Load*

ERP: *Enterprise Resource Planning*

GCP: Google Cloud Platform

GDPR: *General Data Protection Regulation*

GRS: *Geo Redundant Storage*

IaaS: *Infrastructure as a Service*

IaC: *Infrastructure as Code*

PaaS: *Platform as a Service*

SaaS: *Software as a Service*

Introducción

En los últimos años, el crecimiento exponencial de los datos ha transformado la manera en que las organizaciones, tanto públicas como privadas, operan y toman decisiones. Las fuentes de datos son cada vez más variadas, desde transacciones financieras y registros de clientes hasta datos recopilados de dispositivos IoT y redes sociales. Este aumento sin precedentes en el volumen, velocidad y variedad de los datos ha dado lugar a una revolución conocida como la era del Big Data.

Las organizaciones han reconocido rápidamente el valor latente en estos datos. Los datos, en efecto, tienen el potencial de ofrecer perspectivas novedosas, impulsar la eficiencia operativa, personalizar las experiencias del cliente y abrir nuevos canales de ingresos. Sin embargo, el mero acto de recopilar grandes cantidades de datos no es suficiente para garantizar tales beneficios. Es fundamental que estos datos se procesen, almacenen y consuman de manera efectiva para extraer *insights* valiosos.

Históricamente, las arquitecturas de datos estaban diseñadas para gestionar volúmenes relativamente pequeños de datos estructurados, a menudo alojados en sistemas monolíticos y centralizados. Estas estructuras tradicionales, aunque eficientes para sus propósitos originales, se encontraron rápidamente sobrepasadas por la magnitud y complejidad del Big Data. Esta limitación ha llevado a una necesidad imperativa de repensar y reinventar las infraestructuras y arquitecturas de datos.

Además, con la descentralización y diversificación de las fuentes de datos, la ingesta de datos se ha convertido en un desafío en sí mismo. Ya no se trata sólo de extraer datos de un sistema centralizado, sino de consolidar datos de una multitud de fuentes heterogéneas, cada una con sus propias características y desafíos.

Por otro lado, la capacidad de almacenar datos no es el único desafío; la relevancia y oportunidad de los *insights* extraídos de estos datos es esencial. En un entorno empresarial dinámico, donde las decisiones a menudo deben tomarse en tiempo real, la necesidad de procesar datos rápidamente, independientemente de su volumen, ha cobrado una importancia crítica.

A medida que la gestión del dato se convierte en el centro de la transformación digital, la demanda de soluciones integradas y holísticas ha ido en aumento. Las empresas no sólo buscan soluciones *piecemeal*, sino infraestructuras *end-to-end* que aborden de manera coherente todos los aspectos de la gestión del dato, desde la ingesta y el almacenamiento hasta el procesamiento y consumo.

En este escenario, surgen preguntas esenciales: ¿Cómo pueden las organizaciones desplegar infraestructuras que sean tanto flexibles como robustas? ¿Cómo se pueden

integrar las diversas tecnologías de manera coherente? Y, lo más importante, ¿cómo se puede garantizar que estas infraestructuras sean escalables y sostenibles a largo plazo? Además de siempre intentar asegurar un bajo coste para este servicio. Estos son precisamente los desafíos que este trabajo pretende abordar.

Propósito

En el vasto panorama del manejo y gestión de datos, se ha identificado una brecha significativa entre la recopilación de datos y la extracción de valor real de ellos. Si bien se han realizado extensos estudios sobre las técnicas individuales de almacenamiento, procesamiento o ingestión, hay una escasez notoria en la literatura respecto a un enfoque integrado que aborde la gestión del dato desde una perspectiva holística. Este trabajo se embarca en la ambiciosa tarea de llenar ese vacío, proporcionando un marco comprensivo y concreto que unifique las distintas etapas de la cadena de valor del dato interconectando tecnologías ya existentes.

El propósito inicial de este estudio es desarrollar dentro de un marco teórico un entendimiento profundo y exhaustivo de los desafíos inherentes a la gestión de grandes volúmenes de datos y los diferentes competidores en el ecosistema del Big Data, en base a ese entendimiento, proponer una solución tecnológica por cada uno de los elementos de la capa del dato que de respuesta a los objetivos establecidos.

El propósito final de este trabajo es el despliegue de distintos elementos en las capas del dato, asegurando su interconexión eficaz de manera automática y optimizada siguiendo los principios del marco teórico. En concreto en forma de script o ejecutable.

La razón detrás de este enfoque se encuentra en la reconocida necesidad de las organizaciones de contar con soluciones prácticas y directamente aplicables que les permitan superar los desafíos inherentes al manejo del dato en la era actual. Un despliegue automático, además de reducir la carga operativa y los potenciales errores humanos, garantiza una implementación coherente y alineada con los estándares y prácticas óptimas predefinidas.

Es imperativo reconocer que, aunque el objetivo de este trabajo es facilitar el despliegue y la gestión de la infraestructura de datos en las organizaciones, la presencia y la competencia del personal técnico seguirán siendo requisitos esenciales. Un sistema, por más optimizado y automatizado que sea, no elimina la necesidad de expertos que puedan interpretar las métricas, realizar ajustes y tomar decisiones basadas en datos complejos y a menudo ambiguos. En otras palabras, la automatización no es un sustituto de la pericia humana, sino más bien un complemento que permite a los profesionales centrarse en tareas de mayor valor agregado.

En este contexto, la automatización sirve para aliviar el trabajo manual que a menudo consume tiempo y está sujeto a error, como la configuración inicial de servidores, el escalado de recursos y la gestión de fallas. Al reducir la carga operativa, las organizaciones no solo logran una eficiencia significativa sino que también liberan a su personal técnico para que se dedique a actividades más estratégicas. Esto podría incluir la optimización de algoritmos, el análisis de patrones de datos para insights empresariales, o la formulación de políticas de gobernanza de datos más eficaces.

Además, la implementación coherente y alineada con estándares y mejores prácticas, que este proyecto busca asegurar, sirve como una base sólida sobre la cual los técnicos pueden trabajar. Les brinda un marco unificado que es tanto robusto como flexible, permitiéndoles interactuar con el sistema de una manera más segura y eficiente. Esto es particularmente relevante en entornos donde múltiples tecnologías y plataformas están en juego, y donde la coherencia y la interoperabilidad son críticas.

Por lo tanto, aunque la automatización y la optimización son componentes clave de este proyecto, es crucial entender que estos elementos están destinados a potenciar, no a reemplazar, las habilidades y el juicio del personal técnico. De este modo, el proyecto se alinea con la visión más amplia de capacitar a las organizaciones para que no solo manejen sino que también extraigan valor significativo de sus complejas infraestructuras de datos.

Alcance

- I. El alcance de este proyecto será crear una solución para la ingesta, el almacenamiento y el procesamiento de los datos en tecnologías. No incluye el proceso de despliegue de la capa de consumo o visualización.
- II. La infraestructura definida y sus componentes no serán elegibles y se elegirán en el momento de la definición de la solución las tecnologías óptimas siguiendo criterios objetivos de rendimiento, compatibilidad y costes que serán estudiados en el marco teórico del proyecto.
- III. La solución implementará el lanzamiento de los componentes, la interconexión entre todos los componentes y una demo de despliegue.
- IV. La solución no incorporará queries ni pipelines de extract, transform and load de datos entre el origen de datos y las distintas capas. Esto siempre recaerá en el usuario de la infraestructura.
- V. La infraestructura desplegada no tendrá entornos de development y producción, en una primera instancia sólo habrá un único componente de cada tipo.
- VI. Aunque no se desprecia su importancia, no se van a desarrollar ni desplegar medidas para la seguridad de acceso a los componentes montados más que los predeterminados o descritos en el proyecto.

- VII. En el marco teórico sí que se van a exponer otros componentes claves que forman parte de una infraestructura end to end del dato relacionadas con el trabajo y el desarrollo (cómo herramientas CI/CD o control de versionado) aunque no se van a desplegar en la parte práctica.

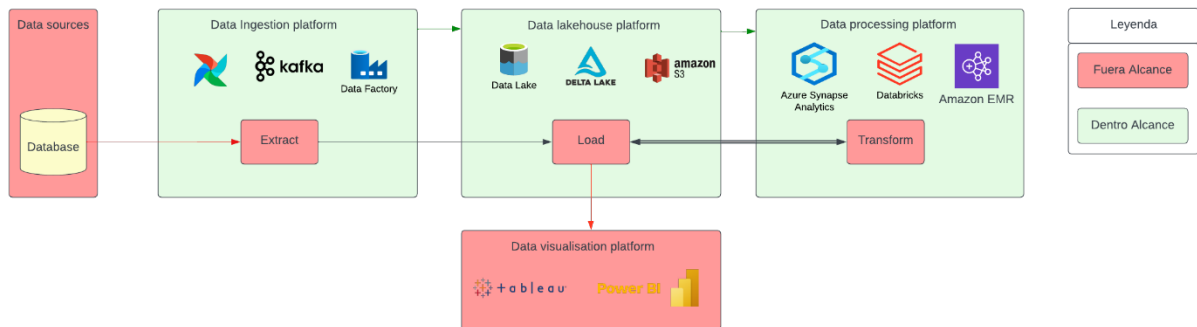


Figura 1. Representación del alcance del proyecto dentro de las capas del dato

Resultados

El proyecto logró desarrollar un sistema automatizado para desplegar una arquitectura de datos de extremo a extremo, cubriendo las fases de ingesta, almacenamiento y procesamiento de datos. Se eliminó la complejidad de tener que integrar múltiples tecnologías manualmente, ofreciendo una solución optimizada y escalable. Aunque se requiere alguna interacción del usuario y hay limitaciones en cuanto a la flexibilidad y la elección de componentes específicos, la infraestructura resultante es robusta y adaptable a diversas necesidades empresariales. Este enfoque promete reducir los costos y acelerar el tiempo de implementación, permitiendo a las organizaciones centrarse más en la extracción de valor de sus datos que en los desafíos técnicos del despliegue.

Además, la solución se integra perfectamente con los objetivos del proyecto, provisionando una arquitectura con gran potencial de futuro, altamente escalable y con costes operacionales optimizados.

Organización del documento

Capítulo 1: Marco Teórico

- **Desafíos en la Gestión de Grandes Volúmenes de Datos:** Se exploran los retos más significativos asociados a la gestión del Big Data. Centrado en la complejidad de la elección de servicios del ecosistema y la interconexión de los componentes en entornos Big Data.

- **Ecosistema del Big Data:** Se presenta una revisión de los principales actores y competidores en este ámbito, así como de las tecnologías y metodologías actuales.

Capítulo 2: Propuesta de Solución

- **Arquitectura Delta Lake:** Se detalla la arquitectura de datos propuesta.
- **Elementos de la Capa del Dato:** Se detallan los componentes clave de la arquitectura de datos propuesta.
- **Interconexión y Despliegue Automático:** Se introduce el diseño y los principios detrás del script o ejecutable propuesto.
- **Seguridad y Gobernanza de datos:** Se definen los elementos de seguridad y gobernanza del diseño.
- **Coste-efectividad de la solución:** Se realiza una definición teórica de los distintos costes de la solución.

Capítulo 3: Desarrollo e Implementación

- **Metodología:** Se describe el proceso seguido para el desarrollo del script o ejecutable.
- **Desarrollo:** descripción técnica de los componentes y detalle del funcionamiento del despliegue.
- **Procedimiento ejecución y despliegue:** se describe el proceso a seguir para desplegar la infraestructura

Capítulo 4: Casos de Estudio

- Se realiza un caso de estudio para mostrar el funcionamiento end-to-end

Capítulo 5: Conclusiones

- Conclusiones del trabajo

Referencias Bibliográficas

- Se enlistan todas las fuentes académicas, libros, artículos y recursos en línea que respaldan y fundamentan el trabajo realizado.

Apéndices y Anexos

- Aquí se incluirán detalles adicionales, códigos fuente, gráficos ampliados, tablas y cualquier otra información relevante que complemente el cuerpo principal del trabajo.

1 Capítulo 1: Marco Teórico

En el paisaje contemporáneo del tratamiento y gestión de la información, el conocimiento sobre cómo navegar y aprovechar eficazmente la vastedad del universo de datos es esencial. El Capítulo 1 se adentra en el análisis del marco teórico que subyace a estos desafíos, se definen las bases en el escenario actual del manejo de grandes volúmenes de datos, identificando los desafíos más significativos y las dinámicas del ecosistema en el que se desenvuelve este fenómeno. La exploración en este capítulo proporcionará una perspectiva sólida y profunda, sentando las bases para las propuestas y soluciones que se presentarán en los capítulos posteriores. Es un primer paso crítico que contextualiza y orienta el estudio, estableciendo las bases sobre las cuales se edificarán las propuestas y desarrollos prácticos.

1.1 Desafíos en la Gestión de Grandes Volúmenes de Datos

El continuo auge en la generación de datos ha convertido la gestión de estos grandes volúmenes en una prioridad estratégica para organizaciones en todo el mundo. Los desafíos inherentes son múltiples y variados, y en este apartado se desentrañarán en profundidad para comprender mejor el paisaje actual del Big Data sobre el que este proyecto pretende trabajar, ligar y optimizar para obtener una solución robusta que tenga en cuenta todas las especificaciones descritas.

1.1.1 Complejidad en la Elección de Servicios y Plataformas

- **Diversidad de Opciones:** El mercado del Big Data es vasto. Desde soluciones de almacenamiento hasta herramientas analíticas, las organizaciones se enfrentan a un mar de opciones que varían en funcionalidad, rendimiento y precio.
- **Consideraciones de Coste:** La elección adecuada no solo implica la eficacia de la herramienta, sino también su relación costo-beneficio. Determinar el retorno de inversión esperado y equilibrarlo con el rendimiento deseado es una tarea desafiante.
- **Compatibilidad y Ecosistema:** Las soluciones no operan en un vacío. La capacidad de una herramienta para integrarse sin problemas con otras aplicaciones y sistemas es vital para evitar silos de datos y garantizar un flujo de datos ininterrumpido.

1.1.2 Interconexión y Cohesión entre Componentes

- **Arquitecturas Distribuidas:** En un entorno de Big Data, es común que los datos estén dispersos en múltiples sistemas o ubicaciones. Asegurar una comunicación eficiente y coherente entre estos sistemas es esencial.

- **Protocolos y Estándares:** Con múltiples herramientas y plataformas en juego, es crucial establecer y adherirse a protocolos y estándares uniformes para asegurar que los datos fluyan correctamente.
- **Latencia y Rendimiento:** La transferencia y procesamiento de grandes volúmenes de datos puede ser susceptible a latencias. Optimizar la infraestructura para minimizar estas latencias garantiza una entrega y análisis de datos en tiempo real o cercano al tiempo real.

1.1.3 Gestión de la Calidad de los Datos

- **Veracidad y Precisión:** En el océano del Big Data, no todos los datos son precisos. Las organizaciones deben establecer mecanismos para verificar la veracidad de los datos recopilados.
- **Procesos de Limpieza:** Las herramientas y técnicas para limpiar y preparar datos para análisis son esenciales. Esto incluye la identificación y corrección de errores, el llenado de datos faltantes y la eliminación de redundancias.
- **Mantenimiento Continuo:** La calidad de los datos no es una tarea única. Requiere un esfuerzo continuo para mantener, actualizar y refinar los conjuntos de datos a lo largo del tiempo.

1.1.4 Escalabilidad y Flexibilidad

- **Infraestructura Adaptable:** Los sistemas deben diseñarse para crecer. A medida que aumenta la generación de datos, la infraestructura debe poder adaptarse sin comprometer el rendimiento.
- **Planificación Futura:** No solo se trata de satisfacer las necesidades actuales, sino también de anticipar las demandas futuras y diseñar soluciones que puedan evolucionar con el tiempo.

1.1.5 Seguridad y Privacidad

- **Resguardo contra Brechas:** Con la creciente cantidad de ciberataques, garantizar la integridad y seguridad de los datos es primordial.
- **Regulaciones y Cumplimiento:** Las leyes y regulaciones relacionadas con la privacidad de datos, como el RGPD en Europa, exigen que las organizaciones sigan directrices estrictas en cuanto a la recopilación, almacenamiento y uso de datos.
- **Concienciación y Capacitación:** La seguridad no se limita solo a las herramientas y tecnologías; el personal debe estar adecuadamente capacitado sobre las mejores prácticas y protocolos de seguridad.

1.1.6 Integración con Infraestructuras Legadas

- **Interoperabilidad:** Las soluciones más antiguas pueden no estar diseñadas para interactuar con tecnologías más recientes, creando desafíos en la integración.
- **Migración de Datos:** Transferir datos de sistemas antiguos a soluciones más modernas puede ser un proceso arduo, que requiere una planificación meticulosa para evitar la pérdida de datos o la degradación de su calidad.

1.1.7 Formación y Capacidad del Personal

- **Actualización Continua:** El ámbito del Big Data está en constante evolución. Las organizaciones deben invertir en la capacitación continua de su personal para mantenerse al día con las últimas tendencias y tecnologías.
- **Cultura de Datos:** Más allá de la formación técnica, es esencial inculcar una cultura donde se entienda y valore la importancia de los datos en la toma de decisiones.

La gestión de grandes volúmenes de datos no es simplemente una cuestión de manejar grandes cantidades de información, sino que implica navegar por una serie de desafíos multifacéticos que abarcan desde decisiones tecnológicas y diseño de sistemas hasta consideraciones de seguridad y formación. Abordar estos desafíos de manera efectiva es esencial para liberar el verdadero potencial del Big Data y convertir estos vastos volúmenes de datos en información valiosa y accionable.

1.2 Ecosistema del Big Data

El ecosistema del Big Data representa un conjunto complejo y multifacético de componentes interconectados que trabajan en sinergia para facilitar el ciclo de vida completo de los datos, desde su generación hasta su análisis y visualización. Para entender cabalmente la intrincada red de elementos que componen este ecosistema, es imperativo desglosar sus distintas capas y examinarlas de forma detallada. El objetivo de esta sección es proporcionar una visión panorámica y comprensiva de estas capas, delineando su importancia, sus desafíos inherentes y las soluciones disponibles en el mercado para cada una de ellas.

Es crucial entender este ecosistema de manera descompuesta para abordar de manera efectiva la complejidad en la gestión de grandes volúmenes de datos. Este entendimiento será el pilar sobre el cual se construirá la propuesta de solución de este proyecto de final de máster, centrado en un despliegue automático y optimizado de elementos en estas capas del dato. Además, se incluirá un análisis de los principales

actores o empresas en cada capa, para contextualizar las opciones disponibles y ofrecer un marco de referencia para futuras implementaciones.

La descripción de los componentes de la arquitectura también abarca descripción y estudio teórico de componentes que quedan fuera del alcance la implementación práctica del proyecto, ya que aunque se hayan obviado para mantener la escalabilidad del trabajo, es importante conocerlas y tenerlas en cuenta para futuros desarrollos.

The 2023 ML, AI, and Data Landscape

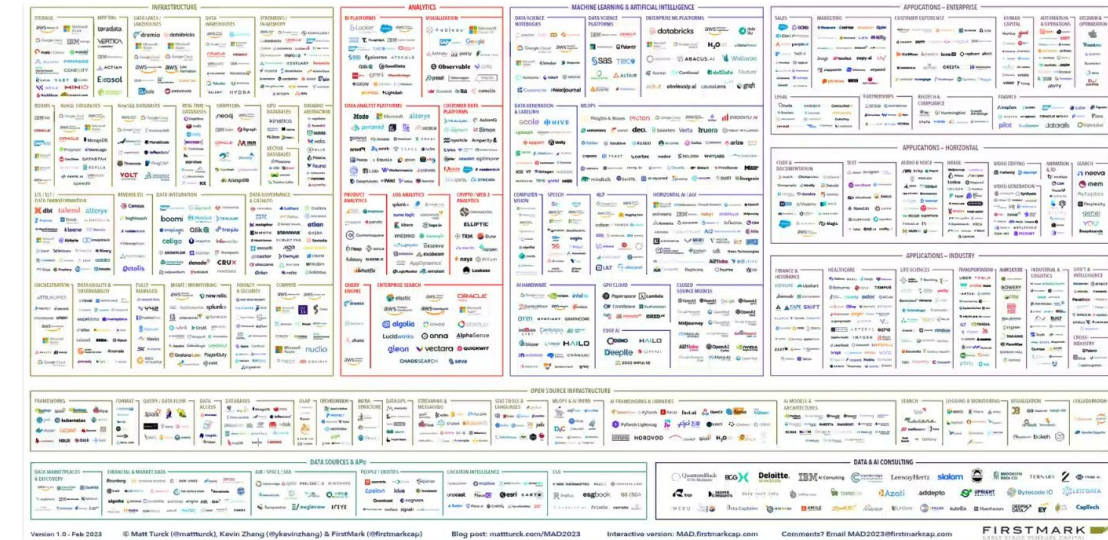


Figura 2. Ecosistema de empresas en el sector de AI, ML y Big Data 2023

1.2.1 Capa de Ingestión y Recopilación de Datos

La capa de ingestión y recopilación de datos es la puerta de entrada al ecosistema de Big Data. Esta capa es responsable de recolectar datos de múltiples fuentes, que pueden ser sistemas de archivos, bases de datos, flujos de datos en tiempo real, sistemas ERP, entre otros, y moverlos hacia el sistema de almacenamiento de datos.

1.2.1.1 Descripción General

La ingestión de datos puede ser una operación simple y unidireccional, o puede involucrar operaciones más complejas como la transformación y limpieza de datos durante el proceso de ingestión. En el contexto de Big Data, se hace uso de tecnologías que pueden manejar grandes volúmenes de datos en diferentes formatos y velocidades, desde datos estructurados en bases de datos hasta flujos de datos en tiempo real generados por sensores o actividades de usuario en una aplicación web.

1.2.1.2 Importancia en el Ecosistema del Big Data

La correcta configuración y operación de la capa de ingestión son cruciales para el funcionamiento eficaz del ecosistema de Big Data. Cualquier fallo o deficiencia en esta capa puede resultar en pérdida de datos, duplicación o incluso corrupción de datos, afectando así la calidad de todo el análisis posterior.

1.2.1.3 Desafíos y Consideraciones

Entre los desafíos significativos de la capa de ingestión están la escalabilidad, la tolerancia a fallos, la latencia y la integridad de los datos. Las soluciones deben ser capaces de manejar picos en el flujo de datos, recuperarse de fallos sin pérdida de datos y garantizar que los datos sean coherentes y completos. Las consideraciones de seguridad y cumplimiento también son primordiales, ya que los datos a menudo contienen información sensible y durante el movimiento de datos entre sistemas se acentúa la vulnerabilidad de los mismos.

1.2.1.4 Principales Actores y Empresas en el Mercado

- **AWS Glue:** Un servicio completamente gestionado que facilita la preparación, el cargado y la transformación de datos. Proporciona tanto capacidades de ETL (extracción, transformación y carga) como de ETL en tiempo real.
- **Azure Data Factory:** Este servicio en la nube de Microsoft permite crear, programar y orquestar flujos de trabajo de datos. Ofrece capacidades tanto para ETL como para ELT (extracción, carga y transformación), y se integra fácilmente con varios servicios de Azure y bases de datos SQL. También puede ser usado como orquestador de otros servicios.
- **Apache Nifi:** Una solución de código abierto que permite la automatización del flujo de datos entre sistemas diferentes.

1.2.2 Capa de Infraestructura y Almacenamiento

1.2.2.1 Descripción general

La capa de infraestructura y almacenamiento es el sustrato físico y lógico sobre el que se construye y opera toda arquitectura de Big Data. Esta capa se encarga del aprovisionamiento de recursos computacionales, gestión del almacenamiento distribuido y la red de interconexión, ofreciendo así la plataforma necesaria para el cómputo en paralelo y el almacenamiento resiliente.

En términos de hardware, estamos hablando de clústeres que incluyen nodos de procesamiento y nodos de almacenamiento, cada uno con especificaciones técnicas diseñadas para satisfacer

requisitos como alta disponibilidad, baja latencia y balanceo de carga. También integra tecnologías de virtualización y contenedorización, como Docker y Kubernetes, para la optimización del uso de recursos y la abstracción del hardware.

Desde el punto de vista del almacenamiento, se recurre a soluciones que varían desde bases de datos NoSQL como MongoDB, Cassandra o HBase, hasta Delta Lakehouse y sistemas de archivos distribuidos como HDFS. Estas soluciones se eligen y configuran en función de las necesidades específicas en cuanto a rendimiento (throughput y latencia), escalabilidad horizontal y consistencia eventual, para así cumplir con los requisitos de los "cinco Vs" del Big Data: Volumen, Velocidad, Variedad, Veracidad y Valor.

Es en esta capa donde se implementan mecanismos de seguridad como la autenticación y el control de acceso, así como se configuran políticas de replicación de datos y estrategias para la recuperación ante fallos. También se establecen las bases para la observabilidad y monitorización del sistema mediante herramientas como Prometheus o Grafana, que facilitan la detección y diagnóstico de cuellos de botella o fallos en el sistema.

La importancia de un diseño y configuración óptimos en esta capa es crítica, ya que cualquier ineficiencia o fallo en esta etapa se propaga y amplifica en las capas superiores, impactando negativamente tanto en el rendimiento como en la fiabilidad del sistema de Big Data en su totalidad. Además, dadas las implicancias de coste y complejidad, las decisiones tomadas en esta fase pueden tener un fuerte impacto económico y operativo en el proyecto.

Dada la naturaleza esencial y compleja de la capa de infraestructura y almacenamiento, este subapartado se enfoca en un análisis detallado y metódico de sus componentes, tecnologías y mejores prácticas. Esto incluirá también un escrutinio de los principales actores del mercado, sus ofertas de valor y cómo se alinean con los requisitos y objetivos de este proyecto. La meta es identificar y justificar las tecnologías y estrategias más adecuadas para el desarrollo del script o ejecutable destinado al despliegue automático y optimizado de las soluciones de Big Data.

1.2.2.2 Importancia en el ecosistema Big Data

En el ecosistema Big Data, la capa de infraestructura y almacenamiento actúa como el pilar fundamental sobre el que descansan todas las demás capas y funciones. Su relevancia no se limita a simplemente proveer recursos físicos o alojar datos; va mucho más allá, determinando en gran medida la efectividad, eficiencia y escalabilidad del entero sistema. En términos más técnicos, podría considerarse como el "kernel" del sistema de Big Data, donde cualquier fallo o ineficiencia podría comprometer la integridad y desempeño de las capas de aplicación y analítica que se superponen sobre ella. A medida que el volumen, la variedad y la velocidad de los datos crecen exponencialmente, la infraestructura se convierte en el factor habilitante que determina el éxito o el fracaso de cualquier proyecto de Big Data.

Sin una infraestructura adecuada, sería imposible llevar a cabo tareas analíticas sofisticadas o ejecutar algoritmos de aprendizaje automático a gran escala. Es la infraestructura la que proporciona el entorno donde las operaciones de datos, desde la ingestión hasta el análisis y la visualización, se llevan a cabo.

Adicionalmente, una infraestructura óptimamente diseñada facilita un flujo de datos no sólo eficiente, sino también efectivo en términos de latencia y throughput. Esta característica es primordial para garantizar la generación y entrega de conocimientos accionables en un marco temporal cercano al tiempo real.

La infraestructura se posiciona como un facilitador estratégico para el despliegue de tecnologías emergentes, como la inteligencia artificial y el aprendizaje automático. En ausencia de una arquitectura robusta capaz de almacenar, procesar y distribuir grandes volúmenes de datos, la implementación efectiva de dichas tecnologías sería inviable.

Una arquitectura de infraestructura sólida y bien orquestada también maximiza la eficiencia en la asignación de recursos computacionales. Esto es especialmente crucial en escenarios que demandan el procesamiento en tiempo real de datos masivos, ya que una asignación eficiente reduce significativamente los tiempos de ejecución y optimiza el rendimiento general.

Con una infraestructura bien diseñada, las organizaciones tienen la flexibilidad para adaptarse a cambios en los requerimientos de negocio o tecnológicos, como pueden ser nuevas fuentes de datos o cambios en las regulaciones. Esto permite a las empresas ser más ágiles y reactivas a las condiciones del mercado.

1.2.2.3 Desafíos y consideraciones

Abordar la capa de Infraestructura y Almacenamiento con precisión técnica es un ejercicio meticuloso que implica varias consideraciones clave:

Consistencia y Disponibilidad

La infraestructura debe garantizar la consistencia de datos en un entorno distribuido, lo cual es un desafío no trivial dada la necesidad de mantener una alta disponibilidad. Las técnicas como el sharding y la replicación pueden ser útiles, pero cada una viene con su propio conjunto de trade-offs.

Escalabilidad Vertical y Horizontal

El sistema debe ser diseñado para soportar tanto la escalabilidad vertical (aumento de la capacidad de un solo nodo) como la horizontal (aumento del número de nodos). Las estrategias para el balanceo de carga y la distribución de datos se vuelven fundamentales aquí.

Tolerancia a Fallos

Dado que estamos hablando de sistemas distribuidos, la tolerancia a fallos se convierte en un criterio de diseño esencial. El sistema debe ser capaz de recuperarse y realizar auto-correcciones sin intervención manual, lo cual a menudo implica implementaciones de mecanismos de consenso como Paxos o Raft.

Latencia y Rendimiento

Las demandas de baja latencia y alto rendimiento en operaciones de lectura y escritura son cruciales. Para lograrlo, se pueden implementar estrategias como la segmentación de datos y la optimización de consultas, utilizando técnicas como el almacenamiento en caché y la indexación.

Seguridad

La capa de infraestructura debe cumplir con las normas de seguridad para la protección de datos. Esto incluye, pero no se limita a, el cifrado en reposo y en tránsito, el control de acceso basado en roles y el seguimiento y auditoría de las operaciones en el sistema.

Administración de Recursos

Un aspecto técnico para considerar es cómo se administrarán los recursos computacionales. Soluciones de orquestación como Kubernetes o proveedores de cloud público pueden ser críticas para el aprovisionamiento dinámico y la administración de contenedores y microservicios.

Interoperabilidad

Finalmente, pero no menos importante, la infraestructura seleccionada debe ser compatible con una variedad de sistemas de almacenamiento y bases de datos, desde almacenes de datos en columnas como Apache Cassandra hasta bases de datos de documentos como MongoDB o bases de datos relacionales.

1.2.2.4 Principales Actores y Empresas en el Mercado

La capa de Infraestructura y Almacenamiento es un componente crucial en el ecosistema de Big Data, y varios actores del mercado se destacan en este ámbito por ofrecer soluciones específicas que responden a los desafíos inherentes a la gestión de grandes volúmenes de datos. A continuación, se presentan estos actores categorizados de acuerdo con su enfoque en el almacenamiento y la infraestructura.

Proveedores de nube pública

En el contexto contemporáneo de la gestión de grandes volúmenes de datos, los proveedores de servicios en la nube pública se han establecido como elementos críticos en la infraestructura de almacenamiento y procesamiento de datos. Estos proveedores ofrecen una gama flexible de servicios que permiten escalar recursos de manera dinámica, reduciendo la necesidad de invertir en hardware físico y en el mantenimiento correspondiente.

Amazon Web Services (AWS)

En el ámbito de la infraestructura de nube pública para Big Data, Amazon Web Services (AWS) se posiciona como un líder indiscutible, ofreciendo una gama de servicios específicamente diseñados para gestionar los complejos desafíos relacionados con el almacenamiento y la infraestructura de datos a gran escala. A continuación, se

describen los servicios más relevantes en la capa de infraestructura y almacenamiento dentro del ecosistema AWS.

Tabla 1. Comparación de servicios de Infraestructura y almacenamiento en AWS

Nombre	Tipo Servicio	Descripción
EC2	IaaS	AWS EC2 ofrece capacidades de computación en la nube escalables y a medida. Los usuarios pueden seleccionar las instancias que mejor se ajusten a sus requisitos de cómputo, con opciones que varían en términos de CPU, memoria, almacenamiento y redes. Estas instancias se pueden escalar vertical y horizontalmente, proporcionando la flexibilidad necesaria para adaptarse a las variaciones en la demanda de recursos. La elasticidad y la gama de instancias disponibles hacen de EC2 una elección sólida para cualquier pila de Big Data.
S3	IaaS	Amazon S3 es el servicio de almacenamiento de objetos de AWS, ampliamente adoptado para almacenar grandes cantidades de datos no estructurados. La alta durabilidad, la capacidad de establecer políticas de ciclo de vida de objetos y la facilidad para configurar bucket policies hacen de S3 una elección robusta para el almacenamiento de datos a gran escala. Las características como la posibilidad de clasificar datos en diferentes clases de almacenamiento (Standard, Infrequent Access, Glacier, etc.) proporcionan una flexibilidad inigualable en la optimización de costos.
EBS	IaaS	Diseñado para ser utilizado con EC2, Amazon EBS proporciona almacenamiento de bloques escalable para el uso con bases de datos y aplicaciones que requieren un acceso a datos rápido y fiable. EBS es especialmente relevante en escenarios donde la latencia es un factor crítico, y ofrece diversas opciones de volumen que se pueden ajustar para cumplir con requisitos específicos de rendimiento.
AWS Glacier	IaaS	AWS Glacier es un servicio de almacenamiento en la nube de bajo costo proporcionado por Amazon Web Services (AWS) diseñado para el archivo de datos a largo plazo y la copia de seguridad. Se caracteriza por ofrecer una alta durabilidad con un coste muy reducido, lo cual lo convierte en una opción ideal para datos que se acceden raramente y que pueden tolerar tiempos de recuperación prolongados.

Microsoft Azure

En la esfera de la nube pública, Microsoft Azure representa una de las plataformas más sólidas y versátiles para las necesidades de infraestructura y almacenamiento de Big Data sobretodo alineadas con las necesidades empresariales.

Tabla 2. Comparación de servicios de Infraestructura y almacenamiento en Azure

Nombre	Tipo Servicio	Descripción
Azure Virtual Machines (VMs)	IaaS	Azure VMs ofrecen una flexibilidad tremenda en términos de opciones de CPU, memoria, almacenamiento y redes, permitiendo a los usuarios escoger instancias que se adecuen a sus necesidades computacionales específicas. Este servicio también admite una amplia variedad de sistemas operativos y se integra estrechamente con otros servicios de Azure, proporcionando una base sólida para cualquier arquitectura de Big Data.
Blob Storage	IaaS	Servicio de almacenamiento de objetos de Azure para manejar datos no estructurados o semi-estructurados a gran escala. Con características como replicación geográfica, niveles de acceso (frío, caliente, y archivado) y políticas de ciclo de vida, Azure Blob Storage se configura como una solución altamente durable y flexible para el almacenamiento de Big Data.
Azure Disk Storage	IaaS	Diseñado para ser usado con Azure VMs, este servicio de almacenamiento en bloque ofrece discos persistentes altamente durables y con disponibilidad asegurada. Con una gama de opciones de rendimiento y tamaño, Azure Disk Storage es particularmente útil para aplicaciones y bases de datos que requieren un acceso rápido y fiable a los datos.
Azure Files	IaaS	Azure Files ofrece un servicio de almacenamiento de archivos basado en la nube que se puede compartir entre varias VMs y servicios. Con soporte para protocolos SMB y NFS, este servicio es particularmente relevante para escenarios de colaboración y flujos de trabajo que necesitan un almacenamiento compartido.

- **Google Cloud Platform (GCP)**

Cómo el 3r competidor, Google cloud también presenta una variedad de soluciones y servicios altamente escalables y flexibles.

Tabla 3. Comparación de servicios de Infraestructura y almacenamiento en GCP

Nombre	Tipo Servicio	Descripción
Compute Engine	IaaS	Se trata del servicio de máquinas virtuales (VMs) de Google Cloud que ofrece una amplia personalización en términos de CPU, memoria y opciones de almacenamiento. Compute Engine es una solución robusta para ejecutar aplicaciones y procesos de Big Data de alta exigencia computacional, especialmente porque se integra fluidamente con otros productos de GCP.
Google Cloud Storage	IaaS	Servicio de almacenamiento de objetos de Google, diseñado para datos no estructurados o semi-estructurados. Ofrece una solución de almacenamiento altamente escalable con características como versionado, durabilidad, y distintas clases de almacenamiento (Standard, Nearline, Coldline y Archive) según el patrón de acceso a los datos.
Persistent Disk	IaaS	Es el servicio de almacenamiento en bloque de Google Cloud, óptimo para bases de datos y aplicaciones que requieren un acceso rápido y

		consistente a grandes volúmenes de datos. Los discos pueden ser SSD o HDD y ofrecen una durabilidad y disponibilidad muy alta.
Google Cloud Filestore	IaaS	Similar a Azure Files y Amazon EFS, Filestore es el servicio de almacenamiento de archivos de Google Cloud. Ofrece una solución NFS-compatible, lo que lo hace adecuado para aplicaciones y flujos de trabajo que requieren almacenamiento de archivos compartidos.

Almacenamiento Distribuido:

El almacenamiento distribuido es un componente crucial en la arquitectura de un ecosistema de Big Data. La necesidad de manejar grandes volúmenes de datos, que pueden estar estructurados, semi-estructurados o no estructurados, requiere soluciones que ofrezcan alta disponibilidad, escalabilidad y tolerancia a fallos.

Hadoop Distributed File System (HDFS): HDFS es una de las soluciones de almacenamiento distribuido más utilizadas en la industria. Fue diseñado para funcionar con hardware de bajo coste y es altamente tolerante a fallos. HDFS se utiliza a menudo en conjunción con el ecosistema Hadoop para el procesamiento de Big Data.

Ceph: Ceph es una solución de almacenamiento distribuido que proporciona escalabilidad horizontal a la par de ofrecer un sistema de archivos robusto. Su arquitectura de objetos y su posibilidad de auto-reparación y auto-equilibrio lo convierten en una opción sólida para entornos empresariales.

Bases de Datos NoSQL y Relacionales

En el ámbito de las bases de datos, la escala y la variedad de los datos han llevado al desarrollo de soluciones tanto NoSQL como relacionales que son críticas en el almacenamiento y la recuperación de datos en un ecosistema de Big Data.

MongoDB: Es una de las bases de datos NoSQL más populares, diseñada para ofrecer escalabilidad horizontal y flexibilidad en el esquema. MongoDB es particularmente útil para almacenar grandes cantidades de datos no estructurados o semi-estructurados.

Cassandra: Desarrollada inicialmente por Facebook, esta base de datos NoSQL es conocida por su escalabilidad y alta disponibilidad sin comprometer el rendimiento.

Amazon DynamoDB: Es una base de datos NoSQL completamente administrada, que ofrece una latencia de milisegundos de un solo dígito a cualquier escala. Su modelo de precios flexible lo convierte en una opción atractiva para empresas de todos los tamaños.

MySQL y PostgreSQL: En el lado de las bases de datos relacionales, tanto MySQL como PostgreSQL se utilizan ampliamente en entornos empresariales. Estas bases de datos son ideales para aplicaciones que requieren transacciones ACID y un esquema de datos bien definido.

Microsoft SQL Server: Este es un sistema de gestión de bases de datos relacionales que proporciona una variedad de capacidades administrativas, de desarrollo y empresariales. Se integra bien con herramientas de BI y otras herramientas de análisis de datos.

1.2.3 Capa de Procesamiento y Transformación

La capa de procesamiento y transformación actúa como el núcleo operativo en un ecosistema de Big Data, encargándose de la manipulación y análisis de los datos almacenados en la capa de infraestructura y almacenamiento. Es aquí donde los datos brutos son transformados en información valiosa mediante algoritmos y operaciones de análisis complejas. Dada su importancia crítica, es imperativo examinar las tecnologías y herramientas que dominan este espacio, sus ventajas, limitaciones y cómo se integran en una arquitectura coherente de Big Data.

1.2.3.1 Descripción General

En la capa de procesamiento y transformación, se aplican algoritmos y se llevan a cabo tareas de limpieza de datos, normalización y transformación mediante procesos de ETL o ELT. El objetivo es preparar los datos para la fase analítica. Este nivel de la arquitectura suele incluir motores de procesamiento de datos, bibliotecas de algoritmos, servicios de transformación y otros componentes que facilitan la manipulación de datos a gran escala. Las soluciones en esta capa pueden clasificarse en dos categorías principales: procesamiento en lotes (batch processing) y procesamiento en tiempo real (stream processing).

Procesamiento en Lotes: Se refiere al procesamiento de grandes volúmenes de datos que se almacenan en un sistema de archivos distribuido. Este enfoque es adecuado para tareas que no requieren una respuesta en tiempo real.

Procesamiento en Tiempo Real: En este modelo, los datos se procesan casi instantáneamente a medida que llegan al sistema, permitiendo que las aplicaciones utilicen la información en tiempo real.

1.2.3.2 Importancia en el Ecosistema del Big Data

La capa de procesamiento y transformación desempeña un papel crítico en el valor que se puede extraer de los datos. La calidad del análisis subsiguiente y las decisiones empresariales resultantes son altamente dependientes del nivel de precisión y eficiencia alcanzado en esta fase.

1.2.3.3 Desafíos y Consideraciones

Los siguientes requerimientos deben cumplirse para poder trabajar en entornos complejos sin problemas en la capa de procesamiento:

Concurrencia: El manejo efectivo de múltiples trabajos y tareas en paralelo es crucial para optimizar los recursos.

Tolerancia a fallos: En entornos productivos de cálculos complejos, donde la interrupción del servicio puede ser costosa, la tolerancia a fallos es un factor crítico.

Latencia y Rendimiento: La capacidad para procesar grandes conjuntos de datos en un tiempo mínimo es otro desafío técnico, especialmente en aplicaciones de tiempo real.

1.2.3.4 Principales Actores y Empresas en el Mercado

Apache Hadoop: Su modelo de programación MapReduce lo convierte en la base de mayoría de *frameworks* de computación en entornos distribuidos.

Apache Spark: Un motor de procesamiento de datos en memoria para operaciones de datos grandes y complejas. Es ampliamente utilizado para tareas que van desde el procesamiento por lotes hasta el análisis de datos en tiempo real.

Databricks: Ofrece un ambiente unificado para trabajar con Spark, facilitando la colaboración entre data engineers y data scientists. Además, Databricks incluye funcionalidades para ML y análisis avanzado.

AWS EMR: Este es un servicio en la nube que utiliza Apache Spark y Hadoop para procesar grandes conjuntos de datos. EMR es una solución de PaaS que permite el procesamiento en paralelo y es altamente escalable.

Azure HDInsight: Es una solución de PaaS que permite procesar grandes cantidades de datos utilizando Apache Spark y Hadoop en el entorno de nube de Azure. HDInsight también soporta R, Python y otros lenguajes, y se integra con diversas herramientas de Azure como Azure Blob Storage y Azure Data Lake Storage.

1.2.4 Capa de Orquestación y Automatización

1.2.4.1 Descripción general

La capa de orquestación y automatización es el espinazo operacional de un ecosistema de Big Data. Esta capa es responsable de la coordinación de tareas y la automatización de flujos de trabajo entre las diversas capas del ecosistema. Actúa como un controlador centralizado para gestionar el almacenamiento, el procesamiento y la distribución de datos. A continuación, se describen sus componentes, su importancia en el ecosistema de Big Data, los desafíos y consideraciones relevantes, y los principales actores y empresas en el mercado que ofrecen soluciones en este ámbito.

1.2.4.2 Importancia de la Integración Efectiva

Esta capa permite que los recursos sean utilizados de manera eficiente y que los flujos de trabajo complejos se ejecuten de manera coherente y confiable. También proporciona un mecanismo para implementar políticas de seguridad y cumplimiento en todas las operaciones de datos aunque su principal funcionalidad es descargar trabajo en el lanzamiento recurrente de trabajos.

1.2.4.3 Desafíos y Consideraciones

Complejidad de Coordinación: La creciente complejidad de los flujos de datos y procesamiento exige una orquestación sofisticada.

Escalabilidad y Elasticidad: Deben soportar el crecimiento dinámico de las tareas, tanto en términos de volumen como de complejidad.

Resiliencia y Recuperación de Desastres: Las soluciones de orquestación deben ser resistentes a fallos y permitir una rápida recuperación en caso de interrupciones.

1.2.4.4 Principales Actores y Empresas en el Mercado

Apache Airflow: Un gestor de flujos de trabajo de código abierto diseñado para programar y monitorizar flujos de trabajo complejos.

Kubernetes: Aunque inicialmente no se diseñó específicamente para Big Data, su capacidad para gestionar contenedores lo hace cada vez más popular para la orquestación de servicios de datos.

AWS Step Functions: Este servicio de AWS coordina componentes de microservicios mediante flujos de trabajo visuales. Es una solución de PaaS.

Azure Logic Apps: Este servicio de Azure permite crear flujos de trabajo automatizados y escalables para la integración de aplicaciones, datos, servicios y sistemas de backend. También es una solución de PaaS.

1.2.5 Capa de Análisis y Consulta

1.2.5.1 Descripción General

La capa de análisis y consulta es una de las partes más cruciales de cualquier arquitectura de big data. Se sitúa generalmente al final del flujo de datos y actúa como el punto de interacción entre los datos procesados y los usuarios finales. Esta capa permite la realización de consultas, generación de informes, visualización de datos, y ejecución de análisis avanzados, a menudo a través de un conjunto diverso de herramientas y plataformas, desde SQL tradicional hasta soluciones especializadas de BI (Business Intelligence).

1.2.5.2 Importancia en el Ecosistema del Big Data

Dado que la gestión y almacenamiento de grandes volúmenes de datos son inútiles si los insights extraídos no son accesibles o utilizables, la capa de análisis y consulta cobra vital importancia. Facilita la traducción de datos crudos en información significativa, impulsando así la toma de decisiones basada en datos. Su capacidad para integrarse con diversas fuentes de datos y su escalabilidad la hacen indispensable en cualquier solución de big data.

1.2.5.3 Desafíos y Consideraciones

Latencia en Consultas: A medida que aumenta el volumen de datos, la velocidad de las consultas puede disminuir, lo que afecta a la eficiencia del análisis.

Integración de Herramientas: Muchas organizaciones ya tienen un ecosistema de herramientas de BI y análisis. Integrar estas herramientas con la capa de análisis y consulta puede ser un desafío técnico.

Seguridad de Datos: Esta capa suele interactuar con datos procesados que podrían contener información sensible. Las medidas de seguridad y cumplimiento son, por lo tanto, cruciales.

Costes: Las soluciones de análisis de datos avanzadas pueden ser caras, especialmente cuando se requiere una gran cantidad de recursos computacionales para consultas y análisis en tiempo real.

1.2.5.4 Principales Actores y Empresas en el Mercado

Tableau: Una de las principales herramientas de visualización de datos y BI, ampliamente utilizada para su capacidad de manejar grandes conjuntos de datos.

Microsoft Power BI: Con fuertes capacidades de integración con otros productos de Microsoft, Power BI es una elección popular para las organizaciones ya arraigadas en el ecosistema de Microsoft.

SAP Analytics Cloud: Ofrece capacidades de planificación, análisis y predicción en una única solución, especialmente popular en entornos empresariales grandes.

Google Data Studio: Proporciona herramientas de visualización de datos de fácil uso y con una curva de aprendizaje más suave, a menudo utilizada para análisis de marketing y métricas web.

1.2.6 Capa de Presentación y Visualización

1.2.6.1 Descripción General

La capa de presentación y visualización es donde los datos procesados y analizados se traducen en gráficos, cuadros y otros formatos visuales para un fácil consumo por parte de los usuarios finales. Esta capa suele ser la interfaz entre los sistemas de big data y los tomadores de decisiones, ofreciendo una forma intuitiva de entender patrones, tendencias y insights que de otra manera podrían pasar desapercibidos en conjuntos de datos sin procesar o tablas complejas.

1.2.6.2 Importancia en el Ecosistema del Big Data

La presentación y visualización no son sólo un 'extra' añadido al final del ciclo de vida del big data, sino una pieza clave para hacer que los datos sean verdaderamente accionables. Al simplificar la complejidad inherente a los grandes conjuntos de datos, la capa de presentación y visualización permite una interpretación más rápida y precisa, lo que puede resultar crucial en escenarios de toma de decisiones a tiempo crítico.

1.2.6.3 Desafíos y Consideraciones

Interactividad y Escalabilidad: Los dashboards y las visualizaciones deben ser no sólo informativos sino también interactivos, lo que plantea desafíos en términos de rendimiento y escalabilidad.

Diseño Intuitivo: La eficacia de una visualización depende en gran medida de su diseño. Un diseño pobre puede llevar a malinterpretaciones y conclusiones erróneas.

Acceso Seguro: Dado que estas interfaces son el punto final de acceso a los datos, la seguridad es una preocupación significativa. La autenticación y el control de acceso son esenciales para proteger la información.

Integración con Otras Capas: La facilidad con la que esta capa puede integrarse con las capas anteriores de análisis y almacenamiento de datos es una consideración crucial para la fluidez del flujo de trabajo.

1.2.6.4 Principales Actores y Empresas en el Mercado

Tableau: Especializado en la creación de dashboards interactivos y de gran impacto visual, Tableau se ha convertido en un estándar de la industria.

Microsoft Power BI: Ofrece integración profunda con otros servicios de Microsoft y tiene una amplia gama de capacidades de visualización.

Looker: Adquirido por Google, se integra perfectamente con Google Cloud y ofrece una plataforma de análisis de datos y visualización muy completa.

Sisense: Famoso por su enfoque en la analítica de negocio, Sisense ofrece soluciones de BI y visualización de datos que se centran en hacer que los datos sean más accesibles para usuarios no técnicos.

QlikView/Qlik Sense: Ofrece un motor de análisis asociativo que permite una exploración de datos más compleja y en profundidad, con visualizaciones avanzadas.

1.2.7 Capa de Seguridad y Cumplimiento

1.2.7.1 Descripción General

La Capa de Seguridad y Cumplimiento se centra en la protección de la integridad, disponibilidad y confidencialidad de los datos. Además de las medidas de seguridad básicas como la encriptación y la autenticación, esta capa también aborda aspectos regulatorios y de cumplimiento, como GDPR, HIPAA, y otros marcos legales pertinentes que regulan cómo se deben manejar y proteger los datos.

1.2.7.2 Importancia en el Ecosistema del Big Data

En un entorno donde se manejan grandes volúmenes de datos, a menudo incluyendo información sensible o personal, la seguridad y el cumplimiento son cruciales. Un solo fallo de seguridad puede resultar en la pérdida de datos críticos, sanciones legales y daño a la reputación de la organización. Por tanto, esta capa es una parte indispensable del ecosistema de Big Data para garantizar que los datos se utilicen de manera responsable y segura.

1.2.7.3 Desafíos y Consideraciones

Escalar Seguridad: A medida que los conjuntos de datos crecen, también lo hace la complejidad de mantenerlos seguros. Las soluciones de seguridad deben ser escalables para adaptarse a volúmenes de datos cada vez mayores.

Cumplimiento Regulatorio: Mantenerse al día con las cambiantes leyes y regulaciones de protección de datos es un desafío constante.

Costo: Las medidas de seguridad robustas requieren una inversión significativa tanto en tecnología como en talento especializado.

Auditorías y Monitorización: Implementar sistemas de auditoría y monitorización para mantener un registro de quién accede a qué datos y cuándo.

1.2.7.4 Principales Actores y Empresas en el Mercado

Symantec: Ofrece una amplia gama de soluciones de seguridad, desde la protección del endpoint hasta la seguridad en la nube.

McAfee: Conocido por sus soluciones de seguridad para empresas, incluidas las destinadas específicamente a la protección de datos.

Palantir: Aunque principalmente una empresa de análisis de datos, Palantir también ofrece soluciones robustas para la seguridad y el cumplimiento de datos.

Okta: Especializado en gestión de identidad y accesos, un componente crítico de la seguridad de datos.

Varonis: Se centra en la seguridad de datos y protección contra amenazas internas, ofreciendo monitorización en tiempo real y alertas para actividades sospechosas.

2 Capítulo 2: Propuesta de solución

Este capítulo se centra en desglosar los elementos que constituyen nuestra propuesta de solución, desarrollado teniendo en cuenta los objetivos del trabajo. La propuesta integra diversas tecnologías y enfoques en un ecosistema coherente, con el objetivo de brindar una infraestructura robusta y escalable para la gestión del dato. Se presentará una arquitectura basada en Delta Lake, apoyada en tres capas de almacenamiento ubicadas en Azure Blob Storage: Bronze, Silver y Gold. Cada una de estas capas tiene un propósito específico y contribuye a la eficiencia y eficacia del sistema en su totalidad.

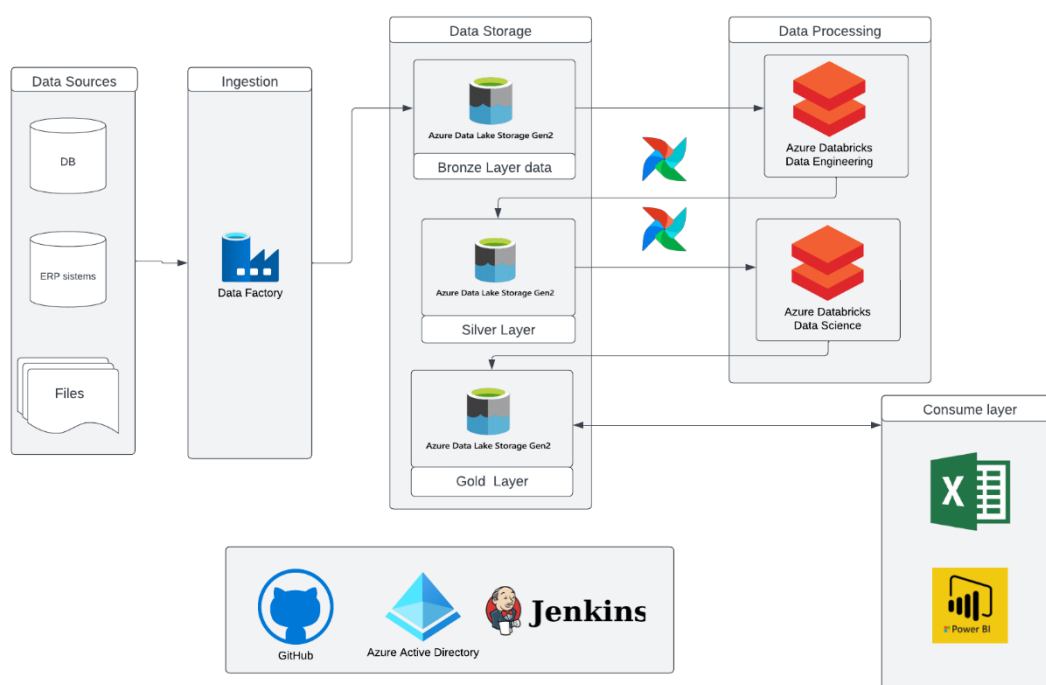


Figura 3. Arquitectura solución

Para asegurar una ingesta de datos sólida y confiable, se utilizará Azure Data Factory. Una vez que los datos están en el sistema, Databricks actuará como el motor de procesamiento para transformar y manipular los datos a medida que se mueven a través de las diferentes capas de almacenamiento. Y para coordinar estos procesos de una manera orquestada, se implementará Apache Airflow. Además, aunque queda fuera del alcance del proyecto, en el esquema se incluyen las herramientas github como control de versionado, Azure AD como método de autenticación entre los servicios y Jenkins como herramienta de CI/CD.

A lo largo de este capítulo, se explicarán en detalle cada uno de estos elementos y se analizará cómo contribuyen al logro del objetivo general del proyecto. Además, se ofrecerán

justificaciones técnicas y económicas para las decisiones de diseño, y se discutirán aspectos como la automatización, la escalabilidad, la seguridad y la gobernanza de los datos.

El enfoque no es solo presentar una solución tecnológica, sino también explicar cómo cada componente se alinea estratégicamente con los objetivos empresariales y técnicos del proyecto.

2.1 Arquitectura Delta Lake

Antes de definir las capas lógicas que constituirán el sistema —Bronze, Silver y Gold— y los elementos técnicos que la van a soportar, es esencial abordar la elección de la arquitectura subyacente que soportará estas capas: Delta Lake. La arquitectura Delta Lake presenta una serie de características técnicas y ventajas operativas que la hacen idónea para el proyecto en cuestión, que aspira a construir una solución robusta, escalable y de alta disponibilidad para la gestión de grandes volúmenes de datos.

2.1.1 Características de la arquitectura Delta Lake

2.1.1.1 Eficiencia de Almacenamiento y Consulta

Delta Lake utiliza un formato de almacenamiento tipo columna (como Parquet), lo cual es extremadamente eficiente tanto en términos de almacenamiento como de velocidad de consulta. Este formato es especialmente beneficioso para las operaciones de análisis de datos, que a menudo requieren la lectura de columnas específicas de los datos en lugar de filas completas. Este formato añade una capa de metadatos sobre el formato que permite realizar una serie de operaciones claves en entornos big data.

2.1.1.2 Transacciones ACID

Uno de los principales retos en la gestión de big data es mantener la integridad de los datos. Delta Lake ofrece soporte para transacciones ACID (Atomicidad, Coherencia, Aislamiento, Durabilidad), lo que garantiza que los datos sean consistentes antes y después de cada operación, evitando así los problemas comunes en los entornos de big data como duplicación o pérdida de datos.

2.1.1.3 Esquema de Evolución y Control

Delta Lake permite una evolución del esquema sin bloqueo, es decir, se pueden realizar cambios en el esquema de los datos sin interrumpir las operaciones en curso. Esto es crucial para la agilidad y la adaptabilidad del proyecto, ya que los requisitos y las fuentes de datos pueden cambiar con el tiempo.

2.1.1.4 Integración con Tecnologías Existentes

Delta Lake se integra sin problemas con tecnologías ampliamente utilizadas en el ecosistema de big data como Spark, Hive, Presto y más. En este proyecto, la integración con Azure Databricks proveerá un entorno unificado para la ingesta, almacenamiento y procesamiento de datos.

2.1.1.5 Optimizaciones de Ingeniería de Datos

Delta Lake incluye diversas optimizaciones específicas para ingeniería de datos como la compactación de archivos pequeños, la mejora de las consultas mediante la indexación Z-Order y la eliminación de archivos no utilizados, que mejoran el rendimiento y reducen los costos.

2.1.1.6 Seguridad y Gobernanza

En un entorno empresarial, la seguridad de los datos es primordial. Delta Lake ofrece múltiples capas de seguridad, que incluyen autenticación, control de acceso basado en roles (RBAC) y encriptación de datos en reposo y en tránsito.

2.1.2 Metadatos y sus funcionalidades en Delta Lake

Es fundamental abordar su enfoque en la gestión de metadatos, una característica que lo distingue y potencia en el ecosistema de big data. Almacenar y gestionar metadatos de manera eficiente es crucial para un sistema de datos de alto rendimiento y confiable, y Delta Lake aborda esta necesidad de manera ejemplar.

Los metadatos, que se podrían describir como "datos acerca de los datos", ofrecen información crucial sobre la estructura, tipo, y relaciones entre los datos almacenados. En Delta Lake, la gestión de metadatos no es un componente secundario, sino una pieza integral de su arquitectura.

Delta Lake mantiene un registro de transacciones que es esencialmente un catálogo completo de todas las operaciones de lectura y escritura realizadas en el sistema de datos. Este registro es utilizado tanto para garantizar transacciones ACID como para proporcionar una vista coherente y unificada de los datos y sus metadatos.

Una de las potentes características de Delta Lake es su habilidad para vincularse con 'external tables' o tablas externas. Estas tablas, que podrían estar almacenadas en otros formatos o incluso en otros sistemas de almacenamiento, pueden ser fácilmente integradas en el ecosistema de Delta Lake gracias a su estructura de metadatos. Esto no solo facilita un acceso rápido y eficiente a una amplia variedad de datos sino que también asegura que los metadatos para esas tablas externas sean mantenidos con el mismo nivel de coherencia y rigor que los datos nativos de Delta Lake.

La gestión de metadatos también permite un descubrimiento de datos más eficiente y proporciona capacidades para realizar optimizaciones en el nivel de consulta. Por ejemplo, los metadatos pueden ser utilizados para realizar la partición de datos, lo que mejora significativamente la velocidad de consulta. Además, al tener un registro completo de los metadatos, es posible hacer seguimiento de las versiones de los datos, facilitando así operaciones como la auditoría y el cumplimiento de normativas.

Este sistema de metadatos resulta especialmente útil cuando se trabaja con múltiples capas de almacenamiento de datos (Bronze, Silver y Gold), que requieren diferentes niveles de transformación y limpieza. Los metadatos ayudan a rastrear el linaje de los datos a medida que pasan por estas capas, ofreciendo transparencia y trazabilidad. Esto no solo facilita el mantenimiento y la depuración sino que también enriquece la calidad y confiabilidad de los datos, que es uno de los objetivos centrales de este proyecto.

2.2 Elementos de la Capa del Dato

2.2.1 Introducción a las capas lógicas del dato: Bronze, Silver y Gold

Una de las piedras angulares de la arquitectura propuesta es la implementación de un enfoque en capas para la gestión de datos, concretamente las capas Bronze, Silver y Gold. Este diseño multicapa permite una separación clara de las responsabilidades y funciones en cada etapa del ciclo de vida del dato, desde su ingestión hasta su consumo para análisis y toma de decisiones. A continuación, se ofrecen definiciones técnicas detalladas de cada una de estas capas, así como un análisis profundo de su función dentro del ecosistema de datos. Es fundamental entender que cada capa representa un nivel diferente de madurez del dato y que, juntas, estas capas conforman una arquitectura de almacenamiento y procesamiento que es tanto escalable como resiliente.

2.2.1.1 Capa Bronze (*Raw Data Layer*)

La capa Bronze es la primera en la jerarquía y actúa como el punto de entrada para todos los datos entrantes en el sistema. Los datos se almacenan en su forma más cruda, es decir, sin procesar o modificar. Esta capa es fundamentalmente un *data lake* en donde se depositan datos de diversas fuentes, sean estructurados, semi-estructurados o no estructurados. Al recopilar todos los datos heterogéneos, la capa Bronze ya comienza a contribuir a la gestión de datos holística. Esta estandarización inicial es crucial para las fases posteriores de transformación y análisis, asegurando que todas las operaciones de datos sean coherentes y comparables, alineándose así con el objetivo de una gestión de datos integral.

Formato de Almacenamiento: En su mayoría, los datos se almacenan en un formato RAW o, en el caso de fuentes de datos relacionales, en formato Parquet para eficiencia de almacenamiento y consultas.

Función: Actúa como un amortiguador que recopila datos en su estado original para que puedan ser procesados y transformados posteriormente. Sirve para la ingesta batch o en tiempo real.

Tecnología Usada: Azure Blob Storage se utiliza para el almacenamiento escalable y Azure Data Factory para la ingestión de datos.

2.2.1.2 Capa Silver (*Cleansed and Enriched Data Layer*)

La capa Silver representa el siguiente nivel de refinamiento, donde los datos se limpian, transforman y enriquecen. Las operaciones ETL (Extracción, Transformación y Carga) se llevan a cabo en esta etapa para preparar los datos para análisis avanzado. Esta capa es escalable tanto en términos de volumen de datos como de complejidad de las transformaciones, lo que facilita una respuesta eficaz a las cambiantes necesidades empresariales y técnicas. La escalabilidad inherente en esta capa es esencial para cumplir con el objetivo de manejar volúmenes de datos en constante evolución.

Formato de Almacenamiento: Los datos generalmente se almacenan en formato Parquet, que es eficiente en términos de almacenamiento y optimizado para operaciones de lectura.

Función: La capa Silver es donde los datos se limpian, se normalizan, y se enriquecen. Aquí, se aplican operaciones como la eliminación de duplicados, la corrección de errores y la incorporación de metadatos.

Tecnología Usada: Azure Databricks se utiliza para realizar todas las operaciones de transformación de datos mediante Spark. Los notebooks de Databricks permiten una colaboración eficiente entre los ingenieros de datos y los científicos de datos.

2.2.1.3 Capa Gold (Aggregated and Business-Ready Data Layer)

La capa Gold es la capa final y contiene datos que han sido transformados, agregados y están listos para ser consumidos por aplicaciones de negocio o para análisis avanzados. Los datos en esta capa están altamente optimizados para consultas rápidas y eficientes, lo que es esencial para cualquier proyecto que aspire a ofrecer insights en tiempo real. Al asegurar que los datos sean de alta calidad, coherentes y de fácil acceso, la capa Gold refuerza el objetivo de una gestión de datos holística.

Formato de Almacenamiento: En la arquitectura delta lake, los datos se almacenan generalmente en formato Delta.

Función: Los datos aquí están optimizados para consultas de alto rendimiento y análisis. Esto incluye tablas agregadas, vistas materializadas y otros conjuntos de datos que son cruciales para la toma de decisiones empresariales.

Tecnología Usada: Los datos en la capa Gold son generalmente accedidos a través de herramientas BI o plataformas de análisis de datos que pueden integrarse con Azure Blob Storage.

Cada una de estas capas no es un compartimento estanco, sino parte de un flujo continuo de datos que se mueve a través de la arquitectura. Este diseño en capas es especialmente útil para garantizar tanto la integridad del dato como la flexibilidad del sistema en su conjunto. Al separar las responsabilidades de la gestión de datos en estas capas distintas pero interconectadas, no solo se mejora la eficiencia operacional, sino que también se alinea estrechamente con los objetivos del proyecto de construir un sistema robusto, escalable y mantenible para la gestión de datos a gran escala.

2.2.2 Azure Blob Storage como almacenamiento principal

Azure Blob Storage es una solución de almacenamiento en la nube diseñada por Microsoft para guardar grandes cantidades de datos no estructurados, como textos o binarios. En el contexto de este proyecto, hemos seleccionado Azure Blob Storage como el almacenamiento principal para nuestras capas Bronze, Silver y Gold.

2.2.2.1 Justificación de la elección de Azure Blob Storage para almacenar las capas Bronze, Silver y Gold.

Azure Blob Storage está diseñado para ser altamente disponible y duradero. Internamente, Azure almacena automáticamente múltiples copias de cada blob (objeto binario grande) en centros de datos localizados, garantizando así la disponibilidad y resiliencia de los datos. Con una durabilidad del 99,9999999% (11 9's) garantizada por Microsoft, esto asegura que los datos en nuestras capas Bronze, Silver y Gold estén prácticamente siempre disponibles y protegidos contra fallos.

El servicio es altamente escalable, es capaz de manejar miles de millones de blobs, o incluso petabytes de datos, sin requerir ningún ajuste previo. Esto se alinea con la necesidad del proyecto de tener una solución que pueda escalar horizontalmente con el crecimiento de datos, sin preocupaciones de limitaciones de infraestructura.

Azure Blob Storage ofrece diferentes niveles de acceso (hot, cool y archive) que permiten optimizar costes según el patrón de acceso a los datos. Por ejemplo, los datos en la capa Bronze, que pueden no ser accedidos con frecuencia pero necesitan ser retenidos, pueden almacenarse en niveles más fríos (cool o archive) para reducir costes. Además permite mantener las copias de seguridad en un sistema unison al resto de datos de la compañía.

Azure Blob Storage ase puede configurar con múltiples capas de seguridad, incluyendo el control de acceso basado en roles (RBAC), la encriptación de datos en reposo y en tránsito, y capacidades de auditoría. Estas características son fundamentales para cumplir con los requisitos de conformidad y gobernanza de datos.

En comparación con soluciones tradicionales de almacenamiento, Azure Blob Storage ofrece un modelo de precio basado en el consumo. Esto significa que sólo se paga por el espacio que realmente se utiliza. Además, al mover datos entre diferentes niveles de acceso (por ejemplo, de hot a archive), se pueden realizar más optimizaciones de coste. Microsoft también ofrece herramientas para monitorizar y gestionar los gastos asociados con el almacenamiento.

Tabla 4. Tabla de coste Azure Blob Storage pay-as-you-go (GRS redundancy)

Data storage prices pay-as-you-go	Hot	Cool	Cold	Archive
First 50 terabyte (TB) / month	€0.0361 per GB	€0.01847 per GB	€0.00748 per GB	€0.00258 per GB
Next 450 TB / month	€0.0348 per GB	€0.01847 per GB	€0.00748 per GB	€0.00258 per GB
Over 500 TB / month	€0.0334 per GB	€0.01847 per GB	€0.00748 per GB	€0.00258 per GB

2.2.2.2 Ventajas técnicas y económicas en contexto con el propósito del proyecto

Ventajas Técnicas

Automatización y Orquestación: Blob Storage se integra perfectamente con servicios de Azure como Data Factory y Databricks, lo que permite una orquestación fluida de flujos de trabajo de datos. Esto es esencial para el objetivo del proyecto de automatizar todo el ciclo de vida de los datos.

Despliegue de Scripts: Es posible automatizar la creación y administración de objetos blob utilizando SDKs y APIs proporcionadas por Azure. Esto se puede hacer, por ejemplo, utilizando un script en Python que haga uso del SDK de Azure para Python. Este nivel de automatización es crucial para el objetivo del proyecto de permitir un manejo flexible y programático de los datos.

Concurrencia: Blob Storage permite operaciones de alta concurrencia, lo que significa que múltiples usuarios y aplicaciones pueden acceder y modificar los datos simultáneamente sin afectar el rendimiento. Esto es crítico para la escalabilidad y la eficiencia del proyecto.

Ventajas Económicas

Modelo de Costo por Uso: Al pagar solo por el almacenamiento que se utiliza efectivamente, se logra un modelo económico flexible que se adapta a las necesidades cambiantes del proyecto, permitiendo así un control más preciso de los costos operativos.

Optimización de Costo a Través de Niveles de Acceso: La posibilidad de cambiar entre diferentes niveles de acceso (hot, cool, archive) permite una gestión económica más eficiente, alineada con la frecuencia y criticidad del acceso a los datos.

Sin Costes Ocultos: Al estar integrado en el ecosistema de Azure, se evitan costes adicionales por transferencia de datos entre Blob Storage y otros servicios de Azure, como Azure Data Factory o Databricks.

Economías de Escala: Al estar dentro del ecosistema de Azure, hay potencial para aprovechar las economías de escala en términos de licenciamiento, mantenimiento y recursos humanos especializados.

2.2.3 Azure Data Factory para la ingesta de datos

Azure Data Factory (ADF) es un servicio de integración de datos en la nube que permite la construcción de flujos de datos ETL (Extracción, Transformación y Carga) y ELT (Extracción, Carga y Transformación) sin servidor. Proporciona una amplia variedad de orígenes y destinos de datos, permitiendo la orquestación de flujos de datos de alta complejidad. En el contexto del proyecto, ADF actúa como el motor principal para la ingestión de datos hacia la capa Bronze en Azure Blob Storage.

La velocidad de ingestión de datos mediante Azure Data Factory puede alcanzar hasta 1 GBps, y la latencia típica para las operaciones de movimiento de datos es del orden de minutos, lo que es altamente eficiente para el propósito del proyecto.

ADF puede ser desplegado y gestionado mediante scripts. Esto se realiza generalmente utilizando plantillas ARM (Azure Resource Manager) o el SDK de Python de Azure. Este nivel de automatización se alinea con el objetivo del proyecto de permitir un manejo programático de los datos.

Ingestión de Datos a la Capa Bronze

La capa Bronze actúa como el "landing zone" o zona de aterrizaje para los datos en su forma más cruda y sin procesar. ADF es el encargado de alimentar esta capa mediante pipelines de datos que extraen información de múltiples fuentes. Estos pipelines son fácilmente configurables y gestionables a través de la interfaz de usuario de ADF o mediante scripts y plantillas ARM (Azure Resource Manager).

El servicio tiene la capacidad de realizar copias de datos a granel y operaciones de transferencia de datos a alta velocidad entre servicios soportados, como bases de datos relacionales, bases de datos NoSQL, data lakes, entre otros. Así, la capa Bronze se alimenta de manera continua y confiable, cumpliendo con los requisitos iniciales del proyecto de proporcionar una base sólida para las fases de transformación de datos posteriores.

Una de las principales fortalezas de ADF es su capacidad para ingerir datos de múltiples fuentes con coherencia y eficacia. Esto incluye desde bases de datos tradicionales como SQL Server, hasta fuentes más modernas como eventos de streaming en Kafka, pasando por archivos en plataformas de almacenamiento en la nube como Azure Blob Storage y Azure Data Lake Storage.

Gracias a su rica biblioteca de conectores, ADF permite unificar diferentes tipos de datos en la capa Bronze. Este hecho es crucial para el proyecto, que tiene como objetivo proporcionar una gestión de datos holística y escalable.

Además, Azure Data Factory ofrece funcionalidades como Data Flow y Mapping Data Flow, permitiendo no solo la ingestión sino también la capacidad de realizar transformaciones de datos complejas en el pipeline, lo cual puede ser especialmente útil para las etapas posteriores de transformación de datos en las capas Silver y Gold.

Costes ADF

Tabla 5. Tabla de coste Azure Data Factory V2 pay-as-you-go

Tipo	Precio de Azure Integration Runtime	Precio de un entorno de ejecución de integración autohospedado
Orquestación	€0,924 por 1.000 ejecuciones	€1,386 por 1.000 ejecuciones
Actividad de movimiento de datos	€0,231/DIU/hora	€0,093/hora
Actividad de canalización	€0,005/hora	€0,001847/hora
Actividad de canalización externa	€0,000231/hora	€0,000093/hora

Orquestación se refiere a ejecuciones de actividades, de desencadenadores y de depuración.

El uso de la actividad de copia para extraer datos de un centro de datos de Azure conlleva cargos adicionales por ancho de banda de red, que aparecerá en la factura como una línea de transferencia de datos de salida aparte.

Las actividades de canalización se ejecutan en el entorno de ejecución de integración. Entre las actividades de canalización, se encuentran las de búsqueda, obtención de metadatos, eliminación y las operaciones de esquema durante la creación (probar la conexión, examinar la lista de carpetas y la lista de tablas, obtener el esquema y ver una vista previa de los datos).

Las actividades de canalización externas se administran en el entorno de ejecución de integración, pero se ejecutan en servicios vinculados. Entre las numerosas actividades externas, se incluyen las de Databricks, procedimientos almacenados y HDInsight.

2.2.4 Databricks cómo Motor de Procesamiento de Datos

Apache Spark es un motor de procesamiento de datos en clúster que proporciona una serie de bibliotecas para el procesamiento de datos distribuido. Databricks es una implementación de Apache Spark que optimiza varios aspectos del motor original, ofreciendo una plataforma unificada para ingeniería de datos, análisis y aprendizaje automático. Databricks es especialmente relevante para este proyecto, ya que funciona como el motor de procesamiento de datos encargado de la transformación y manipulación de datos entre las capas Bronze, Silver y Gold.

Databricks proporciona un entorno que permite el procesamiento de datos a gran escala, lo cual es crucial para cumplir con los requisitos de escalabilidad del proyecto. Además, el soporte nativo para Delta Lake en Databricks permite la fácil manipulación de grandes conjuntos de datos mientras se mantienen características esenciales como la ACID compliance (Atomicity, Consistency, Isolation, Durability).

Databricks emplea notebooks que pueden programar tareas de transformación en diferentes lenguajes (Python, Scala, SQL), y que son fácilmente orquestados utilizando herramientas como Apache Airflow. Estos notebooks pueden realizar operaciones complejas como joins, agregaciones y transformaciones de columnas en datos almacenados en la capa Bronze, y luego moverlos a las capas Silver y Gold según corresponda.

Para lograr una gestión de datos holística y escalable, en este proyecto se proponen dos workspaces de Databricks distintos, diseñados para cumplir con diferentes aspectos de la transformación y preparación de datos:

En el contexto de este proyecto, los distintos workspace de Databricks se usarán respectivamente para las operaciones que involucran transformaciones de la capa Bronze a la capa Silver, y de la capa Silver a la capa Gold.

Las ventajas de separar las tareas de transformación en dos workspaces son las siguientes:

Separación de Preocupaciones: Usar dos workspaces diferentes garantiza una clara división de responsabilidades. El primer workspace se centra en operaciones de limpieza de datos, desnormalización y transformaciones iniciales necesarias para pasar de Bronze a Silver. El segundo workspace se enfoca en optimizaciones más avanzadas, como transformaciones para análisis, agregaciones y generación de vistas materializadas, esenciales para mover los datos de Silver a Gold.

Seguridad y Gobernanza: Mantener workspaces distintos permite un nivel más granular de control de acceso. Los ingenieros de datos que trabajan en las transformaciones iniciales pueden no necesitar acceso a las transformaciones más sensibles o complejas realizadas en la capa Silver a Gold.

Optimización de Recursos: Diferentes workspaces pueden ser configurados con diferentes tamaños y tipos de clústeres para adaptarse mejor a las necesidades de computación específicas de cada etapa de transformación. Por ejemplo, las transformaciones de

Bronze a Silver podrían requerir una mayor cantidad de nodos de memoria, mientras que las transformaciones de Silver a Gold podrían ser más intensivas en CPU.

Trazabilidad y Auditoría: Al separar las transformaciones en diferentes workspaces, se facilita el proceso de auditoría y diagnóstico en caso de errores o problemas de rendimiento. Cada workspace puede tener sus propios logs, métricas y alertas, lo que simplifica el monitoreo y la resolución de problemas.

2.2.4.1 Descripción de los Dos Workspaces de Databricks

1. Workspace 1: Transformación de Datos de Bronze a Silver

Este workspace está destinado específicamente para manipular los datos brutos en la capa Bronze y transformarlos en datos más limpios, estructurados y enriquecidos en la capa Silver. Las operaciones en este workspace se enfocan principalmente en:

- Limpieza de datos: Eliminación de duplicados, manejo de valores nulos o faltantes, etc.
- Desnormalización: Combinar diferentes conjuntos de datos para crear estructuras más amplias y útiles.
- Validación: Aseguramiento de la calidad de los datos mediante reglas de validación.
- Transformación inicial: Operaciones como la conversión de tipos de datos, codificación y decodificación, entre otras.

Configuración del Cluster

Para este workspace, se podría emplear un clúster con alta memoria, dado que las operaciones de limpieza y transformación pueden ser intensivas en memoria. Podemos configurar el clúster con nodos de tipo `D12 v2` (o mayor) en Azure, que ofrecen una buena relación costo-eficiencia para tareas que requieren alta memoria.

Tabla 6. Tabla de coste Azure Databricks con clúster optimizado para memoria

Instance	vCPU(s)	RAM	DBU Count	DBU Price	Pay As You Go Total Price
D12 v2	4	28.00 GiB	1.00	€0.278/hour	€0.628/hour
D13 v2	8	56.00 GiB	2.00	€0.555/hour	€1.255/hour
D14 v2	16	112.00 GiB	4.00	€1.109/hour	€2.510/hour
D15 v2	20	140.00 GiB	5.00	€1.386/hour	€3.137/hour

2. Workspace 2: Transformación de Datos de Silver a Gold

Este workspace tiene como objetivo tomar los datos ya limpios y estructurados de la capa Silver y transformarlos en un formato óptimo para análisis y consultas en la capa Gold. Este workspace se centra en:

- Agregación: Sumarización de datos para análisis de alto nivel.
- Generación de vistas materializadas: Creación de tablas que almacenan resultados de consultas para mejorar la eficiencia.
- Enriquecimiento: Unión de diferentes fuentes de datos para crear conjuntos de datos más informativos.
- Indexación y particionado: Optimización para acelerar las consultas en la capa Gold.

Configuración del Cluster

Para las tareas de agregación y optimización, un clúster con un alto número de CPUs puede ser más apropiado. En este contexto, podríamos optar por nodos de tipo `SFs_v2` en Azure, que proporcionan una alta capacidad de CPU.

Tabla 7. Tabla de coste Azure Databricks con clúster optimizado para cómputo

Instance	vCPU(s)	RAM	DBU Count	DBU Price	Pay As You Go Total Price
F4s v2	4	8.00 GiB	0.75	€0.208/hour	€0.387/hour

F8s v2	8	16.00 GiB	1.50	€0.416 /hour	€0.774 /hour
F16s v2	16	32.00 GiB	3.00	€0.832 /hour	€1.548 /hour
F32s v2	32	64.00 GiB	6.00	€1.663 /hour	€3.096 /hour
F64s v2	64	128.00 GiB	12.00	€3.325 /hour	€6.191 /hour

2.2.5 Apache Airflow para la Orquestación

Apache Airflow es una plataforma de código abierto que permite a los desarrolladores describir, planificar, programar y monitorizar flujos de trabajo mediante programación en Python. Estos flujos de trabajo, denominados DAGs (Directed Acyclic Graphs), describen cómo se ejecutan las tareas, el orden de ejecución y las dependencias entre ellas. Fue desarrollado inicialmente por Airbnb, Apache Airflow ha ganado relevancia en la industria debido a su escalabilidad, rica interfaz de usuario y robustez en la definición de pipelines de datos.

Uno de los aspectos más destacados de Apache Airflow es su capacidad para integrarse con una amplia variedad de sistemas, tanto en la nube como fuera de ella. Esto es especialmente relevante para proyectos que requieren un alto grado de interoperabilidad y flexibilidad en sus flujos de trabajo.

2.2.5.1 Integración con Databricks

Apache Airflow se integra de forma transparente con Azure Databricks a través de la Databricks Airflow Operator. Esto permite desencadenar trabajos en Databricks al alcanzar ciertos nodos en un DAG de Airflow. Las ventajas son claras: un control más granular sobre cuándo y cómo se inician las transformaciones de datos, la capacidad de reintentar o escalar automáticamente tareas y la posibilidad de parametrizar los trabajos, lo que aporta una mayor flexibilidad y reutilización del código. Estas capacidades de orquestación efectiva son críticas cuando se trata de transformaciones de datos que deben ejecutarse en un orden específico o que tienen dependencias complejas, como es el caso del flujo de datos desde la capa Bronze a la capa Gold.

A pesar de su integración nativa con servicios de Azure, Apache Airflow tiene la ventaja de ser una herramienta agnóstica a la nube. Esto significa que podría orquestar flujos de trabajo que interactúan con múltiples plataformas y servicios, no sólo los alojados en Azure. Por ejemplo, podría coordinar flujos de trabajo que implican bases de datos on-premise, almacenes de datos en otras nubes públicas o incluso servicios de terceros para el análisis de datos. Esta agnosticidad a la nube provee un nivel adicional de flexibilidad que es invaluable para organizaciones que operan en entornos híbridos o multicloud. Además, al ser una solución de código abierto, Airflow no incurre en costos de licencia, lo que podría ser una consideración económica significativa a medida que el proyecto escala.

Un aspecto técnico digno de mención es que Apache Airflow permite la orquestación distribuida. Esto significa que Airflow puede coordinar tareas que se ejecutan en un conjunto distribuido de trabajadores, lo cual es altamente compatible con el paralelismo inherente en operaciones de big data. Tal capacidad es vital para garantizar que los trabajos de transformación de datos se completen en marcos de tiempo aceptables, incluso cuando los volúmenes de datos son significativos.

Costes

Apache Airflow es una solución de código abierto y no tiene asociados costes de licencia o uso. El coste asociado a este elemento se basará en la máquina dónde se ejecute el programa y las tareas de Airflow.

2.2.6 Comunicación entre los componentes

La cohesión entre los distintos componentes de la arquitectura propuesta no sólo garantiza una gestión fluida y coherente de los flujos de datos, sino que también posibilita el despliegue automatizado y la escalabilidad de la infraestructura. En esta sección, se describirá la interconexión técnica y la comunicación entre cada elemento, subrayando cómo se establecen estos enlaces y por qué son cruciales para el éxito del proyecto.

2.2.6.1 Protocolos y APIs

Las conexiones entre los componentes se establecen predominantemente a través de interfaces de programación de aplicaciones (APIs) robustas. Estas APIs permiten que las soluciones, como Azure Data Factory o Databricks, se integren sin problemas con servicios externos y herramientas, como Apache Airflow. La utilización de APIs RESTful, en particular, garantiza una comunicación estandarizada y segura entre estos componentes.

2.2.6.2 Azure Data Factory y Azure Blob Storage

La comunicación entre Azure Data Factory y Azure Blob Storage es fundamental para el flujo de datos en esta arquitectura, particularmente en la etapa inicial de ingestión de datos. Este mecanismo de comunicación se orquesta mediante una serie de interacciones que aprovechan tanto las APIs como los SDKs proporcionados por Azure, estableciendo un flujo de trabajo altamente optimizado y seguro. Al ser dos componentes del ecosistema de Azure, se puede aprovechar de sistemas nativos de autenticación que facilitan la conexión entre ambos elementos.

Es crucial destacar que toda la comunicación entre Azure Data Factory y Azure Blob Storage se realiza a través de una conexión segura TLS/SSL. Además, los datos pueden ser cifrados en tránsito y en reposo para proporcionar una capa adicional de seguridad.

Azure Data Factory requiere permisos para leer o escribir en los contenedores de Azure Blob Storage. Esto se logra utilizando Managed Identity o Shared Access Signature (SAS) tokens, que garantizan que sólo las entidades autorizadas tengan acceso a los recursos adecuados.

2.2.6.3 Azure Blob Storage y Databricks

El Azure Blob Storage actúa como un repositorio centralizado para los datos en las capas Bronze, Silver y Gold. Databricks, al tener una integración nativa con Azure, puede acceder fácilmente a estos datos para realizar operaciones de lectura y escritura. Esta integración se realiza utilizando la biblioteca DBFS (Databricks File System) que permite montar contenedores de blob y acceder a ellos como si fueran sistemas de archivos locales dentro de los notebooks de Databricks.

Esto se consigue mediante la creación de external storage locations y credenciales dentro del Databricks Unity Catalog.

2.2.6.4 Apache Airflow y Databricks

La comunicación entre Apache Airflow y Databricks es esencial para la automatización de los flujos de trabajo. El flujo de conexión tiene los siguientes componentes:

Uso de Operadores y Hooks

Apache Airflow incluye un conjunto de operadores específicos para Databricks, como *DatabricksSubmitRunOperator* y *DatabricksRunNowOperator*, que facilitan la interacción con la API de Databricks. Estos operadores se basan en "hooks" subyacentes, que son interfaces para conectarse a plataformas externas. En este caso, el *DatabricksHook* se utiliza para encapsular la lógica de la API de Databricks, asegurando que las credenciales, tokens y otros aspectos de la autenticación se manejen de manera segura y eficiente.

Flujo de Trabajo

Cuando un DAG (Directed Acyclic Graph) en Apache Airflow es activado, los operadores de Databricks inician trabajos de transformación en el workspace de Databricks correspondiente. Para ser más específicos, el *DatabricksSubmitRunOperator* crea un nuevo 'run' a partir de una tarea predefinida en Databricks. Este operador utiliza el endpoint de la API de Databricks para enviar una solicitud que contiene todas las especificaciones del trabajo, como el notebook a ejecutar, los parámetros a pasar y las configuraciones del clúster.

Monitorización y Registro

Después de que se lanza un trabajo, Apache Airflow monitoriza su estado mediante el método GET de la API de Databricks. Si el trabajo falla por alguna razón, Apache Airflow tiene la capacidad de reintentar el trabajo según la política de reintentos configurada. Además, los logs y métricas generados durante la ejecución del trabajo se capturan y se pueden visualizar dentro de la interfaz de usuario de Apache Airflow para facilitar la depuración y el análisis del rendimiento.

Seguridad y Control de Acceso

Es importante señalar que toda la comunicación entre Apache Airflow y Databricks se realiza a través de una conexión segura (HTTPS), y se utiliza la autenticación basada en tokens para garantizar que sólo las entidades autorizadas puedan iniciar o modificar trabajos.

2.2.6.5 Infraestructura cómo Código (IaC)

Para lograr un despliegue automatizado de la infraestructura, es esencial adoptar prácticas de Infraestructura como Código. Herramientas como Azure ARM templates permiten describir y provisionar recursos de la nube de manera declarativa. Esto significa que se puede definir la estructura deseada de la infraestructura en archivos de código, y estas herramientas se encargan de crear o modificar los recursos para que coincidan con esa estructura. Esto garantiza que la interconexión entre componentes se establezca de manera coherente y reproducible en diferentes entornos o etapas del proyecto.

El diseño de esta interconexión satisface la necesidad de una infraestructura end-to-end por varias razones clave. Primero, permite una automatización completa desde la ingestión hasta el almacenamiento y la transformación de datos, cumpliendo con los objetivos de escalabilidad y mantenibilidad. Segundo, al tener todos los componentes interconectados, se reduce la latencia en el flujo de datos y se mejora la eficiencia global del sistema. Finalmente, este diseño soporta una alta disponibilidad y recuperación ante desastres, ya que cada componente ha sido seleccionado y configurado para garantizar un funcionamiento robusto.

2.3 Interconexión y definición del despliegue automático

El despliegue automático representa un paradigma emergente en la ingeniería de datos que aborda la complejidad intrínseca de los ecosistemas de Big Data. En esta metodología, los scripts o ejecutables se diseñan para instalar, configurar y conectar componentes de software de forma autónoma, minimizando la intervención humana y, por ende, reduciendo el margen de error. En un entorno de Big Data, donde el volumen, la velocidad y la variedad de los datos son inherentemente elevados, la automatización del despliegue no solo acelera la implementación de la arquitectura sino que también garantiza un grado más alto de coherencia, seguridad y eficacia operacional.

En el contexto del presente proyecto, el despliegue automático se erige como una piedra angular, dado que aborda una brecha significativa en el mercado: la ausencia de soluciones que ofrezcan una arquitectura de datos end-to-end totalmente automatizada. Esta carencia en el mercado actual impide que las organizaciones desplieguen infraestructuras de Big Data de manera rápida y segura, limitando así su capacidad para aprovechar al máximo los beneficios del análisis de datos a gran escala.

El despliegue automático en este proyecto tiene como objetivo facilitar la rápida instauración de una arquitectura de datos robusta, compuesta por múltiples elementos descritos en el apartado anterior: Azure Blob Storage, Azure Data Factory, Databricks y Apache Airflow. Mediante la automatización de la instanciación y la configuración de estos componentes, el proyecto busca ofrecer una solución que no solo sea replicable y escalable, sino también segura y compatible con los estándares actuales de gobernanza de datos.

Este apartado tiene como objetivo describir cómo los componentes individuales del ecosistema de datos se interconectan y comunican entre sí. Además, se detallará la estrategia para automatizar el despliegue de la infraestructura a través de scripts o ejecutables.

2.3.1 Principios de diseño del ejecutable

En el diseño y desarrollo de cualquier sistema o solución, los principios que guían su creación son esenciales para garantizar que cumple con las expectativas y necesidades del proyecto. Para el script o ejecutable que se propone en este proyecto, se han establecido una serie de principios basados en la funcionalidad, la integridad, la eficiencia y la seguridad. Estos principios, detallados a continuación, buscan crear un marco que permita un despliegue automático eficaz y seguro de la infraestructura de Big Data.

2.3.1.1 Unicidad de Ejecución

Dado que el script está diseñado para ser ejecutado una sola vez al inicio, su construcción garantiza que no haya efectos secundarios ni ejecuciones múltiples. Esto evita la sobresaturación de recursos y garantiza la idempotencia del proceso.

2.3.1.2 Idempotencia

El script está diseñado para ser idempotente, es decir, su ejecución múltiple produce el mismo resultado que su ejecución única. Esto garantiza la integridad de la arquitectura, permitiendo que el script pueda ser reejecutado en caso de fallos parciales sin efectos secundarios adversos.

2.3.1.3 Atomicidad

Cada operación dentro del script es atómica, asegurando que se complete en su totalidad o se revierta en caso de fallo. Esta característica es crucial para mantener la coherencia del sistema, especialmente cuando se manipulan componentes interdependientes como en una arquitectura de Big Data.

2.3.1.4 Complejidad Oculta

Aunque la infraestructura subyacente y los procesos son complejos, el script se diseñó para ocultar esta complejidad al usuario. Esto permite a las empresas centrarse en el uso de los componentes desplegados sin tener que preocuparse por los detalles técnicos de la implementación.

2.3.1.5 Eficiencia en el Despliegue

El script optimiza el orden y la manera en que se despliegan los componentes, garantizando que los recursos se utilicen de manera eficiente y que los tiempos de despliegue sean lo más cortos posible.

2.3.1.6 Modularidad

Aunque el script se ejecuta como una entidad única, su diseño subyacente es modular. Esto permite que, en futuras iteraciones o adaptaciones, se puedan añadir, modificar o eliminar componentes de manera sencilla sin afectar al funcionamiento global.

2.3.1.7 Despliegue de Apache Airflow en Docker

Una de las decisiones técnicas clave es el uso de contenedores Docker para desplegar Apache Airflow. Esta aproximación garantiza una mayor portabilidad y facilita el escalado horizontal, permitiendo que el sistema de orquestación se adapte de manera eficiente a las demandas de carga variable. Además, la encapsulación en un contenedor Docker añade una capa extra de seguridad al aislar el entorno de ejecución.

2.3.1.8 Uso de Plantillas ARM para Componentes Azure:

Para asegurar un despliegue coherente y reproducible de los componentes en Azure, se utilizarán Plantillas ARM (Azure Resource Manager). Estas plantillas permiten definir la infraestructura como código, lo cual no sólo mejora la eficiencia del despliegue sino que también posibilita un control más estricto sobre las configuraciones, ayudando a mantener altos niveles de seguridad y cumplimiento.

2.3.2 Componentes claves para la interconexión

Para lograr una interconexión eficaz y coherente entre los distintos elementos de la arquitectura, se requiere de varios componentes y tecnologías específicas que actúan como puentes y facilitadores. Estos componentes son esenciales para garantizar que la información fluya de manera segura, eficiente y confiable entre las distintas capas y servicios del ecosistema.

La manera exacta cómo estos elementos se despliegan se expone más en profundidad en el Capítulo 3.

2.3.2.1 Control de Acceso Basado en Identidad (IAM) y Managed Identities en Azure

El Control de Acceso Basado en Identidad es un marco de políticas y tecnologías para garantizar que el acceso apropiado a los recursos en una red sea proporcionado a las personas o sistemas que lo requieran. IAM va más allá del simple control de acceso, involucrando múltiples características como la autenticación, la autorización, los roles y las políticas para proporcionar un control de acceso robusto y granular. En el contexto de Azure, el IAM permite la administración de identidades y permisos para los recursos y servicios de Azure, mediante Azure Active Directory (Azure AD).

Las Managed Identities en Azure son una característica de Azure AD que simplifica la administración de identidades para aplicaciones que requieren un acceso seguro a los recursos de Azure. En lugar de manejar credenciales como claves de acceso, tokens o certificados, se utiliza una identidad gestionada. Estas identidades son manejadas completamente por Azure AD y son seguras, ya que no hay necesidad de que el usuario gestione ni almacene secretos de forma explícita.

Una identidad gestionada se asocia con un recurso de Azure (como una instancia de Azure VM, una aplicación web o una instancia de Azure Data Factory) y se utiliza para autenticar o autorizar dicho recurso cuando accede a otro recurso de Azure, como al Azure Blob.

- **System Assigned:** Esta identidad está ligada al ciclo de vida del recurso de Azure al cual está asociada. Cuando el recurso se elimina, Azure elimina automáticamente la identidad.

- **User Assigned:** Esta identidad se crea como un recurso de Azure separado y se asocia manualmente a los recursos que la requieren. Esta identidad puede ser reutilizada y no está ligada al ciclo de vida de los recursos a los que está asignada.

Para lograr una autenticación y autorización seguras entre Azure Data Factory y Azure Blob Storage, se utilizarán políticas IAM (Identity and Access Management) y Managed Identities. Esto permitirá que Azure Data Factory tenga los permisos necesarios para leer y escribir en Azure Blob Storage, manteniendo al mismo tiempo una alta granularidad en las políticas de acceso.

En este caso, se creará una managed identity del componente Azure Datafactory y se le asignarán permisos a la cuenta de storage en la layer bronce (*landing*).

Las plantillas ARM son archivos JSON que definen los recursos que se necesitan desplegar para una solución. Estas plantillas son esenciales para la infraestructura como código (IaC) en Azure. Mediante las plantillas ARM, el script desplegará y configurará Azure Blob Storage, Azure Data Factory, y otros recursos necesarios en Azure.

Se utilizan para asegurar que cada recurso se configure con las especificaciones adecuadas, como el Access connector para Databricks

2.3.2.2 Docker compose para Apache Airflow

Docker Compose es una herramienta de orquestación que facilita la definición y ejecución de aplicaciones multi-contenedor en Docker. Utiliza archivos de configuración en formato YAML para especificar los servicios, redes y volúmenes de una aplicación, permitiendo que todo el ecosistema de la aplicación pueda ser desplegado y administrado de una manera estructurada y coherente.

Desde un punto de vista técnico, Docker Compose maneja el ciclo de vida del contenedor de manera integral. En lugar de tener que construir, configurar y ejecutar contenedores de forma individual mediante comandos de Docker, con Docker Compose es posible hacer todo esto con un único comando (`docker-compose up`), que lee el archivo de configuración y realiza las operaciones necesarias.

Características técnicas:

Definición de Servicios: Los servicios se definen en el archivo **docker-compose.yml**. Cada servicio representa un contenedor y se puede configurar para usar una imagen específica, puertos, variables de entorno, volúmenes, y otros.

Orquestación: Docker Compose permite iniciar, detener y escalar servicios definidos en el archivo **docker-compose.yml** de forma conjunta o individual.

Networking: Facilita la interconexión de contenedores mediante redes definidas por el usuario, permitiendo una comunicación más segura y eficiente entre los componentes.

Volúmenes y Datos Persistentes: Los volúmenes se pueden definir en el archivo de configuración para gestionar datos persistentes.

Variables de entorno: Se pueden especificar en el archivo de configuración o en archivos `.env` separados, permitiendo una configuración más dinámica.

En el contexto de Apache Airflow, Docker Compose será utilizado para orquestar y gestionar los contenedores necesarios para la operación del sistema de orquestación, incluyendo el servidor web, el planificador y la base de datos.

2.3.3 Secuencia de despliegue

La secuencia de despliegue se realiza de manera estratégica, siguiendo una estructura planificada que se agrupa por grupos de recursos en Azure. Esta secuencia es crucial para la integridad, eficacia y seguridad de toda la infraestructura.

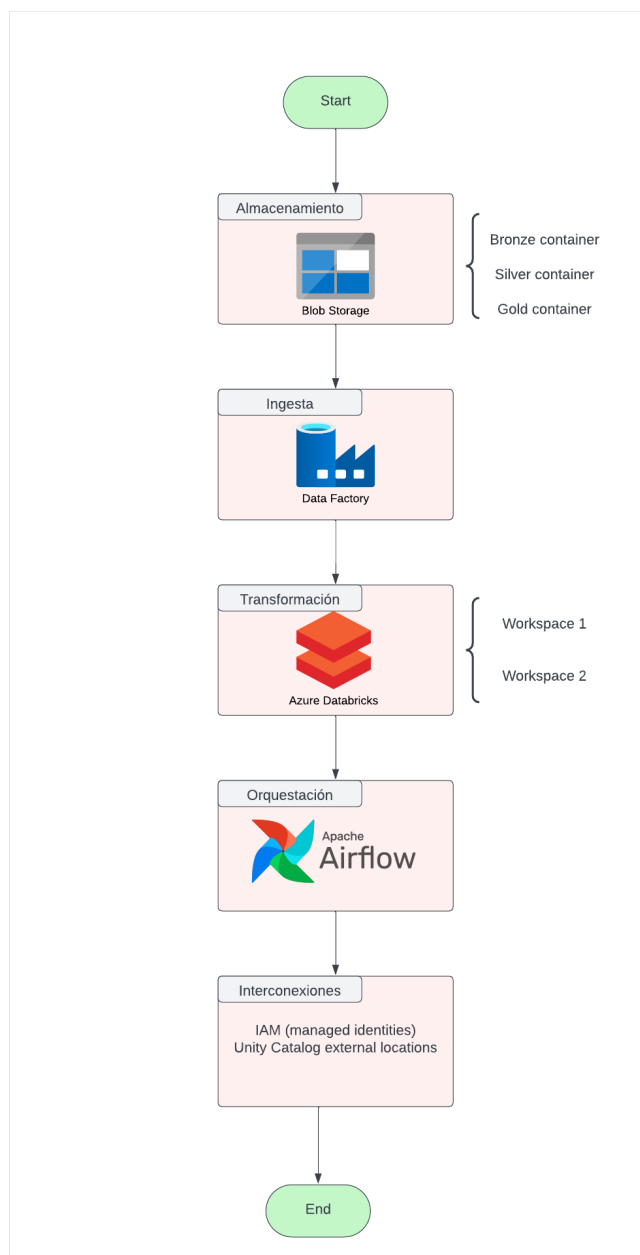


Figura 5. Orden de ejecución del proceso de despliegue

Grupo de Recursos 1: Almacenamiento de Datos

1. Azure Blob Storage

Despliegue de una Storage Account en Azure, con tres contenedores distintos en tipo Azure Datalake Storage Gen2 cada una correspondiendo a una capa específica: Bronze, Silver y Gold.

Creación de un contenedor en cada Storage Account para alojar los datos de las respectivas capas.

Grupo de Recursos 2: Ingesta de Datos

1. Azure Data Factory

Implementación de Azure Data Factory para la ingestión de datos desde múltiples fuentes externas.

Configuración de Data Pipelines para transferir datos al contenedor Bronze en Azure Blob Storage.

Grupo de Recursos 3: Transformación de Datos

1. Azure Databricks

Despliegue de dos workspaces de Databricks: uno destinado para las transformaciones de Bronze a Silver, y otro para las transformaciones de Silver a Gold.

Configuración del Access Connector para vincular cada workspace con los contenedores correspondientes en Azure Blob Storage.

Grupo de Recursos 4: Orquestación de Datos

1. Apache Airflow en Docker

Despliegue de Apache Airflow utilizando Docker Compose.

Configuración para la interconexión con Azure Databricks para lanzar trabajos de transformación de datos.

Grupo de Recursos 5: Seguridad y Gobernanza

1. IAM (Identidad y Gestión de Accesos)

Implementación de Managed Identities para la autenticación y autorización entre Azure Data Factory, Azure Databricks y Azure Blob Storage.

2. Unity Catalog en Databricks

Establecimiento del Unity Catalog para la gobernanza de datos, actuando como la piedra angular para la administración de metadatos y la seguridad.

El script o ejecutable de despliegue utilizará diversas tecnologías como Plantillas ARM para Azure, APIs REST, y Docker Compose para automatizar esta secuencia. Cada componente se configura y se interconecta de forma automática, asegurando que el sistema esté listo para operaciones de datos end-to-end al finalizar el despliegue. Esta secuencia garantiza un proceso de implementación estructurado que reduce las dependencias y facilita la gestión.

2.3.3.1 Justificación secuencia de despliegue

La justificación del orden de despliegue se basa en una serie de factores críticos que incluyen las dependencias entre componentes, los requisitos de inicialización y configuración, y las necesidades de seguridad y gobernanza.

1. Almacenamiento de Datos

- Se inicia con el despliegue del Azure Blob Storage porque es la base sobre la cual se apoyarán todos los demás servicios para almacenar y recuperar datos. Es crucial tener los contenedores de almacenamiento preparados primero para asegurar que las siguientes etapas tengan un lugar donde ingresar o transformar datos.

2. Ingesta de Datos

- Azure Data Factory se implementa a continuación para llenar el contenedor Bronze en Azure Blob Storage. La ingestión de datos debe ser el primer proceso operativo una vez que el almacenamiento está en su lugar. Además, este paso establece las bases para las transformaciones de datos subsiguientes.

3. Transformación de Datos

- Los workspaces de Azure Databricks se despliegan para las transformaciones de datos porque son dependientes de la existencia de datos en la capa Bronze. Además, se necesitan para preparar datos para las capas Silver y Gold, que son contenedores previamente configurados en el Azure Blob Storage.

4. Orquestación de Datos

- Apache Airflow se despliega después de la configuración de Databricks porque se encargará de la orquestación de los trabajos de transformación de datos que se ejecutarán en Databricks.

5. Seguridad y Gobernanza

- Finalmente, las características de seguridad y gobernanza como IAM y Unity Catalog se implementan. Estos componentes son cruciales para asegurar que sólo los servicios y personas autorizados tengan acceso a los recursos y datos apropiados. Pero como estos controles de acceso suelen ser transversales y dependen de la existencia previa de otros recursos, se configuran al final. Esta parte no es un grupo de recursos sino una serie de configuraciones que se aplican sobre los componentes ya desplegados.

2.4 Seguridad y Gobernanza de datos

La seguridad y la gobernanza de datos constituyen piedras angulares en la arquitectura global de este proyecto, alineándose estrechamente con los objetivos de proporcionar una gestión de datos holística, escalable y segura. El componente clave para la gestión eficaz de la gobernanza de datos en este entorno es el Unity Catalog de Databricks, que sirve como un repositorio unificado para la metainformación y el linaje de datos a través de las distintas capas de almacenamiento: Bronze, Silver y Gold.

2.4.1 Unity Catalog

El Unity Catalog en Databricks actúa como el núcleo central de la gobernanza y la administración de metadatos en nuestra arquitectura de datos, proporcionando un marco unificado para la gestión de todos los recursos de datos y metadatos que residen en las capas Bronze, Silver y Gold almacenadas en Azure Blob Storage. En términos técnicos, Unity Catalog se basa en un esquema de metadatos extensible que permite la indexación, la clasificación y el etiquetado de datasets, lo cual es crucial para las operaciones de búsqueda, descubrimiento y linaje de datos.

Unity Catalog genera automáticamente y mantiene un sistema de metadatos enriquecido que incluye detalles como la estructura de los datos, tipos de datos, transformaciones aplicadas, y otros atributos clave. Esto es especialmente relevante para cumplir con normativas de cumplimiento como GDPR, ya que el linaje de datos permite rastrear el flujo y las transformaciones que se aplican a cada fragmento de datos desde su origen hasta su estado final.

Uno de los aspectos más importantes de Unity Catalog es su capacidad para integrarse estrechamente con otros componentes de la arquitectura, incluidos Azure Data Factory y Apache Airflow. Por ejemplo, cuando Azure Data Factory ingiere datos en la capa Bronze, los metadatos asociados con esa ingestión son automáticamente indexados y almacenados en Unity Catalog. Esto facilita la administración de la calidad de los datos y la ejecución de trabajos de transformación de datos posteriores que pueden ser automatizados y orquestados mediante Apache Airflow.

Unity Catalog incorpora un mecanismo de Control de Acceso Basado en Roles (RBAC) que permite una gestión más granular de permisos. Se pueden asignar roles y permisos específicos para diferentes tipos de usuarios en función de sus responsabilidades en el ciclo de vida de los datos, lo que añade una capa adicional de seguridad y reduce el riesgo de manipulación o exposición de datos no autorizados.

2.4.2 Seguridad en la transmisión y almacenamiento de datos

La seguridad de la transmisión de datos está asegurada mediante el uso de protocolos seguros como TLS/SSL, tanto en la fase de ingestión de datos con Azure Data Factory como en las transformaciones realizadas a través de Databricks. Adicionalmente, se aplican políticas de cifrado en reposo en Azure Blob Storage, añadiendo una capa de seguridad adicional.

2.4.3 Control de Acceso Basado en Roles

Para administrar eficientemente quién tiene acceso a qué recursos y en qué capacidad, se implementa el Control de Acceso Basado en Roles (RBAC) en todos los componentes de la arquitectura. En el contexto de Databricks y su Unity Catalog, esto significa que los usuarios sólo pueden acceder o modificar los datasets y notebooks a los que se les ha dado permiso explícito, reduciendo así el riesgo de acceso no autorizado o manipulación de datos.

2.5 Coste-Efectividad de la solución

La coste-efectividad de la arquitectura propuesta se analiza considerando tanto los costes iniciales de implementación como los costes operativos recurrentes, incluyendo almacenamiento, cómputo y transferencia de datos. El enfoque es coherente con el objetivo estratégico del proyecto de optimizar la eficiencia financiera sin comprometer la calidad, la seguridad y la escalabilidad de la solución.

2.5.1 Costes de implementación

Son los costos asociados al despliegue y servicio base de los componentes. El hecho de usar la mayoría de servicios en la nube pública como SaaS permite no tener que hacer ninguna inversión grande inicial de hardware o costes de licencias.

Azure Blob Storage: En términos de almacenamiento, Azure Blob Storage opera con un modelo de coste basado en la capacidad utilizada y las operaciones de lectura/escritura. Al utilizar tres contenedores separados para las capas Bronze, Silver y Gold, se optimiza la eficiencia del almacenamiento, ya que cada contenedor se ajusta a diferentes niveles de rendimiento y coste.

Azure Data Factory: Para la ingestión de datos, Azure Data Factory tiene un modelo de facturación basado en el rendimiento del servicio y el volumen de datos transferidos. Con las capacidades de optimización de tasa de transferencia de Data Factory, se reduce la necesidad de recursos de cómputo adicionales, disminuyendo así los costes.

Databricks: En cuanto al procesamiento de datos, Databricks tiene un modelo de precios basado en unidades Databricks (DBU). La utilización de dos workspaces distintos para las transformaciones de Bronze a Silver y Silver a Gold permite una asignación más eficiente de los recursos de cómputo.

Apache Airflow: Este componente se puede ejecutar tanto en un entorno en la nube como en un entorno local. Al estar fuera del entorno de Azure, la escalabilidad y la eficiencia del coste pueden mejorarse mediante técnicas de planificación de tareas.

2.5.2 Costes operativos recurrentes

Son aquellos costes asociados al uso de los distintos servicios. Podemos distinguir:

Almacenamiento y Acceso: A medida que los datos crecen, también lo hacen los costes asociados con Azure Blob Storage. Sin embargo, la utilización de políticas de ciclo de vida y la automatización de la purga de datos obsoletos o innecesarios puede mantener estos costes bajo control.

Procesamiento y Transformación: La optimización continua de los trabajos de Databricks y la monitorización de los recursos consumidos pueden ayudar a mantener un equilibrio entre el rendimiento y el coste.

Orquestación y Automatización: Mantener Apache Airflow actualizado y optimizar los DAGs para garantizar que se ejecuten en tiempos óptimos contribuye a una gestión coste-efectiva del flujo de trabajo.

La arquitectura propuesta, a pesar de su naturaleza compleja y multifacética, ha sido diseñada con la eficiencia en mente. Utilizando servicios basados en el consumo y herramientas optimizables, la solución permite un ajuste granular de los costes sin comprometer el rendimiento o la funcionalidad. Es fundamental tener en cuenta que una inversión inicial bien gestionada en una arquitectura robusta puede resultar en ahorros a largo plazo, al reducir la necesidad de rediseños costosos o soluciones de contingencia imprevistas.

Para presentar datos concretos, la creación de todo el proyecto y el análisis y ejecución del Caso de Estudio, resultó en un coste total de 1.28€ de servicios, estableciendo así un baremo muy asequible de infraestructura.

2.6 Conclusión del capítulo

En el presente capítulo, se ha delineado la arquitectura propuesta para la gestión integral y escalable de datos, abarcando desde la capa de almacenamiento hasta la orquestación de flujos de trabajo. Se han seleccionado componentes tecnológicos clave, como Azure Blob Storage, Azure Data Factory, Databricks, y Apache Airflow, fundamentados en criterios técnicos y económicos alineados con los objetivos del proyecto.

El concepto de despliegue automático se ha erigido como un elemento fundamental de esta arquitectura. La justificación reside en su capacidad para agilizar la implementación de una infraestructura compleja, mitigar errores humanos y garantizar una configuración uniforme, factores críticos para cualquier organización que busca una gestión de datos eficiente y robusta. La estrategia de interconexión y las tecnologías asociadas, tales como IAM para el control de acceso y Docker para el contenedorización, desempeñan un papel crucial en la cohesión de esta arquitectura multicapa.

En el diseño del script o ejecutable para el despliegue, se han establecido varios principios y objetivos orientados a la eficiencia, seguridad, y robustez del sistema. Este diseño tiene en cuenta tanto la secuencia como las dependencias entre los componentes, estableciendo un orden lógico de despliegue que optimiza la interconexión y reduce los riesgos.

Este capítulo sienta las bases para el siguiente, que se enfocará en la metodología de desarrollo y la implementación de la solución propuesta. Allí se abordarán aspectos prácticos del despliegue automático, desde el ciclo de desarrollo hasta las pruebas en entornos controlados, proporcionando un análisis más profundo de cómo cada componente y mecanismo contribuye a una infraestructura de datos holística y escalable.

3 Capítulo 3: Desarrollo e implementación

Tras establecer en el capítulo anterior la arquitectura propuesta, sus componentes y las premisas de diseño, el capítulo 3 se concentra en el ámbito práctico del proyecto. Este enfoque operativo es esencial para trasladar las especificaciones teóricas a una implementación tangible que pueda ser evaluada y validada. El desarrollo de la solución no sólo requiere una comprensión técnica profunda, sino también una metodología rigurosa que garantice que los principios de diseño se traduzcan en código y configuraciones que funcionen como se espera.

En este capítulo, se desglosará el proceso adoptado para la creación del script o ejecutable, profundizando en las etapas, herramientas y prácticas adoptadas. Posteriormente, se examinarán los resultados preliminares obtenidos, proporcionando una visión inicial de la eficacia de la solución en entornos controlados y su alineación con los objetivos del proyecto.

3.1 Metodología

La metodología, por lo tanto, se enfoca en asegurar que el código y las configuraciones se adhieran a los principios de diseño previamente establecidos, con particular énfasis en la integridad, eficiencia y seguridad del sistema global. Este enfoque es intrínseco al cumplimiento de los objetivos del proyecto, los cuales priorizan la automatización como vehículo para el despliegue ágil y robusto de componentes interconectados.

3.1.1 Entorno de Desarrollo

Para el desarrollo del script o ejecutable se han seleccionado un conjunto de herramientas, lenguajes y frameworks en función de su adecuación a los requisitos técnicos, su capacidad de integración y su escalabilidad. A continuación, se describen los elementos clave del entorno de desarrollo:

3.1.1.1 Lenguaje de Programación para el despliegue de recursos en Azure: Python

La elección de Python como lenguaje de programación para desplegar recursos en Azure se fundamenta en varias consideraciones técnicas y estratégicas que se alinean estrechamente con los objetivos del proyecto. Primero y ante todo, Python ofrece una biblioteca estándar robusta y un ecosistema de paquetes de terceros que facilitan la interacción con APIs y servicios web. Este aspecto es crucial dado que el despliegue en Azure a menudo implica invocaciones de API para configurar y gestionar recursos.

Además, Python posee bibliotecas específicas como `azure-mgmt-resource` para interactuar directamente con Azure Resource Manager, lo que simplifica enormemente la tarea de

desplegar y administrar recursos en Azure. Este nivel de soporte nativo disminuye la curva de aprendizaje y acelera el tiempo de desarrollo, lo que es especialmente beneficioso en un entorno empresarial donde la eficiencia y la velocidad son cruciales.

Desde el punto de vista de la automatización, Python ofrece capacidades excepcionales para el scripting, lo cual es esencial para este proyecto, que se centra en el despliegue automático de una arquitectura end-to-end. Las bibliotecas de Python permiten una alta granularidad en la automatización, lo que facilita una configuración más detallada de los recursos, algo que se alinea directamente con los objetivos de integridad y seguridad del sistema global.

Otro factor relevante es la portabilidad del lenguaje. Dado que Python es interpretado, el script de despliegue se puede ejecutar en cualquier sistema que tenga instalado un intérprete de Python, incluidos los contenedores Docker, que son una parte fundamental de la arquitectura del proyecto.

Finalmente, la comunidad activa y el extenso soporte en línea para Python aseguran que cualquier desafío técnico o conceptual que surja durante el desarrollo pueda abordarse de manera efectiva, beneficiando así tanto la robustez como la sostenibilidad del proyecto.

Por lo tanto, Python no solo cumple con los requisitos técnicos para la interacción con Azure y la automatización del despliegue, sino que también ofrece ventajas estratégicas que optimizan el desarrollo, la escalabilidad y la sostenibilidad del proyecto.

3.1.1.2 Contenedores: Docker

Docker se utiliza para crear, desplegar y ejecutar aplicaciones en contenedores. En este proyecto, se crean dos contenedores Docker: uno para Apache Airflow y otro para la ejecución del script de Python que despliega los recursos en Azure. Docker garantiza que las aplicaciones se ejecuten de manera idéntica en diferentes entornos al empaquetar todas las dependencias necesarias.

3.1.1.3 ARM Templates

Las plantillas ARM (Azure Resource Manager) se utilizan para definir los recursos que se necesitan para la aplicación en Azure. Estas plantillas son invocadas por el script de Python para realizar el despliegue automatizado en Azure.

3.1.1.4 Gestión de la configuración: Variables de entorno

Las variables de entorno se utilizan para almacenar configuraciones que varían entre los entornos de despliegue. Esto incluye, las credenciales de acceso para la suscripción de Azure que se requieren para poder ejecutar el script de Python de despliegue. Esta metodología aumenta la seguridad del sistema al no almacenar información sensible en el código fuente.

3.1.1.5 Mecanismo de Ejecución: Docker Compose

Docker Compose se emplea para definir y ejecutar aplicaciones Docker de varios contenedores. En este caso, Docker Compose orquestará tanto el contenedor de Airflow como el contenedor de Python, permitiendo una inicialización coordinada y sincronizada de ambos en una sola llamada.

3.1.1.6 Repositorio de Código: Git

Git se utiliza para el control de versiones del código, permitiendo un desarrollo colaborativo y un seguimiento eficiente de las modificaciones.

El uso de Git en un proyecto de desarrollo de software individual no es menos importante que en un entorno de equipo, y se justifica por varias razones técnicas y metodológicas que alinean con los objetivos del proyecto. En primer lugar, Git ofrece un sistema de control de versiones robusto que permite realizar un seguimiento meticuloso de los cambios en el código fuente. Esto es especialmente útil en un proyecto complejo y multifacético como el despliegue automático de una arquitectura de Big Data en Azure. Cada cambio en el script o en los archivos de configuración puede ser documentado y revertido si es necesario, lo cual proporciona un nivel de robustez y trazabilidad que sería difícil de lograr de otro modo.

Además, Git facilita la implementación de prácticas de desarrollo de software como la integración continua y la entrega continua (CI/CD), incluso en un proyecto individual. A medida que el proyecto evoluciona, la capacidad de realizar pruebas automatizadas en cada "commit" asegura que los cambios no introduzcan errores o vulnerabilidades en el sistema. Este aspecto es crítico cuando se trata de desplegar y gestionar recursos en un entorno de nube, donde errores menores pueden tener implicaciones significativas en términos de costos y seguridad.

Git también ofrece facilidades para la documentación. Los mensajes de commit, bien estructurados, actúan como un registro en tiempo real del progreso del proyecto, lo que podría ser invaluable para futuras etapas de desarrollo o para la revisión del proyecto.

Por último, aunque el proyecto sea individual, la realidad empresarial podría requerir que más desarrolladores colaboren en el futuro. En ese contexto, haber utilizado Git desde el principio simplifica enormemente la incorporación de nuevos miembros al proyecto, permitiendo una transición más suave y efectiva hacia un enfoque de equipo.

Link del proyecto: https://github.com/pgraciae/MBD_LakeHouseDeploymentModel.git

3.1.1.7 IDE: Visual Studio Code

El uso de Visual Studio como entorno de desarrollo integrado (IDE) en este proyecto también tiene méritos técnicos y prácticos que se alinean con los objetivos globales. Visual Studio proporciona un ambiente unificado para el desarrollo, la depuración y el despliegue de aplicaciones, y su compatibilidad nativa con lenguajes como Python lo convierte en una elección idónea para un proyecto que involucra tanto el desarrollo de scripts de Python como la orquestación con Apache Airflow.

Una de las características destacadas de Visual Studio es su integración directa con Git, lo que permite una gestión del control de versiones más fluida directamente desde la interfaz del IDE.

Además, Visual Studio ofrece capacidades avanzadas de depuración y pruebas unitarias, herramientas que son fundamentales para asegurar la calidad del código en un sistema complejo como el que se está desarrollando. El IDE también es compatible con una variedad de extensiones que pueden facilitar el trabajo con tecnologías específicas, como plantillas ARM de Azure o contenedores Docker, elementos centrales de este proyecto.

Por último, la facilidad de integración entre Visual Studio y los servicios de Azure ofrece un flujo de trabajo cohesivo para el desarrollo y despliegue de recursos en la nube. Esto facilita tareas como el monitoreo de recursos en tiempo real, la revisión de métricas de rendimiento y la gestión de costos, aspectos que son vitales para la implementación efectiva y sostenible de una arquitectura de Big Data en Azure.

3.2 Desarrollo

Este apartado aborda en detalle el proceso técnico asociado con el desarrollo del despliegue automático de la infraestructura end-to-end para Big Data. Cada subsección describe distintos componentes del sistema y los procedimientos para su implementación.

Es necesario mencionar que todos los nombres de los recursos siguen la nomenclatura estándar de azure.

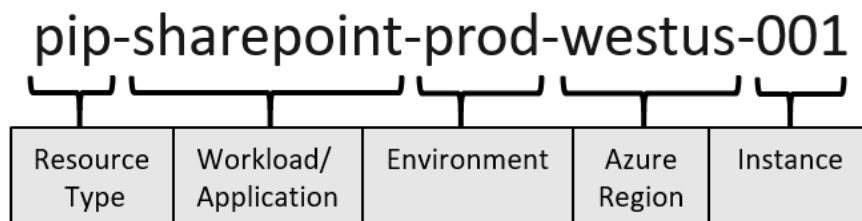


Figura 6. Nomenclatura estándar recursos azure

3.2.1 Autenticación de la aplicación en Azure

Este subapartado aborda el proceso de autenticación de la aplicación de Python dentro del contenedor Docker para permitir el despliegue automatizado de recursos en Azure. La autenticación se lleva a cabo mediante un servicio principal de Azure Active Directory (Azure AD), el cual es un tipo de identidad gestionada que se utiliza para permitir que las aplicaciones interactúen con los recursos de Azure de forma segura.

3.2.1.1 Configuración de Servicio Principal en Azure AD

Para configurar un nuevo servicio principal, es necesario acceder al Azure Portal y abrir la consola de Azure Cloud Shell. En el shell, se ejecuta el siguiente comando:

```
az ad sp create-for-rbac --name AzureDeploy --role Contributor --scopes /subscriptions/{subscription-id}
```

Este comando crea un nuevo servicio principal con el nombre "AzureDeploy" y le otorga el rol de "Contributor" en el ámbito de la suscripción especificada. Los parámetros son los siguientes:

--name: Especifica el nombre del servicio principal. En este caso, se ha utilizado "AzureDeploy" para coherencia y fácil identificación.

--role: Define el rol que se asignará al servicio principal. Se ha elegido "Contributor" para otorgar los permisos necesarios para crear, leer, actualizar y eliminar recursos.

--scopes: Establece el alcance dentro del cual el servicio principal tiene permisos. En este contexto, el alcance está limitado a la suscripción Azure identificada por {subscription-id}.

3.2.1.2 Uso de Variables de Entorno para la Autenticación

Una vez creado el servicio principal, Azure devuelve un objeto JSON con varias propiedades como `appId`, `password`, y `tenant`. Estas se utilizan como variables de entorno en el archivo `.env` que se carga en el contenedor Docker para que el script de Python pueda inicializar el objeto **EnvironmentCredential** del SDK de Azure.

Las variables son accesibles una sola vez por motivos de seguridad así que es necesario inmediatamente posteriormente a crear el service principal, añadir la configuración al fichero de *environment*.

El uso de un servicio principal en combinación con variables de entorno ofrece un enfoque seguro para la autenticación por diversas razones: Primero, el servicio principal actúa como una identidad de aplicación autónoma, eliminando la necesidad de almacenar credenciales de usuario en el código, lo que reduce significativamente el riesgo de exposición de datos sensibles. Además, al asignar un rol específico al servicio principal, como "Contributor", se aplica el principio de mínimo privilegio, permitiendo sólo las operaciones estrictamente necesarias para cumplir con las tareas designadas.

En cuanto al uso de variables de entorno, éstas se almacenan fuera del código fuente y se inyectan en el contenedor en tiempo de ejecución. Esto significa que las credenciales y otros datos sensibles no se almacenan en el repositorio de código, lo que mejora la seguridad al evitar posibles filtraciones. Además, el manejo de las variables de entorno se puede controlar de forma centralizada, facilitando la rotación de credenciales y la revocación de acceso en caso de compromiso.

Este enfoque de utilizar un servicio principal y variables de entorno proporciona un método seguro y escalable para gestionar la autenticación en aplicaciones automatizadas.

3.2.2 Despliegue infraestructura de Almacenamiento

El despliegue se realiza de forma automatizada mediante un contenedor Docker que ejecuta un script Python, el cual se encarga de crear y configurar los recursos en Azure a partir de una plantilla ARM (Azure Resource Manager).

3.2.2.1 Creación de grupo de recursos

La primera etapa del despliegue implica la creación de un grupo de recursos en Azure, que actuará como contenedor lógico para agrupar y gestionar los recursos asociados con la infraestructura de almacenamiento. Los detalles técnicos son los siguientes:

- **Nombre del grupo de recursos:** El nombre del grupo de recursos se define como "rg-storage-lakehouse-test-westeu-002" para garantizar una nomenclatura coherente y fácilmente identificable.
- **Localización grupo de recursos:** La localización geográfica seleccionada para el grupo de recursos es "West Europe".

Estos detalles se implementan mediante el uso del SDK de Azure para Python. El objeto **ResourceManagementClient** del SDK se inicializa con credenciales proporcionadas a través de variables de entorno. A continuación, se crea o se actualiza el grupo de recursos utilizando el método `create_or_update`.

3.2.2.2 Creación plantilla ARM

La siguiente etapa del despliegue implica la utilización de una plantilla ARM para definir y configurar la infraestructura de almacenamiento, concretamente una Storage Account.

La plantilla ARM se encuentra en el archivo `storage.json`, que es leído por el script Python. Este archivo JSON contiene la definición detallada de la infraestructura de almacenamiento:

- **Nombre storage account:** El nombre de la storage account creada es: **sa-storage-layer-test-02**. Este nombre debe ser único en todo el dominio de azure.
- **Creación 3 contenedores:** El script genera los 3 contenedores de tipo `datalakestoragev2` con nomenclatura jerárquica.
 - **adl-bronzelayer-test-01**
 - **adl-silverlayer-test-01**
 - **adl-goldlayer-test-01**

El script Python luego utiliza el objeto **ResourceManagementClient** para comenzar el despliegue de la plantilla ARM con el método `begin_create_or_update`. El modo de despliegue se establece como **DeploymentMode.incremental** para asegurar que los recursos existentes no sean eliminados.

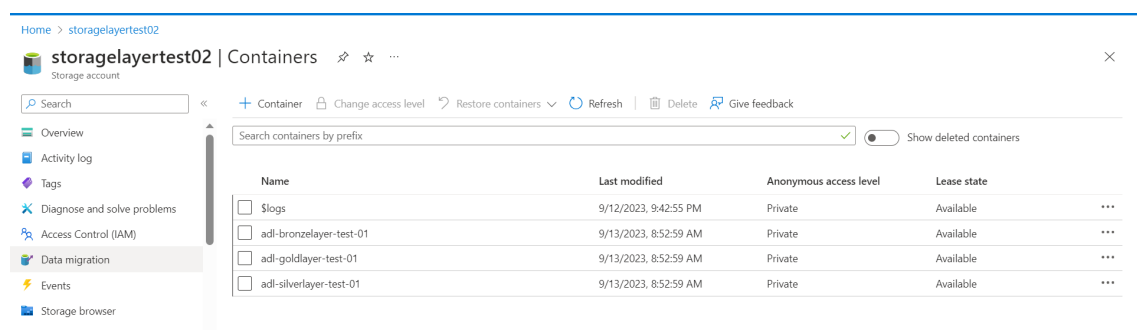


Figura7. Resultado del deployment de la capa de storage con 3 contenedores

3.2.3 Despliegue de componentes de Ingesta de datos

El proceso de despliegue para los componentes de ingesta de datos se inicia con la creación de un grupo de recursos específico en Azure. Este grupo de recursos servirá como contenedor lógico para los servicios y componentes necesarios para la ingestión de datos.

3.2.3.1 Creación grupo de recursos

- **Nombre del grupo de recursos:** El nombre del grupo de recursos es rg-ingestionlakehouse-test-westeurope-002. Se ha seleccionado de forma que refleje su función específica y cumpla con las normativas de nomenclatura empresarial.
- **Localización del grupo de recursos:** La localización del grupo de recursos se establece en "West Europe", para optimizar la latencia y los costos de datos al coincidir con la localización geográfica de los servicios y usuarios finales.

3.2.3.2 Creación plantilla ARM

Tras la creación del grupo de recursos, el siguiente paso es la implementación de la infraestructura necesaria utilizando una plantilla ARM (Azure Resource Manager). En este proyecto, se ha desarrollado una plantilla ARM personalizada que se encuentra en un archivo JSON denominado ingestion.json. La plantilla configura:

- **Nombre Data factory:** El nombre del Datafactory creado es: **df-ingestionlayer-test-02**. Este nombre debe ser único en todo el dominio de azure.
- **Tipo de Datafactory:** v2

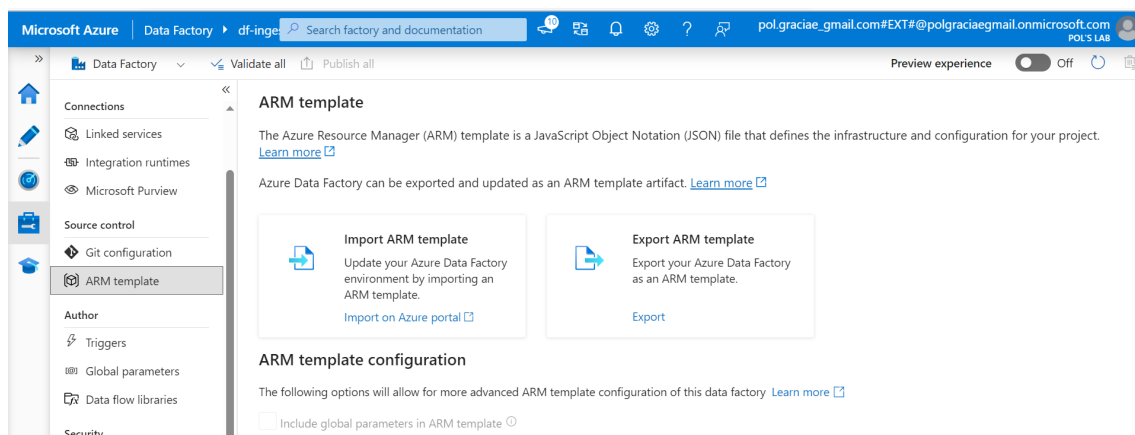


Figura 8. Generación plantilla ARM Data Factory

En el caso de Datafactory, la plantilla base tiene que ser obtenida des de un estudio de Datafactory.

3.2.4 Despliegue de componentes de Transformación y Procesado

3.2.4.1 Creación de grupo de recursos

- **Nombre del grupo de recursos:** El nombre del grupo de recursos es rg-processinglakehouse-test-westeurope-002. Se ha seleccionado de forma que refleje su función específica y cumpla con las normativas de nomenclatura empresarial.
- **Localización del grupo de recursos:** La localización del grupo de recursos se establece en "West Europe", para optimizar la latencia y los costos de datos al coincidir con la localización geográfica de los servicios y usuarios finales.

Una característica significativa del despliegue de un Azure Databricks Workspace es la creación automática de dos grupos de recursos gestionados, o Managed Resource Groups (MRGs). Estos MRGs desempeñan funciones especializadas y contienen varios recursos críticos.

- **Databricks Managed Resource Group:** Este grupo de recursos aloja diversos componentes que son esenciales para el funcionamiento del Azure Databricks Workspace. Entre ellos se encuentran:
 - **Clústers de Databricks:** Instancias de máquinas virtuales que ejecutan tareas de cálculo.
 - **Blob Storage:** Almacenes de objetos para guardar datos y artefactos.
 - **Event Hubs:** Para gestión de eventos en tiempo real.
 - **VNET Injection:** Posible configuración de red virtual para garantizar una comunicación segura.

Este grupo de recursos es gestionado íntegramente por Azure Databricks y no requiere intervención manual para su mantenimiento.

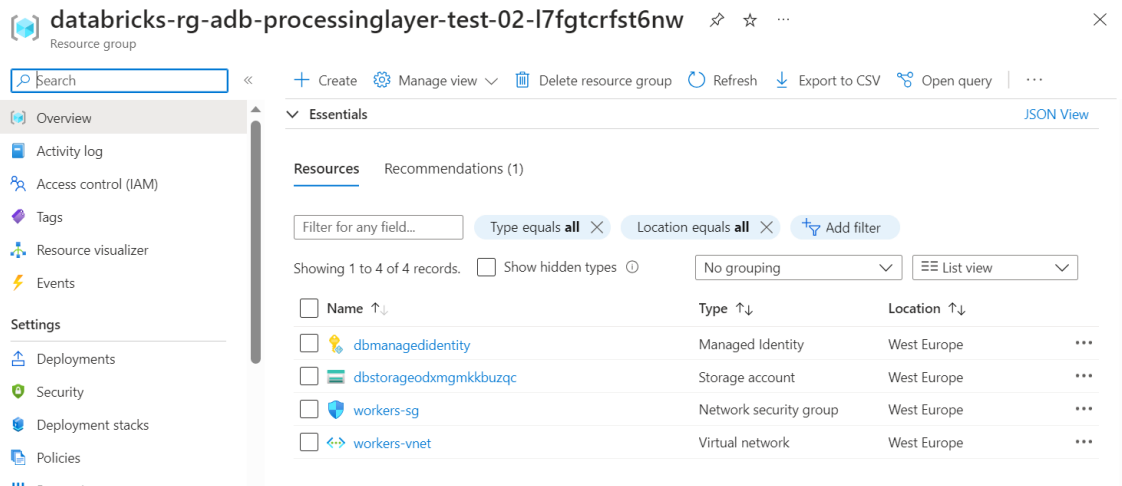


Figura 9. Managed resource group para la gestión del workspace de databricks

- **Network Managed Resource Group:** Este grupo de recursos se especializa en contener los elementos relacionados con la red y la seguridad. Los componentes típicos son:
 - **Virtual Network (VNet):** Red virtual que permite la comunicación entre los recursos de Azure.
 - **Network Security Group (NSG):** Contiene reglas de seguridad para regular el tráfico de red entrante y saliente a la red virtual y subredes.
 - **Public IP addresses:** Direcciones IP públicas asignadas a los recursos que requieren accesibilidad desde Internet.

Este MRG también es de gestión automática y ofrece una capa adicional de seguridad y aislamiento para los recursos del Databricks Workspace a nivel de red.

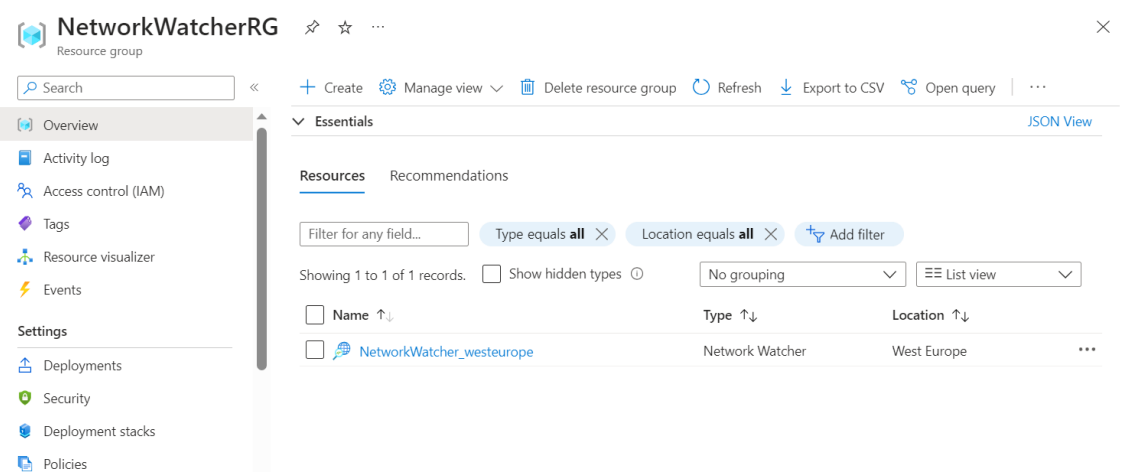


Figura 10. Managed resource group para la gestión del workspace de databricks

La creación de estos Managed Resource Groups es automatizada y se lleva a cabo durante el proceso de despliegue del Databricks Workspace, sin requerir intervenciones manuales adicionales.

3.2.4.2 Creación de plantilla ARM

Tras la creación del grupo de recursos, el siguiente paso es la implementación de la infraestructura necesaria utilizando una plantilla ARM (Azure Resource Manager). En este proyecto, se ha desarrollado una plantilla ARM personalizada que se encuentra en un archivo JSON denominado `processing.json`. Es necesario mencionar que los dos managed resource groups mencionados se crean de manera transparente al usuario y al script ARM, se configura el workspace y lo demás se crea de forma automática:

- **Nombre Databricks workspace:** El nombre del Databricks creado es: **db-processinglayer-test-02**. Este nombre debe ser único en todo el dominio de azure.
- **Tipo de Databricks:** Premium (permite el uso autenticación con azure AD).

También se genera mediante la plantilla ARM un recurso Azure Databricks Connector que va a permitir asignar roles mediante Managed Identity al workspace principal.

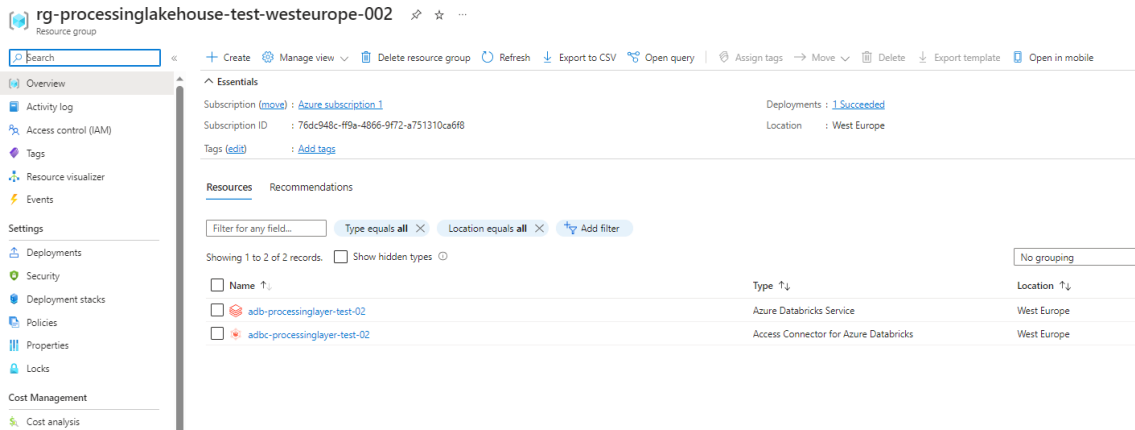


Figura 11. Despliegue grupo de recursos Databricks

3.2.5 Contenerización de Airflow

En esta sección, se aborda la contenerización del entorno de Apache Airflow mediante el uso de Docker. Dado que Airflow consta de varios componentes que trabajan en conjunto, la estrategia de contenerización implica la creación de múltiples contenedores, cada uno de los cuales aloja un componente específico.

3.2.5.1 Ventajas de la Contenerización como Microservicios

La arquitectura de microservicios es especialmente adecuada para aplicaciones complejas y escalables como Airflow. Al contenerizar cada componente crítico de Airflow en su propio contenedor Docker, se obtienen múltiples ventajas:

- **Aislamiento:** Cada contenedor opera en un entorno aislado, lo que minimiza las interferencias entre componentes y facilita el diagnóstico de problemas y la optimización del rendimiento.
- **Escalabilidad:** Los contenedores se pueden escalar de manera independiente. Por ejemplo, si hay un alto volumen de tareas pendientes, se pueden escalar horizontalmente los workers sin afectar otros servicios.
- **Despliegue y Mantenimiento:** Los contenedores permiten un ciclo de desarrollo más rápido al facilitar el despliegue, actualización y rollback de cada componente de forma independiente.
- **Reutilización de Recursos:** Permite el uso eficiente de los recursos del sistema al compartir servicios y recursos del sistema operativo subyacente, sin comprometer el aislamiento.

3.2.5.2 Componentes de airflow

Los microservicios desplegados en Docker son:

Airflow Common

Este bloque de configuración actúa como una plantilla que se reutiliza para configurar múltiples servicios de Airflow. Definido bajo **x-airflow-common**, este bloque establece la imagen de Docker a usar (**apache/airflow:2.5.1**) y varios parámetros de configuración mediante variables de entorno.

- **Executor:** Se usa ejecución local, lo que significa que todas las tareas se ejecutan en un solo nodo. Este es un modo adecuado para pruebas y desarrollo.
- **Base de datos:** Se configura una cadena de conexión a Postgres para el almacenamiento del metadata de Airflow. Esta cadena de conexión usa psycopg2 como controlador de Postgres.
- **Fernet Key y Secret Key:** Estas claves se utilizan para cifrar y descifrar información sensible en la base de datos y las cookies de la sesión, respectivamente.
- **Paralelismo, Workers, Zona Horaria, etc.:** Se configuran varios parámetros adicionales, como el número de tareas que pueden ejecutarse en paralelo, la zona horaria del servidor y otros.

Web Server

El servidor web es el componente de interfaz de usuario en Airflow y se utiliza para todo tipo de interacciones con el sistema. Se ha optado por contenerizar el servidor web para aislar la interfaz de usuario de otros componentes. Esto es crucial desde un punto de vista de seguridad, ya que cualquier vulnerabilidad en la interfaz web no afectará directamente a otros componentes del sistema. Además, este aislamiento permite escalar el servidor web independientemente en caso de altas cargas de trabajo de usuario.

Este servicio utiliza la configuración común definida anteriormente y agrega su propia configuración específica.

- **Puertos:** Se expone el puerto 8080 para el acceso a la interfaz web de Airflow.
- **Chequeo de Salud:** Se configura un chequeo de salud para este servicio, permitiendo saber si está funcionando como se espera.

Postgres Database

Airflow necesita una base de datos relacional para almacenar el estado del sistema, y se ha elegido Postgres por su robustez y características de ACID (Atomicity, Consistency, Isolation, Durability). Al contenerizar la base de datos, se facilita su mantenimiento, copias de seguridad y escalabilidad sin afectar al resto del sistema.

Se define un volumen para persistir los datos y se establece un chequeo de salud para asegurarse de que el servicio está funcionando correctamente. Además, se crea un usuario que utilizará el servicio de Airflow para interactuar con postgres.

Servicio de Inicio (init)

Este servicio realiza operaciones iniciales como las migraciones de la base de datos y la creación del usuario de Airflow. Utiliza la misma imagen que los otros servicios de Airflow, pero su objetivo principal es preparar el entorno antes de que los otros servicios comiencen.

Redes y volúmenes

Se crea una red personalizada llamada airflow para aislar los servicios de la red predeterminada de Docker. Además, se utiliza un volumen para persistir los datos de la base de datos Postgres.

Esto permite a Airflow utilizar una red distinta a la *default* de Docker, quedando así aislada de cualquier otro contenedor del sistema y aumentando la seguridad de los componentes.

3.2.6 Configuración de interconexiones entre los componentes

En este proyecto, una de las capas críticas es la gestión de identidades y el control de acceso, asegurando que los diferentes servicios puedan interactuar entre sí de forma segura y eficiente. Es decir, no sólo la implementación de los recursos sino la creación de interconexiones entre ellos para el uso end-to-end del dato. Esto se ha implementado a través de la infraestructura de gestión de identidades y accesos (IAM) de Azure y las identidades gestionadas (Managed Identities).

3.2.6.1 Implementación de Conexiones y Managed Identities

Lamentablemente, no todas las tareas pueden ser completamente automatizadas. Algunas configuraciones, como el establecimiento de políticas de acceso más granulares en nuestro caso la asignación de roles entre los recursos aún no es posible de hacer mediante el Python sdk y sólo se puede desarrollar mediante el azure CLI.

Para ello, algunos de los pasos requerirán la entrada al portal y la ejecución de unos scripts mediante el cloud Shell.

Conexión Datafactory a Blob storage

En el desarrollo de este proyecto, la conexión entre Azure Data Factory y Azure Blob Storage se ha establecido mediante dos fases distintas: la creación de un Linked Service a través de una plantilla ARM y la asignación de roles de acceso utilizando Azure CLI desde el Azure Cloud Shell.

- **Creación de Linked Service mediante Plantilla ARM**

La creación del Linked Service es la primera etapa y se ha realizado utilizando una plantilla ARM (Azure Resource Manager). La plantilla ARM se contiene dentro del despliegue del componente de ingesta y que contiene el datafactory y todas las configuraciones y parámetros necesarios para instanciar el Linked Service en Azure Data Factory. Este Linked Service actúa como un puente, permitiendo a Data Factory acceder al Blob Storage creado anteriormente de manera segura.

- **Asignación de Rol mediante Azure CLI**

Una vez creado el Linked Service, la siguiente etapa involucra la asignación de roles de acceso para garantizar que Azure Data Factory tenga los permisos necesarios para interactuar con Azure Blob Storage. Para lograr esto, se ha utilizado Azure CLI, específicamente desde el entorno de Azure Cloud Shell.

Se debe ejecutar un comando (descrito en el subapartado 3) para asignar el rol de "Storage Blob Data Contributor" a la identidad gestionada de Data Factory en el ámbito de Blob Storage. Este rol le confiere a Data Factory los permisos necesarios para realizar operaciones de lectura, escritura en el Blob Storage.

Conexión Databricks a Blob storage

La conexión entre Databricks y Azure Blob Storage se ha establecido siguiendo un proceso en dos fases: la primera implica la creación de un Databricks Access Connector a través de una plantilla ARM, y la segunda abarca la asignación de roles de acceso, ejecutada utilizando Azure CLI desde el Azure Cloud Shell.

- **Creación del Databricks Access Connector mediante Plantilla ARM**

La plantilla ARM (Azure Resource Manager) se ha utilizado para crear el Databricks Access Connector, que actúa como una capa intermedia para facilitar el acceso seguro entre el entorno de Databricks y Blob Storage. En esta plantilla, se definen todas las configuraciones necesarias para establecer el acceso desde Databricks hasta el Blob Storage de Azure. El despliegue de esta plantilla ARM se ha automatizado y forma parte de la infraestructura como código que alimenta el pipeline de CI/CD. Al aplicar esta plantilla, se instancian los recursos necesarios, configurando el Access Connector en el workspace de Databricks con los parámetros y permisos apropiados.

- **Asignación de Rol mediante Azure CLI**

Después de crear el Databricks Access Connector, el siguiente paso es asegurar que el entorno de Databricks tenga los permisos necesarios para acceder y manipular los datos en Blob Storage. Esto se ha logrado mediante el uso del Azure CLI en el Azure Cloud Shell. Un comando específico de Azure CLI se ha utilizado para asignar el rol de "Blob Data Contributor" al servicio de Databricks a la Storage Account. Este rol le proporciona al entorno de Databricks los permisos necesarios para ejecutar operaciones de lectura, escritura y eliminación en Blob Storage.

3.2.6.2 Configuración de Unity Catalog en Databricks y acceso a blob storage mediante ABFSS

Una vez establecidas las conexiones y los permisos necesarios entre Databricks y Azure Blob Storage, el siguiente paso es configurar el Unity Catalog dentro del entorno de Databricks para facilitar la gestión y consulta de datos almacenados en Blob Storage. Dado que la autorización se ha realizado a través de IAM (Identity and Access Management), no es necesario el uso de secretos para acceder al almacenamiento (se ha usado la práctica recomendada por azure).

Creación de Storage Credentials

Aunque no se necesitan secretos para la autorización, sí es necesario configurar las credenciales de almacenamiento para que Databricks pueda acceder a Blob Storage. Estas credenciales se generan dentro del workspace de Databricks y se asocian con la identidad gestionada que ya tiene asignado el rol de "Blob Data Contributor". De esta manera se crea una storage credential

que requiere solamente el id del databricks connector ya configurado para acceder a la storage account y todos los contenedores.

Este paso se debe realizar de manera manual desde el workspace de Databricks.

Creación de External Locations

Después de crear la storage credential, el siguiente paso es crear external locations para poder acceder a los contenedores desde el código. Las ubicaciones externas en Databricks permiten definir rutas específicas en el Blob Storage donde se almacenan los datos.

Para este proyecto, se crean 3 ubicaciones externas apuntando a los contenedores **bronze, silver y gold**.

Esta práctica también mejora la gobernanza en nuestra base de datos, permitiendo granular el acceso de futuros usuarios a cada layer.

Acceso a Blob Storage mediante abfss://

Una vez configurado el Unity Catalog y establecidas las localizaciones externas y la storage credential, se puede acceder a los datos en Blob Storage utilizando el protocolo abfss://. Este protocolo asegura que la comunicación entre Databricks y Blob Storage se realice de manera segura y eficiente.

Para acceder a un contenedor específico, se utiliza una URI que sigue la siguiente estructura: abfss://<container-name>@<storage-account-name>.dfs.core.windows.net/. Esto se puede hacer directamente desde las notebooks de Databricks para leer o escribir datos, o bien a través de trabajos programados que utilicen este protocolo.

Por ejemplo, para acceder al contenedor Bronze, se utilizaría una URI como **abfss://adl-bronzelayer-test-01@myStorageAccount.dfs.core.windows.net/**.

3.2.7 Unificación del proceso mediante Docker compose

Docker Compose es una herramienta para definir y gestionar aplicaciones de contenedores Docker de múltiples servicios. Utiliza archivos YAML para especificar los servicios, redes y volúmenes que conforman la aplicación, y ofrece un conjunto de comandos para gestionar el ciclo de vida de los componentes. En el contexto del despliegue de soluciones empresariales que implican múltiples componentes, como bases de datos, aplicaciones web y servicios backend, Docker Compose presenta una ventaja estratégica para simplificar y estandarizar el proceso de despliegue.

Para realizar el despliegue automático, se usa el siguiente esquema de dockerización de componentes:

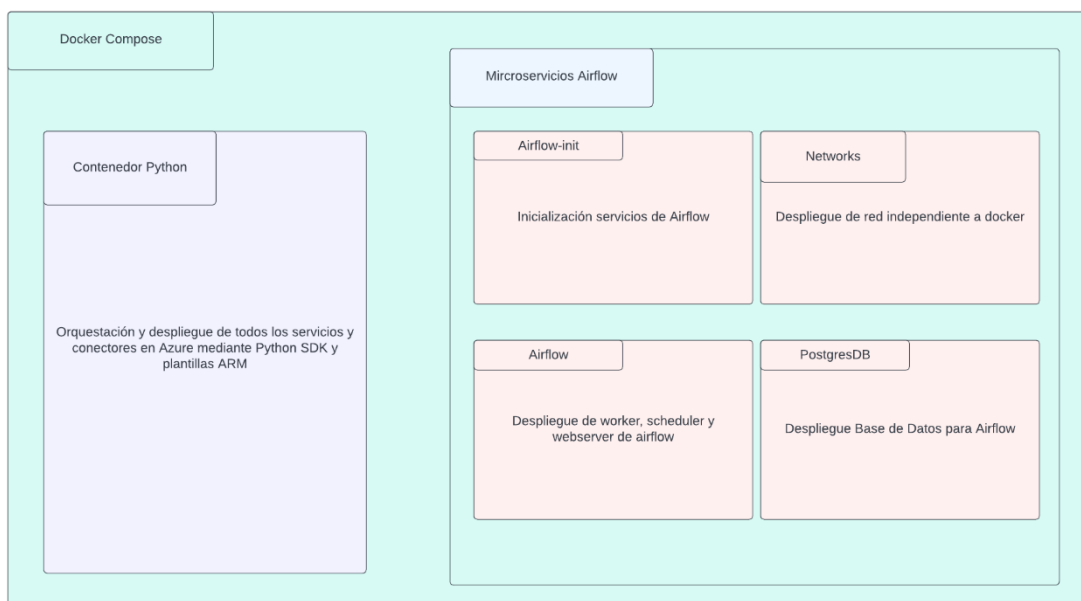


Figura 11. Estructura dockerización de los componentes

3.2.7.1 Funcionamiento Técnico de Docker Compose

Docker Compose funciona leyendo un archivo **docker-compose.yml**, que es un archivo de texto en formato YAML. Este archivo describe detalladamente los servicios que componen una aplicación, sus configuraciones, sus relaciones y sus dependencias. Cuando se ejecuta el comando **docker-compose up** en un entorno bash, la herramienta realiza una serie de acciones:

1. **Creación de la Red:** Docker Compose crea automáticamente una red aislada para la aplicación, lo que facilita la comunicación segura y eficiente entre los contenedores.
2. **Creación de Volumen:** Si se definen volúmenes en el archivo **docker-compose.yml**, se crean para permitir la persistencia de datos.
3. **Construcción de Imágenes:** Si se especifican instrucciones para construir imágenes de contenedor, Docker Compose construye esas imágenes antes de iniciar los contenedores.
4. **Inicio de Contenedores:** Docker Compose inicia los contenedores en el orden especificado, respetando las dependencias declaradas entre ellos.
5. **Orquestación:** Asegura que los contenedores se inicien, detengan y escalen juntos, ofreciendo comandos para gestionar todo el ciclo de vida de la aplicación.

3.2.7.2 Ventajas para el Proceso de Despliegue

1. **Estandarización:** Docker Compose permite definir una única "receta" para el despliegue que es fácil de entender y replicar, asegurando la coherencia a lo largo de diferentes entornos (desarrollo, prueba, producción).
2. **Simplicidad y Rapidez:** Con un solo comando, se pueden levantar todos los servicios, reduciendo la complejidad y el tiempo necesario para poner en marcha una solución completa.
3. **Gestión de Dependencias:** Los servicios que dependen entre sí se pueden coordinar de manera efectiva mediante la opción **depends_on**, asegurando que los servicios dependientes se inicien en el orden correcto.
4. **Seguridad:** Al utilizar redes y volúmenes específicos, Docker Compose garantiza el aislamiento y la seguridad de los datos y servicios.
5. **Flexibilidad:** Los archivos **docker-compose.yml** son fácilmente parametrizables, lo que permite adaptar la configuración a diferentes requisitos sin tener que cambiar el código de la aplicación.
6. **Monitoreo y Escalabilidad:** Aunque Docker Compose se usa comúnmente para entornos de desarrollo y prueba, su integración con Docker Swarm y Kubernetes permite gestionar aplicaciones en producción a gran escala.

3.3 Procedimiento de ejecución del despliegue

El procedimiento para la ejecución del despliegue de la arquitectura está diseñado para ser robusto y automatizado, reduciendo al mínimo la intervención manual. Cada paso está meticulosamente planificado para garantizar un proceso fluido y seguro. A continuación se detallan los pasos que debe llevar a cabo un usuario para lanzar el despliegue:

1 Clonar el Repositorio

Para iniciar el proceso, primero es necesario obtener una copia local del repositorio GitHub que contiene el código fuente y los archivos de configuración.

- **Acción:** Hacer pull del repositorio:

```
Git pull https://github.com/pgraciae/MBD\_LakeHouseDeploymentModel
```

2 Creación de la Cuenta en Azure

Para desplegar recursos en Azure, es necesario tener una cuenta que brinde acceso al portal de Azure y a los servicios ofrecidos.

- **Acción:** Acceder al portal de Azure y seguir los pasos de registro.

3 Configuración de la Suscripción

Una vez registrados en Azure, se debe establecer una suscripción. La suscripción es esencial para administrar y pagar por los recursos que consumimos.

- **Acción:** En el portal de Azure, navegar a "Suscripciones" y seleccionar o crear una nueva.

4 Ejecución de Azure CLI en cloud shell

Azure CLI (Command-Line Interface) es una serie de comandos utilizados para gestionar recursos de Azure.

- **Acción:** Dentro del portal, iniciar el **cloud Shell**.

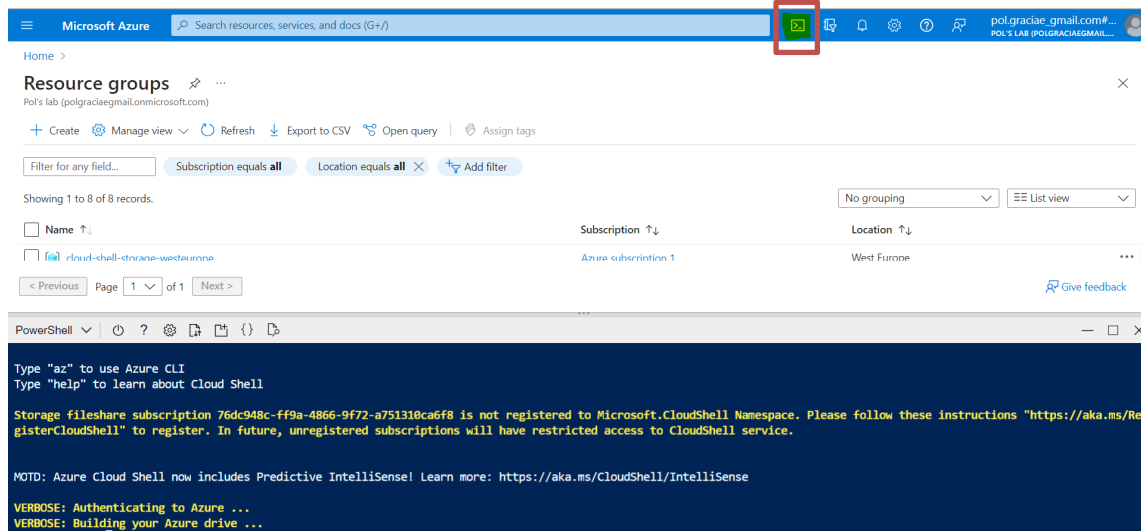


Figura 12. Ejemplificación lanzamiento del Azure Cloudshell

5 Creación de Service Principal

Los Service Principals permiten a las aplicaciones interactuar con Azure Active Directory y, por extensión, con Azure. Este paso es crucial para garantizar que el script que se va a ejecutar posteriormente tenga los permisos adecuados.

- **Acción:** Ejecutar el siguiente comando, reemplazando {subscription-id} por el ID de suscripción de la cuenta:

```
az ad sp create-for-rbac --name AzureDeploy --role Contributor --scopes /subscriptions/{subscription-id}
```

6 Configuración del Archivo .env

Este archivo almacenará variables de entorno que la aplicación necesita para funcionar correctamente.

- **Acción:** En la terminal o editor de texto, navegar a la carpeta ./azure/ y crear un archivo denominado .env.

7 Configuración de las variables de entorno

Después de crear el Service Principal, Azure CLI devolverá un conjunto de credenciales. Es fundamental copiar estos datos, ya que Azure no volverá a mostrar la contraseña.

- **Acción:** Añadir las credenciales al archivo `.env` siguiendo el formato:

```
AZURE_CLIENT_ID=appId  
AZURE_TENANT_ID=tenant  
AZURE_CLIENT_SECRET=password  
AZURE_SUBSCRIPTION_ID=subscriptionId
```

8 Ejecución de Docker Compose

Finalmente, se navega hasta el directorio dónde se encuentra el repositorio y se inicia el proceso de despliegue.

- **Acción:** Ejecutar el siguiente comando:

```
docker-compose up --build
```

Este comando se encargará de construir las imágenes de Docker, crear los contenedores, establecer las redes y montar los volúmenes según lo definido en el archivo **docker-compose.yml**. El comando solo necesita ser ejecutado una vez para configurar todo el ambiente y desplegar los servicios.

9 Establecer conexión entre los componentes

Una vez el proceso de Docker-compose ha terminado, se deben establecer los permisos entre los componentes. Como ya se ha mencionado, este paso no se puede hacer desde el sdk de Python, así que desde el cloud Shell, en el portal de Azure, en formato bash, se debe ejecutar el fichero llamado `./azure/authorise.sh`, que automáticamente concederá los permisos de contributor al databricks access connector y al data factory al blob storage.

10 Configuración Unity Catalog

El último paso es la configuración del Unity Catalog y la creación de external locations en databricks. Lamentablemente, este paso debe ser hecho de manera manual dentro del workspace de Azure Databricks. Se ha proporcionado un notebook llamado `./Databricks/db_environment_setup.ipynb` que realiza todos los pasos que son automatizables (4 –):

1. Habilitar Unity Catalog mediante la creación de una metastore

- a. Login account Management <https://accounts.azuredatabricks.net/login/>
- b. Dentro de catalog, crear metastore

- c. Añadir detalles de:
 - i. Nombre metastore
 - ii. Región: westeurope
 - iii. Link blob storage account: adl-bronzelayer-test-01@sastoragelayer-test02.dfs.core.windows.net/
 - iv. id del connector de databricks
- d. Asignar metastore al workspace de databricks

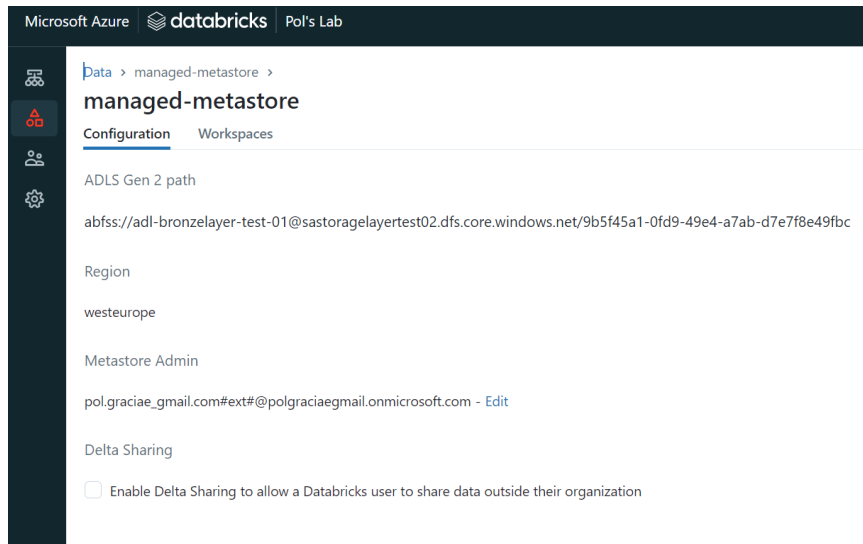


Figura 13. Ejemplificación configuración managed-metastore

2. Creación de un clúster de databricks

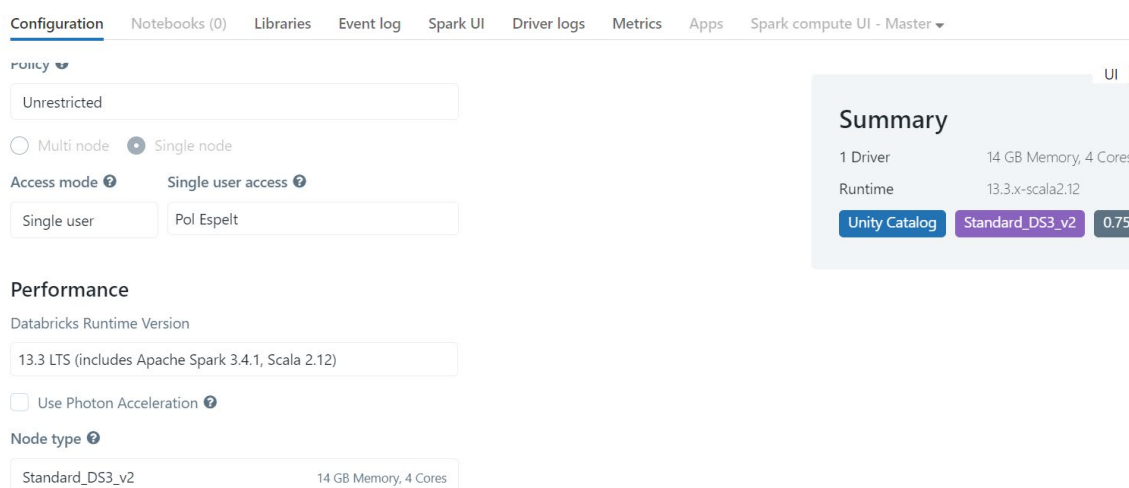


Figura 14. Ejemplificación configuración clúster databricks

3. Creación de storage credential

- a. Iniciar workspace y entrar con la cuenta administrativa
- b. Dentro de catalog, seleccionar external locations

- c. Seleccionar create storage account
- d. Añadir detalles:
 - i. Nombre credencial
 - ii. Id del conector de databricks

Name ↕	Credential Type	Properties	Owner	Comment
deltalake_credential	Managed Identi...	Connector Id: /subscriptions/76dc948c-ff9a-4866-9f72-a75 User Assigned Managed Identity Id:	pol.graciae_gmail.com#ext...	Storage credential para con...

Figura 15. Visualización Storage Credential

4. Creación de external location (una por cada layer):
 - a. Conexión a bronze layer
 - b. Conexión a silver layer
 - c. Conexión a gold layer

Name	Credential	URL ↕	Owner
silverlayer	deltalake_cred...	abfss://adl-silverlayer-test-01@sastoragelayertest02....	pol.graciae_gmail.com#ext#@polgraci...
goldlayer	deltalake_cred...	abfss://adl-goldlayer-test-01@sastoragelayertest02....	pol.graciae_gmail.com#ext#@polgraci...
bronzelayer	deltalake_cred...	abfss://adl-bronzelayer-test01@sastoragelayertest02...	pol.graciae_gmail.com#ext#@polgraci...

Figura 16. Visualización External Location

5. Creación de schemas (notebook)

El notebook despliega un schema llamado **test_schema** para validar la conexión.

6. Creación de tablas (notebook)

El notebook crea una tabla externa en formato delta llamada **test_table** y guardada en la bronze layer.

El notebook inserta unos datos en ellos y se deberían poder ver tanto des de databricks cómo des del blob storage (bronze layer).

```

1 %sql
2 select *
3 from test_schema.test_table

```

▶ (3) Spark Jobs

▶ _sqlidf: pyspark.sql.dataframe.DataFrame = [id: integer, nom: string ... 1 more field]

	id	nom	location
1	30	Bernat	MAIDSTONE
2	40	Victor	DARLINGTON
3	50	Carles	BIRMINGHAM
4	20	Joan	PADDINGTON
5	10	Pol	EDINBURGH

↓ 5 rows | 2.95 seconds runtime

Figura 17. Visualización output tabla

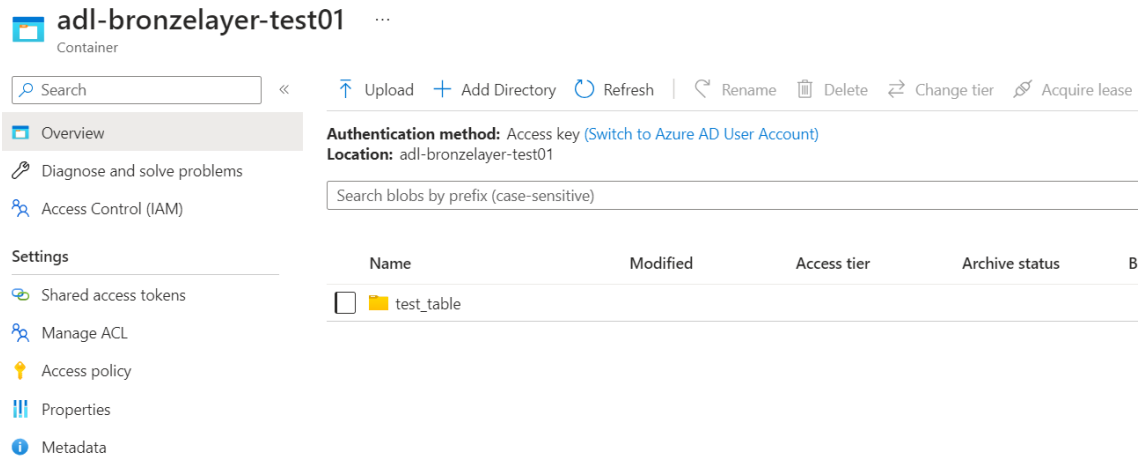


Figura 18. Visualización ficheros generados para la creación de la tabla

4 Capítulo 4: Casos de estudio y validación

El Capítulo 4 se centra en los casos de estudio y en la validación del proceso end-to-end diseñado y desplegado en los capítulos anteriores. A través de escenarios de uso realistas y pruebas de concepto, este capítulo busca no solo demostrar la funcionalidad integral del sistema, sino también evaluar su robustez, escalabilidad y eficiencia.

El diseño teórico y la implementación práctica del sistema se someten a para asegurar que todos los componentes individuales - desde la ingestión de datos con Azure Data Factory hasta el procesamiento de datos en Azure Databricks, orquestado por Apache Airflow - funcionan en una sinergia operacional coherente. Esta validación es esencial tanto para verificar la correcta implementación del diseño como para identificar áreas potenciales para mejoras y optimizaciones.

Este caso de estudio se complementa del código en el repositorio de github del proyecto, para entender y ver cómo se ha realizado con código cada paso del caso es recomendable mirarlo conjuntamente.

4.1 Caso de estudio: Validación del proceso end-to-end

Objetivo del Caso de Estudio:

El principal objetivo de este caso de estudio es validar el flujo completo del sistema, desde la ingestión de datos mediante Azure Data Factory hasta el procesamiento de datos en Azure Databricks, todo ello orquestado por Apache Airflow.

Escenario:

Suponga que una organización desea analizar los patrones de ventas en diferentes regiones a lo largo del tiempo. Los datos brutos de ventas se almacenan en un blob storage en la capa de bronce y deben ser transformados y enriquecidos antes de realizar cualquier análisis.

Dataset:

Se ha elegido un dataset de Kaggle llamado 'Amazon Sales Dataset' [<https://www.kaggle.com/datasets/karkavelraj/amazon-sales-dataset?resource=download>]. Cabe destacar que no se ha usado ningún código de Kaggle para el presente caso de estudio, sólo se ha utilizado como fuente de datos.

Este dataset contiene los de datos de más de 1000 productos de amazon vendidos, con la estructura de usuario, reseñas y puntuación y nos servirá para ejemplificar que tranformaciones y análisis se llevarían a cabo en un escenario real.

El dataset contiene los siguientes campos:

- **product_id** - ID del producto

- **product_name** - Nombre del producto
- **category** - Categoría del producto
- **discounted_price** - Precio con descuento del producto
- **actual_price** - Precio real del producto
- **discount_percentage** - Porcentaje de descuento para el producto
- **rating** - Calificación del producto
- **rating_count** - Número de personas que votaron por la calificación de Amazon
- **about_product** - Descripción acerca del producto
- **user_id** - ID del usuario que escribió la reseña para el producto
- **user_name** - Nombre del usuario que escribió la reseña para el producto
- **review_id** - ID de la reseña del usuario
- **review_title** - Reseña corta
- **review_content** - Reseña larga
- **img_link** - Enlace de la imagen del producto
- **product_link** - Enlace al sitio web oficial del producto

Pasos del Proceso:

Para la simplificación del desarrollo de las explicaciones, en los apartados 2, 3 en primer lugar se van a explicar los funcionamientos de los notebooks y transformaciones que se hacen en cada lugar y en el apartado 5 se va a desarrollar cómo estos procesos se entrelazan con la orquestación de airflow.

1. **Ingestión de Datos:** Los datos de ventas se ingieren en la capa de bronce del blob storage mediante un pipeline de Azure Data Factory.
2. **Pre-procesamiento en Databricks:** La primera etapa del workflow de Airflow ejecuta un notebook de Databricks que realiza transformaciones iniciales en los datos y los mueve a la capa de plata.
3. **Transformación y Enriquecimiento de Datos:** Una segunda etapa en el workflow de Airflow dispara otro notebook de Databricks para enriquecer los datos con información adicional, como tasas de cambio y datos demográficos, antes de moverlos a la capa de oro.

4. **Análisis de Datos:** Finalmente, un tercer notebook en Databricks realiza análisis de patrones de ventas y visualizaciones.
5. **Orquestación des de airflow:** Se describe cómo se usa airflow para realizar las tareas de preprocesado y transformación y enriquecimiento de Datos.
6. **Validación:** Se realizan diversas pruebas para validar si los procedimientos se han ejecutado correctamente de principio a fin (del proceso end-to-end).


Recordar que todos los notebooks y código del caso de estudio se puede encontrar en el repositorio del proyecto.

4.1.1 Ingestión de datos

En primer lugar, se reproducirá la casuística de la extracción de datos de ventas del origen (podría ser un ERP o una fuente de datos en tiempo real) y la inserción de estos datos a la capa de bronce mediante Data Factory.

Para simplificar el proceso, ya que queda fuera del alcance del proyecto la conexión a fuentes externas de datos, se subirán los datos manualmente a un contenedor nuevo de la storage account (que interpretaremos cómo la source) y con el Data Factory se realizará una operación de COPY hacia la bronce layer.

Es necesario mencionar que este procedimiento se puede hacer de manera inmediata ya que la interconexión entre los componentes ya está realizada y el Data Factory tiene permisos de **Blob Storage Data Contributor** sobre la storage account del proyecto.



<input type="checkbox"/> adl-bronzelayer-test01	9/14/2023, 2:08:52 PM	Private	Available	...
<input type="checkbox"/> adl-goldlayer-test-01	9/13/2023, 8:20:36 PM	Private	Available	...
<input type="checkbox"/> adl-silverlayer-test-01	9/13/2023, 8:20:36 PM	Private	Available	...
<input type="checkbox"/> source-test	9/14/2023, 5:56:59 PM	Private	Available	...

Figura 19. Transferencia de datos entre source-test y adl-bronzelayer-test01

Para ello, en primer lugar se abre el Data Factory Management Studio y se crea una nueva pipeline.

Dentro del pipeline se inserta la actividad **COPY** que se encarga de mover datos (sin hacer transformaciones) entre los distintos orígenes de datos. La actividad requiere la configuración de 2 parámetros:

- **Source:** Es el origen de los datos: el fichero amazon_sales.csv dentro blob source-test. En este caso se selecciona el linked service a Azure Blob storage ya creado y de manera sencilla se puede seleccionar el tipo de archivo de origen **csv** y se selecciona la ruta del archivo (en este caso source-test/amazon_sales.csv)
- **Sink:** Es el destino de los datos: el blob adl-bronzelayer-test01. En este caso se selecciona el linked service a Azure Blob storage ya creado y se selecciona que queremos hacer una

transformación de datos a tipo **parquet** ya que és el formato estándar para delta lake en la capa de bronce, mucho más optimizado que el csv.

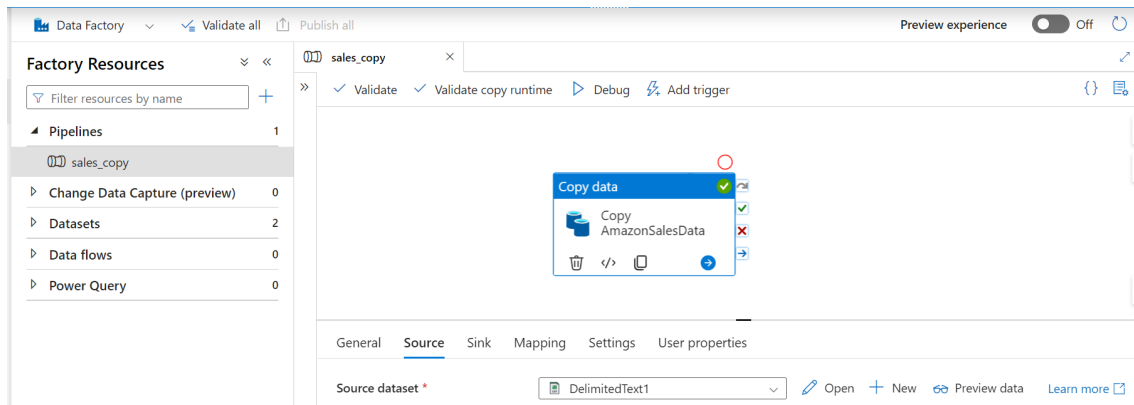


Figura 20. Pipeline Data Factory con actividad de Copy

A continuación, se ejecuta de manera manual la pipeline con la opción **Trigger now**. En un futuro, esta actividad se podría programar para que se lanzara de manera recurrente de manera automática.

Una vez la pipeline se ha ejecutado sin errores, podemos ver que hay un nuevo fichero llamado `amazon_sales.parquet` en la bronce layer.

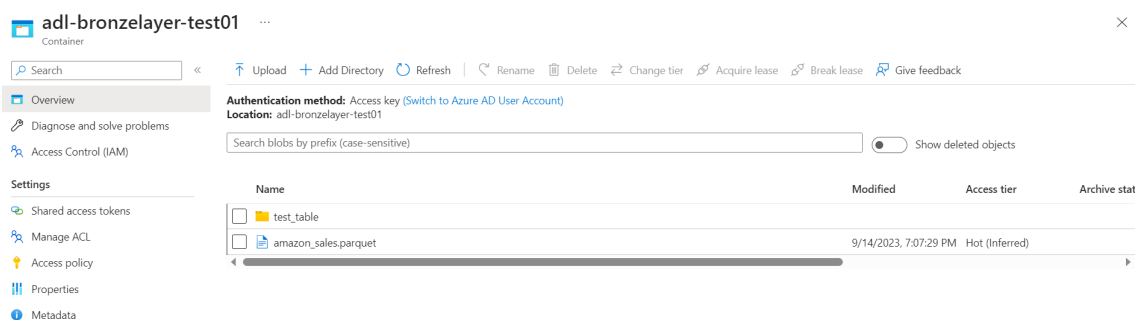


Figura 21. Datos transferidos a bronce layer en formato parquet

4.1.2 Pre-procesamiento de datos

La transformación de datos desde la capa Bronze hasta la capa Silver es un paso crítico que a menudo involucra diversas operaciones de limpieza, transformación y enriquecimiento de datos. Sin embargo, en el contexto del caso de estudio, las transformaciones son relativamente sencillas y se limitan a cambiar el tipo de datos y almacenar los datos transformados.

En un escenario real también se harían operaciones de filtrado y data consistency. El notebook que realiza las transformaciones se llama

4.1.2.1 Transformaciones realizadas:

1. **Tipo de Datos:** En la capa Bronze, todos los datos se almacenan en formato string para maximizar la flexibilidad y garantizar que los datos en bruto se capturen tal como están. Sin embargo, para facilitar el análisis y el procesamiento posteriores, se convierten los tipos de datos a formas más apropiadas en la capa Silver. Por ejemplo:
 - Los campos como **actual_price**, **discounted_price** y **discount_percentage** se convierten de string a tipo numérico (double).
 - Los campos **rating** y **rating_count** también se convierten a tipos numéricos (double e int, respectivamente).
 - El campo **ingestion_timestamp** se introduce como tipo timestamp para marcar cuándo se ingirieron los datos.
 - El campo **review_id** identifica de manera única los campos de la tabla **ProductReviews**.
2. **Particionamiento:** Se incorpora una estrategia de particionamiento en las tablas de la capa Silver para optimizar el rendimiento de las consultas. La tabla **ProductInfo** se particiona por la columna **category**, y la tabla **ProductReviews** por **product_id**. Esto facilita la recuperación rápida de datos relacionados con una categoría de producto o un producto específico.
3. **Estructura de Tabla:** Las tablas de la capa Silver se diseñan de tal manera que reflejen una estructura más normalizada y optimizada para consultas analíticas. En este caso, las tablas están organizadas en **ProductInfo** y **ProductReviews** para separar la información del producto y las revisiones de los usuarios.
4. **Almacenamiento Delta:** Al usar el formato Delta para las tablas de la capa Silver, se benefician de características como ACID transactions, versionado y optimización del rendimiento de lectura, lo cual es crucial para la analítica y las operaciones de Machine Learning que puedan seguir.

El paso de la capa Bronze a la capa Silver asegura que los datos sean más accesibles y manejables para análisis futuros y tareas de procesamiento. Este proceso es esencial para añadir estructura y significado a los datos en bruto de la capa Bronze

Home > Resource groups > rg-storagehouse-test-west-europe-002 > sastorage-test02 | Containers >

adl-silverlayer-test-01 ...

Container

Search < > Upload + Add Directory Refresh | Rename Delete Change tier Acquire lease Break lease Give feedback

Overview

Diagnose and solve problems

Access Control (IAM)

Settings

Shared access tokens

Manage ACL

Access policy

Properties

Metadata

Authentication method: Access key (Switch to Azure AD User Account)

Location: adl-silverlayer-test-01

Search blobs by prefix (case-sensitive) Show deleted objects

	Name	Modified	Access tier	Archive status	Blob type	Size	Lease state
<input type="checkbox"/>	ProductInfo						- ...
<input type="checkbox"/>	ProductReviews						- ...

Figura 22. Tablas creadas en la silver layer posterior al procesado

Se puede apreciar cómo las tablas se han creado como externas en la storage account en la silver layer.

4.1.3 Transformación y enriquecimiento de datos

En las arquitecturas Delta Lake, la transición de la capa Silver a la capa Gold generalmente implica un conjunto de transformaciones de datos más avanzadas y agregaciones para preparar los datos para análisis y presentación. La capa Silver ya contiene datos limpios y enriquecidos, pero la capa Gold se centra en resumir y organizar estos datos de manera que sean fácilmente consumibles por las aplicaciones de análisis y los usuarios finales.

En el caso de estudio, la etapa de transformación entre la capa Silver y la capa Gold tiene como objetivo crear dos tablas en la capa Gold: **ProductSummary** y **UserActivity**. Esta etapa implica una serie de transformaciones y agregaciones realizadas en las tablas de origen de la capa Silver (**ProductInfo** y **ProductReviews**).

4.1.3.1 Transformaciones para la tabla ProductSummary

La tabla **ProductSummary** se genera a partir de un join interno entre las tablas **ProductInfo** y **ProductReviews** en la capa Silver utilizando el campo **product_id** como clave.

1. **Join Interno:** Se realiza un join interno entre **ProductInfo** y **ProductReviews** usando **product_id**.
2. **Agrupación:** Se agrupan los datos por **product_id** y **product_name**.
3. **Agregación:** Se llevan a cabo las siguientes agregaciones:
 - **average_rating:** Calcula la media de las puntuaciones utilizando la función **avg** sobre el campo **rating**.
 - **total_reviews:** Cuenta el número total de reseñas utilizando la función **count** en el campo **review_id**.
 - **total_rating_count:** Suma el total de votos de calificación utilizando la función **sum** sobre el campo **rating_count**.

Estas métricas agregadas forman los campos de la nueva tabla **ProductSummary** en la capa Gold.

4.1.3.2 Transformaciones para la tabla UserActivity

La tabla **UserActivity** se genera a partir de la tabla **ProductReviews** de la capa Silver.

1. **Agrupación:** Se agrupan los datos por **user_id** y **user_name**.
2. **Agregación:** Se llevan a cabo las siguientes agregaciones:
 - **total_reviews_written:** Calcula el número total de reseñas escritas por cada usuario utilizando la función **count** sobre el campo **review_id**.
 - **average_rating_given:** Calcula la calificación promedio dada por cada usuario utilizando la función **avg** sobre el campo **rating**.

Estas métricas agregadas forman los campos de la nueva tabla **UserActivity** en la capa Gold.

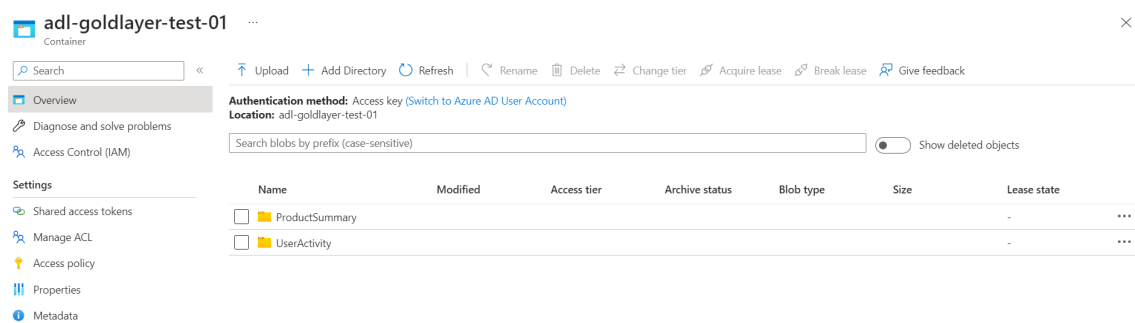


Figura 23. Tablas creadas en la gold layer posterior al procesado

De esta manera, en la capa gold tenemos valores agregados y listos para el consumo de los departamentos más funcionales.

4.1.4 Análisis de datos

En una configuración de análisis de datos típica, la capa Gold representa una fuente optimizada para el análisis y la toma de decisiones. Se crean notebooks o se linkean a herramientas que van a permitir sacar valor del dato.

En este caso específico, nos hemos centrado en un análisis más descriptivo y exploratorio utilizando los datos en las tablas **ProductSummary** y **UserActivity** de la capa Gold. Aquí hay un resumen de lo que se ha hecho:

1. **Top 10 Productos por Calificación Promedio:** Se identificaron los 10 productos con las calificaciones más altas para entender qué productos son más apreciados por los clientes. Esto podría ser útil para estrategias de promoción y stock.
2. **Top 5 Usuarios más Activos:** Se identificaron los 5 usuarios más activos en términos de reseñas escritas. Conocer a los usuarios más activos puede ser beneficioso para estrategias de engagement y marketing.
3. **Resumen de Categorías:** Se generó un resumen que muestra la calificación promedio y el número total de reseñas por categoría de producto. Esto proporciona una vista de

alto nivel del rendimiento de diferentes categorías, lo cual es esencial para la toma de decisiones estratégicas.

4. **Visualizaciones:** Cada uno de estos análisis fue acompañado de una visualización para facilitar la interpretación de los datos.

Las transformaciones y análisis se realizaron utilizando PySpark, y se generaron visualizaciones básicas para acompañar los insights. Estas son operaciones relativamente sencillas y representan solo la punta del iceberg de lo que podría hacerse en términos de análisis de datos en la capa Gold.

4.1.5 Orquestación con airflow

Para la orquestación con airflow, en primer lugar se configuran las credenciales de los clústers de Databricks para poder acceder a los notebooks desde el contenedor de Airflow ejecutándose en el entorno local.

Una vez se ha configurado la conexión, se crea un script de Python que se encargará de llamar a los distintos notebooks de databricks y permite organizar su orquestación.

Al crear un dag hay que configurar los siguientes parámetros (entre otros):

- **Fecha de inicio:** Para el caso práctico se va a usar el primer día de 2023, aunque se podrían usar fechas anteriores si se tuvieran datos de esos periodos.
- **Schedule Interval:** La recurrencia de la ejecución del DAG en formato Chron expression. En este caso se ha hecho un Schedule diario y se ha parado manualmente la ejecución para evitar repetir el proceso.

Para llamar los notebooks es necesario usar el operador '**DatabricksSubmitRunOperator**' que indicando la ruta y el clúster de cómputo a usar, realiza una llamada que lanza el proceso. El estado de los Jobs se puede visualizar tanto desde airflow como desde el workspace de Databricks.

Para el caso de estudio, se diseñó un DAG que consta de varias tareas:

1. **Pre-procesamiento en Databricks:** Ejecuta el notebook de Databricks **ETL_bronze_silver** transformar los datos de la capa de bronce a la capa de plata.
2. **Transformación y Enriquecimiento de Datos:** Ejecuta el notebook de Databricks **ETL_silver_gold** que realiza las transformaciones adicionales y el enriquecimiento de datos para mover los datos de la capa de plata a la capa de oro.

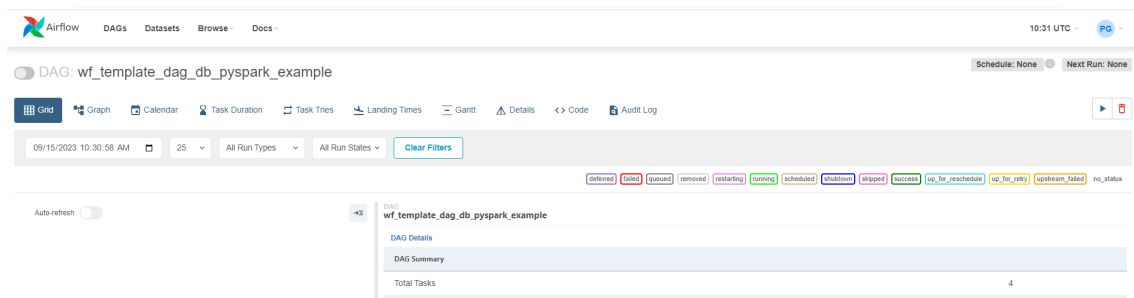


Figura 24. Ejemplo DAG en el webserver de Airflow

4.1.6 Validación

La validación en un proyecto de creación de infraestructura end to end de datos como este es crítica para asegurar que los datos se muevan y transformen correctamente a través de las distintas capas y que los insights generados sean fiables. Aquí se describe cómo se llevó a cabo la validación en este caso de estudio.

4.1.6.1 Validación del Código

- **Revisiones de Código:** Se realizó una revisión de código minuciosa para asegurarse de que las transformaciones de datos y las operaciones de análisis se implementaran según las especificaciones.
- **Pruebas Unitarias:** Se ejecutaron pruebas unitarias en fragmentos específicos de código, particularmente aquellos que llevan a cabo transformaciones importantes de datos, para asegurarse de que funcionan como se esperaba.
- **Logs y Alertas:** Se configuraron logs y alertas para capturar y notificar cualquier error o anomalía durante la ejecución.

4.1.6.2 Validación de Datos en Blob Storage

- **Consistencia de Datos en Contenedores:** Se verificó que los datos se cargaran en los contenedores de blob correctos correspondientes a cada capa (Bronze, Silver, Gold).
- **Integridad de Datos:** Se realizó una revisión manual de una muestra de registros para asegurarse de que los datos se transformaran y almacenaran correctamente a través de las capas.
- **Verificación de Particiones:** Dado que las tablas se habían particionado por ciertos campos (como **category** para productos), se verificó que la partición se llevara a cabo correctamente.

4.1.6.3 Validación de la Ejecución del Workflow

- **Monitorización en Tiempo Real:** Se utilizó la ui de Data Factory y Data Bricks para rastrear la ejecución de los workflows en tiempo real.
- **Verificación End-to-End:** Se ejecutó el flujo de trabajo de principio a fin en un ambiente de pruebas para asegurarse de que todos los componentes (Data Factory, Databricks, Airflow) trabajaran juntos sin problemas.

5 Conclusiones

5.1 Conclusiones del trabajo realizado

El proyecto presentado ha sido una tarea altamente compleja y desafiante, principalmente por la falta de documentación y ejemplos previos en los que apoyarse. La necesidad de innovar e idear nuevas formas de abordar problemas ha sido un elemento constante en este trabajo. Por ejemplo, la implementación de dos contenedores Docker distintos, uno para manejar el despliegue de los componentes de Azure a través del SDK de Python y otro para gestionar el orquestador Airflow, fue una solución original y eficiente que permitió desacoplar distintas responsabilidades del sistema e integrar una solución con microservicios de cara a Airflow.

Esta complejidad no solo añadió una capa de dificultad sino que también presentó una oportunidad para la innovación.

El hecho de que este trabajo sea pionero en su ámbito tiene múltiples implicaciones. Primero, siendo de código abierto, establece un precedente y puede servir como referencia para futuros proyectos similares. Segundo, al ser uno de los primeros en abordar estos desafíos, hay una oportunidad de liderazgo y de establecimiento de estándares en la comunidad.

Aunque se han hecho avances significativos, el proyecto aún está en una etapa temprana, en particular con respecto a la automatización total del flujo de trabajo. Esta limitación es en parte debida a las capacidades actuales del SDK de Python proporcionado por Azure, lo cual restringe el grado de automatización que se puede alcanzar en este momento. Esto, sin embargo, también señala un área para futuras mejoras, tanto por parte de los servicios de Azure como en futuras iteraciones del proyecto.

La elección de utilizar una arquitectura moderna como Delta Lake merece una mención especial. Delta Lake ofrece robustez, escalabilidad y rendimiento, y se está posicionando como el estándar para los lagos de datos. La decisión de utilizar esta arquitectura no solo valida la elección técnica sino que también apunta a una tendencia más amplia en la industria del tratamiento de datos. Estamos en un momento de transición hacia arquitecturas más eficientes y flexibles, y este proyecto es un claro indicativo de hacia dónde se dirige el futuro de los ecosistemas de datos.

En general, este proyecto ha sido tanto un desafío técnico como una oportunidad para innovar en el espacio de la gestión de datos y la automatización. Ha demostrado las limitaciones actuales de las herramientas disponibles, pero también ha mostrado lo que es posible cuando se aplica ingenio y pensamiento estratégico. La modernidad de la arquitectura elegida coloca al proyecto en la vanguardia de las tendencias actuales, mientras que su enfoque pionero ofrece múltiples oportunidades para futuras investigaciones y mejoras en este espacio en rápido desarrollo.

5.2 Puntos fuertes y débiles del trabajo realizado

5.2.1 Puntos fuertes

1. Innovación y Pionería:

- Este proyecto es uno de los primeros en abordar el tipo de flujo de datos y arquitectura elegidos, lo cual lo convierte en una fuente de referencia invaluable para futuros desarrolladores en este ámbito. La falta de documentación y ejemplos previos ha hecho que este proyecto sea una forja de nuevos caminos, lo cual tiene un gran valor inherente.

Para añadir otro ejemplo de innovación, el tipo de conexión usado para conectar azure blob storage y Databricks hace apenas unos pocos meses que existe.

2. Cobertura Integral del Ciclo de Vida de los Datos:

- El proyecto no se limita a un aspecto particular del flujo de datos, sino que cubre todo el ciclo de vida, desde la ingesta hasta el análisis. Esta visión integral es crucial para entender los matices y las interdependencias dentro del ecosistema de datos. Permite, además, una comprensión más profunda de cómo cada componente afecta al todo.

3. Rigurosidad del Caso de Estudio:

- A través de un caso de estudio muy completo, el proyecto no solo demuestra la viabilidad de la arquitectura elegida, sino que también ilustra su potencial para resolver problemas reales. El caso de estudio actúa como una validación práctica de la teoría y la tecnología empleadas, lo que aporta un valor adicional al trabajo.

5.2.2 Puntos Débiles

1. Dependencia de Azure:

- Aunque Azure es una plataforma robusta y ampliamente utilizada, la alta dependencia del proyecto en sus servicios podría ser vista como una limitación. Esto reduce la portabilidad del proyecto hacia otras plataformas de la nube y podría excluir a aquellos interesados que utilicen otras infraestructuras.

2. Interacción Manual Requerida:

- Uno de los objetivos deseables en cualquier proyecto de automatización de flujo de datos es minimizar la necesidad de intervención manual. En este caso, no se ha logrado una automatización completa, lo que puede ser una barrera para implementaciones donde la eficiencia y la escalabilidad son críticas.

3. Complejidad Técnica y Barreras de Entrada:

- Dado el nivel de complejidad técnica del proyecto, se requiere un grado considerable de conocimientos especializados tanto en ciencia de datos como en tecnologías de la nube para implementar y mantener la solución. Esta barrera de entrada podría limitar su adopción entre pequeñas organizaciones o individuos con recursos más limitados.

Al evaluar los puntos fuertes frente a los débiles, se observa un claro patrón de compromisos entre la innovación y la aplicabilidad práctica. Si bien el proyecto rompe nuevos terrenos y ofrece un profundo nivel de comprensión, también resalta áreas donde existen desafíos prácticos y limitaciones inherentes.

5.3 Limitaciones del trabajo realizado

En un mundo ideal, cualquier sistema de procesamiento y análisis de datos funcionaría de manera completamente autónoma, sin requerir la intervención manual del usuario. Sin embargo, este proyecto aún necesita cierto grado de interacción humana para su funcionamiento. Aunque la automatización es alta, la necesidad de ciertas entradas manuales podría considerarse un cuello de botella, especialmente en un entorno empresarial donde la eficiencia y la velocidad son cruciales. Esta limitación también plantea preguntas sobre la escalabilidad del sistema. Si se requiere intervención manual a medida que el sistema se escala, esto podría llevar a ineficiencias y aumentar la probabilidad de errores humanos.

El proyecto es una implementación compleja que involucra una serie de tecnologías avanzadas y conceptos de arquitectura. Desde la comprensión de los contenedores Docker hasta la interacción con Azure y el dominio del procesamiento de datos a gran escala, la barrera de entrada es alta. Esto significa que la solución es menos accesible para organizaciones o individuos que no tienen los recursos para emplear o formar personal con estas habilidades especializadas. La complejidad técnica, aunque es un reflejo de la robustez y la profundidad del proyecto, también es una limitación en términos de quién puede implementar o adaptar el trabajo realizado.

Aunque el proyecto se ha centrado en utilizar Azure como su principal proveedor de servicios en la nube, esto en sí mismo puede considerarse una limitación. Primero, se reduce la flexibilidad para integrar componentes de otros proveedores de servicios en la nube que puedan ofrecer características más adecuadas para ciertos escenarios. En segundo lugar, esta dependencia implica que las organizaciones interesadas en el proyecto deben estar dispuestas a comprometerse con el ecosistema de Azure. Esto podría ser problemático para organizaciones que ya han invertido en otra infraestructura de nube o que tienen requisitos que no se pueden cumplir con los servicios de Azure disponibles.

5.4 Líneas de contexto de trabajo

5.4.1 Ejecución en Varios Proveedores de Nube

La capacidad de ejecutar la solución en múltiples proveedores de servicios en la nube es un gran paso hacia la portabilidad y la resiliencia del sistema. Si bien el proyecto actual está centrado en Azure, diversificarlo hacia otros proveedores como AWS o GCP añadiría una capa extra de flexibilidad y seguridad. Además, permitiría a las empresas evitar el bloqueo de proveedores y aprovechar las mejores ofertas y servicios de cada plataforma.

5.4.2 Elección de Componentes Personalizados

Permitir una selección más amplia de componentes dentro del stack tecnológico brinda más opciones de personalización y optimización. Esto puede ser particularmente útil para organizaciones que tienen requisitos muy específicos en términos de almacenamiento de datos, rendimiento o seguridad. Por ejemplo, algunos componentes podrían ser más adecuados para manejar grandes volúmenes de datos, mientras que otros podrían estar optimizados para un acceso más rápido.

5.4.3 Despliegue de Entornos Dev y Pro Interconectados

Tener entornos de desarrollo y producción separados pero interconectados ofrece una serie de ventajas. Permite un ciclo de desarrollo más rápido, donde los cambios pueden ser probados en un entorno controlado antes de ser lanzados en producción. También minimiza los riesgos asociados con el lanzamiento de nuevas características o cambios en el código, ya que estos se pueden probar exhaustivamente en el entorno de desarrollo antes de que afecten al entorno de producción.

5.4.4 Integración con Herramientas de CI/CD como Travis

La adopción de prácticas de Integración Continua y Despliegue Continuo (CI/CD) mediante herramientas como Travis automatiza el proceso de desarrollo y despliegue, aumentando así la eficiencia y la confiabilidad del sistema. Esta integración asegura que cualquier cambio en el código se pruebe automáticamente, lo que reduce el margen de error humano y acelera el tiempo de llegada al mercado de nuevas características o correcciones.

5.4.5 Integración con Herramientas de Control de Versiones como GitHub

Incorporar un sistema de control de versiones como GitHub mejora la colaboración y la gestión del proyecto. Además de rastrear los cambios en el código, permite la colaboración en tiempo real entre los miembros del equipo, lo que es crucial para proyectos complejos y multifacéticos. También añade un nivel de seguridad al permitir que los cambios se revisen y aprueben antes de su incorporación final al proyecto.

Cada una de estas líneas de continuación añade una capa adicional de robustez, eficiencia y flexibilidad al proyecto, adaptándolo mejor a un paisaje tecnológico en constante evolución y haciéndolo más atractivo para una gama más amplia de aplicaciones y contextos empresariales

6 Referencias

- [1] <https://www.bing.com/search?pglt=161&q=big+data+layers&cvid=e7c8000591fa4c4ca1c551836469d8c7&aqs=edge.0.0l9j69i11004.5302j0j1&FORM=ANNAB1&PC=U531>
- [2] <https://learn.microsoft.com/en-us/azure/azure-resource-manager/templates/template-tutorial-create-first-template?tabs=azure-powershell>
- [3] [Definición de su convención de nomenclatura - Cloud Adoption Framework | Microsoft Learn](#)
- [4] <https://learn.microsoft.com/en-us/cli/azure/create-an-azure-service-principal-azure-cli>
- [5] <https://learn.microsoft.com/en-us/azure/developer/python/sdk/examples/azure-sdk-example-virtual-machines?tabs=cmd#3-write-code-to-provision-a-virtual-machine>
- [6] <https://learn.microsoft.com/en-us/azure/azure-resource-manager/templates/deploy-python>
- [7] <https://learn.microsoft.com/en-us/python/api/azure-core/azure.core.polling.lropoller?view=azure-python&viewFallbackFrom=azure-python-preview>
- [8] <https://learn.microsoft.com/en-us/azure/role-based-access-control/role-assignments-cli>
- [9] <https://learn.microsoft.com/en-us/azure/databricks/data-governance/unity-catalog/azure-managed-identities>
- [10] <https://learn.microsoft.com/es-es/azure/data-factory/connector-azure-blob-storage?tabs=data-factory>
- [11] [CREATE TABLE \[USING\] - Azure Databricks - Databricks SQL | Microsoft Learn](#)
- [12] [Connect to Azure Data Lake Storage Gen2 and Blob Storage - Azure Databricks | Microsoft Learn](#)

7 Apéndices y Anexos

Se puede encontrar todo el desarrollo del proyecto en el siguiente repositorio:
https://github.com/pgraciae/MBD_LakeHouseDeploymentModel.git

Dentro del repositorio se puede encontrar documentación y explicación detallada del funcionamiento y estructura del mismo.