1. Justificación del Uso de la Nube

1.1. Recordatorio del proyecto y contexto de migración

El proyecto desarrolla un flujo completo de Inteligencia de Negocios para monitorear los niveles de madurez **Practitioner** y **Continuous Integration (CI)** del BBVA, utilizando en su versión local:

- Ingesta manual de archivos CSV exportados desde el **Marco Playbook**.
- Procesos ETL en **PySpark** sobre un entorno Hadoop/Hortonworks con **Zeppelin**.
- Un **Data Warehouse relacional** con modelo estrella implementado en **PostgreSQL**.
- Una **API en Flask** y un **dashboard web en React** para la visualización de KPIs.

La migración a la nube de Azure busca:

- Escalar el procesamiento distribuido de datos sin administrar servidores físicamente.
- Simplificar la gestión de datos mediante un **Data Lake** y un **DW administrado en PostgreSQL**.
- Mejorar la **seguridad, segmentación de red, cumplimiento y auditoría** usando servicios como
  Virtual Network, Network Security Groups, Azure Firewall, Private Endpoints, Private DNS Zone, Azure Key Vault, Azure Monitor, Log Analytics Workspace, Recovery Services Vault y Audit Logs.
- Mantener una capa de exposición moderna basada en **Container App + Static Web Apps** para la interfaz Flask/React.
- Reducir el mantenimiento de infraestructura on-premise y riesgos asociados (hardware, energía, respaldos manuales, etc.).
- Garantizar **costos controlables y trazables** mediante monitoreo y métricas centralizadas.

Para seleccionar el proveedor cloud se evaluaron tres opciones líderes del mercado: **AWS**, **Microsoft Azure** y **Google Cloud Platform (GCP)**, considerando siete características críticas para soluciones de **BI** y **Big Data**.

1.2. Criterios de evaluación utilizados

Los 7 criterios aplicados en la comparativa fueron:

1. Seguridad y cumplimiento normativo
2. Escalabilidad y disponibilidad
3. Pricing y modelo de costos
4. Ecosistema de BI y análisis de datos
5. Servicios Big Data y procesamiento distribuido

6. Soporte y madurez de servicios
7. Facilidad de migración y compatibilidad

1.3. Análisis por criterio

1.3.2. Criterio 1: Seguridad y cumplimiento normativo

Los tres proveedores cuentan con catálogos amplios de certificaciones y servicios de seguridad. Sin embargo, en este proyecto el énfasis está en:

- **Aislamiento de red** (segmentación por VNet, subredes y reglas de tráfico).
- **Acceso privado a servicios PaaS** (Private Endpoints + Private DNS Zone).
- **Gestión segura de secretos** (Key Vault).
- **Supervisión centralizada** (Azure Monitor + Log Analytics + Audit Logs).
- **Respaldo y recuperación ante desastres** (Recovery Services Vault).

Comparación resumida

Aspecto clave	**AWS**	**Azure**	**GCP**
Enfoque de seguridad	Amplio portafolio de servicios de seguridad y compliance	Enfoque de plataforma integrada: red privada (VNet), segmentación (NSG), Firewall, endpoints privados, Key Vault y monitoreo unificado	Fuerte foco en protección de datos y seguridad gestionada para workloads
Segmentación de red	VPC, Security Groups, NACLs	Virtual Network, subredes, Network Security Groups y Azure Firewall para definir perímetros y políticas de tráfico	VPC, firewall de red, reglas de IAM y políticas de organización
Acceso privado a servicios gestionados	Endpoints privados para ciertos servicios PaaS	Private Endpoints + Private DNS Zone para exponer servicios solo dentro de la VNet	Private Service Connect para acceso privado
Gestión de secretos	AWS KMS y Secrets Manager	Azure Key Vault como almacén central cifrado de secretos, claves y certificados, integrado con los servicios de la solución	Cloud KMS y Secret Manager

Aspecto clave	AWS	Azure	GCP
Monitoreo y auditoría	CloudTrail, CloudWatch	Azure Monitor + Log Analytics Workspace + Audit Logs para métricas, logging y trazabilidad de acciones sobre recursos	Cloud Logging, Cloud Monitoring
Respaldo y recuperación	Backup y servicios de DR específicos	Recovery Services Vault para backups y recuperación de máquinas y bases de datos, con retención configurable	Backup/DR gestionado por servicios individuales
Alineamiento con entorno bancario	Alto	Muy alto: modelo pensado para entornos enterprise regulados, con integración nativa de red privada, claves, monitoreo, auditoría y respaldo central	Medio-alto

Conclusión

En un contexto similar al de BBVA, donde la seguridad, el aislamiento de red y la trazabilidad son críticos, **Azure** ofrece una combinación sólida de:

- **Segmentación de red** (VNet, NSG, Firewall).
- **Acceso privado a datos y servicios** (Private Endpoints, Private DNS Zone).
- **Protección de secretos** (Key Vault).
- **Monitoreo y auditoría centralizados** (Azure Monitor, Log Analytics, Audit Logs).
- **Respaldo y recuperación** (Recovery Services Vault).

Esto permite construir un flujo BI de uso interno con un nivel de gobierno y cumplimiento alineado a entornos financieros.

1.3.3. Criterio 2: Escalabilidad y disponibilidad

Los tres proveedores dominan el mercado de nube y ofrecen altos niveles de disponibilidad. Para este proyecto, los aspectos más relevantes son:

- Escalar el **procesamiento distribuido** (PySpark en Databricks).
- Escalar el **Data Warehouse relacional** (PostgreSQL administrado).
- Publicar el **backend Flask** y el **frontend React** en servicios gestionados, sin administrar servidores físicos.

- Proteger todo dentro de una **VNet** con controles de tráfico y monitoreo.

Comparación resumida

Aspecto clave	**AWS**	**Azure**	**GCP**
Cobertura global	Líder histórico, gran número de regiones y zonas	Número de regiones muy alto, con fuerte presencia enterprise y escenarios híbridos	Menos regiones, red muy optimizada
Escalabilidad de cómputo	EC2 + Auto Scaling, ECS/EKS, Lambda	Servicios de cómputo y contenedores desplegados dentro de una VNet, con reglas de NSG y Firewall; escalado horizontal configurado según la demanda	GKE, Cloud Run, App Engine, Cloud Functions
Escalabilidad de datos	S3 + Redshift/Aurora/RDS	Azure Data Lake + PostgreSQL administrado, expuestos por Private Endpoints dentro de la VNet	Cloud Storage + BigQuery/Cloud SQL
Disponibilidad gestionada	Alta, con SLAs por servicio	Alta, con SLAs competitivos y posibilidad de combinarlo con Recovery Services Vault para mejorar la estrategia de continuidad	Muy alta en servicios analíticos serverless

Conclusión

Para un flujo **ETL + Data Lake + DW + API + frontend**, Azure permite:

- Desplegar los componentes críticos dentro de una **red privada (VNet)**.
- Proteger el acceso con **NSG + Azure Firewall**.
- Exponer servicios PaaS mediante **Private Endpoints**.
- Monitorizar el comportamiento con **Azure Monitor + Log Analytics**.

Todo ello sin necesidad de administrar hardware propio y manteniendo la posibilidad de escalar los recursos a medida que crecen los volúmenes de datos o usuarios.

1.3.4. Criterio 3: Pricing y modelo de costos

AWS, Azure y GCP utilizan un modelo de **pago por uso**, con la opción de descuentos por compromisos de uso (reservas) y herramientas de control de costos.

Para este proyecto interesa:

- Poder **empezar pequeño** y crecer con el tiempo.
- Pagar principalmente por **ejecuciones de Databricks** y capacidad del **DW PostgreSQL**.
- Tener trazabilidad de los recursos que más consumen mediante **métricas y logs centralizados**.

Comparación resumida

Aspecto clave	**AWS**	**Azure**	**GCP**
Modelo base	Pay-as-you-go + Reserved Instances / Savings Plans	Pay-as-you-go + reservas por capacidad, con métricas unificadas vía Azure Monitor / Log Analytics	Pay-as-you-go + descuentos por uso sostenido
Visibilidad de costos	Cost Explorer, Budgets	Portal unificado con vistas de consumo por recurso, grupo de recursos o etiqueta, apoyado en métricas y logs de Azure Monitor	Herramientas de presupuesto y uso
Ajuste a cargas BI tipo batch	Viable, cuidando recursos siempre encendidos	Viable: se combinan SKUs pequeños, escalado por demanda y políticas de apagado de entornos no productivos	Muy competitivo en escenarios 100 % serverless

Conclusión

Aunque **GCP** resulta muy atractivo en escenarios totalmente serverless, en un proyecto que combina **Spark + DW relacional + aplicaciones web**, **Azure** ofrece un equilibrio razonable entre:

- Flexibilidad para crecer de forma gradual.
- Herramientas nativas para entender dónde se está gastando (métricas y logs centralizados).

- Opciones de reservas en aquellas capas que luego se estabilicen.

1.3.5. Criterio 4: Ecosistema de BI y análisis de datos

El proyecto actual construye dashboards y tablas interactivas mediante un **frontend React** publicado en **Static Web Apps**, consumiendo datos procesados por Databricks y almacenados en PostgreSQL.

Aun así, se consideró el **ecosistema de BI** de cada nube a nivel estratégico (posible evolución futura):

- AWS: QuickSight como solución BI principal.
- Azure: ecosistema BI de Microsoft, integrado con plataformas de datos y servicios de Azure.
- GCP: Looker y Looker Studio, muy orientados a entornos data/analytics.

Conclusión

Aunque la versión actual del proyecto utiliza un **dashboard propio (React)**, elegir Azure mantiene abierta la posibilidad de integrar, a futuro, herramientas BI del ecosistema Microsoft de forma natural, sin cambiar de proveedor de nube.

1.3.6. Criterio 5: Servicios Big Data y procesamiento distribuido

La arquitectura local está basada en **Spark sobre Hadoop**. En la nube se busca:

- Mantener la lógica en **PySpark**.
- Dejar de manejar clústeres manualmente.
- Aprovechar un **Data Lake** con capas Bronze/Silver/Gold.

En Azure, esto se resuelve con:

- **Azure Databricks** como entorno gestionado para notebooks PySpark.
- **Azure Data Lake** para el almacenamiento en crudo y transformado.

Comparación resumida

Aspecto	AWS	Azure	GCP
Spark gestionado	EMR, Glue	Azure Databricks (servicio de primer nivel en la plataforma)	Dataproc
Integración con Data Lake	S3	Azure Data Lake con soporte natural para arquitectura Medallion	Cloud Storage

Aspecto	AWS	Azure	GCP
Continuidad con arquitectura local	Requiere adaptar entorno	Migra notebooks PySpark y reemplaza HDFS por Data Lake con cambios mínimos	Requiere adaptar a Dataproc

Conclusión

Para un flujo **HDFS + PySpark + DW** ya existente, **Azure Databricks + Azure Data Lake** permiten migrar la lógica ETL casi de forma directa, añadiendo además:

- Mejor gobernanza de datos.
- Mejor integración con la capa de seguridad y red (VNet, Key Vault, Monitor).

1.3.7. Criterio 6: Soporte y madurez de servicios

En cuota de mercado y madurez:

- **AWS** sigue liderando en participación.
- **Azure** se sitúa en segundo lugar con fuerte foco enterprise.
- **GCP** ocupa el tercer lugar, con foco en datos e IA.

Los tres son proveedores "tier 1", pero Azure está especialmente bien posicionado en entornos corporativos que ya utilizan tecnologías Microsoft.

Conclusión

Para un proyecto que desea parecerse a cómo trabajaría una empresa grande (como un banco) que ya usa herramientas Microsoft, **Azure** ofrece una alineación más directa en:

- Prácticas recomendadas.
- Documentación y ejemplos.
- Ecosistema de partners y soporte corporativo.

1.3.8. Criterio 7: Facilidad de migración y compatibilidad

El stack actual está conformado por:

- **PySpark + Zeppelin** (ETL).
- **HDFS** (staging / data lake).
- **PostgreSQL** (Data Warehouse relacional con modelo estrella).
- **Flask** (backend).
- **React** (frontend).

Comparación resumida

Componente actual	AWS	Azure	GCP
PySpark + Zeppelin	EMR Notebooks / Glue	Azure Databricks (notebooks PySpark gestionados)	Dataproc + Notebooks
HDFS (Hortonworks)	S3	Azure Data Lake	Cloud Storage
PostgreSQL (DW)	RDS PostgreSQL / Redshift	PostgreSQL administrado en Azure (servicio PaaS)	Cloud SQL / BigQuery
Flask + React	ECS/EKS, Elastic Beanstalk, Amplify	Azure Container Apps + Static Web Apps	Cloud Run / GKE

Conclusión

La migración "uno a uno" del stack actual es especialmente directa hacia **Azure**:

- ETL PySpark → notebooks en **Azure Databricks**.
- HDFS → **Azure Data Lake**.
- PostgreSQL local → **PostgreSQL administrado en Azure**.
- Flask + React → **Container Apps + Static Web Apps**, con tráfico protegido dentro de una **VNet** y expuesto sólo a través de los puntos que se definan.

Esto reduce retrabajo y permite concentrarse en mejorar el flujo BI, no en reescribir toda la solución desde cero.

1.4. Matriz de Decisión

Se asignaron pesos porcentuales a cada uno de los 7 criterios según su relevancia para el proyecto y se calificó a cada proveedor en una escala 1–4:

- 1 = Regular
- 2 = Bueno
- 3 = Muy bueno
- 4 = Excelente

1.4.1. Evaluación ponderada por criterio

Característica	Peso	GCP	AWS	Azure
Seguridad y Cumplimiento Normativo	20 %	3	4	4

Característica	Peso	GCP	AWS	Azure
Escalabilidad y Disponibilidad	18 %	4	4	4
Pricing y Modelo de Costos	16 %	3	3	4
Ecosistema de BI y Análisis de Datos	15 %	3	3	4
Servicios Big Data y Procesamiento Distribuido	15 %	4	3	4
Soporte y Madurez de Servicios	10 %	3	4	4
Facilidad de Migración y Compatibilidad	6 %	3	4	4

Resumen de resultados

Proveedor	Puntuación ponderada (1–4)
GCP	3.33
AWS	3.54
Azure	4.00

Azure obtiene la puntuación máxima posible (**4.00**), superando a **AWS (3.54)** y **GCP (3.33)** en la evaluación global, de acuerdo con los pesos definidos para los criterios del proyecto.

1.5. Decisión Final: Microsoft Azure

1.5.1. Justificación técnica

Azure obtiene la puntuación más alta porque ofrece la combinación óptima de características para este caso de uso:

1. **Plataforma de datos y procesamiento alineada con el stack actual**
    - **Azure Databricks** permite ejecutar los notebooks PySpark existentes con cambios mínimos.
    - **Azure Data Lake** soporta la arquitectura **Medallion (Bronze, Silver, Gold)** para almacenar datos crudos y transformados.
    - Un **PostgreSQL administrado en Azure** permite mantener el modelo estrella para KPIs Practitioner/CI.
2. **Capa de red y seguridad de nivel corporativo**

- Todos los componentes se despliegan dentro de una **Virtual Network** con subredes específicas.
- El tráfico se controla con **Network Security Groups** y **Azure Firewall**, reduciendo la exposición a internet.
- El acceso a servicios PaaS se realiza mediante **Private Endpoints** y resolución mediante **Private DNS Zone**, evitando endpoints públicos.
- Las credenciales sensibles se almacenan en **Azure Key Vault**.
- La operación se supervisa con **Azure Monitor**, **Log Analytics Workspace** y **Audit Logs**.
- Los backups y la recuperación ante desastres se gestionan con **Recovery Services Vault**.

3. **Arquitectura de exposición moderna**
   - El **backend Flask** se ejecuta en **Azure Container Apps**.
   - El **frontend React** se publica en **Azure Static Web Apps**, optimizado para SPAs.
   - Ambos se integran con la VNet y con la capa de seguridad para exponer únicamente lo necesario.

En conjunto, Azure permite reproducir y mejorar la arquitectura local, pero con una red privada, controles de seguridad más finos, observabilidad centralizada y servicios gestionados.

1.5.2. Justificación financiera

A nivel financiero, Azure aporta:

- Un modelo **pay-as-you-go** que permite comenzar con tamaños de clúster y bases de datos modestos y crecer según la volumetría real.
- Posibilidad de aplicar **descuentos por reserva de capacidad** en los componentes que se estabilicen (por ejemplo, DW PostgreSQL o capacidad de contenedores).
- **Métricas y logs centralizados** (Azure Monitor + Log Analytics) para identificar rápidamente qué servicios son responsables de la mayor parte del consumo y ajustar el dimensionamiento.
- Capacidad de **apagar o reducir** entornos no productivos (dev/test) fuera de horario, evitando pagar por recursos ociosos.

Esto facilita mantener controlado el presupuesto del proyecto y justificar los costos frente al equivalente on-premise.

1.5.3. Alineación con requerimientos del proyecto

El proyecto requiere migrar el pipeline:

Ingesta CSV → Procesamiento Spark → Data Lake (Bronze/Silver/Gold) → DW PostgreSQL (modelo estrella) → API Flask → Dashboard React

Mapeo a Azure:

- **Ingesta y Data Lake**
    - Los CSV del Marco Playbook se cargan y almacenan en **Azure Data Lake**, donde se organizan en capas Bronze/Silver/Gold.
- **Procesamiento distribuido**
    - La lógica ETL en PySpark se migra a **Azure Databricks**, manteniendo el mismo lenguaje y patrón de notebooks, pero aprovechando un servicio gestionado.
- **Data Warehouse relacional**
    - El modelo estrella se implementa en **PostgreSQL administrado en Azure**, conservando la estructura de hechos y dimensiones.
- **Capa de exposición**
    - El **backend Flask** se despliega en **Azure Container Apps**.
    - El **frontend React** se hospeda en **Azure Static Web Apps**, consumiendo la API expuesta desde Container Apps.
- **Seguridad, red y operación**
    - Todo se ejecuta dentro de una **VNet** protegida con **NSG** y **Azure Firewall**.
    - Los servicios PaaS se consumen mediante **Private Endpoints** y **Private DNS Zone**.
    - Los secretos se gestionan en **Azure Key Vault**.
    - La observabilidad y trazabilidad se centralizan en **Azure Monitor, Log Analytics Workspace y Audit Logs**.
    - Los respaldos y la recuperación ante desastres se gestionan con **Recovery Services Vault**.

La combinación de:

- **Alineación técnica** con el stack actual,
- **Arquitectura segura de red y datos**, y
- **Modelo económico flexible y monitorizable**,

Justifica de forma sólida la selección de **Microsoft Azure** como plataforma cloud para este caso de uso.

2. SELECCIÓN DE SERVICIOS CLOUD

2.1. Visión general de servicios Azure seleccionados

La siguiente tabla resume la correspondencia entre la arquitectura local y los servicios de Azure utilizados en la migración, organizada por capas del flujo BI.

Tabla 2.1 – Mapeo de componentes locales a servicios Azure

Capa funcional	Componente local	Servicio Azure seleccionado	Rol dentro del flujo BI
Ingesta / almacenamiento bruto	Sistema de archivos local (uploads de CSV)	**Azure Data Lake Storage Gen2 – Capa Bronze**	Punto de entrada de los archivos CSV del Marco Playbook; almacena los datos crudos tal como llegan.
Data Lake	HDFS (Hortonworks)	**Azure Data Lake Storage Gen2 – Capas Bronze, Silver y Gold**	Almacenamiento distribuido para datos crudos, limpios y curados bajo el modelo Medallion.
Procesamiento ETL	Apache Zeppelin + Apache Spark	**Azure Databricks**	Ejecución de notebooks PySpark para limpieza, transformación y cálculo de KPIs.
Data Warehouse (DW)	PostgreSQL en contenedor Docker	**Azure Database for PostgreSQL (servicio PaaS)**	Almacenamiento relacional del modelo dimensional tipo estrella.
Backend API	Flask en contenedor Docker	**Azure Container Apps**	Hosting del backend (API Flask) que expone KPIs y métricas hacia el frontend y otros consumidores.
Frontend web	React servido desde el mismo contenedor	**Azure Static Web Apps**	Hosting del frontend React como SPA, que consume la API de Container Apps.
Gestión de secretos	Variables de entorno / archivos de configuración	**Azure Key Vault**	Almacenamiento seguro de secretos y cadenas de conexión utilizados por los distintos componentes.

Capa funcional	Componente local	Servicio Azure seleccionado	Rol dentro del flujo BI
Monitoreo y logging	Logs en stdout de contenedores / archivos sueltos	**Azure Monitor + Log Analytics Workspace + Audit Logs**	Monitoreo unificado de rendimiento, disponibilidad y auditoría de acciones sobre los recursos.
Red y seguridad perimetral	Red local, firewall físico, VLANs	**Virtual Network, Network Security Groups, Azure Firewall**	Segmentación de red, control de tráfico entrante/saliente y protección de la superficie de ataque.
Acceso privado a servicios PaaS	Conexiones directas en la red interna	**Private Endpoints + Private DNS Zone**	Exposición privada de servicios PaaS (Data Lake, PostgreSQL, etc.) sólo dentro de la VNet.
Respaldo y recuperación ante desastres	Scripts y discos externos	**Recovery Services Vault**	Gestión centralizada de backups y estrategias de recuperación de los componentes críticos.

Nota: en esta arquitectura no se utiliza Power BI ni servicios adicionales de BI cloud.
Toda la visualización se realiza mediante el **dashboard React** desplegado en Azure Static Web Apps y consumiendo la API Flask en Azure Container Apps.

2.2. Servicios Azure por capa funcional

2.2.1. Almacenamiento y Data Lake

Azure Data Lake Storage Gen2

Función

Actúa como almacenamiento distribuido jerárquico que implementa la arquitectura **Medallion** (capas Bronze, Silver y Gold), permitiendo organizar los datos según su grado de procesamiento:

- **Bronze:** datos crudos (raw) tal como llegan desde los archivos CSV del Marco Playbook.
- **Silver:** datos limpios y estandarizados luego de las primeras transformaciones.
- **Gold:** datos curados y agregados, listos para el consumo por el Data Warehouse y la API.

Justificación

- Reemplaza el **HDFS** del entorno Hortonworks por un servicio totalmente gestionado, eliminando la necesidad de administrar nodos y discos físicos.
- Proporciona **jerarquía de directorios y ACLs granulares**, lo que facilita separar dominios (Practitioner vs CI) y controlar permisos a nivel de archivo/carpeta.
- Ofrece **alto rendimiento** de lectura/escritura e integración nativa con Spark mediante el driver **ABFS**, simplificando la configuración en Azure Databricks.
- En combinación con **Delta Lake**, permite disponer de:
    - Transacciones **ACID** sobre archivos.
    - Manejo de históricos y **time travel**.
    - Soporte para actualizaciones incrementales y corrección de datos.
- Permite recibir directamente los archivos CSV desde la interfaz de carga, por lo que **no se requiere un Blob Storage intermedio**: la capa Bronze del Data Lake asume el rol de zona de aterrizaje inicial.

Equivalente local

- **Hadoop HDFS** dentro del contenedor Hortonworks y sistema de archivos local usado para almacenar los CSV.

2.2.2. Procesamiento y transformación

Azure Databricks

Función

Plataforma basada en **Apache Spark** que ejecuta los notebooks PySpark de:

- Limpieza y validación de los datos provenientes de la capa Bronze.
- Transformación y enriquecimiento hacia la capa Silver.
- Agregación y cálculo de KPIs para la capa Gold y el Data Warehouse PostgreSQL.

Justificación

- Reemplaza directamente el stack **Apache Zeppelin + Apache Spark** del entorno local.
- Proporciona **clusters gestionados** con:
    - Escalado automático (autoscaling) según la carga.
    - Auto-terminación tras periodos de inactividad, lo que ayuda a controlar costos.
- Ofrece **notebooks colaborativos** con integración a sistemas de control de versiones (por ejemplo, GitHub), facilitando el trabajo en equipo y la trazabilidad del código ETL.
- Se integra de forma nativa con **Azure Data Lake Storage Gen2**, leyendo y escribiendo sobre rutas ABFS en las capas Bronze, Silver y Gold.

- El uso de **Delta Lake** mejora la calidad y confiabilidad del Data Lake (upserts, manejo de históricos, corrección de registros).

Equivalente local

- Conjunto **Apache Zeppelin + Apache Spark** ejecutándose sobre Hortonworks.

2.2.3. Base de datos analítica

Azure Database for PostgreSQL

Función

Actúa como **Data Warehouse relacional**, almacenando el **modelo dimensional tipo estrella** (tablas de hechos y dimensiones) que consolida la información de Practitioner y Continuous Integration tras las transformaciones de Databricks.

Justificación

- Reemplaza el **PostgreSQL local en contenedor Docker** por un servicio PaaS totalmente administrado.
- Permite mantener prácticamente el **mismo modelo lógico** ya diseñado (esquema, tablas, tipos de datos y consultas SQL).
- Ofrece ventajas gestionadas:
    - **Backups automáticos** y restauración dentro de una ventana de retención configurable.
    - **Alta disponibilidad** mediante réplicas y SLA garantizado por la plataforma.
    - Escalado vertical (vCores, memoria y almacenamiento) sin necesidad de reinstalar ni migrar manualmente.
- Es consumido directamente por la **API Flask** desplegada en Azure Container Apps para responder las consultas del dashboard React.

Equivalente local

- **PostgreSQL** desplegado dentro de un contenedor Docker en la infraestructura on-premise.

2.2.4. Visualización y capa de aplicación

Backend – Azure Container Apps

Función

Servicio gestionado para ejecutar el **backend Flask** dentro de contenedores, expuesto mediante endpoints HTTP/HTTPS que consumen los datos del Data Warehouse PostgreSQL y del Data Lake según sea necesario.

Justificación

- Reemplaza el **contenedor local** que ejecutaba Flask en la infraestructura on-premise.
- Permite desplegar la aplicación directamente desde una imagen de contenedor (por ejemplo, desde un registro de contenedores) sin administrar máquinas virtuales.
- Soporta **escalabilidad automática** basada en métricas (por ejemplo, número de peticiones o CPU), lo que permite acompañar el crecimiento de usuarios del dashboard.
- Se integra con la **Virtual Network**, lo que posibilita que el backend acceda al Data Lake y a PostgreSQL a través de **Private Endpoints**, manteniendo el tráfico dentro de la red privada.
- Envía logs y métricas a **Azure Monitor** y al **Log Analytics Workspace**, lo que simplifica la observabilidad del backend.

Equivalente local

- Contenedor Docker con la aplicación **Flask** ejecutándose en servidores on-premise.

Frontend – Azure Static Web Apps

Función

Servicio diseñado para hospedar el **frontend React** como **Single Page Application (SPA)** estática. La aplicación consume las APIs expuestas por el backend Flask en Azure Container Apps.

Justificación

- Se ajusta perfectamente al patrón SPA utilizado en el dashboard actual.
- Integra de forma nativa flujos de **CI/CD con GitHub** (build + deploy automático), simplificando los despliegues del frontend.
- Incluye CDN y cacheo en el edge, reduciendo la latencia de carga del dashboard para los usuarios.
- Gestiona automáticamente **HTTPS** y las rutas de SPA (por ejemplo, /dashboard, /detalle), sin necesidad de configurar servidores adicionales.
- Permite desacoplar claramente la capa de presentación del backend, facilitando la evolución independiente de ambas.

Equivalente local

- Parte frontend del contenedor que servía la aplicación **React** junto con Flask.

2.2.5. Red, seguridad y gobernanza

En la arquitectura local, la seguridad y operación dependían de:

- Un servidor físico, firewall y configuración de red manual.

- Scripts de backup y discos externos.
- Logs dispersos por contenedores y servidores sin un punto central de monitoreo.

En Azure, estas responsabilidades se distribuyen en varios servicios especializados.

Tabla 2.2 – Servicios de red, seguridad y gobernanza

Servicio	Función principal	Problema que resuelve frente al entorno local
Virtual Network (VNet)	Define la red privada lógica donde se ubican Container Apps, PostgreSQL y demás recursos.	Sustituye la red local física, permitiendo aislar la solución del resto de internet.
Network Security Groups	Reglas de seguridad a nivel de subred e interfaz para permitir o denegar tráfico.	Reemplaza reglas manuales en firewall/servidor; facilita segmentar tráfico por capas.
Azure Firewall	Firewall administrado y centralizado para controlar tráfico entrante/saliente hacia/desde la VNet.	Aporta un punto único de inspección y registro, sin necesidad de hardware dedicado.
Private Endpoints	Publican servicios PaaS (Data Lake, PostgreSQL, etc.) como direcciones privadas dentro de la VNet.	Evitan exponer endpoints públicos; el acceso a datos pasa por la red privada.
Private DNS Zone	Resuelve los nombres de los servicios PaaS a sus direcciones privadas asociadas a Private Endpoints.	Permite que las aplicaciones usen nombres de host “normales” pero con resolución privada.
Azure Key Vault	Almacén seguro de secretos, claves y certificados.	Elimina credenciales hardcodeadas en código o variables de entorno sin protección.
Azure Monitor	Plataforma de monitoreo de métricas de los servicios.	Proporciona una vista centralizada de rendimiento y disponibilidad.
Log Analytics Workspace	Repositorio y motor de consulta para logs de aplicaciones e infraestructura.	Centraliza los logs antes dispersos en stdout y archivos de cada contenedor/servicio.

Servicio	Función principal	Problema que resuelve frente al entorno local
Audit Logs / Activity Logs	Registro de operaciones administrativas sobre los recursos de Azure.	Aporta trazabilidad y auditoría, inexistente en la solución on-premise.
Recovery Services Vault	Servicio para gestionar backups y restauraciones de recursos críticos.	Sustituye soluciones de backup manuales basadas en scripts y discos externos.

A continuación se describen brevemente los más relevantes.

Azure Key Vault

Función

Almacena de forma segura secretos (connection strings, claves de acceso, credenciales de servicio) utilizados por:

- Azure Databricks.
- Azure Container Apps.
- Otros componentes que requieran credenciales para conectarse a Data Lake o PostgreSQL.

Justificación

- Evita guardar secretos en código o archivos de configuración sin cifrar.
- Permite **rotar claves** sin necesidad de cambiar el código de las aplicaciones.
- Ofrece **auditoría** sobre el acceso a secretos.
- Se integra con identidades administradas (Managed Identities), reduciendo el uso de contraseñas explícitas.

Equivalente local

- Variables de entorno y archivos de configuración sin mecanismos formales de auditoría ni rotación centralizada.

Azure Monitor + Log Analytics Workspace + Audit Logs

Función

- Recopilar métricas de rendimiento (CPU, memoria, latencia, uso de almacenamiento, etc.) de los servicios de la solución.
- Centralizar los logs de aplicación (Container Apps, Static Web Apps, Databricks) y de infraestructura.

- Registrar las operaciones administrativas realizadas sobre los recursos (creación, borrado, cambios de configuración).

Justificación

- Proporciona una **visión unificada** del estado de los componentes de la arquitectura.
- Permite definir **alertas** (por ejemplo, ante fallos, timeouts, saturación de CPU o uso excesivo de almacenamiento).
- Log Analytics ofrece consultas avanzadas mediante KQL para análisis de errores y patrones de uso.
- Los **Audit Logs** facilitan demostrar quién hizo qué cambio y cuándo, algo clave en contextos regulados.

Equivalente local

- Logs dispersos por contenedores y servidores sin panel unificado ni mecanismos formales de alertas ni auditoría.

Virtual Network, Network Security Groups, Private Endpoints, Private DNS Zone y Azure Firewall

Función conjunta

- Definir una **red privada** donde residen los componentes de la solución.
- Controlar qué tráfico entra, sale y se mueve entre subredes mediante **NSG** y **Azure Firewall**.
- Exponer el Data Lake, PostgreSQL y otros servicios PaaS sólo mediante **Private Endpoints**, con nombres resueltos por **Private DNS Zone**.

Justificación

- Aísla la solución del tráfico público por defecto; sólo se publica lo estrictamente necesario (por ejemplo, el acceso al dashboard).
- Reduce la superficie de ataque al evitar endpoints públicos en servicios de datos.
- Permite aplicar políticas de segmentación por capas (ingesta, procesamiento, datos, exposición) similares o superiores a las que se tenían en la red local.

Recovery Services Vault

Función

Administrador central de **backups** y restauraciones de los recursos más críticos de la solución.

Justificación

- Sustituye scripts manuales y copias en discos externos.
- Permite definir políticas de respaldo y retención consistentes en el tiempo.

- Facilita la recuperación ante desastres con tiempos de restauración más predecibles.

3. MATRIZ DE COSTOS Y PROYECCIÓN DE USO

3.1. Metodología de estimación

Herramienta utilizada: Azure Pricing Calculator oficial (https://azure.microsoft.com/en-us/pricing/calculator/)

Supuestos de uso y volumetría

1. **Frecuencia de procesamiento**

- Mensual (12 ejecuciones/año).

2. **Volumen de datos**

- CSV Practitioner: 50 MB/mes (600 MB/año).
- CSV Continuous Integration: 80 MB/mes (960 MB/año).
- **Total ingesta:** 130 MB/mes ≈ 1.56 GB/año.

3. **Azure Data Lake Storage Gen2**

- Capa Bronze: 5 GB acumulado (CSV crudos históricos).
- Capa Silver: 8 GB acumulado (archivos Parquet limpios).
- Capa Gold: 3 GB acumulado (tablas Delta que soportan el modelo estrella).
- **Total almacenamiento estimado:** 16 GB.

4. **Base de datos gestionada en Azure (PostgreSQL)**

- Servicio: *Azure Database for PostgreSQL* (configuración General Purpose equivalente a 4 vCores).
- Tamaño de base de datos: 10 GB inicial → 25 GB proyectado a 12 meses.
- Consultas del backend/dashboard: 50,000 queries/mes.
- Conexiones concurrentes: ~10 usuarios simultáneos.

5. **Azure Databricks**

- Tipo de clúster: Standard_DS3_v2 (4 vCores, 14 GB RAM).
- Autoscaling: 2–8 workers (promedio 4 workers en ejecución).
- Horas de ejecución: 4 horas/mes (limpieza, transformaciones y carga al DW).

6. **Azure Container Apps (backend Flask)**

- Backend expuesto como contenedor (API Flask) desplegado en Azure Container Apps.
- Consumo equivalente a 1 instancia base con 1–2 vCPU y ~2 GB RAM.
- Autoscaling: 1–3 réplicas (promedio 1.5 réplicas activas).
- Usuarios concurrentes en el backend: 20–30 (vía dashboard React).

7. **Azure Static Web Apps (frontend React)**

- Plan: Free.
- Uso: despliegue del frontend React (SPA) sin costo adicional de infraestructura dentro de los límites del plan gratuito.

8. **Servicios de seguridad, monitoreo y gobierno**

- **Azure Key Vault:** almacén centralizado de secretos.
- **Azure Monitor + Log Analytics Workspace:** observabilidad unificada (métricas, logs y alertas) de Databricks, Container Apps, base de datos, Static Web Apps, Firewall, etc.
- **Componentes de red y seguridad:** Virtual Network, Network Security Groups, Private Endpoints, Private DNS Zone, Azure Firewall, Recovery Services Vault y Audit Logs.
    - En esta estimación se modelan explícitamente los costos de *Log Analytics* y *Key Vault*; el resto de componentes introduce un costo marginal adicional frente a los servicios de cómputo y se considera dentro del margen de error de la proyección.

Consideraciones especiales

- **Región:** East US (referencia de precios; regiones con costos competitivos).
- **Redundancia:** LRS (Locally Redundant Storage) en esta estimación (entorno académico / desarrollo).
- **Reservas:** no se consideran Reserved Instances en el primer año (modelo pay-as-you-go para validar la volumetría real).
- **Descuentos:** se asume un descuento corporativo efectivo de aproximadamente 40 % en la base de datos gestionada (similar al efecto de acuerdos de licenciamiento en entornos empresariales), solo con fines estimativos.

3.2. Costos por servicio

3.2.1. Servicio 1: Azure Data Lake Storage Gen2

Componente	SKU/Tier	Volumen mensual	Costo mensual (USD)	Costo anual (USD)	Justificación
Almacenamiento Hot Tier	Standard LRS	16 GB	0.37	4.44	Capas Bronze/Silver/Gold con acceso mensual. Ref: ~0.0184 USD/GB/mes × 16 GB

Componente	SKU/Tier	Volumen mensual	Costo mensual (USD)	Costo anual (USD)	Justificación
Operaciones Lectura (Class 2)	–	10,000 ops/mes	0.01	0.12	Databricks lee datos en Bronze y Silver (CSV/Parquet).
Operaciones Escritura (Class 1)	–	5,000 ops/mes	0.10	1.20	Proceso de ingesta y Databricks escriben en Bronze/Silver/Gold.
Total Data Lake Gen2			**0.48**	**5.76**	

Justificación de dimensionamiento

- 16 GB son suficientes para almacenar ~12 meses históricos:
    - Bronze: ~1.56 GB/año (CSV crudos).
    - Silver: ~3 GB/año (Parquet comprimido).
    - Gold: ~1 GB/año (Delta).
- Hot Tier adecuado por el patrón de acceso (lecturas/escrituras mensuales para procesamiento).
- Operaciones por ciclo de ETL (aprox.):
    - 1 escritura principal de datos crudos a Bronze.
    - Varias lecturas/escrituras de Databricks para capas Silver y Gold.

3.2.2. Servicio 2: Azure Databricks (Premium Tier)

Componente	SKU/Tier	Volumen mensual	Costo mensual (USD)	Costo anual (USD)	Justificación
Databricks Units	Premium Jobs	16 DBU/mes	0.90	10.80	4 h de ejecución × 4 DBU/hora. Ref: ~0.15

Componente	SKU/Tier	Volumen mensual	Costo mensual (USD)	Costo anual (USD)	Justificación
					USD/DBU × 4 DBU × 4 h
Compute VM (driver)	Standard_DS3_v2	4 horas/mes	0.68	8.16	Nodo driver activo durante las ejecuciones (~0.17 USD/h × 4 h).
Compute VM (workers)	Standard_DS3_v2	16 worker-hours/mes	2.72	32.64	Autoscaling promedio 4 workers × 4 h (0.17 USD/h × 4 × 4).
Total Databricks			**4.30**	**51.60**	

Justificación de dimensionamiento

- Standard_DS3_v2 (4 vCores, 14 GB RAM) es suficiente para procesar ~130 MB/mes + histórico sin clústeres grandes.
- Premium Tier aporta:
    - RBAC y auditoría avanzados.
    - Integración con Delta Lake (ACID, optimizaciones de lectura/escritura).
- 4 horas/mes como estimación conservadora para:
    - Transformaciones Bronze → Silver.
    - Transformaciones Silver → Gold.
    - Preparación/carga hacia la base de datos PostgreSQL.
- Autoscaling 2–8 workers (promedio 4) equilibra rendimiento y costo.

3.2.3. Servicio 3: Base de datos gestionada en Azure (Azure Database for PostgreSQL – General Purpose)

Componente	SKU/Tier	Volumen mensual	Costo mensual (USD)	Costo anual (USD)	Justificación
Compute (vCore)	General Purpose, 4 vCores	730 horas/mes (24/7)	423.60	5,083.20	4 vCores suficientes para ~50,000 queries/mes de backend/dashboard.
Storage	Premium SSD	25 GB	6.25	75.00	10 GB inicial → 25 GB a 12 meses (modelo estrella + índices).
Backup Storage (LRS)	Backups automatizados	25 GB	2.50	30.00	Retención típica 35 días.
Subtotal antes de descuento			**432.35**	**5,188.20**	
Descuento corporativo (≈40 %)	–	–	**– 172.94**	**– 2,075.28**	Descuento efectivo similar a acuerdos de licenciamiento en entornos banca.
Total base de datos			**259.41**	**3,112.92**	

Justificación de dimensionamiento

- **4 vCores General Purpose:**
    - Balance adecuado para una carga analítica moderada (≈50,000 consultas/mes ≈ 69 consultas/hora promedio).
    - Permite cierto crecimiento sin cambiar de tier inmediatamente.
- **25 GB de almacenamiento:**
    - Incluye crecimiento del modelo dimensional (hechos, dimensiones, índices) a ~12 meses.
- **Descuento corporativo ~40 %:**

    - Representa un escenario realista de negociación de precios/licenciamiento en organizaciones grandes.
- Servicios como Azure Synapse SQL Pool se descartan en esta fase por costo base más alto y por estar sobredimensionados para <100 GB y concurrencia limitada.

3.2.4. Servicio 4: Azure Container Apps (backend Flask)

Componente	SKU/Tier	Volumen mensual	Costo mensual (USD)	Costo anual (USD)	Justificación
Cómputo base (réplica 1)	Consumo equivalente	730 horas/mes (24/7)	69.35	832.20	Consumo aproximado equivalente a 1 instancia pequeña (~1–2 vCPU, 2 GB RAM).
Autoscaling (réplicas extra)	Consumo equivalente	365 horas/mes adicionales	34.68	416.10	Promedio 1.5 réplicas (1 base + 0.5 extra en picos de uso).
HTTPS y gestión de certificados	Incluido	–	0.00	0.00	Certificados TLS gestionados por la plataforma.
Total Container Apps			**104.03**	**1,248.30**	

Justificación de dimensionamiento

- Se aproxima el costo al de un plan Standard S1 de App Service para una carga similar (backend Flask de tamaño moderado).
- Permite autoscaling horizontal sin gestionar VMs, alineado con la arquitectura de contenedores.
- Los 20–30 usuarios concurrentes previstos se consideran una carga ligera-moderada.

3.2.5. Servicio 5: Azure Static Web Apps (frontend React)

Componente	SKU/Tier	Volumen mensual	Costo mensual (USD)	Costo anual (USD)	Justificación
Static Web Apps Plan	Free	Dentro de límites del plan	0.00	0.00	Hosting de la SPA React dentro de los límites del plan gratuito.

Justificación de dimensionamiento

- Para un escenario académico/prototipo, el plan **Free** es suficiente:
    - Soporta despliegue de la SPA React.
    - Incluye HTTPS y CI/CD básico.
- La carga principal recae en el backend (Container Apps) y la base de datos, no en Static Web Apps.

3.2.6. Servicio 6: Azure Key Vault

Componente	SKU/Tier	Volumen mensual	Costo mensual (USD)	Costo anual (USD)	Justificación
Secret Operations	Standard Tier	10,000 ops/mes	0.03	0.36	Backend y Databricks leen secretos varias veces al día.
Secrets Storage	Standard Tier	5 secretos	0.00	0.00	Primeros 10,000 secretos sin costo adicional.
Total Key Vault			**0.03**	**0.36**	

Justificación de dimensionamiento

- 10,000 operaciones/mes:
    - Estimación conservadora (~300 accesos/día × 30 días ≈ 9,000 ops).
- 5 secretos (ejemplos):
    - Cadena de conexión a PostgreSQL.
    - Credenciales de Storage Account.

- Secretos de acceso a Databricks.
- Credenciales de Service Principal, etc.

3.2.7. Servicio 7: Azure Monitor + Log Analytics Workspace

Componente	SKU/Tier	Volumen mensual	Costo mensual (USD)	Costo anual (USD)	Justificación
Log Analytics Ingestion	Pay-as-you-go	5 GB logs/mes	11.50	138.00	Logs de Databricks, Container Apps, DB, Static Web Apps, Firewall, etc.
Log Analytics Retention	90 días	5 GB almacenados	0.50	6.00	Días 1–31 gratis; días 32–90 con costo (~0.10 USD/GB).
Alertas (Action Groups)	Standard Tier	10 alertas/mes	0.20	2.40	Alertas por fallos, alta latencia o consumo anómalo.
Total Azure Monitor			**12.20**	**146.40**	

Justificación de dimensionamiento

- 5 GB de logs/mes estimados:
  - Databricks: ~2 GB (logs de notebooks/jobs).
  - Azure Container Apps: ~1.5 GB (requests, errores).
  - Base de datos PostgreSQL gestionada: ~0.5 GB.
  - Static Web Apps, Firewall y otros componentes: ~1 GB.
- Retención 90 días:
  - Compromiso entre requisitos de auditoría/diagnóstico y costo.
- 10 alertas/mes:

- Fallos de ejecución críticos.
- Latencia elevada en la API.
- Uso de CPU/memoria anómalo.
- Costos acercándose a un presupuesto definido.

3.3. Costo total estimado

3.3.1. Desglose por servicio

Servicio Azure	**Costo mensual (USD)**	**Costo anual (USD)**	**% del total aprox.**
Base de datos gestionada (Azure Database for PostgreSQL)	259.41	3,112.92	68.2 %
Azure Container Apps (backend Flask)	104.03	1,248.30	27.3 %
Azure Monitor + Log Analytics	12.20	146.40	3.2 %
Azure Databricks	4.30	51.60	1.1 %
Azure Data Lake Storage Gen2	0.48	5.76	0.1 %
Azure Key Vault	0.03	0.36	<0.1 %
Azure Static Web Apps	0.00	0.00	0.0 %
TOTAL MENSUAL	**380.45**		**100 %**
TOTAL ANUAL		**4,565.40**	

3.3.2. Desglose por categoría

Separando el costo de la base de datos en cómputo y almacenamiento/backups tras aplicar el descuento.

Categoría	Costo mensual (USD)	Costo anual (USD)	% del total aprox.
Cómputo (Databricks + Azure Container Apps + cómputo DB)	362.49	4,349.88	95.3 %
Almacenamiento (Data Lake + almacenamiento/backups de la DB)	5.73	68.76	1.5 %
Servicios gestionados (Key Vault + Azure Monitor/Log Analytics)	12.23	146.76	3.2 %
TOTAL	**380.45**	**4,565.40**	**100 %**

Nota: para simplificar, el costo total de la base de datos se ha desagregado internamente entre cómputo y almacenamiento/backups, aunque Azure lo factura como un único servicio gestionado.

3.3.3. Observaciones clave

1. **La base de datos gestionada en Azure es el principal driver de costos**, representando alrededor del **68 %** del costo mensual total.
    - Esto se justifica por:
        - Alta disponibilidad (SLA gestionado).
        - Backups automáticos y capacidad de restauración.
        - Capacidad suficiente para manejar ~50,000 queries/mes del backend/dashboard.
2. **El descuento corporativo (~40 %) aporta un ahorro aproximado de 2,075 USD/año** sobre la base de datos gestionada.
    - Sin este descuento, el costo anual de la base de datos pasaría de **3,112.92 USD** a ~**5,188.20 USD**, elevando el TCO del entorno en torno a un 45 %.
3. **El almacenamiento es extremadamente económico**, representando cerca del **1.5 %** del costo total anual (~**68.76 USD/año**):
    - Incluye Data Lake Gen2 (capas Bronze, Silver, Gold).
    - Incluye almacenamiento y backups de la base de datos PostgreSQL gestionada.
4. **Los servicios gestionados de seguridad y observabilidad (Key Vault + Azure Monitor/Log Analytics)** suponen alrededor del **3.2 %** del costo total:
    - Aportan seguridad centralizada de secretos (Key Vault).
    - Monitoreo, logging y alertas unificadas (Azure Monitor + Log Analytics).

    - Permiten incorporar también logs de red y seguridad (Firewall, NSG, Private Endpoints, Audit Logs) en el mismo workspace.

5. **No se incurre en costos adicionales de BI cloud** (como Power BI Service):
    - La visualización se realiza exclusivamente mediante el frontend React en Azure Static Web Apps (plan Free) consumiendo la API en Azure Container Apps.
    - Esto mantiene el costo de BI cloud en **0 USD**, a cambio de concentrar la lógica de presentación y reporting dentro de la propia aplicación web.

1. Arquitectura avanzada en la nube (Azure)

1.1. Objetivo de la sección

La arquitectura en Azure despliega un **flujo completo de Business Intelligence** para medir los niveles de madurez *Practitioner* y *Continuous Integration (CI)* a partir de archivos CSV generados por el **Marco Playbook**.

La solución cloud mantiene la lógica funcional de la arquitectura on-premise (ETL en Spark + modelo estrella + dashboard web), pero la mejora en:

- Escalabilidad y elasticidad (autoscaling de cómputo y servicios gestionados).
- Alta disponibilidad y recuperación ante desastres.
- Seguridad, redes privadas y gobierno de datos.
- Observabilidad centralizada (métricas, logs y alertas).

1.2. Diagrama general de la arquitectura

El siguiente diagrama de arquitectura muestra lo siguiente:

- **Sources**: archivos CSV del Marco Playbook.
- **Process / Backend**: interfaz de carga, Databricks y base de datos PostgreSQL gestionada.
- **Store**: Data Lake con capas Bronze, Silver y Gold.
- **Serve**: backend en Azure App Service y frontend en Azure Static Web Apps.
- **Servicios transversales**: VNet, Private Endpoints, NSG, Azure Firewall, Key Vault, Azure Monitor, Log Analytics, Private DNS Zone, Recovery Services Vault y Audit Logs.

1.3. Capas funcionales de la solución

La arquitectura se organiza en capas lógicas, siguiendo el flujo de datos de extremo a extremo.

1.3.1. Fuentes de datos (Sources)

- **Origen principal**: archivos CSV exportados manualmente desde el **Marco Playbook**.
- Cada mes se cargan dos archivos:
    - CSV Practitioner.
    - CSV Continuous Integration.

1.3.2. Capa de proceso (Backend)

Componentes principales:

- **Interfaz de carga**
  Formulario web que permite al *Service Owner* subir los CSV al entorno cloud.
- **Azure Databricks**
  Ejecuta los notebooks PySpark que realizan:
    - Limpieza y validación de datos.
    - Estandarización de tipos y formatos.
    - Cálculo de KPIs de Practitioner y CI.
    - Generación de las tablas finales para el modelo estrella.
- **PostgreSQL gestionado en Azure**
  Base de datos analítica donde se implementa el **modelo dimensional tipo estrella** (tablas de hechos y dimensiones) que será consumido por el dashboard y por futuros casos de uso.

1.3.3. Capa de almacenamiento (Store)

La solución utiliza un **Data Lake en Azure** con una arquitectura tipo **Medallion**:

- **Capa Bronze (raw)**
  Almacena los CSV tal como llegan desde el Marco Playbook, sin transformar.
  Es la "fuente de la verdad" histórica.
- **Capa Silver (trusted)**
  Contiene los datos limpios y tipificados en formato optimizado (por ejemplo, Parquet).
  Aquí ya se han aplicado reglas de calidad (nulos, duplicados, tipos).
- **Capa Gold (refined)**
  Concentra los datos agregados y listos para consumo analítico y carga al modelo estrella en PostgreSQL.

1.3.4. Capa de presentación (Serve)

- **Azure App Service (backend Flask)**
  Expone una API REST que consulta el modelo estrella y sirve los KPIs y métricas necesarios para el dashboard.
- **Azure Static Web Apps (frontend React)**
  Hospeda una Single Page Application (SPA) desarrollada en React que:
    - Consume las APIs del backend.
    - Muestra tablas, gráficos y KPIs de madurez Practitioner/CI.
    - Actúa como "visor BI" personalizado, en lugar de utilizar un servicio de BI cloud externo (Power BI, Looker, etc.).

1.4. Servicios Azure utilizados

La siguiente tabla resume los servicios clave de la arquitectura:

Capa / Dominio	Servicio Azure	Rol principal en la solución
Red y seguridad	**Virtual Network (VNet)**	Red privada que aísla los recursos de la solución.
	Subredes + Network Security Groups	Segmentación de servicios (datos, backend, seguridad) y filtrado L4.
	Private Endpoints	Acceso privado a cuentas de almacenamiento y base de datos.
	Azure Firewall	Firewall centralizado para tráfico saliente/entrante controlado.
	Private DNS Zone	Resolución DNS interna para endpoints privados.
	Audit Logs	Auditoría de acciones administrativas y de seguridad.
Procesamiento	**Azure Databricks**	Ejecución de notebooks PySpark para el ETL completo.
Almacenamiento	**Azure Data Lake (Bronze/Silver/Gold)**	Data Lake estructurado con arquitectura Medallion.
	Blob Storage	Zona de aterrizaje inicial para los CSV del Marco Playbook.
Datos relacionales	**PostgreSQL gestionado en Azure**	Data Warehouse con modelo estrella (hechos/dimensiones).
Aplicación / API	**Azure App Service**	Backend Flask que expone KPIs y métricas vía API.
Presentación	**Azure Static Web Apps**	SPA React que actúa como visor BI en la nube.

Capa / Dominio	Servicio Azure	Rol principal en la solución
Seguridad de secretos	**Azure Key Vault**	Almacén seguro de cadenas de conexión y credenciales.
Monitoreo y logging	**Azure Monitor + Log Analytics**	Métricas, logs centralizados y alertas.
Backup / DR	**Recovery Services Vault**	Copias de seguridad y recuperación ante desastres.

1.5. Flujo de datos end-to-end

1. **Carga de archivos**
   El *Service Owner* descarga los CSV del Marco Playbook y los sube a través de la interfaz web.
   La aplicación los envía a **Blob Storage** (zona de staging).
2. **Ingesta al Data Lake (Bronze)**
   Un proceso controlado carga los CSV desde Blob a la **capa Bronze** del Data Lake, preservando la estructura original.
3. **Procesamiento en Databricks (Silver)**
   Azure Databricks:
   - Lee los archivos de Bronze.
   - Aplica reglas de limpieza y tipificación.
   - Escribe los resultados limpios en la **capa Silver**.
4. **Enriquecimiento y agregación (Gold)**
   Nuevos notebooks en Databricks:
   - Realizan joins, agregaciones y cálculo de KPIs.
   - Escriben las tablas refinadas en la **capa Gold**.
5. **Carga al Data Warehouse (PostgreSQL)**
   Databricks inserta/actualiza las tablas de hechos y dimensiones en PostgreSQL utilizando los datos refinados de Gold.
6. **Exposición vía API (App Service)**
   El backend Flask, desplegado en App Service, expone endpoints REST que consultan el modelo estrella.
7. **Visualización (Static Web Apps)**
   La SPA React, hospedada en Azure Static Web Apps:
   - Consume las APIs del backend.
   - Presenta KPIs, gráficos y tablas al *Service Owner* y al resto de usuarios internos.

1.6. Escalabilidad, elasticidad y alta disponibilidad

Escalabilidad y elasticidad

- **Azure Databricks**
  - Clústeres con **autoscaling** (por ejemplo, 2–8 workers).
  - Auto-termination tras la ventana de procesamiento mensual, evitando costo de recursos ociosos.
- **Azure App Service**

    - Plan Standard S1 con **escalado horizontal** basado en métricas (CPU/uso de memoria).
    - Capacidad de aumentar instancias del backend durante picos de uso del dashboard.
- **PostgreSQL en Azure**
    - Escalado **vertical** ajustando vCores y almacenamiento sin downtime significativo.

Alta disponibilidad y DR

- Servicios gestionados (App Service, PostgreSQL, almacenamiento) con SLA elevado (hasta 99.95–99.99 % según configuración).
- **Recovery Services Vault**:
    - Almacena las copias de seguridad de recursos críticos.
    - Permite restauraciones ante fallos graves o desastres.
- Uso de almacenamiento con redundancia local y, opcionalmente, redundancia geográfica si se requiere un escenario multi-región.

1.7. Seguridad, redes y gobernanza (visión general)

El detalle técnico se documenta en la carpeta 2_Seguridad_IAM_Redes_y_Gobernanza. Aquí se presenta un resumen a nivel de arquitectura.

- **VNet + subredes privadas**
  Todos los servicios de datos (Data Lake, PostgreSQL, Databricks) se ubican en subredes privadas, sin exposición directa a Internet.
- **Network Security Groups y Azure Firewall**
  Controlan el tráfico entrante y saliente por puertos/protocolos, restringiendo el acceso sólo a los servicios y orígenes necesarios.
- **Private Endpoints + Private DNS Zone**
  El acceso a cuentas de almacenamiento y bases de datos se realiza mediante **endpoints privados**, evitando uso de IPs públicas.
- **Azure Key Vault**
  Las cadenas de conexión y secretos no se guardan en código ni variables de entorno sin protección; se leen de Key Vault.
- **Audit Logs**
  Registra operaciones administrativas y de seguridad sobre los recursos, permitiendo auditoría y trazabilidad.

1.8. Monitoreo y observabilidad

- **Azure Monitor + Log Analytics Workspace**

- Centralizan métricas de:
  - App Service (latencia, errores HTTP, CPU).
  - Databricks (estado de jobs y clústeres).
  - PostgreSQL (conexiones, tiempos de respuesta).
  - Storage y Static Web Apps (errores, tráfico).
- Permiten definir **alertas** (por ejemplo, fallos de job, latencia alta, costo proyectado).

2. Escenarios de Escalabilidad, Elasticidad y Recuperación ante Desastres (DR)

2.1. Objetivo de la sección

Esta sección describe cómo la arquitectura en Azure está preparada para:

- Escalar ante distintos niveles de carga.
- Ajustar automáticamente los recursos (elasticidad) para no pagar por capacidad ociosa.
- Mantener alta disponibilidad del servicio.
- Recuperarse ante fallos graves mediante mecanismos de **backup y Disaster Recovery (DR)**.

Se centra en los servicios clave del proyecto:

- Azure Databricks
- Data Lake en Azure (Bronze / Silver / Gold)
- PostgreSQL gestionado en Azure
- Azure App Service (backend Flask)
- Azure Static Web Apps (frontend React)
- Virtual Network (VNet), NSG, Azure Firewall, Private Endpoints, Private DNS Zone
- Azure Key Vault
- Azure Monitor + Log Analytics Workspace
- Recovery Services Vault
- Audit Logs

2.2. Escenarios de carga

Para diseñar la escalabilidad, se consideran tres escenarios de carga típicos:

2.2.1. Escenario A – Carga base (operación normal)

- **Frecuencia de ETL**: mensual.
- **Volumen de datos**:

    - CSV Practitioner: ~50 MB/mes.
    - CSV CI: ~80 MB/mes.
    - Total ingesta: ~130 MB/mes.
- **Usuarios concurrentes**:
    - 20–30 usuarios internos consultando el dashboard.
- **Patrón de acceso**:
    - Picos de uso en horario laboral (consultas al dashboard).
    - Procesamiento intensivo solo en la ventana ETL mensual.
    -

Este escenario corresponde al dimensionamiento base usado en la matriz de costos.

2.2.2. Escenario B – Crecimiento moderado

- **Volumen de datos**:
    - +20 % de crecimiento trimestral → ~156 MB/mes.
- **Usuarios concurrentes**:
    - 30–40 usuarios internos.
- **Impacto esperado**:
    - Más tiempo de cómputo en Databricks.
    - Más consultas y almacenamiento en PostgreSQL.
    - Más tráfico hacia App Service y Static Web Apps.

La arquitectura debe soportar este crecimiento con **ajustes menores de configuración** (escalado vertical/horizontal).

2.2.3. Escenario C – Pico puntual de consumo

- Auditoría interna, presentaciones o cierres trimestrales.
- Múltiples usuarios consultan el dashboard en paralelo.
- Consultas más pesadas sobre históricos completos.

En este escenario es clave poder **escalar temporalmente** App Service y, si es necesario, los recursos de PostgreSQL.

2.3. Estrategia de escalabilidad y elasticidad

2.3.1. Azure Databricks – Escalabilidad para el ETL

- **Tipo de clúster**: Standard_DC4as_v5 (4 vCores, 16 GB RAM).
- **Autoscaling de workers**:
    - Rango propuesto: Single node

Configuración recomendada:

- Política de auto-terminación: apagar el clúster tras 20 minutos sin actividad.
- Horario de ejecución del job:
    - Ventana ETL mensual programada (por ejemplo, madrugada o fuera de horario pico).

Beneficios:

- El clúster solo consume recursos durante:
    - Ingesta y limpieza (Bronze → Silver).
    - Transformaciones de negocio (Silver → Gold).
    - Carga al DW (PostgreSQL).
- El resto del tiempo no hay costo de cómputo en Databricks.

2.3.2. PostgreSQL en Azure – Escalabilidad vertical

PostgreSQL en Azure se dimensiona inicialmente con:

- **vCores**: 4 vCores (General Purpose).
- **Almacenamiento**: 25 GB (modelo estrella + índices).

La escalabilidad se resuelve principalmente de forma **vertical**:

- Aumento de vCores (por ejemplo, de 4 → 8) ante:
    - Incremento sostenido de consultas.
    - Picos de uso en cierres mensuales/trimestrales.
- Aumento de almacenamiento (por ejemplo, 25 → 50 GB) ante:
    - Crecimiento de históricos.
    - Nuevas tablas de hechos/dimensiones.

La plataforma permite aplicar estos cambios con mínimo downtime, manteniendo la lógica del modelo estrella intacta.

2.3.3. Azure App Service – Escalabilidad horizontal

- **Plan**: Standard S1.
- **Instancias**: 1 instancia base, con posibilidad de escalar horizontalmente.

Reglas de autoscaling recomendadas:

- Métrica: **CPU Percentage** o **HTTP Queue Length**.
- **Scale-out**:
    - Si CPU > 70 % durante 5 minutos → agregar 1 instancia (hasta un máximo de 3).
- **Scale-in**:
    - Si CPU < 40 % durante 10 minutos → eliminar 1 instancia (mínimo 1).

Esto permite:

- **Escalar horizontalmente** el backend Flask cuando hay más usuarios.
- Volver automáticamente a la capacidad mínima fuera de picos (elasticidad).

2.3.4. Azure Static Web Apps – Escalabilidad gestionada

Azure Static Web Apps ofrece:

- **Escalado automático gestionado por la plataforma** para el contenido estático.
- Uso de CDN y edge caching para mejorar tiempos de respuesta sin que el equipo tenga que gestionar servidores.

En la práctica:

- El costo y el esfuerzo de escalado del frontend son mínimos.
- La principal carga de procesamiento recae en App Service y PostgreSQL.

Colocar captura del recurso Static Web Apps mostrando el plan utilizado.

2.3.5. Almacenamiento – Data Lake y Blob

El Data Lake está dimensionado para:

- Volumen inicial bajo (< 20 GB).
- Crecimiento proyectado controlado (decenas de GB).

La escalabilidad es **prácticamente ilimitada**, y se gestiona mediante:

- Aumento de capacidad de la cuenta de almacenamiento (automático en la práctica).
- Opcionales reglas de Lifecycle Management para mover datos antiguos a tiers más baratos (Cool).

2.4. Alta disponibilidad (HA)

Aunque se trata de un entorno académico/prototipo, la arquitectura sigue principios de alta disponibilidad:

2.4.1. Servicios gestionados con SLA

- **PostgreSQL en Azure**:
    - Alta disponibilidad gestionada por la plataforma.
    - SLA elevado (99.95–99.99 % según configuración).
- **App Service y Static Web Apps**:
    - Plataformas PaaS con múltiples instancias gestionadas por Azure.
- **Cuenta de almacenamiento (Data Lake + Blob)**:
    - Redundancia **LRS** por defecto (tres réplicas dentro de la misma región).

2.4.2. Red y acceso

- **Virtual Network + subredes**:
    - Segmentación lógica entre servicios de datos, backend y componentes de seguridad.
- **Network Security Groups + Azure Firewall**:
    - Reglas explícitas de entrada/salida reducen la superficie de ataque.
- **Private Endpoints + Private DNS Zone**:
    - Acceso interno a recursos de datos (Data Lake, PostgreSQL) sin exposición pública.

Este diseño disminuye el riesgo de caídas por ataques externos o configuraciones inseguras.

2.5. Estrategia de backup y Disaster Recovery (DR)

2.5.1. Objetivos de DR

- **RPO (Recovery Point Objective)** esperado:
    - Horas, dependiendo de la frecuencia de backups configurada.
- **RTO (Recovery Time Objective)** esperado:
    - Horas, dependiendo de los procedimientos de restauración.

Estos valores son adecuados para un entorno académico/prototipo, pero el diseño puede ajustarse a escenarios productivos.

2.5.2. Recovery Services Vault

Se utiliza **Recovery Services Vault** para gestionar copias de seguridad de recursos críticos:

- Backups de **PostgreSQL** (base de datos de modelo estrella).
- Opcionalmente, copias de App Service (configuración) y de la cuenta de almacenamiento.

Configuración típica:

- Frecuencia de backup: diaria.
- Retención: 30–35 días.
- Almacenamiento redundante: LRS (en este entorno).

2.5.3. Estrategia de backup por componente

Componente	Estrategia de backup	Comentarios clave
Data Lake (Bronze/Silver/Gold)	Copias redundantes + export ocasional a otra cuenta / región opcional	Datos pueden regenerarse desde históricos CSV si es necesario.
CSV originales en Blob	Conservación de los últimos meses como “fuente de verdad”	Permiten reprocesar el ETL desde cero.
PostgreSQL (DW)	Backups automáticos gestionados por Azure + Recovery Services Vault	Punto más crítico desde el punto de vista analítico.
App Service (backend)	Código almacenado en GitHub + backups de configuración opcionales	La infraestructura se puede recrear vía IaC/portal.
Static Web Apps	Código en GitHub (fuente de despliegue)	Se vuelve a desplegar a partir del repositorio.
Configuración de secretos (Key Vault)	Export de configuración (con cuidado) + documentación	Los secretos se reemiten en caso extremo.

2.5.4. Procedimiento de recuperación (ejemplo)

Escenario: pérdida de la base de datos PostgreSQL (corrupción lógica o fallo grave).

1. **Detección**
    - Azure Monitor dispara una alerta de error crítico (por ejemplo, fallos de conexión constantes).
2. **Identificación del punto de restauración**
    - El equipo revisa los backups disponibles en **Recovery Services Vault**.
    - Se elige un punto de restauración anterior al incidente (según RPO).
3. **Restauración de la base de datos**
    - Se realiza la restauración a un nuevo servidor o sobre el actual (según la política).
    - Se actualizan las cadenas de conexión en **Azure Key Vault** si cambia el endpoint.

4. **Validación**
    - Se ejecutan consultas de validación sobre el modelo estrella.
    - Se verifica que el dashboard vuelve a mostrar métricas correctas.
5. **Reprocesamiento opcional**
    - Si es necesario, se reprocesa el ETL en Databricks a partir de las capas Bronze/Silver/Gold o desde los CSV originales.

- Key Vault + Audit Logs
  refuerza la seguridad y la gobernanza de la solución ante incidentes operativos y de seguridad.

Esta documentación, junto con las capturas de Azure, demuestra que la arquitectura no sólo funciona en condiciones normales, sino que está preparada para crecer, ajustarse a la demanda y recuperarse ante fallos, cumpliendo con los criterios de **escalabilidad, elasticidad, alta disponibilidad y DR** de la rúbrica de la práctica.

Despliegue de Dashboard BBVA en la Nube

0. Descargar Archivos

Obtenga los archivos aqui y extraigalos en el escritorio.

0.5. Loguearse

az login

Luego ingresar con una cuenta autorizada y dar enter (para seleccionar la subcripción por default)

1. Asignar Roles AIM

Obtener UPN

az ad user list --query "[].{name:displayName, upn:userPrincipalName}" -o table

Obtener los Object ID

$PERSONA1 = (az ad user show --id "UPN_DE_LA_PERSONA1" --query id -o tsv)

$PERSONA2 = (az ad user show --id "UPN_DE_LA_PERSONA2" --query id -o tsv)

$PERSONA3 = (az ad user show --id "UPN_DE_LA_PERSONA3" --query id -o tsv)

Asignar Roles

- Persona 1

$RG_ID = (az group show -n rg-bbva-dashboard --query id -o tsv)

$ACR_ID = (az acr show -n acrbbvadashboard -g rg-bbva-dashboard --query id -o tsv)

$BACKEND_ID = (az containerapp show -n bbva-backend-api -g rg-bbva-dashboard --query id -o tsv)

$FRONTEND_ID = (az staticwebapp show -n bbva-dashboard-frontend -g rg-bbva-dashboard --query id -o tsv)

$ENV_ID = (az containerapp env show -n managedEnvironment-vnet -g rg-bbva-dashboard --query id -o tsv)

$LAW_ID = (az monitor log-analytics workspace show -n law-bbva-dashboard -g rg-bbva-dashboard --query id -o tsv)

az role assignment create --assignee $PERSONA1 --role "Contributor" --scope $RG_ID

az role assignment create --assignee $PERSONA1 --role "AcrPull" --scope $ACR_ID

az role assignment create --assignee $PERSONA1 --role "AcrPush" --scope $ACR_ID

az role assignment create --assignee $PERSONA1 --role "Container Apps Contributor" --scope $ENV_ID

az role assignment create --assignee $PERSONA1 --role "Contributor" --scope $BACKEND_ID

az role assignment create --assignee $PERSONA1 --role "Contributor" --scope $FRONTEND_ID

az role assignment create --assignee $PERSONA1 --role "Log Analytics Contributor" --scope $LAW_ID

- Persona 2

$DBW_ID = (az databricks workspace show -n dbw-bbva-dashboard -g rg-bbva-dashboard --query id -o tsv)

$KV_ID = (az keyvault show -n kv-bbva-dashboard --query id -o tsv)

$PG_ID = (az postgres flexible-server show -n pg-bbva-dashboard -g rg-bbva-dashboard --query id -o tsv)

$BKP_ID = (az dataprotection backup-vault show -n rg-bbva-dashboard-backup -g rg-bbva-dashboard --query id -o tsv)

$ST_ID = (az storage account show -n stbbvadatalake -g rg-bbva-dashboard --query id -o tsv)

az role assignment create --assignee $PERSONA2 --role "Owner" --scope $DBW_ID

az role assignment create --assignee $PERSONA2 --role "Key Vault Secrets Officer" --scope $KV_ID

az role assignment create --assignee $PERSONA2 --role "Contributor" --scope $PG_ID

az role assignment create --assignee $PERSONA2 --role "Backup Contributor" --scope $BKP_ID

az role assignment create --assignee $PERSONA2 --role "Storage Account Contributor" --scope $ST_ID

az role assignment create --assignee $PERSONA2 --role "Storage Blob Data Contributor" --scope $ST_ID

- Persona 3

$VNET_ID = (az network vnet show -n vnet-bbva-dashboard -g rg-bbva-red --query id -o tsv)

$FW_ID = (az network firewall show -n fw-bbva-dashboard -g rg-bbva-red --query id -o tsv)

$FW_PIP_ID = (az network public-ip show -n pip-firewall -g rg-bbva-red --query id -o tsv)

$RT_ID = (az network route-table show -n rt-firewall -g rg-bbva-red --query id -o tsv)

$FWPOL_ID = (az network firewall policy show -n policy-bbva-firewall -g rg-bbva-red --query id -o tsv)

$DNS_ACR = (az network private-dns zone show -n privatelink.azurecr.io -g rg-bbva-red --query id -o tsv)

$DNS_BLOB = (az network private-dns zone show -n privatelink.blob.core.windows.net -g rg-bbva-red --query id -o tsv)

$DNS_DFS = (az network private-dns zone show -n privatelink.dfs.core.windows.net -g rg-bbva-red --query id -o tsv)

$DNS_PG = (az network private-dns zone show -n privatelink.postgres.database.azure.com -g rg-bbva-red --query id -o tsv)

$DNS_KV = (az network private-dns zone show -n privatelink.vaultcore.azure.net -g rg-bbva-red --query id -o tsv)

$PE_ACR = (az network private-endpoint show -n pe-acr -g rg-bbva-red --query id -o tsv)

$PE_KV = (az network private-endpoint show -n pe-keyvault -g rg-bbva-red --query id -o tsv)

$PE_PG = (az network private-endpoint show -n pe-postgresql -g rg-bbva-red --query id -o tsv)

$PE_BLOB = (az network private-endpoint show -n pe-storage-blob -g rg-bbva-red --query id -o tsv)

$PE_DFS = (az network private-endpoint show -n pe-storage-dfs -g rg-bbva-red --query id -o tsv)

$NIC_ACR = (az network nic show -n pe-acr.nic.6d2eb155-47cf-42d6-81a1-c410338567d8 -g rg-bbva-red --query id -o tsv)

$NIC_KV = (az network nic show -n pe-keyvault.nic.b5962fc2-ffae-45a5-817e-361e905d86cf -g rg-bbva-red --query id -o tsv)

$NIC_PG = (az network nic show -n pe-postgresql.nic.98fda535-3b45-4cbd-aa88-bce0137ebc0c -g rg-bbva-red --query id -o tsv)

$NIC_BLOB = (az network nic show -n pe-storage-blob.nic.fe524dcc-3035-4f13-a043-962b5e72edbd -g rg-bbva-red --query id -o tsv)

$NIC_DFS = (az network nic show -n pe-storage-dfs.nic.dff6e04d-160d-4a36-8273-4fc845f54b7a -g rg-bbva-red --query id -o tsv)

$NSG_CA = (az network nsg show -n nsg-containerapp -g rg-bbva-red --query id -o tsv)

$NSG_DB = (az network nsg show -n nsg-databricks -g rg-bbva-red --query id -o tsv)

$NSG_KV = (az network nsg show -n nsg-keyvault -g rg-bbva-red --query id -o tsv)

$NSG_PG = (az network nsg show -n nsg-postgresql -g rg-bbva-red --query id -o tsv)

$NSG_ST = (az network nsg show -n nsg-storage -g rg-bbva-red --query id -o tsv)

az role assignment create --assignee $PERSONA3 --role "Network Contributor" --scope $VNET_ID

az role assignment create --assignee $PERSONA3 --role "Network Contributor" --scope $FW_ID

az role assignment create --assignee $PERSONA3 --role "Network Contributor" --scope $RT_ID

az role assignment create --assignee $PERSONA3 --role "Network Contributor" --scope $FWPOL_ID

az role assignment create --assignee $PERSONA3 --role "Network Contributor" --scope $FW_PIP_ID

az role assignment create --assignee $PERSONA3 --role "Private DNS Zone Contributor" --scope $DNS_ACR

az role assignment create --assignee $PERSONA3 --role "Private DNS Zone Contributor" --scope $DNS_BLOB

az role assignment create --assignee $PERSONA3 --role "Private DNS Zone Contributor" --scope $DNS_DFS

az role assignment create --assignee $PERSONA3 --role "Private DNS Zone Contributor" --scope $DNS_PG

az role assignment create --assignee $PERSONA3 --role "Private DNS Zone Contributor" --scope $DNS_KV

az role assignment create --assignee $PERSONA3 --role "Network Contributor" --scope $PE_ACR

az role assignment create --assignee $PERSONA3 --role "Network Contributor" --scope $PE_KV

az role assignment create --assignee $PERSONA3 --role "Network Contributor" --scope $PE_PG

az role assignment create --assignee $PERSONA3 --role "Network Contributor" --scope $PE_BLOB

az role assignment create --assignee $PERSONA3 --role "Network Contributor" --scope $PE_DFS

az role assignment create --assignee $PERSONA3 --role "Network Contributor" --scope $NIC_ACR

az role assignment create --assignee $PERSONA3 --role "Network Contributor" --scope $NIC_KV

az role assignment create --assignee $PERSONA3 --role "Network Contributor" --scope $NIC_PG

az role assignment create --assignee $PERSONA3 --role "Network Contributor" --scope $NIC_BLOB

az role assignment create --assignee $PERSONA3 --role "Network Contributor" --scope $NIC_DFS

az role assignment create --assignee $PERSONA3 --role "Network Contributor" --scope $NSG_CA

az role assignment create --assignee $PERSONA3 --role "Network Contributor" --scope $NSG_DB

az role assignment create --assignee $PERSONA3 --role "Network Contributor" --scope $NSG_KV

az role assignment create --assignee $PERSONA3 --role "Network Contributor" --scope $NSG_PG

az role assignment create --assignee $PERSONA3 --role "Network Contributor" --scope $NSG_ST

Crear Resource Group

az group create --name rg-bbva-dashboard --location eastus

1.5 Crear key vault

- Registrar el provider

az provider register --namespace Microsoft.KeyVault

- Crear el vault

az keyvault create --name kv-bbva-dashboard --resource-group rg-bbva-dashboard --location eastus

- Configurar la policy

$USER_OBJECT_ID = az ad signed-in-user show --query id -o tsv

$KV_ID = az keyvault show --name kv-bbva-dashboard --query id -o tsv

az role assignment create --assignee $USER_OBJECT_ID --role "Key Vault Secrets Officer" --scope $KV_ID

2. Crear Storage Account

az storage account create --name stbbvadatalake --resource-group rg-bbva-dashboard --location eastus --sku Standard_LRS `

--kind StorageV2 --hns true

Se crean los contenedores Bronce y Silver

az storage container create --account-name stbbvadatalake --name bronze --auth-mode login

az storage container create --account-name stbbvadatalake --name silver --auth-mode login

Se crean las carpetas Data Sucia y Data Limpia

az storage fs directory create --account-name stbbvadatalake --file-system bronze `

--name data_sucia --auth-mode login

az storage fs directory create --account-name stbbvadatalake --file-system bronze `

--name data_sucia/data_sucia_continuous_integration --auth-mode login

az storage fs directory create --account-name stbbvadatalake --file-system bronze `

--name data_sucia/data_sucia_practitioner --auth-mode login

az storage fs directory create --account-name stbbvadatalake --file-system silver `

--name data_limpia --auth-mode login

az storage fs directory create --account-name stbbvadatalake --file-system silver `

--name data_limpia/data_limpia_continuous_integration --auth-mode login

az storage fs directory create --account-name stbbvadatalake --file-system silver `

--name data_limpia/data_limpia_practitioner --auth-mode login

Obtenga los archivos data sucia aqui y extraigalos en el escritorio.

Guardar Connection String en Key Vault

$AZURE_STORAGE_CONNECTION_STRING = az storage account show-connection-string --name stbbvadatalake `

--resource-group rg-bbva-dashboard --query connectionString -o tsv

az keyvault secret set --vault-name kv-bbva-dashboard --name "azure-storage-connection-string" `

--value "$AZURE_STORAGE_CONNECTION_STRING"

3. Crear PostgreSQL Flexible Server

az postgres flexible-server create --name pg-bbva-dashboard --resource-group rg-bbva-dashboard --location centralus `

--admin-user adminuser --admin-password "SecurePass123!" --sku-name Standard_D2ds_v5 --tier GeneralPurpose `

--storage-size 128 --version 18 --public-access all

Guardar credenciales de PostgreSQL en Key Vault

az keyvault secret set --vault-name kv-bbva-dashboard --name "db-host" --value "pg-bbva-dashboard.postgres.database.azure.com"

az keyvault secret set --vault-name kv-bbva-dashboard --name "db-port" --value "5432"

az keyvault secret set --vault-name kv-bbva-dashboard --name "db-user" --value "adminuser"

az keyvault secret set --vault-name kv-bbva-dashboard --name "db-password" --value "SecurePass123!"

az keyvault secret set --vault-name kv-bbva-dashboard --name "db-name-practitioner" --value "data_oro_practitioner"

az keyvault secret set --vault-name kv-bbva-dashboard --name "db-name-ci" --value "data_oro_ci"

Crear la bases de datos data_oro_practitioner y data_oro_ci

az postgres flexible-server db create --resource-group rg-bbva-dashboard --server-name pg-bbva-dashboard `

--database-name data_oro_practitioner

az postgres flexible-server db create --resource-group rg-bbva-dashboard --server-name pg-bbva-dashboard `

--database-name data_oro_ci

Crear las tablas practitioner (Ingresar la contraseña: SecurePass123!)

psql `

-h pg-bbva-dashboard.postgres.database.azure.com `

-p 5432 `

-U adminuser `

-d data_oro_practitioner `

-f "$env:USERPROFILE\Desktop\Querys\query_practitioner.sql" `

--set=sslmode=require

Crear las tablas continuous integration (Ingresar la contraseña: SecurePass123!)

psql `

-h pg-bbva-dashboard.postgres.database.azure.com `

-p 5432 `

-U adminuser `

-d data_oro_ci `

-f "$env:USERPROFILE\Desktop\Querys\query_ci.sql" `

--set=sslmode=require

Obtenga los archivos query aqui y extraigalos en el escritorio.

4. Crear Databricks

az databricks workspace create --name dbw-bbva-dashboard --resource-group rg-bbva-dashboard `

--location eastus --sku standard --managed-resource-group rg-bbva-dashboard-db-managed --public-network-access Enabled

Obtener acceso admin (Borrar Cache si es necesario)

az role assignment create `

--role "Owner" `

--assignee $(az ad signed-in-user show --query id -o tsv) `

--scope $(az databricks workspace show --name dbw-bbva-dashboard --resource-group rg-bbva-dashboard --query id -o tsv)

Obtener la Key del Datalake

$DL_KEY = az storage account keys list --account-name stbbvadatalake --resource-group rg-bbva-dashboard --query "[0].value" `

-o tsv

Guardar la Key del Datalake en Key Vault

az keyvault secret set --vault-name kv-bbva-dashboard --name "datalake-key" --value "$DL_KEY"

Crear el cluster - Obtener el Token (User Settings → Generate Token)

$DATABRICKS_TOKEN = "PONER AQUI EL TOKEN"

$WORKSPACE_URL = az databricks workspace show --name dbw-bbva-dashboard --resource-group rg-bbva-dashboard `

--query workspaceUrl -o tsv

$headers = @{

"Authorization" = "Bearer $DATABRICKS_TOKEN"

"Content-Type" = "application/json"

}

```
$body = @{
   cluster_name = "cluster-bbva"
   spark_version = "15.4.x-scala2.12"
   node_type_id = "Standard_DS3_v2"
   num_workers = 0
   autotermination_minutes = 20
   custom_tags = @{
      "ResourceClass" = "SingleNode"
   }
   enable_photon = $true
   runtime_engine = "PHOTON"
   spark_conf = @{
      "fs.azure.account.key.stbbvadatalake.dfs.core.windows.net" = $DL_KEY
   }
} | ConvertTo-Json -Depth 10

Invoke-RestMethod `
   -Uri "https://$WORKSPACE_URL/api/2.1/clusters/create" `
   -Method Post `
   -Headers $headers `
   -Body $body
```

Guardar credenciales de Databricks en Key Vault

```
az keyvault secret set --vault-name kv-bbva-dashboard --name "databricks-token" --value "$DATABRICKS_TOKEN"

az keyvault secret set --vault-name kv-bbva-dashboard --name "databricks-workspace-url" --value "https://$WORKSPACE_URL"
```

Configurar Databricks CLI

```
$configContent = @"
[DEFAULT]
host = https://$WORKSPACE_URL
token = $DATABRICKS_TOKEN
"@
```

[System.IO.File]::WriteAllText("$env:USERPROFILE\.databrickscfg", $configContent, [System.Text.Encoding]::ASCII)

Subir los Notebooks (En este caso estan en el escritorio en una carpeta Notebooks)

$USER_EMAIL = az account show --query user.name -o tsv

databricks workspace import-dir "$env:USERPROFILE\Desktop\Notebooks" "/Workspace/Users/$USER_EMAIL"

Guardar el Cluster ID en Key Vault

$CLUSTER_ID = (databricks clusters list --output JSON | ConvertFrom-Json)[0].cluster_id

az keyvault secret set --vault-name kv-bbva-dashboard --name "databricks-cluster-id" --value "$CLUSTER_ID"

5. Crear Container Registry

az acr create --name acrbbvadashboard --resource-group rg-bbva-dashboard --location eastus --sku Standard `

  --admin-enabled true

Guardar credenciales de ACR en Key Vault

$ACR_PASSWORD = az acr credential show --name acrbbvadashboard --query "passwords[0].value" -o tsv

az keyvault secret set --vault-name kv-bbva-dashboard --name "acr-password" --value "$ACR_PASSWORD"

az keyvault secret set --vault-name kv-bbva-dashboard --name "acr-username" --value "acrbbvadashboard"

az keyvault secret set --vault-name kv-bbva-dashboard --name "acr-server" --value "acrbbvadashboard.azurecr.io"

Abrir una terminal en la carpeta del backend para:

- Construir la imagen desde tu Dockerfile

docker build -t bbva-backend:v1 .

- Loguearse al ACR

az acr login --name acrbbvadashboard

- Taguear la imagen con el ACR

docker tag bbva-backend:v1 acrbbvadashboard.azurecr.io/bbva-backend:v1

- Subirla al ACR

docker push acrbbvadashboard.azurecr.io/bbva-backend:v1

Obtenga el backend aqui y extraelo en el escritorio.

6. Crear Container App

Obtener el id del log-analytics workspace

$LAW_ID = az monitor log-analytics workspace show --resource-group rg-bbva-dashboard --workspace-name law-bbva-dashboard `

--query customerId -o tsv

$LAW_KEY = az monitor log-analytics workspace get-shared-keys --resource-group rg-bbva-dashboard --workspace-name law-bbva-dashboard `

--query primarySharedKey -o tsv

Obtener el IDs

$SUBNET_ID = az network vnet subnet show `

--name snet-containerapp-infra `

--vnet-name vnet-bbva-dashboard `

--resource-group rg-bbva-red `

--query id -o tsv

$LAW_ID = az monitor log-analytics workspace show `

--resource-group rg-bbva-dashboard `

--workspace-name law-bbva-dashboard `

--query customerId -o tsv

$LAW_KEY = az monitor log-analytics workspace get-shared-keys `

--resource-group rg-bbva-dashboard `

--workspace-name law-bbva-dashboard `

--query primarySharedKey -o tsv

Crear el nuevo environment con VNET integration

az containerapp env create `

--name managedEnvironment-vnet `

--resource-group rg-bbva-dashboard `

--location eastus `

--infrastructure-subnet-resource-id $SUBNET_ID `

--internal-only false `

--logs-workspace-id $LAW_ID `

--logs-workspace-key $LAW_KEY

Crear variables

$DB_HOST = az keyvault secret show --vault-name kv-bbva-dashboard --name "db-host" --query value -o tsv

$DATABRICKS_WORKSPACE_URL = az keyvault secret show --vault-name kv-bbva-dashboard --name "databricks-workspace-url" --query value -o tsv

$AZURE_STORAGE_CONNECTION_STRING = az keyvault secret show --vault-name kv-bbva-dashboard --name "azure-storage-connection-string" --query value -o tsv

$ACR_PASSWORD = az keyvault secret show --vault-name kv-bbva-dashboard --name "acr-password" --query value -o tsv

$CLUSTER_ID = az keyvault secret show --vault-name kv-bbva-dashboard --name "databricks-cluster-id" --query value -o tsv

$DATABRICKS_TOKEN = az keyvault secret show --vault-name kv-bbva-dashboard --name "databricks-token" --query value -o tsv

$DB_PASSWORD = az keyvault secret show --vault-name kv-bbva-dashboard --name "db-password" --query value -o tsv

Crear el contenedor

$DB_HOST = az keyvault secret show --vault-name kv-bbva-dashboard --name "db-host" --query value -o tsv

Crear el Container App

az containerapp create --name bbva-backend-api --resource-group rg-bbva-dashboard --environment managedEnvironment-vnet `

--image acrbbvadashboard.azurecr.io/bbva-backend:v1 --registry-server acrbbvadashboard.azurecr.io `

--registry-username acrbbvadashboard --registry-password $ACR_PASSWORD --target-port 5000 --ingress external `

--cpu 0.5 --memory 1.0Gi --system-assigned --min-replicas 1 --max-replicas 1 `

--env-vars `

"DB_HOST=$DB_HOST" `

"DB_PORT=5432" `

"DB_USER=adminuser" `

"DB_PASSWORD=$DB_PASSWORD" `

"DB_NAME_PRACTITIONER=data_oro_practitioner" `

"DB_NAME_CI=data_oro_ci" `

"DATABRICKS_WORKSPACE_URL=$DATABRICKS_WORKSPACE_URL" `

"DATABRICKS_TOKEN=$DATABRICKS_TOKEN" `

"FLASK_ENV=production" `

"AZURE_STORAGE_CONNECTION_STRING=$AZURE_STORAGE_CONNECTION_STRING" `

"DATABRICKS_CLUSTER_ID=$CLUSTER_ID" `

"KEY_VAULT_NAME=kv-bbva-dashboard"

Configurar permisos de Key Vault para Container App

$CONTAINER_APP_PRINCIPAL_ID = az containerapp show --name bbva-backend-api --resource-group rg-bbva-dashboard `

--query identity.principalId -o tsv

$KV_ID = az keyvault show --name kv-bbva-dashboard --query id -o tsv

az role assignment create --assignee $CONTAINER_APP_PRINCIPAL_ID --role "Key Vault Secrets User" --scope $KV_ID

Actualizar la imagen

az containerapp update --name bbva-backend-api --resource-group rg-bbva-dashboard `

--image acrbbvadashboard.azurecr.io/bbva-backend:v1

7. Crear Static Web App

az staticwebapp create --name bbva-dashboard-frontend --resource-group rg-bbva-dashboard --location eastus2 --sku Free

Obtener el URL del backend para reemplazarlo en el config del frontend

$API_BASE_URL = "https://$(az containerapp show --name bbva-backend-api --resource-group rg-bbva-dashboard `

--query properties.configuration.ingress.fqdn -o tsv)"

Write-Host $API_BASE_URL

Obtenga el frontend aqui y extraelo en el escritorio.

Abrir una terminal en la carpeta del frontend para contruir la carpeta dist

npm install

npm run build

Subir la carpeta dist

$token = az staticwebapp secrets list --name bbva-dashboard-frontend --resource-group rg-bbva-dashboard `

--query properties.apiKey -o tsv

swa deploy --app-location "$env:USERPROFILE\Desktop\frontend\dist" --deployment-token "$token" --env "production"

8. Crear Logs Analitics WorkSpace

az monitor log-analytics workspace create `

--resource-group rg-bbva-dashboard `

--workspace-name law-bbva-dashboard `

--location eastus

Ruta para ejecutar cualquier archivo

Para los demas

Obtenga los KQLs aqui y ejecutelos.

Para el Databricks (Desde el CLI)

- Cluster Events

databricks clusters events 1129-051945-6zvtepb6

- Para Jobs

databricks jobs list

databricks jobs list-runs --job-id <ID>

databricks jobs get-run-output --run-id <RUN_ID>

9. Crear Alertas

az group create --name rg-bbva-alerts --location eastus

Para el Datalake

- Error rate anormal (transacciones fallidas)

az monitor metrics alert create `

--name alert-storage-errors `

--resource-group rg-bbva-alerts `

--scopes $STORAGE_ID `

--condition "total Transactions > 5 where ResponseType includes Error" `

--description "Errores de Storage > 5 en 5 minutos" `

--evaluation-frequency 1m `

--window-size 5m `

--severity 2

- Latencia alta

az monitor metrics alert create `

--name alert-storage-latency `

--resource-group rg-bbva-alerts `

--scopes $STORAGE_ID `

--condition "avg SuccessE2ELatency > 0.2" `

--description "Latencia del Storage > 200ms" `

--evaluation-frequency 5m `

--window-size 15m `

--severity 3

- Costo inesperado por datos salientes

az monitor metrics alert create `

--name alert-storage-egress `

--resource-group rg-bbva-alerts `

--scopes $STORAGE_ID `

--condition "total Egress > 200" `

--description "Egress del Storage mayor a 200 MB (posible fuga de datos)" `

--evaluation-frequency 30m `

--window-size 24h `

--severity 3

Para el container app

- Error rate en el backend

$API_ID = az containerapp show --name bbva-backend-api --resource-group rg-bbva-dashboard --query id -o tsv

az monitor metrics alert create `

--name alert-backend-5xx `

--resource-group rg-bbva-alerts `

--scopes $API_ID `

--condition "total Requests > 5 where statusCodeCategory includes 5XX" `

--description 'Más de 5 errores HTTP 5xx en 5 min' `

--evaluation-frequency 1m `

--window-size 5m `

--severity 1

- Backend sin requests (posible caída)

az monitor metrics alert create `

--name alert-backend-no-requests `

--resource-group rg-bbva-alerts `

--scopes $API_ID `

--condition "total Requests < 1" `

--description "0 requests por 15 minutos → posible caída del backend" `

--evaluation-frequency 5m `

--window-size 15m `

--severity 2

- CPU muy alta (>70%)

az monitor metrics alert create `

--name alert-backend-cpu-high `

--resource-group rg-bbva-alerts `

--scopes $API_ID `

--condition "avg CpuPercentage > 70" `

--description "CPU del backend mayor al 70%" `

--evaluation-frequency 1m `

--window-size 5m `

--severity 2

Para el postgresql

- Consumo de CPU alto

$PG_ID = az postgres flexible-server show --name pg-bbva-dashboard --resource-group rg-bbva-dashboard --query id -o tsv

az monitor metrics alert create `

--name alert-postgres-cpu-high `

--resource-group rg-bbva-alerts `

--scopes $PG_ID `

--condition "avg cpu_percent > 70" `

--description "CPU de PostgreSQL mayor al 70%" `

--evaluation-frequency 5m `

--window-size 15m `

--severity 2

- Conexiones máximas (>80%)

az monitor metrics alert create `

--name alert-postgres-connections-high `

--resource-group rg-bbva-alerts `

--scopes $PG_ID `

--condition "avg active_connections > 80" `

--description 'Conexiones activas por encima del 80% del máximo permitido' `

--evaluation-frequency 5m `

--window-size 15m `

--severity 2

- Espacio en disco bajo (<20%)

az monitor metrics alert create `

--name alert-postgres-storage-low `

--resource-group rg-bbva-alerts `

--scopes $PG_ID `

--condition "avg storage_percent > 80" `

--description "PostgreSQL está usando más del 80% del almacenamiento" `

--evaluation-frequency 30m `

--window-size 1h `

--severity 2

10. Virtual Network

- Crear la Virtual Network con la primera subnet

az network vnet create `

--name vnet-bbva-dashboard `

--resource-group rg-bbva-red `

--location eastus `

--address-prefix 10.0.0.0/16 `

--subnet-name snet-containerapp `

--subnet-prefix 10.0.1.0/24

Explicación de parámetros: • --address-prefix 10.0.0.0/16: Rango total de IPs (65,536 direccionesdisponibles) • --subnet-prefix 10.0.1.0/24: Subred para Container Apps (256 IPs)

- Crear subnet para el enviroment del ACR

az network vnet subnet create

--name snet-containerapp-infra

--resource-group rg-bbva-red

--vnet-name vnet-bbva-dashboard

--address-prefix 10.0.8.0/23

--delegations Microsoft.App/environments

Crear subnet para PostgreSQL

az network vnet subnet create `

--name snet-postgresql `

--resource-group rg-bbva-red `

--vnet-name vnet-bbva-dashboard `

--address-prefix 10.0.2.0/24

Crear subnet para Databricks (necesita 2 subnets)

az network vnet subnet create `

--name snet-databricks-public `

--resource-group rg-bbva-red `

--vnet-name vnet-bbva-dashboard `

--address-prefix 10.0.3.0/24

```
az network vnet subnet create `
  --name snet-databricks-private `
  --resource-group rg-bbva-red `
  --vnet-name vnet-bbva-dashboard `
  --address-prefix 10.0.4.0/24
```

Crear subnet para Storage Account (Private Endpoints)

```
az network vnet subnet create `
  --name snet-storage `
  --resource-group rg-bbva-red `
  --vnet-name vnet-bbva-dashboard `
  --address-prefix 10.0.5.0/24
```

Crear subnet para Azure Firewall

```
az network vnet subnet create `
  --name AzureFirewallSubnet `
  --resource-group rg-bbva-red `
  --vnet-name vnet-bbva-dashboard `
  --address-prefix 10.0.6.0/24
```

Nota: El nombre AzureFirewallSubnet es obligatorio para el firewall.

Crear subnet para Key Vault (Private Endpoint)

```
az network vnet subnet create `
  --name snet-keyvault `
  --resource-group rg-bbva-red `
  --vnet-name vnet-bbva-dashboard `
  --address-prefix 10.0.7.0/24
```

11. Network Security Groups

- NSG para Container App

Crear NSG

```
az network nsg create `
  --name nsg-containerapp `
  --resource-group rg-bbva-red `
  --location eastus
```

Regla 1: Permitir HTTPS desde internet (puerto 443)

```
az network nsg rule create `
  --name AllowHTTPS `
  --nsg-name nsg-containerapp `
  --resource-group rg-bbva-red `
  --priority 100 `
  --source-address-prefixes Internet `
  --destination-port-ranges 443 `
  --protocol Tcp `
  --access Allow `
  --direction Inbound `
  --description "Permitir tráfico HTTPS público"
```

Explicación: • --priority 100: Menor número = mayor prioridad (rango: 100-4096) • --source-address-prefixes Internet: Desde cualquier IP pública • --destination-port-ranges 443: Puerto HTTPS

Regla 2: Permitir HTTP desde internet (puerto 80)

```
az network nsg rule create `
  --name AllowHTTP `
  --nsg-name nsg-containerapp `
  --resource-group rg-bbva-red `
  --priority 110 `
  --source-address-prefixes Internet `
  --destination-port-ranges 80 `
  --protocol Tcp `
  --access Allow `
  --direction Inbound `
  --description "Permitir tráfico HTTP público"
```

Regla 3: Permitir puerto 5000 para el backend Flask

```
az network nsg rule create `
  --name AllowBackendPort `
  --nsg-name nsg-containerapp `
  --resource-group rg-bbva-red `
```

--priority 120 `

--source-address-prefixes 10.0.0.0/16 `

--destination-port-ranges 5000 `

--protocol Tcp `

--access Allow `

--direction Inbound `

--description "Permitir puerto 5000 del backend desde VNET"

Asociar NSG a la subnet

az network vnet subnet update `

--name snet-containerapp `

--resource-group rg-bbva-red `

--vnet-name vnet-bbva-dashboard `

--network-security-group nsg-containerapp

- NSG para PostgreSQL

Crear NSG

az network nsg create `

--name nsg-postgresql `

--resource-group rg-bbva-red `

--location eastus

Permitir conexiones PostgreSQL solo desde la VNET

az network nsg rule create `

--name AllowPostgreSQL `

--nsg-name nsg-postgresql `

--resource-group rg-bbva-red `

--priority 100 `

--source-address-prefixes 10.0.0.0/16 `

--destination-port-ranges 5432 `

--protocol Tcp `

--access Allow `

--direction Inbound `

--description "Permitir PostgreSQL solo desde la VNET"

Explicación: • --source-address-prefixes 10.0.0.0/16: Solo desde tu VNET, NO desde internet • Puerto 5432 es el puerto estándar de PostgreSQL

Denegar todo el tráfico público entrante

```
az network nsg rule create `
  --name DenyAllInbound `
  --nsg-name nsg-postgresql `
  --resource-group rg-bbva-red `
  --priority 4096 `
  --source-address-prefixes '*' `
  --destination-port-ranges '*' `
  --protocol '*' `
  --access Deny `
  --direction Inbound `
  --description "Denegar todo el tráfico público"
```

Asociar NSG a la subnet

```
az network vnet subnet update `
  --name snet-postgresql `
  --resource-group rg-bbva-red `
  --vnet-name vnet-bbva-dashboard `
  --network-security-group nsg-postgresql
```

- NSG para Storage Account

Crear NSG

```
az network nsg create `
  --name nsg-storage `
  --resource-group rg-bbva-red `
  --location eastus
```

Permitir acceso HTTPS solo desde la VNET

```
az network nsg rule create `
  --name AllowStorageFromVNET `
  --nsg-name nsg-storage `
  --resource-group rg-bbva-red `
```

```
  --priority 100 `
  --source-address-prefixes 10.0.0.0/16 `
  --destination-port-ranges 443 `
  --protocol Tcp `
  --access Allow `
  --direction Inbound `
  --description "Permitir Storage solo desde VNET"
```

Denegar acceso público

```
az network nsg rule create `
  --name DenyPublicAccess `
  --nsg-name nsg-storage `
  --resource-group rg-bbva-red `
  --priority 4096 `
  --source-address-prefixes Internet `
  --destination-port-ranges '*' `
  --protocol '*' `
  --access Deny `
  --direction Inbound `
  --description "Denegar acceso público al Storage"
```

Asociar NSG a la subnet

```
az network vnet subnet update `
  --name snet-storage `
  --resource-group rg-bbva-red `
  --vnet-name vnet-bbva-dashboard `
  --network-security-group nsg-storage
```

- NSG para Databricks

Crear NSG

```
az network nsg create `
  --name nsg-databricks `
  --resource-group rg-bbva-red `
  --location eastus
```

Regla 1: Permitir comunicación con el control plane de Databricks

```
az network nsg rule create `
  --name AllowDatabricksControlPlane `
  --nsg-name nsg-databricks `
  --resource-group rg-bbva-red `
  --priority 100 `
  --source-address-prefixes AzureDatabricks `
  --destination-port-ranges 443 `
  --protocol Tcp `
  --access Allow `
  --direction Inbound `
  --description "Permitir comunicación con control plane de Databricks"
```

Regla 2: Permitir comunicación interna entre nodos de Databricks

```
az network nsg rule create `
  --name AllowDatabricksInternal `
  --nsg-name nsg-databricks `
  --resource-group rg-bbva-red `
  --priority 110 `
  --source-address-prefixes VirtualNetwork `
  --destination-address-prefixes VirtualNetwork `
  --destination-port-ranges '*' `
  --protocol '*' `
  --access Allow `
  --direction Inbound `
  --description "Permitir comunicación interna de Databricks"
```

Asociar NSG a ambas subnets de Databricks

```
az network vnet subnet update `
  --name snet-databricks-public `
  --resource-group rg-bbva-red `
  --vnet-name vnet-bbva-dashboard `
  --network-security-group nsg-databricks
```

```
az network vnet subnet update `
  --name snet-databricks-private `
  --resource-group rg-bbva-red `
  --vnet-name vnet-bbva-dashboard `
  --network-security-group nsg-databricks
```

- NSG para Key Vault

Crear NSG

```
az network nsg create `
  --name nsg-keyvault `
  --resource-group rg-bbva-red `
  --location eastus
```

Permitir HTTPS solo desde la VNET

```
az network nsg rule create `
  --name AllowKeyVaultFromVNET `
  --nsg-name nsg-keyvault `
  --resource-group rg-bbva-red `
  --priority 100 `
  --source-address-prefixes 10.0.0.0/16 `
  --destination-port-ranges 443 `
  --protocol Tcp `
  --access Allow `
  --direction Inbound `
  --description "Permitir Key Vault solo desde VNET"
```

Denegar acceso público

```
az network nsg rule create `
  --name DenyPublicAccess `
  --nsg-name nsg-keyvault `
  --resource-group rg-bbva-red `
  --priority 4096 `
  --source-address-prefixes Internet `
  --destination-port-ranges '*' `
```

--protocol '*' `

--access Deny `

--direction Inbound `

--description "Denegar acceso público al Key Vault"

Asociar NSG a la subnet

az network vnet subnet update `

--name snet-keyvault `

--resource-group rg-bbva-red `

--vnet-name vnet-bbva-dashboard `

--network-security-group nsg-keyvault

12. Private DNS Zone

- DNS Zone para Storage Account (Blob)

Crear Private DNS Zone para Blob Storage

az network private-dns zone create `

--name privatelink.blob.core.windows.net `

--resource-group rg-bbva-red

Vincular la DNS Zone a la VNET

az network private-dns link vnet create `

--name link-blob-to-vnet `

--resource-group rg-bbva-red `

--zone-name privatelink.blob.core.windows.net `

--virtual-network vnet-bbva-dashboard `

--registration-enabled false

Explicación: • --registration-enabled false: No registra automáticamente VMs en el DNS

- DNS Zone para Storage Account (DFS - Data Lake)

Crear Private DNS Zone para Data Lake Storage (DFS)

az network private-dns zone create `

--name privatelink.dfs.core.windows.net `

--resource-group rg-bbva-red

Vincular la DNS Zone a la VNET

az network private-dns link vnet create `

--name link-dfs-to-vnet `

--resource-group rg-bbva-red `

--zone-name privatelink.dfs.core.windows.net `

--virtual-network vnet-bbva-dashboard `

--registration-enabled false

- DNS Zone para PostgreSQL

Crear Private DNS Zone para PostgreSQL

az network private-dns zone create `

--name privatelink.postgres.database.azure.com `

--resource-group rg-bbva-red

Vincular la DNS Zone a la VNET

az network private-dns link vnet create `

--name link-postgres-to-vnet `

--resource-group rg-bbva-red `

--zone-name privatelink.postgres.database.azure.com `

--virtual-network vnet-bbva-dashboard `

--registration-enabled false

- DNS Zone para Key Vault

Crear Private DNS Zone para Key Vault

az network private-dns zone create `

--name privatelink.vaultcore.azure.net `

--resource-group rg-bbva-red

Vincular la DNS Zone a la VNET

az network private-dns link vnet create `

--name link-keyvault-to-vnet `

--resource-group rg-bbva-red `

--zone-name privatelink.vaultcore.azure.net `

--virtual-network vnet-bbva-dashboard `

--registration-enabled false

- DNS Zone para Container Registry

Crear Private DNS Zone para ACR

az network private-dns zone create `

--name privatelink.azurecr.io `

--resource-group rg-bbva-red

Vincular la DNS Zone a la VNET

az network private-dns link vnet create `

--name link-acr-to-vnet `

--resource-group rg-bbva-red `

--zone-name privatelink.azurecr.io `

--virtual-network vnet-bbva-dashboard `

--registration-enabled false

13. Private Endpoints

- Deshabilitar acceso público a los servicios

Para las llaves

az keyvault update `

--name kv-bbva-dashboard `

--resource-group rg-bbva-dashboard `

--public-network-access Disabled

Para el contenedor

az acr update `

--name acrbbvadashboard `

--resource-group rg-bbva-dashboard `

--sku Premium

--public-network-enabled false

- Private Endpoint para Storage Account (Blob)

Deshabilitar políticas de red en la subnet (requerido para Private Endpoints)

az network vnet subnet update `

--name snet-storage `

--resource-group rg-bbva-red `

--vnet-name vnet-bbva-dashboard `

--disable-private-endpoint-network-policies true

Crear Private Endpoint para Blob

az network private-endpoint create `

--name pe-storage-blob `

--resource-group rg-bbva-red `

--vnet-name vnet-bbva-dashboard `

--subnet snet-storage `

--private-connection-resource-id $(az storage account show --name stbbvadatalake --resource-group rg-bbva-dashboard --query id -o tsv) `

--group-id blob `

--connection-name conn-storage-blob `

--location eastus

Explicación: • --group-id blob: Tipo de servicio (blob, dfs, table, queue, file) • --private-connection-resource-id: ID del Storage Account

Crear registro DNS automático para Blob

az network private-endpoint dns-zone-group create `

--name zg-storage-blob `

--resource-group rg-bbva-red `

--endpoint-name pe-storage-blob `

--private-dns-zone privatelink.blob.core.windows.net `

--zone-name blob

- Private Endpoint para Storage Account (DFS - Data Lake)

Crear Private Endpoint para DFS

az network private-endpoint create `

--name pe-storage-dfs `

--resource-group rg-bbva-red `

--vnet-name vnet-bbva-dashboard `

--subnet snet-storage `

--private-connection-resource-id $(az storage account show --name stbbvadatalake --resource-group rg-bbva-dashboard --query id -o tsv) `

--group-id dfs `

--connection-name conn-storage-dfs `

--location eastus

Crear registro DNS automático para DFS

```
az network private-endpoint dns-zone-group create `
  --name zg-storage-dfs `
  --resource-group rg-bbva-red `
  --endpoint-name pe-storage-dfs `
  --private-dns-zone privatelink.dfs.core.windows.net `
  --zone-name dfs
```

- Private Endpoint para PostgreSQL

Deshabilitar políticas de red en la subnet de PostgreSQL

```
az network vnet subnet update `
  --name snet-postgresql `
  --resource-group rg-bbva-red `
  --vnet-name vnet-bbva-dashboard `
  --disable-private-endpoint-network-policies true
```

Crear Private Endpoint para PostgreSQL

```
az network private-endpoint create `
  --name pe-postgresql `
  --resource-group rg-bbva-red `
  --vnet-name vnet-bbva-dashboard `
  --subnet snet-postgresql `
  --private-connection-resource-id $(az postgres flexible-server show --name pg-bbva-dashboard --resource-group rg-bbva-dashboard --query id -o tsv) `
  --group-id postgresqlServer `
  --connection-name conn-postgresql `
  --location eastus
```

Crear registro DNS automático

```
az network private-endpoint dns-zone-group create `
  --name zg-postgresql `
  --resource-group rg-bbva-red `
  --endpoint-name pe-postgresql `
  --private-dns-zone privatelink.postgres.database.azure.com `
  --zone-name postgres
```

- Private Endpoint para Key Vault

Deshabilitar políticas de red en la subnet de Key Vault

```
az network vnet subnet update `
  --name snet-keyvault `
  --resource-group rg-bbva-red `
  --vnet-name vnet-bbva-dashboard `
  --disable-private-endpoint-network-policies true
```

Crear Private Endpoint para Key Vault

```
az network private-endpoint create `
  --name pe-keyvault `
  --resource-group rg-bbva-red `
  --vnet-name vnet-bbva-dashboard `
  --subnet snet-keyvault `
  --private-connection-resource-id $(az keyvault show --name kv-bbva-dashboard --resource-group rg-bbva-dashboard --query id -o tsv) `
  --group-id vault `
  --connection-name conn-keyvault `
  --location eastus
```

Crear registro DNS automático

```
az network private-endpoint dns-zone-group create `
  --name zg-keyvault `
  --resource-group rg-bbva-red `
  --endpoint-name pe-keyvault `
  --private-dns-zone privatelink.vaultcore.azure.net `
  --zone-name vault
```

- Private Endpoint para Container Registry

Crear Private Endpoint para ACR

```
az network private-endpoint create `
  --name pe-acr `
  --resource-group rg-bbva-red `
  --vnet-name vnet-bbva-dashboard `
```

--subnet snet-storage `

--private-connection-resource-id $(az acr show --name acrbbvadashboard --resource-group rg-bbva-dashboard --query id -o tsv) `

--group-id registry `

--connection-name conn-acr `

--location eastus

Crear registro DNS automático

az network private-endpoint dns-zone-group create `

--name zg-acr `

--resource-group rg-bbva-red `

--endpoint-name pe-acr `

--private-dns-zone privatelink.azurecr.io `

--zone-name registry

14. Firewall

- Crear IP Pública para el Firewall

El Firewall necesita una IP pública estática

az network public-ip create `

--name pip-firewall `

--resource-group rg-bbva-red `

--location eastus `

--allocation-method Static `

--sku Standard

Explicación: • --allocation-method Static: IP fija (no cambia) • --sku Standard: Requerido para Firewall

- Crear Azure Firewall Policy

Crear política de firewall

az network firewall policy create `

--name policy-bbva-firewall `

--resource-group rg-bbva-red `

--location eastus `

--sku Standard

Crear colección de reglas de aplicación (URLs permitidas)

```
az network firewall policy rule-collection-group create `
  --name rcg-app-rules `
  --policy-name policy-bbva-firewall `
  --resource-group rg-bbva-red `
  --priority 100
```

Regla 1: Permitir acceso a servicios de Azure

```
az network firewall policy rule-collection-group collection add-filter-collection `
  --name rc-allow-azure `
  --policy-name policy-bbva-firewall `
  --resource-group rg-bbva-red `
  --rcg-name rcg-app-rules `
  --collection-priority 100 `
  --action Allow `
  --rule-name AllowAzureServices `
  --rule-type ApplicationRule `
  --target-fqdns "*.azure.com" "*.microsoft.com" "*.windows.net" `
  --source-addresses 10.0.0.0/16 `
  --protocols Https=443
```

Explicación: • --target-fqdns: Dominios permitidos • *.azure.com: Servicios de Azure • --protocols Https=443: Solo HTTPS

Regla 2: Permitir acceso a Databricks

```
az network firewall policy rule-collection-group collection rule add `
  --name AllowDatabricks `
  --policy-name policy-bbva-firewall `
  --resource-group rg-bbva-red `
  --rcg-name rcg-app-rules `
  --collection-name rc-allow-azure `
  --rule-type ApplicationRule ` rule-type
  --target-fqdns "*.databricks.net" "*.azuredatabricks.net" `
  --source-addresses 10.0.3.0/24 10.0.4.0/24 `
  --protocols Https=443
```

Regla 3: Permitir npm y paquetes de Python (para Databricks)

```
az network firewall policy rule-collection-group collection rule add `
  --name AllowPackages `
  --policy-name policy-bbva-firewall `
  --resource-group rg-bbva-red `
  --rcg-name rcg-app-rules `
  --collection-name rc-allow-azure `
  --rule-type ApplicationRule `
  --target-fqdns "*.pypi.org" "*.npmjs.org" "*.github.com" `
  --source-addresses 10.0.0.0/16 `
  --protocols Https=443
```

Regla 4: Permitir el dominio del Container App

```
az network firewall policy rule-collection-group collection rule add `
  --name AllowContainerAppBackend `
  --policy-name policy-bbva-firewall `
  --resource-group rg-bbva-red `
  --rcg-name rcg-app-rules `
  --collection-name rc-allow-azure `
  --rule-type ApplicationRule `
  --target-fqdns "*.azurecontainerapps.io" `
  --source-addresses 10.0.0.0/16 `
  --protocols Https=443
```

Regla 5: Permitir tráfico desde el dominio del Static Web App

```
az network firewall policy rule-collection-group collection rule add `
  --name AllowStaticWebApp `
  --policy-name policy-bbva-firewall `
  --resource-group rg-bbva-red `
  --rcg-name rcg-app-rules `
  --collection-name rc-allow-azure `
  --rule-type ApplicationRule `
  --target-fqdns "*.azurestaticapps.net" `
```

--source-addresses "*" `

--protocols Https=443

- Crear el Azure Firewall

Crear el Firewall

az network firewall create `

--name fw-bbva-dashboard `

--resource-group rg-bbva-red `

--location eastus `

--vnet-name vnet-bbva-dashboard `

--firewall-policy policy-bbva-firewall

Explicación: • --enable-dns-proxy true: El Firewall actúa como servidor DNS

Asociar la IP pública al Firewall

az network firewall ip-config create `

--name fw-config `

--firewall-name fw-bbva-dashboard `

--resource-group rg-bbva-red `

--vnet-name vnet-bbva-dashboard `

--public-ip-address pip-firewall

- Obtener la IP privada del Firewall

$FIREWALL_PRIVATE_IP = az network firewall show `

--name fw-bbva-dashboard `

--resource-group rg-bbva-red `

--query 'ipConfigurations[0].privateIPAddress' -o tsv

Write-Host "Firewall Private IP: $FIREWALL_PRIVATE_IP"

- Crear Route Table para dirigir tráfico al Firewall

Crear Route Table

az network route-table create `

--name rt-firewall `

--resource-group rg-bbva-red `

--location eastus

Crear ruta: Todo el tráfico de salida va al Firewall

az network route-table route create `

--name route-to-firewall `

--resource-group rg-bbva-red `

--route-table-name rt-firewall `

--address-prefix 0.0.0.0/0 `

--next-hop-type VirtualAppliance `

--next-hop-ip-address $FIREWALL_PRIVATE_IP

Explicación: • --address-prefix 0.0.0.0/0: Todo el tráfico de internet • --next-hop-type VirtualAppliance: El Firewall es un "appliance virtual"

Asociar Route Table a las subnets

az network vnet subnet update `

--name snet-containerapp `

--resource-group rg-bbva-red `

--vnet-name vnet-bbva-dashboard `

--route-table rt-firewall

az network vnet subnet update `

--name snet-databricks-public `

--resource-group rg-bbva-red `

--vnet-name vnet-bbva-dashboard `

--route-table rt-firewall

az network vnet subnet update `

--name snet-databricks-private `

--resource-group rg-bbva-red `

--vnet-name vnet-bbva-dashboard `

--route-table rt-firewall

- Configurar Diagnostic Settings para el Firewall (enviar logs a Log Analytics)

$FIREWALL_ID = az network firewall show `

--name fw-bbva-dashboard `

--resource-group rg-bbva-red `

--query id -o tsv

```
$LAW_ID = az monitor log-analytics workspace show `
  --resource-group rg-bbva-dashboard `
  --workspace-name law-bbva-dashboard `
  --query id -o tsv
az monitor diagnostic-settings create `
  --name diag-firewall `
  --resource $FIREWALL_ID `
  --workspace $LAW_ID `
  --logs
'[{"category":"AzureFirewallApplicationRule","enabled":true},{"category":"AzureFirewallNetwo
rkRule","enabled":true}
  --metrics '[{"category":"AllMetrics","enabled":true}]'
```

Algunas reglas adicionales

Esto hace que el firewall deje pasar la subida de archivos.

```
az network firewall policy rule-collection-group collection rule add `
  --name AllowBlobStorage `
  --policy-name policy-bbva-firewall `
  --resource-group rg-bbva-red `
  --rcg-name rcg-app-rules `
  --collection-name rc-allow-azure `
  --rule-type ApplicationRule `
  --target-fqdns "*.blob.core.windows.net" "*.dfs.core.windows.net" `
  --source-addresses 10.0.0.0/16 `
  --protocols Https=443
```

Asignar permisos a la identidad del Container App

```
$CONTAINER_APP_PRINCIPAL_ID = az containerapp show `
    --name bbva-backend-api `
    --resource-group rg-bbva-dashboard `
    --query identity.principalId -o tsv
az role assignment create `
  --assignee $CONTAINER_APP_PRINCIPAL_ID `
  --role "Storage Blob Data Contributor" `
```

--scope $(az storage account show --name stbbvadatalake --resource-group rg-bbva-dashboard --query

az network firewall policy rule-collection-group collection rule add `

--name AllowStorageADLS `

--policy-name policy-bbva-firewall `

--resource-group rg-bbva-red `

--rcg-name rcg-app-rules `

--collection-name rc-allow-azure `

--rule-type ApplicationRule `

--target-fqdns "*.blob.core.windows.net" "*.dfs.core.windows.net" "login.microsoftonline.com" "*.login.microsoftonline.com" "*.aadcdn.microsoftonline-p.com" `

--source-addresses 10.0.0.0/16 `

--protocols Https=443

az network firewall policy rule-collection-group collection rule add `

--name AllowAzureManagement `

--policy-name policy-bbva-firewall `

--resource-group rg-bbva-red `

--rcg-name rcg-app-rules `

--collection-name rc-allow-azure `

--rule-type ApplicationRule `

--target-fqdns "management.azure.com" `

--source-addresses 10.0.0.0/16 `

--protocols Https=443

- Backup Vault

Registrar los providers

az provider register --namespace Microsoft.DataProtection

az provider register --namespace Microsoft.RecoveryServices

Crear Backup Vault

az dataprotection backup-vault create `

--resource-group rg-bbva-dashboard `

--vault-name rg-bbva-dashboard-backup `

--location eastus `

```
  --storage-settings type="GeoRedundant" datastore-type="VaultStore"
```

Habilitar identidad administrada (System Assigned)

```
az dataprotection backup-vault update `
  --resource-group rg-bbva-dashboard `
  --vault-name rg-bbva-dashboard-backup `
  --set identity.type="SystemAssigned"
```

Activar Soft Delete

```
az storage account blob-service-properties update `
  --account-name stbbvadatalake `
  --resource-group rg-bbva-dashboard `
  --enable-delete-retention true `
  --delete-retention-days 30
```

Container Soft Delete

```
az storage account blob-service-properties update `
  --account-name stbbvadatalake `
  --resource-group rg-bbva-dashboard `
  --enable-container-delete-retention true `
  --container-delete-retention-days 30
```

Restore Status

```
az storage container restore-status show `
  --account-name stbbvadatalake `
  --resource-group rg-bbva-dashboard
```

Account Redundancy (GRS check)

```
az storage account show `
  --name stbbvadatalake `
  --resource-group rg-bbva-dashboard `
  --query sku.name -o tsv
```

Flujo de Implementación

1. Asignar los roles (IAM)

Antes de crear recursos en Azure, se definen claramente las responsabilidades de cada miembro del equipo y se asignan los **roles mínimos necesarios (principio de mínimo privilegio)** usando Azure RBAC.

En este entorno se trabaja con tres personas ficticias: **Persona 1, Persona 2 y Persona 3**, cada una con un ámbito de administración bien delimitado.

1.1 Persona 1 – Propietario del despliegue de aplicaciones

Responsabilidades principales

Persona 1 se encarga de todo lo relacionado con el despliegue y operación de las aplicaciones:

- **Azure Container Registry**: acrbbvadashboard
- **Backend**: Container App bbva-backend-api
- **Frontend**: Static Web App bbva-dashboard-frontend
- **Container Apps Environment**: managedEnvironment-vnet
- **Log Analytics Workspace**: law-bbva-dashboard

Objetivo: poder construir, versionar y desplegar el backend y frontend, además de revisar la observabilidad básica de la solución.

Roles asignados a Persona 1

Recurso	**Rol**
Resource Group rg-bbva-dashboard	Contributor (base)
ACR acrbbvadashboard	AcrPush, AcrPull
Container App bbva-backend-api	Contributor
Static Web App bbva-dashboard-frontend	Contributor
Container Apps Environment managedEnvironment-vnet	Container Apps Contributor
Log Analytics law-bbva-dashboard	Log Analytics Contributor

1.2 Persona 2 – Dueño de datos y analytics (Data & Security Engineer)

Responsabilidades principales

Persona 2 administra todo lo que tiene que ver con **datos, gobernanza y respaldo**:

- Databricks Workspace: dbw-bbva-dashboard
- Key Vault: kv-bbva-dashboard
- PostgreSQL Flexible Server: pg-bbva-dashboard
- Backup Vault: rg-bbva-dashboard-backup
- Storage Account / Data Lake: stbbvadatalake (incluye niveles **Bronze / Silver / Gold**)

Objetivo: garantizar que los datos estén seguros, respaldados, encriptados y disponibles para los procesos de ingeniería y analytics.

Roles asignados a Persona 2

Recurso	**Rol**
Databricks Workspace	Owner
Key Vault kv-bbva-dashboard	Key Vault Secrets Officer
PostgreSQL Flexible Server	Contributor
Backup Vault	Backup Contributor
Storage Account stbbvadatalake	Storage Account Contributor
Blobs / Data Lake del Storage	Storage Blob Data Contributor

1.3 Persona 3 – Seguridad, redes y conectividad (Network & Firewall Admin)

Responsabilidades principales

Persona 3 es responsable de toda la **capa de red y perímetro de seguridad**:

- Virtual Network: vnet-bbva-dashboard
- Network Security Groups (NSGs) de todas las subnets
- Private Endpoints
- Private DNS Zones
- Firewall: fw-bbva-dashboard
- Firewall Policy: policy-bbva-firewall
- Route Table: rt-firewall
- Public IPs asociadas al firewall y otros componentes de red

Objetivo: asegurar que toda la comunicación entre servicios sea **privada, controlada y trazable**, aplicando buenas prácticas de segmentación y filtrado.

Roles asignados a Persona 3

Recurso	**Rol**
VNet completa	Network Contributor
NSGs	Network Contributor
Private Endpoints	Network Contributor
Private DNS Zones	Private DNS Zone Contributor
Public IPs	Network Contributor
Firewall	Network Contributor
Firewall Policy	Network Contributor
Route Table	Network Contributor

2. Crear los Resource Group

Como primer paso se definieron tres grupos de recursos para separar responsabilidades y facilitar la gobernanza:

1. rg-bbva-alerts

Propósito: Concentrar recursos de monitoreo y alertas. **Recursos incluidos:**

- Log Analytics
- Azure Monitor
- Action Groups

2. rg-bbva-dashboard

Propósito: Agrupar servicios directamente relacionados al dashboard. **Recursos incluidos:**

- Storage Account del Data Lake
- Servidor PostgreSQL flexible
- Workspace de Databricks
- Otros componentes de aplicación (ej. backend/frontend) ligados al proyecto

3. rg-bbva-red

Propósito: Reservado para componentes de red y seguridad. **Recursos incluidos:**

- Virtual Network y subredes privadas
- Network Security Groups (NSG)
- Private Endpoints
- Azure Firewall
- Recursos de gobernanza (Private DNS Zone, Recovery Services Vault)

Beneficios de esta estructura:

- **Separación por dominio** (alertas, carga de trabajo y red)
- Permite aplicar **políticas y permisos diferenciados** por equipo
- **Simplifica la administración** y gobernanza

3. Crear key vault

Se creó el recurso **kv-bbva-dashboard** dentro del grupo de recursos **rg-bbva-dashboard**.

Propósito

Centralizar todos los secretos sensibles utilizados por la solución.

Secretos almacenados

- Cadenas de conexión hacia **PostgreSQL Flexible Server**
- Claves de acceso del **Storage Account**
- Credenciales de **service principal** o **identidades administradas** que consumen el Data Lake
- Cualquier otro secreto necesario para el **backend** o los **jobs de Databricks**

Configuración destacada

- **SKU:** Standard (suficiente para escenarios internos y académicos)
- **Soft delete:** Habilitado (protege contra eliminaciones accidentales)
- **Control de acceso:** Integración con Access Policies / IAM

Políticas de acceso

El acceso está limitado únicamente a:

- Identidades de **Databricks**
- Identidades del **backend** / scripts de administración
- Usuarios **administradores** del entorno

Beneficio de seguridad

Evita almacenar credenciales en:

- Código fuente
- Archivos .env
- Variables de entorno sin protección

4. Crear el Storage Account

Para implementar el Data Lake se creó el Storage Account **stbbvadatalake** con las siguientes características:

- **Tipo:** StorageV2 (general purpose v2)
- **Hierarchical namespace:** Enabled, para habilitar Azure Data Lake Storage Gen2
- **Replicación:** LRS (Locally Redundant Storage), suficiente para el ámbito académico del proyecto
- **Access tier:** Hot, ya que los datos se leen y escriben en cada ciclo de carga mensual

Acceso seguro mediante:

- Require secure transfer for REST API operations = Enabled
- Integración con Private Endpoints (se observan conexiones privadas configuradas en el panel de Networking)

Dentro del Storage Account se definieron varios contenedores:

- **bronze:** capa de data sucia / raw, donde se almacenan los CSV tal como llegan
- **silver:** capa de data limpia / transformada, en formato parquet u otro formato optimizado tras el ETL
- **slogs** y **$blobchangefeed:** contenedores técnicos para logs/cambios de blobs (propios del servicio)

Esta estructura respeta la separación raw/trusted y facilita el control de acceso por capa.

Estructura Bronze – Data Sucia

En el contenedor **bronze** se creó una carpeta lógica **data_sucia** que separa los archivos según el dominio funcional:

- data_sucia_practitioner
- data_sucia_continuous_integration

En estas ubicaciones se cargan los archivos CSV exportados desde el Marco Playbook (o sus equivalentes de prueba en este entorno).

Cada subcarpeta representa el **"raw zone"** de su dominio, sin transformaciones ni limpieza aplicada.

Estructura Silver – Data Limpia

En el contenedor **silver** se creó la carpeta **data_limpia**, también separada por dominio:

- data_limpia_practitioner
- data_limpia_continuous_integration

Aquí Databricks escribe los resultados de las transformaciones: datos limpios, tipificados y listos para ser cargados en la capa Oro (PostgreSQL).

Esta separación permite trazar claramente qué dataset se encuentra en qué nivel de calidad.

5. Crear PostgreSQL Flexible Server

Para la capa de datos Oro se optó por **Azure Database for PostgreSQL – Flexible Server**, alineado con el motor usado en el entorno local.

El servidor creado es:

- **Nombre:** pg-bbva-dashboard
- **Ubicación:** Central US
- **Tipo:** General Purpose, tamaño D2ds v5 (2 vCores, 8 GB RAM, 128 GB storage)
- **Acceso restringido** mediante reglas de red y Private Endpoints (conectado a la VNet de rg-bbva-red)
- **Alta disponibilidad** deshabilitada en este entorno académico (pero documentada como recomendación para entornos productivos)

Este servidor se utiliza como **Data Warehouse relacional** donde se modela la capa Oro con esquemas en estrella para Practitioner y CI.

Creación de las bases de datos Oro

Dentro del servidor pg-bbva-dashboard se crearon las bases de datos de la capa Oro:

- data_oro_practitioner
- data_oro_ci

Cada una contiene:

- Tablas de hechos y dimensiones necesarias para los KPIs del dominio
- Índices y claves foráneas que soportan las consultas del dashboard
- Esquemas diseñados para facilitar las consultas analíticas desde el backend y, si se necesitara, desde herramientas externas de BI

Estas bases de datos son el **destino final** del ETL ejecutado en Databricks.

6. Crear Databricks

Para el procesamiento distribuido se aprovisionó un workspace de **Azure Databricks**:

- **Nombre:** dbw-bbva-dashboard
- **Resource group:** rg-bbva-dashboard
- **Tipo de workspace:** Hybrid
- **Enable No Public IP:** Yes, lo que obliga a consumir Databricks a través de la red privada y endpoints seguros, alineado con los requisitos de seguridad del proyecto

Este workspace es el **punto central** para desarrollar notebooks PySpark, ejecutar el ETL y orquestar los jobs.

Creación del clúster de Databricks

Dentro del workspace se configuró el clúster **cluster-bbva**, con las siguientes características:

- **Runtime:** Databricks 16.4 LTS (incluye Apache Spark 3.5.2)
- **Tipo de nodo:** Standard_D4as_v5 (16 GB Memory, 4 Cores)
- **Modo:** Single node (suficiente para los volúmenes actuales)
- **Auto-termination:** 20 minutos de inactividad, para evitar costos innecesarios
- **Access mode:** Custom, permitiendo ajustar permisos de acceso a datos según las necesidades del ETL

Este clúster es el que ejecuta los notebooks de limpieza y transformación que mueven la información desde:

1. **Bronze → Silver** en el Data Lake
2. **Silver → Oro** en PostgreSQL Flexible Server

Creación de Jobs de ETL

Sobre el clúster cluster-bbva se diseñaron dos Jobs principales (ilustrativos en este entorno académico):

Job ETL Practitioner

- Lee los CSV crudos desde bronze/data_sucia_practitioner
- Aplica limpieza, tipificación y joins necesarios
- Escribe los datos limpios en silver/data_limpia_practitioner
- Finalmente carga la capa Oro en la base de datos data_oro_practitioner (tablas de hechos y dimensiones)

Job ETL Continuous Integration (CI)

- Mismo patrón que el anterior, pero usando:
    - bronze/data_sucia_continuous_integration
    - silver/data_limpia_continuous_integration
    - Base de datos data_oro_ci como destino final

Nota: En un escenario productivo, ambos jobs se programarían para ejecutarse automáticamente 1 vez al mes, alineados con la frecuencia de exportación del Marco Playbook. En este proyecto se dejan configurados de forma ilustrativa, demostrando la lógica de orquestación y la trazabilidad del pipeline, aunque no estén conectados a un origen real productivo.

Subir los notebooks a Databricks

Una vez creado el workspace y el clúster de Databricks, se suben los notebooks que implementan el ETL completo sobre las tablas de Practitioner y Continuous Integration.

En la siguiente captura se observa la carpeta de trabajo del usuario con los seis notebooks creados:

- Limpiar Practitioner
- Limpiar Continuous Integration
- Transformar Practitioner
- Transformar Continuous Integration
- Vista Practitioner
- Vista Continuous Integration

Los notebooks siguen el patrón Medallion:

- Los notebooks de **Limpieza** leen los CSV ubicados en la capa Bronze y escriben datos tipificados y depurados en la capa Silver

- Los notebooks de **Transformación** enriquecen la información Silver (joins, derivación de campos, normalización) y preparan las tablas de Gold que luego se cargan en PostgreSQL
- Los notebooks de **Vista** generan las vistas de negocio (KPIs) que se consultan desde el backend

Nota: Estos notebooks pueden ejecutarse manualmente durante el laboratorio o programarse como Jobs para que se disparen automáticamente (por ejemplo, una vez al mes cuando Marco Playbook publique nuevos archivos).

7. Crear Container Registry

Para empaquetar y versionar la API de backend en contenedores se crea un **Azure Container Registry (ACR)** dentro del resource group rg-bbva-dashboard.

En la captura se aprecia el registro privado con:

- **Pricing plan:** tipo Premium (útil para escenarios corporativos y mayor throughput)
- **Integración** con la suscripción y el grupo de recursos del proyecto
- **Soft delete** deshabilitado (para simplificar el laboratorio)

Este registro actúa como **repositorio central de imágenes Docker internas**, evitando exponer el backend en registries públicos y permitiendo controlar versiones, acceso y políticas de seguridad.

Publicar la imagen del backend en el Registry

A partir del código del backend Flask se construye una imagen Docker y se publica en el ACR.

En la captura del repositorio bbva-backend se ve la etiqueta v1, que corresponde a la primera versión estable del servicio:

La imagen incluye:

- Código Flask de la API
- Dependencias Python (requirements.txt)
- Configuración de Gunicorn para servir la aplicación

El tag v1 permite gestionar versiones posteriores (v2, v3, etc.) y facilitar rollbacks en caso de errores de despliegue.

El flujo típico es:

1. docker build de la imagen del backend
2. docker tag apuntando al login server del ACR
3. docker push para publicar la versión en el registro

8. Crear Container Apps Environment

Como capa de ejecución para el backend se despliega un **Azure Container Apps Environment** asociado a una VNet (managedEnvironment-vnet).

En la captura se observa:

- El entorno ejecutándose en la región **East US**
- La integración con la red virtual y la subred de infraestructura, lo que permite:
    - Exponer el backend de forma controlada
    - Conectarlo de manera segura a PostgreSQL y al Storage Account
- Un contador de aplicaciones donde, para este laboratorio, se despliega una sola API: bbva-backend-api

Este entorno funcionará como **"cluster lógico"** para todas las Container Apps relacionadas con el dashboard.

9. Crear el Conteiner App

Sobre el environment anterior se crea la Container App **bbva-backend-api**, que ejecuta la imagen bbva-backend:v1 alojada en el ACR.

La captura muestra:

- Estado **Running** y URL pública de la aplicación
- Asociación al environment managedEnvironment-vnet
- Suscripción y resource group rg-bbva-dashboard

En esta Container App se configuran:

- Recursos (CPU/RAM) apropiados para una API ligera
- Variables de entorno con las cadenas de conexión a PostgreSQL y Storage, obtenidas desde Azure Key Vault
- Escalamiento automático basado en métricas (por ejemplo, número de requests o porcentaje de CPU), cumpliendo con los criterios de escalabilidad de la rúbrica

Esta API es el **punto de entrada** para el frontend React y otros posibles consumidores internos.

10. Crear el Static Web App

Para la capa de presentación se utiliza **Azure Static Web Apps**, donde se despliega el dashboard web desarrollado en React.

En la primera captura se ve el recurso **bbva-dashboard-frontend**:

- Plan **Free**, suficiente para el laboratorio
- Dominio generado por Azure (URL de acceso público)
- Estado **Ready**, indicando que el último despliegue fue exitoso

En la segunda captura (sección **Environments**) se observa:

- Un entorno de producción activo al **100 %** del tráfico
- La posibilidad de usar entornos **preview** vinculados a ramas o Pull Requests (útil para futuras mejoras de CI/CD)

El flujo de despliegue es:

1. Clonar el repositorio del frontend
2. Ejecutar npm install y npm run build para generar la carpeta de salida (dist o build)
3. Configurar Static Web Apps para que tome esa carpeta como origen del contenido estático
4. Cada **push a la rama principal** dispara un nuevo build y despliegue

11. Crear el Log Analytics workspace

Para centralizar métricas y logs de toda la solución se crea un **Log Analytics workspace** denominado law-bbva-dashboard. Se puede observar:

- Workspace en la región **East US**
- Modelo de facturación **Pay-as-you-go**
- Integración con la suscripción y el resource group del proyecto

Este workspace es el **backend de observabilidad** sobre el que se apoyan:

- Azure Monitor
- Las métricas y logs de:
    - Storage Account (Data Lake)
    - Container Apps (backend)
    - Azure Database for PostgreSQL
    - Otros recursos que se deseen monitorear

Una vez configurado el envío de diagnósticos al workspace, es posible ejecutar consultas **Kusto Query Language (KQL)** para analizar el comportamiento del entorno. En la imagen se muestra un ejemplo de consulta sobre la tabla **StorageBlobLogs**.

Esta consulta permite:

- Ver las últimas operaciones de lectura/escritura realizadas sobre la capa **Bronze** del Data Lake
- Identificar errores (códigos 4xx/5xx) y patrones de acceso durante la ejecución del ETL

12. Crear las Alertas

Se configuran reglas de alerta en Azure Monitor para reaccionar ante problemas de rendimiento, disponibilidad o capacidad. En la captura de **Alert rules** se visualizan, entre otras, las siguientes reglas:

- **alert-backend-5xx**: se dispara cuando el backend devuelve más de 5 respuestas 5xx, indicando errores en la API
- **alert-backend-cpu-high**: alerta cuando el CPU de la Container App supera el 70 % durante un periodo sostenido
- **alert-backend-no-requests**: detecta ausencia de requests (posible caída de tráfico o problemas de conectividad)
- **alert-postgres-connections-high**: monitorea conexiones activas al servidor PostgreSQL, evitando saturar el límite configurado
- **alert-postgres-cpu-high** y **alert-postgres-storage-low**: vigilan el uso de CPU y el porcentaje de almacenamiento consumido
- **alert-storage-egress**, **alert-storage-errors** y **alert-storage-latency**: verifican volumen de tráfico, número de errores y latencia en el Data Lake

Cada regla define:

- Condición (métrica, umbral y ventana de tiempo)
- Severidad (Error, Warning o Informational)
- Action Group asociado (por ejemplo, envío de correo al equipo o notificación a Teams)

Con este set de alertas la solución cumple con los requisitos de monitoreo y gobernanza, permitiendo detectar incidentes con anticipación y tomar acciones correctivas.

13. Audit Logs

Para cumplir con la parte de auditoría y gobernanza se habilitó el uso de **Audit Logs en Microsoft Entra ID (Azure AD)**.

Estos logs permiten rastrear:

- Creación y eliminación de service principals usados por Databricks y Container Apps
- Cambios en asignaciones de roles y permisos
- Acciones de administración realizadas en el tenant

Nota: En un escenario corporativo, estos registros se enviarían también a Log Analytics o a un SIEM para retención a largo plazo y análisis

avanzado.

14. Crear el Backup Vault

Para la estrategia de respaldo y recuperación ante desastres (DR) se configuró un **Azure Backup Vault** dedicado al proyecto. En este vault se centraliza la política de backup de:

- Azure Database for PostgreSQL flexible server
- (Opcional) Máquinas virtuales o recursos adicionales si el entorno creciera

En la propiedades del vault se puede apreciar:

Aspectos relevantes:

- **Redundancia:** Geo-redundant (GRS) para disponer de copias en una región secundaria
- **Soft delete habilitado** para proteger contra eliminaciones accidentales
- **Cross Region Restore habilitado**, lo que permite restaurar backups en otra región en caso de caída total de la principal

15 . Crear la Red

La conectividad interna del entorno se organiza alrededor de la VNet **vnet-bbva-dashboard**.

Características principales:

- **Address space:** 10.0.0.0/16
- **Región:** East US
- **Integración con:**
    - Azure Firewall para inspección y filtrado de tráfico saliente
    - Private Endpoints de Storage, PostgreSQL, ACR y Key Vault
    - Subnets específicas por servicio (según se detalla a continuación)

Esta VNet también se enlaza con otras capacidades de red que se crean más adelante (Private DNS Zones, Private Endpoints, etc.), de forma que los servicios PaaS se consumen por rutas privadas.

Dentro de vnet-bbva-dashboard se definieron subredes dedicadas por tipo de servicio:

Subnets más relevantes:

- **AzureFirewallSubnet** – Reservada para el servicio de Azure Firewall
- **snet-postgresql** – Aloja el Private Endpoint hacia el servidor PostgreSQL
- **snet-keyvault** – Aloja el Private Endpoint de Key Vault
- **snet-containerapp** y **snet-containerapp-infra** – Dedicadas al entorno de Container Apps y su infraestructura
- **snet-storage** – Donde residen los Private Endpoints de Blob y DFS para el Data Lake
- **snet-databricks-public** y **snet-databricks-private** – Redes usadas por el workspace de Databricks (nodos de cluster y conectividad interna)

El objetivo es **segmentar el tráfico por tipo de servicio**, aplicar reglas de seguridad específicas y evitar que recursos con diferentes perfiles de riesgo compartan la misma subnet.

Crear los Network Security Groups (NSG)

Para controlar el tráfico a nivel de capa 4 se crearon Network Security Groups por tipo de servicio. **Ejemplos de reglas:**

- **En nsg-postgresql:**
    - Permitir tráfico entrante desde snet-containerapp y snet-databricks-private al puerto 5432
    - Denegar tráfico desde Internet
- **En nsg-storage:**
    - Permitir acceso desde subnets de Databricks y Container Apps a los endpoints de Storage
    - Restringir otros orígenes
- **En nsg-databricks:**
    - Controlar el tráfico saliente de los clusters hacia servicios internos y externos

Con esto se consigue un **modelo zero-trust** donde solo se permiten las comunicaciones necesarias entre componentes.

Crear las Private DNS Zones

Para que los servicios PaaS sean accesibles por nombres privados se crearon varias Private DNS Zones:

Estas zonas se agregaron a la VNet vnet-bbva-

dashboard:

De esta forma:

- Cuando el backend o Databricks resuelven, por ejemplo, stbbvadatalake.dfs.core.windows.net, obtienen una IP privada en la VNet
- Todo el tráfico hacia ACR, Storage, Key Vault y PostgreSQL fluye por la red privada de Azure, sin exponer endpoints públicos

Crear los Private Endpoints

Finalmente, se definieron Private Endpoints para los servicios críticos, mapeándolos a las subnets correspondientes.

Private Endpoints configurados:

- **pe-acr** → Azure Container Registry (acrbbvadashboard) en snet-storage o subnet específica
- **pe-keyvault** → Key Vault del proyecto en snet-keyvault
- **pe-postgresql** → Servidor PostgreSQL flexible pg-bbva-dashboard en snet-postgresql
- **pe-storage-blob** y **pe-storage-dfs** → Cuenta de Storage stbbvadatalake (Blob y Data Lake Gen2) en snet-storage

Gracias a estos endpoints:

- Container Apps, Databricks y otros servicios acceden a los recursos PaaS sin salir a Internet
- Se reduce la superficie de ataque al deshabilitar (o minimizar) el acceso público
- Se facilita el cumplimiento de políticas de seguridad y gobernanza de red

Network Interfaces

Las **Network Interfaces (NIC)** son los recursos que vinculan la red virtual con los servicios privados como Storage, PostgreSQL, Key Vault y el propio Azure Container Apps. Cada private endpoint crea de manera automática una NIC dentro del grupo de recursos de red, lo que permite controlar en detalle el tráfico que entra y sale de cada servicio.

En la figura se puede apreciar que se han creado cinco interfaces de red, cada una asociada a

un private endpoint específico:

- **pe-acr-nic** – interfaz asociada al private endpoint del Azure Container Registry
- **pe-keyvault-nic** – interfaz asociada al private endpoint del Key Vault
- **pe-postgresql-nic** – interfaz asociada al private endpoint de la base de datos PostgreSQL flexible
- **pe-storage-blob-nic** y **pe-storage-dfs-nic** – interfaces asociadas a los private endpoints de Blob Storage y Data Lake (DFS)

Tener estas NIC dedicadas permite:

- Aislar el tráfico de cada servicio dentro de la VNet
- Aplicar reglas de seguridad por subnet y por Network Security Group (NSG)
- Auditar de forma más precisa los accesos a nivel de red

Creación del Firewall y tabla de ruteo

Para centralizar el control del tráfico saliente y proteger el entorno frente a accesos no deseados desde Internet, se implementa un **Azure Firewall** dedicado, ubicado en el mismo grupo de recursos de red.

En la captura se observan los recursos principales relacionados con el firewall:

- **fw-bbva-dashboard** – instancia de Azure Firewall que inspecciona el tráfico
- **pip-firewall** – IP pública asociada al firewall, necesaria para el tráfico de salida controlado
- **policy-bbva-firewall** – Firewall Policy donde se definen las reglas de:
    - Acceso a servicios de Azure (Storage, PostgreSQL, ACR, etc.)
    - Restricciones por FQDN o rangos de IP
- **rt-firewall** – tabla de ruteo que fuerza el tráfico de determinadas subnets (por ejemplo, Databricks, Container Apps, Storage) a pasar por el firewall

De esta manera, cualquier salida hacia Internet o hacia servicios PaaS expuestos por IP pública se canaliza a través del firewall, donde se pueden aplicar reglas de filtrado, logging y futuras políticas de inspección más avanzadas.

Visualización general de la topología de red

Para validar la configuración y evidenciar la conectividad entre los distintos componentes, se utiliza la funcionalidad de **Resource Visualizer** sobre la red virtual principal. Esta vista permite mostrar de forma gráfica la relación entre:

- La VNet **vnet-bbva-dashboard**
- Las subnets especializadas (storage, key vault, PostgreSQL, container app, Databricks)
- Los Network Security Groups (NSG) que protegen cada subnet
- Los Private Endpoints que conectan los servicios PaaS a la VNet
- El firewall y su tabla de ruteo

En el diagrama se observa cómo:

- La VNet actúa como **nodo central** desde el cual se conectan los NSG, el firewall y los private endpoints
- El firewall se posiciona como **punto de salida controlada**, recibiendo tráfico desde las subnets de aplicación y redirigiéndolo hacia Internet o servicios externos
- Los private endpoints **garantizan** que PostgreSQL, Storage, Key Vault y el ACR sean consumidos de forma privada, sin exponer endpoints públicos

Esta visualización se incluye para que se tenga una comprensión rápida de la topología de red, apoyando la explicación previa de subnets, NSG, Private DNS Zones, Private Endpoints, Network Interfaces y Firewall.

Funcionamiento Detallado de la Pantalla "SUBIR DATOS"

Esta es la pantalla de inicio del dashboard. Su único propósito es permitirte cargar los datos que necesitas para que el sistema funcione. Es como la puerta de entrada al mundo de los reportes y análisis.

1. La Barra de Navegación Superior

En la parte superior, tienes tres botones que te permiten moverte por el dashboard:

- SUBIR DATOS (Botón Azul): Este es el botón activo en esta pantalla. Al hacer clic aquí, llegas a esta misma vista, donde puedes cargar nuevos archivos. Es el punto de partida.
- VER DASHBOARD GLOBAL: Este botón está desactivado (apagado) cuando no hay datos cargados. Una vez que subas un archivo y el sistema lo procese, podrás hacer clic aquí para ver una vista general de todos los servicios, con gráficos y tablas resumen.
- VER DASHBOARD POR SERVICIO: Este botón también está desactivado inicialmente. Cuando tengas datos cargados, te llevará a una vista más profunda, donde podrás analizar el rendimiento de un servicio específico en detalle.

2. El Formulario de Carga: Tu Área de Trabajo Principal

Este es el corazón de la pantalla. Aquí es donde tú, como usuario, interactúas con el sistema para subir la información.

a) Campo: DESTINO DEL DATO

- Qué es: Un menú desplegable (una cajita con una flechita abajo).
- Qué hace: Te permite decirle al sistema a qué nivel de madurez pertenecen los datos que vas a subir. Solo tienes dos opciones:
    - Practitioner: Si tu archivo CSV contiene métricas sobre este nivel.
    - Continuous Integration: Si tu archivo CSV contiene métricas sobre este otro nivel.
- Por qué es importante: El sistema necesita saber esto porque los archivos de estos dos niveles tienen estructuras diferentes. Al seleccionar uno, el sistema prepara todo su interior para recibir y procesar ese tipo de datos específicamente. Es como elegir el idioma en una máquina antes de usarla.

b) Campo: SELECCIONAR ARCHIVO CSV

- Qué es: Un botón que dice NINGÚN ARCHIVO SELECCIONADO.
- Qué hace: Cuando haces clic en él, se abre una ventana de tu computadora (como el Explorador de Archivos de Windows o Finder de Mac) para que busques y selecciones el archivo .csv que descargaste desde el Marco Playbook.
- Qué pasa después: Una vez que eliges un archivo, el texto en el botón cambiará para mostrar el nombre del archivo que seleccionaste (por ejemplo, datos_practitioner_marzo.csv). Si cambias de opinión, puedes volver a hacer clic para seleccionar otro.

c) Botón: SUBIR ARCHIVO

- Qué es: El botón azul grande y brillante en el centro.
- Qué hace: Este es el botón mágico. Cuando haces clic en él, inicias el proceso de carga.
- Qué pasa cuando lo presionas:

1. Validación: El sistema primero revisa que el archivo que seleccionaste tenga el formato correcto (.csv) y que las columnas dentro del archivo coincidan con lo que espera para el nivel que elegiste (Practitioner o Continuous Integration). Si algo no coincide, te mostrará un mensaje de error explicando qué salió mal.
2. Carga y Procesamiento: Si todo está bien, el sistema toma tu archivo y lo envía a su "laboratorio interno" (el backend). Allí, el archivo será limpiado, organizado y transformado para que pueda ser usado en los gráficos.
3. Feedback al Usuario: Mientras el sistema trabaja (lo cual puede tomar unos segundos), la pantalla se actualizará para mostrarte el progreso. Verás mensajes como "Limpiando...", "Subiendo...", "Procesando..." hasta que finalmente aparezca "¡Carga completada!".
4. Resultado Final: Una vez terminado, los otros dos botones de la barra superior (VER DASHBOARD GLOBAL y VER DASHBOARD POR SERVICIO) se activarán, y podrás navegar a las vistas de análisis.

📊 **Funcionamiento Detallado de la Pantalla "VER DASHBOARD GLOBAL"**

1. La Barra de Navegación Superior

Al igual que en la pantalla anterior, aquí tienes los tres botones principales:

- SUBIR DATOS (Botón Inactivo): Este botón está apagado porque estás en la vista de análisis. Si necesitas cargar nuevos datos, puedes hacer clic aquí para volver a la pantalla de carga.
- VER DASHBOARD GLOBAL (Botón Activo): Este es el botón que te trajo a esta pantalla. Está resaltado en azul brillante, indicando que estás viendo la vista global.
- VER DASHBOARD POR SERVICIO (Botón Inactivo): Este botón te llevará a la vista de análisis profundo de un servicio específico. Lo usarás cuando quieras profundizar en un caso concreto.

2. Selección del Nivel de Madurez

Justo debajo del título de la pantalla, encontrarás dos botones que te permiten cambiar entre los dos niveles de certificación:

- PRACTITIONER: Al hacer clic en este botón, la pantalla se actualiza para mostrar todos los datos, gráficos y tablas relacionados con el nivel Practitioner. El botón se pondrá en azul brillante para indicar que está seleccionado.
- CONTINUOUS INTEGRATION: Al hacer clic en este botón, la pantalla cambia para mostrar los datos del nivel Continuous Integration. El botón se resaltará y el contenido de la pantalla se actualizará completamente.

3. Tarjetas Resumen (Summary Cards)

En la parte superior de la pantalla, hay tres tarjetas que te dan una visión rápida de los datos clave:

- Total de Registros: Muestra el número total de registros (filas) que se han cargado para el nivel de madurez seleccionado. Por ejemplo, para Practitioner, muestra 3095.
- Geografías Analizadas: Indica cuántas regiones o países diferentes están siendo evaluadas. En el ejemplo, son 14.
- Adopción Total Promedio: Este es el KPI más importante. Muestra el promedio ponderado de adopción de todas las métricas para el nivel seleccionado. Para Practitioner, es 81.92%, y para CI, es 79.53%.

4. Filtros de Análisis

Debajo de las tarjetas, encontrarás dos filtros que te permiten ajustar la vista de los datos:

- Filtro FECHA: Un menú desplegable que te permite seleccionar un mes específico o elegir Todas para ver todos los datos disponibles. Esto es útil si quieres analizar el rendimiento de un periodo concreto.
- Filtro GEOGRAFÍA: Otro menú desplegable que te permite seleccionar una región específica (por ejemplo, ARGENTINA, PERU, MEXICO) o elegir Todas para ver la vista global.

5. Gráfico: "Niveles de Certificación por % de Adopción"

Este es un gráfico de tarta que te muestra cómo se distribuyen todos los servicios según su nivel de certificación.

- Qué representa: El gráfico divide el círculo en cuatro sectores, cada uno representando un nivel de certificación:
    - LEVEL 3: Servicios con 90% o más de adopción. (Color Azul Claro)
    - LEVEL 2: Servicios con entre 80% y 89% de adopción. (Color Azul Medio)
    - LEVEL 1: Servicios con entre 70% y 79% de adopción. (Color Azul Oscuro)
    - No Certificado: Servicios con menos del 70% de adopción. (Color Gris)

- Qué ves en el centro: El número total de servicios que cumplen con los criterios de filtro (por ejemplo, 3021).
- Leyenda: A la derecha, una leyenda explica qué color corresponde a cada nivel y muestra el porcentaje y el número de servicios en cada categoría.

6. Gráfico: "Niveles de Certificación por Geografía"

Este es un gráfico de barras apiladas que te permite comparar el rendimiento entre diferentes regiones.

- Qué representa: Cada barra horizontal corresponde a una geografía (por ejemplo, ARGENTINA, PERU, ESPAÑA). Dentro de cada barra, los colores de los segmentos muestran la cantidad de servicios en cada nivel de certificación (LEVEL 3, LEVEL 2, LEVEL 1, No Certificado).
- Qué ves al final de cada barra: El número total de servicios analizados en esa geografía.
- Cómo leerlo: Puedes ver fácilmente qué regiones tienen una mayor proporción de servicios en LEVEL 3 (azul claro) y cuáles tienen muchos servicios sin certificar (gris).

7. Tabla: "Cálculo de los KPI's"

Esta es la tabla detallada que muestra los datos granulares de cada servicio.

- Qué contiene: La tabla tiene columnas para:
    - FECHA: El mes al que corresponden los datos.
    - GEOGRAFÍA: La región del servicio.
    - SERVICE1 NAME: El nombre del servicio.
    - RFO OK PCT, DEP PCT, ADOPCION SN2 PCT, CALIDAD FEATURES PCT, SEGURIDAD PCT: Los porcentajes individuales de cada KPI.
    - ADOPCION TOTAL PCT: El porcentaje de adopción total ponderado, que es el resultado final.
- Paginación: Debajo de la tabla, hay botones Anterior y Siguiente para navegar por las páginas. También muestra cuántos registros se están mostrando (por ejemplo, Mostrando 1 a 10 de 3095).

📈 **Funcionamiento Detallado de la Pantalla "VER DASHBOARD POR SERVICIO"**

1. La Barra de Navegación Superior

Al igual que en las otras pantallas, aquí tienes los tres botones principales:

- SUBIR DATOS (Botón Inactivo): Este botón te permitirá volver a la pantalla de carga si necesitas actualizar los datos.
- VER DASHBOARD GLOBAL (Botón Inactivo): Te llevará de vuelta a la vista macro, donde puedes ver todos los servicios juntos.
- VER DASHBOARD POR SERVICIO (Botón Activo): Este es el botón que te trajo a esta pantalla. Está resaltado en azul brillante, indicando que estás viendo la vista de análisis profundo por servicio.

2. Filtros "Service Owner": Tu Panel de Control Personalizado

En esta pantalla, los filtros son el corazón del sistema. Son los controles que te permiten elegir exactamente qué servicio quieres analizar. El diseño es inteligente porque los filtros se actualizan en cascada, lo que significa que la selección de uno afecta las opciones disponibles en el siguiente.

a) NIVEL DE MADUREZ

- Qué es: Un menú desplegable que te permite seleccionar el nivel de certificación que deseas analizar: Practitioner o Continuous Integration.
- Qué hace: Al hacer clic en este filtro, el sistema consulta la base de datos y te muestra solo las geografías que tienen datos para ese nivel de madurez. Esto evita que veas opciones inválidas.
- Por qué es importante: Es el primer paso para enfocar tu análisis. Si quieres ver cómo va un servicio en Practitioner, debes seleccionar este nivel primero.

b) GEOGRAFÍA

- Qué es: Otro menú desplegable que te permite seleccionar una región específica (por ejemplo, ARGENTINA, PERU, URUGUAY).
- Qué hace: Una vez que has seleccionado un nivel de madurez, este filtro se actualiza automáticamente para mostrarte solo las geografías que tienen datos para ese nivel.

Por ejemplo, si seleccionas Practitioner, no podrás elegir una geografía que solo tenga datos de CI.

- Por qué es importante: Te permite acotar tu análisis a una región específica. Si eres el Service Owner de Uruguay, puedes filtrar solo por esa geografía.

c) SERVICIO N1

- Qué es: Un tercer menú desplegable que te permite seleccionar un servicio específico dentro de la geografía que elegiste.
- Qué hace: Al seleccionar una geografía, este filtro se actualiza para mostrarte solo los servicios que existen en esa región para el nivel de madurez seleccionado. Por ejemplo, si eliges URUGUAY y Practitioner, te mostrará servicios como ENGINEERING & DATA o RETAIL CLIENT SOLUTIONS.
- Por qué es importante: Este es el filtro final que te lleva al análisis de un servicio concreto. Es aquí donde empieza la magia del dashboard.

3. Área de Visualización: El Análisis Profundo del Servicio

Una vez que has seleccionado un nivel de madurez, una geografía y un servicio específico, el dashboard se actualiza para mostrarte una serie de gráficos y métricas detalladas. Aquí es donde puedes ver el rendimiento de ese servicio en profundidad.

a) Gráfico de Evolución de Adopción Total Este es un gráfico de líneas que te muestra cómo ha cambiado el porcentaje de adopción total del servicio a lo largo del tiempo.

*Qué representa: Cada punto en la línea corresponde al porcentaje de adopción total del servicio para un mes específico. Por ejemplo, puedes ver cómo fue en marzo, abril, mayo, etc.

- Modos de Visualización:
    - MENSUAL: Muestra los datos puntuales de cada mes. Es útil para ver los picos y valles.
    - ACUMULADO: Muestra un promedio progresivo. Es útil para ver la tendencia general a lo largo del tiempo.
- Cómo funciona: El sistema toma los datos de los últimos 12 meses del servicio seleccionado y los grafica. Puedes ver claramente si el servicio está mejorando, empeorando o manteniéndose estable.

b) Gauge Chart (Velocímetro) de Adopción Total Este es un medidor tipo velocímetro que te muestra el valor más reciente de la adopción total del servicio.

- Qué representa: El número grande en el centro (por ejemplo, 49.06%) es el porcentaje de adopción total del servicio en el último mes disponible.
- Colores: El velocímetro tiene zonas de color:
    - Verde: Indica un buen estado (>= 90%).
    - Amarillo: Indica un estado medio (70-89%).
    - Rojo: Indica un estado crítico (< 70%).
- Texto: Debajo del número, te dice en qué categoría de certificación se encuentra el servicio (por ejemplo, NO CERTIFICADO).

c) Selector de KPI's Este es un conjunto de botones que te permiten cambiar el gráfico de evolución para ver otros KPIs individuales del servicio.

- Qué hace: Al hacer clic en un botón (por ejemplo, RFO OK, DEP, CALIDAD FEATURES), el gráfico de evolución cambia para mostrar la tendencia de ese KPI específico en lugar de la adopción total.
- Por qué es importante: Te permite profundizar en un aspecto concreto del servicio. Si ves que la adopción total está baja, puedes usar este selector para ver si el problema está en la calidad de las features, en las dependencias o en otro KPI.

d) Gráfico Radar de KPIs Este es un gráfico tipo radar que te muestra una vista holística de todos los KPIs del servicio para un mes específico.

- Qué representa: Cada eje del radar corresponde a un KPI diferente (por ejemplo, RFO OK, DEP, ADOPCIÓN SN2, CALIDAD FEATURES, SEGURIDAD). La distancia desde el centro indica el valor del KPI.
- Comparación: Puedes seleccionar un mes pasado (por ejemplo, Julio 2025) para compararlo con el mes actual (Agosto 2025). El gráfico superpondrá ambos períodos, permitiéndote ver fácilmente si el servicio ha mejorado o empeorado en cada KPI.
- Leyenda: A la derecha, una leyenda explica qué color corresponde a cada período.