**Portafolio 9: Tecnología Cloud para ingenieros de datos.**

**M9 AE 1: Mapa comparativo de modelos cloud**

Completar cuadro comparativo entre cuatro servicios en la nube (IaaS, PaaS, SaaS y FaaS).

**1. Definiciones breves**

**IaaS (Infrastructure as a Service)**: Proporciona infraestructura básica como servidores, redes y almacenamiento en la nube. Un tercero le presta los servicios de infraestructura, como el almacenamiento y la virtualización.

**PaaS (Platform as a Service)**: Ofrece una plataforma completa para desarrollar, ejecutar y administrar aplicaciones sin gestionar la infraestructura. El proveedor aloja el hardware y el software en su propia infraestructura y ofrece la plataforma al usuario como una solución integrada.

**SaaS (Software as a Service)**: Permite a los usuarios acceder a software y aplicaciones a través de Internet sin instalarlas localmente. Ofrece una aplicación integral que gestiona el proveedor, a través de un explorador web.

**FaaS (Function as a Service)**: Permite ejecutar funciones individuales en la nube en respuesta a eventos, sin preocuparse por servidores o infraestructura. Es una plataforma que permite ejecutar código sin necesidad de aprovisionar, administrar ni escalar servidores.

**2. Identificar 1 Proveedor y 2 ejemplos reales**

| **Modelo** | **Proveedor** | **Ejemplos reales** |
| --- | --- | --- |
| IaaS | Amazon Web Services (AWS)  Microsoft Azure  Google Cloud | Amazon EC2,  Google Compute  Engine |
| PaaS | Google Cloud  Microsoft Azure  Salesforce  IBM | Google App Engine  Azure App Service  Heroku  Red Hat OpenShift. |
| SaaS | Microsoft  Google  Fundación Dropbox | Microsoft 365  Google Workspace  Dropbox |
| FaaS | Amazon Web Services (AWS)  Microsoft Azure  Google Cloud | AWS Lambda  Azure Functions  Google Cloud Functions |

**3. Indicar quien gestiona qué Enumerar 2 ventajas clave y 1 desafío**

| **Modelo** | **Gestión (Cliente vs Proveedor)** | **2 ventajas** | **1 desafío** |
| --- | --- | --- | --- |
| IaaS | Cliente: gestiona: SO y aplicaciones y datos  Proveedor: Red, Hardware, Almacenamiento | Alta flexibilidad, escalabilidad. | Requiere conocimientos técnicos |
| PaaS | Cliente: Código y datos Proveedor: gestiona infra y entorno | Acelera el desarrollo, menos mantenimiento | Dependencia del proveedor |
| SaaS | Cliente: sólo usa el software.  Todo gestionado por el proveedor | Fácil de usar, acceso desde cualquier lugar | Menos personalización, se usa como está implementado |
| FaaS | Cliente: Carga códigos y gestiona eventos  Proveedor: gestiona todo excepto lógica | Escalabilidad automática, múltiples códigos | Difícil depuración y monitoreo, pues no hay manejo del servidor, dependencia del proveedor |

**Recomendación de solución cloud**

**Una empresa e-commerce quiere escalar su aplicación rápidamente, Manejar eventos masivos, reducir costos de infraestructura**. **Proponer una combinación de modelos de servicio de la nube y tecnologías adecuadas**.

FaaS: En este caso el cliente crea y carga los códigos permite el pago por uso y es escalable de acuerdo con los requerimientos.

**Preguntas a la clase:**

**¿Por qué crees que tantas empresas están migrando sus servicios a la nube?**

Porque evitan el pago de implementación propia, delegan la administración al proveedor y les permite el uso desde cualquier lugar.

**¿Qué implicancias tiene delegar el control de la infraestructura a un proveedor cloud?**

Dentro de las ventajas está el ahorro del costo de implementar y administrar infraestructura propia. Desventajas, dependencia del proveedor, seguridad de los datos ya que se entregan al proveedor y se suben a la nube, ante posibles problemas del servicio el cliente queda sin poder funcionar.

**¿Cómo se equilibra la necesidad de escalabilidad con el riesgo del vendor lock-in?**

Evitando la dependencia de un solo proveedor, buscando diversificar en diferentes servicios. Otra opción es desarrollar soluciones propias que limiten el uso de los servicios en la nube a tareas que puedan ser migradas o no limiten el funcionamiento del cliente.

**M9 AE2: Tecnologías de Almacenamiento en la nube:**

**Actividad: Selección del servicio adecuado según caso de uso:**

**1. Identificar los tipos de datos que se manejan en el caso. Algunos ejemplos mencionados son datos estructurados, archivos colaborativos, datos históricos y copias de seguridad (backups).**

**2. Asociar cada tipo de dato con la tecnología más adecuada.**

**3. Justificar la elección de tecnología basándose en factores como costos, accesibilidad, rendimiento y resiliencia.**

**Escenario 1: Startup de e-commerce**

Un startup de moda digital está lanzando su primera tienda en línea. El equipo fundador es pequeño y no tienen especialistas en infraestructura. Al inicio esperan pocas visitas, pero saben que durante campañas en redes sociales la cantidad de clientes puede dispararse de manera inesperada. Además, necesitan cuidar al máximo el presupuesto.

**Tipos de datos:** Se espera datos estructurados, en bases como clientes, productos y ventas. Además, las imágenes de productos serían no estructurados

**Tecnología:** Amazon S3 (Simple Storage Service) con Dynamo DB.

**Justificación:** Permite crecer de acuerdo con la necesidad del negocio, soporta altas demandas por períodos, es de fácil implementación y de bajo costo

**Escenario 2: Banco tradicional**

Un banco con operaciones en todo el país está migrando parte de sus sistemas de atención en línea hacia la nube. La plataforma maneja millones de transacciones diarias y debe cumplir con estrictas normas de seguridad y auditoría. El área de TI cuenta con un equipo grande y especializado que administra infraestructura de forma constante y está acostumbrado a tener control detallado de los sistemas.

**Tipos de datos:** Se espera datos estructurados y la mantención de archivos históricos.

**Tecnología:** Amazon EC2 (Elastic Compute Cloud), con Amazon Aurora

**Justificación:** Entrega alto nivel de seguridad, y control del sistema. Requiere de un equipo con mayor conocimiento y su costo es más alto debido al requerimiento de seguridad del caso.

**Escenario 3: ONG de salud**

Una organización sin fines de lucro que ofrece atenciones médicas rurales quiere desplegar una aplicación en la nube para registrar datos de pacientes y generar reportes. El uso del sistema no es constante: algunos días casi no se ingresa información y en otras ocasiones, durante campañas masivas, la demanda se multiplica. El equipo técnico es muy reducido y dependen de donaciones, por lo que buscan soluciones simples y económicas.

**Tipos de datos:** estructurados pacientes, atenciones y no estructurados (exámenes e imágenes)

**Tecnología:** Se utilizará almacenamiento en S3 para el acceso inicial de la información y después de un tiempo, por ejemplo 30 días se almacenará como archivos históricos de acceso poco frecuente en Amazon Glacier.

**Justificación:** se tendrá acceso directo a la información por un tiempo determinado y posteriormente se almacenará como archivos históricos en Glacier para su acceso poco frecuente.

**Contexto: Diseña una estrategia de almacenamiento escalonada para una empresa que maneja grandes volúmenes de datos (por ejemplo, videos educativos, registros de usuarios, logs de sistema y backup mensuales).**

**Datos:**

**Videos educativos** no estructurados, de uso frecuente. Se almacena en S3, por ser archivos no estructurados y de acceso frecuente.

**Registros de usuarios** datos estructurados de uso frecuente. Se almacenan en EBS con alto rendimiento y acceso frecuente

**Logs de sistema** datos estructurados de uso frecuente. Se almacenan en EBS con alto rendimiento y acceso frecuente.

**Backup,** hibrido, pues considera los videos, logs y datos de usuarios para consultas históricas con acceso ocasional. Usa Glacier.

El flujo de datos se puede representar de la siguiente manera:

1. Los **videos educativos** se cargan a **S3** para su visualización inmediata.
2. Los **registros de usuarios** se almacenan en una base de datos de **DynamoDB** en EBS para acceso constante de la aplicación.
3. Los **logs del sistema** se escriben inicialmente en S3 Standard y, después de 30 días, se usa una regla del ciclo de vida de S3 para moverlos automáticamente a **S3 IA**.
4. Los **backups mensuales** se crean y se mueven directamente a **Glacier Deep Archive** para su archivo a largo plazo.

**Clase M9 AE3 Servicios de cómputo en la nube:**

|  |  |  |  |
| --- | --- | --- | --- |
| Criterio | Máquinas Virtuales (VMs) | Contenedores | Cómputo sin Servidor (Serverless) |
| Control | Alto. El usuario gestiona el sistema operativo, dependencias y aplicaciones. | Medio. Se gestiona la aplicación y sus dependencias, pero el sistema operativo subyacente es gestionado por el proveedor. | Bajo. El usuario solo se preocupa por el código. El proveedor gestiona todo el hardware y el entorno de ejecución. |
| Escalabilidad | Manual/Lenta. Se requiere aprovisionar y configurar nuevas instancias, lo que toma tiempo. | Rápida. La orquestación (como Kubernetes) permite escalar rápidamente agregando más réplicas de contenedores. | Automática/Instantánea. El proveedor escala automáticamente en función de la demanda. |
| Costos | Fijos/Por hora. Se paga por la instancia encendida, incluso si no se usa. | Eficiente. Se paga solo por los recursos que utilizan los contenedores. Mayor densidad de aplicaciones por servidor. | Por evento/Uso. Se paga solo por el tiempo de ejecución del código. Extremadamente rentable para cargas de trabajo intermitentes. |
| Facilidad de Implementación | Baja. Requiere configurar el SO, el entorno y la aplicación. | Media. Se debe crear una imagen de contenedor y usar un orquestador, lo que simplifica la gestión de dependencias. | Alta. Simplemente se sube el código y se configura un "trigger" (disparador). |
| Tiempo de Respuesta | Constante. La instancia está siempre disponible. | Constante. Los contenedores están listos para responder rápidamente. | Variable. Puede haber un "cold start" (inicio en frío) en la primera ejecución después de un período de inactividad. |
| Mantenimiento | Alto. Se debe parchar el SO, gestionar actualizaciones de software y dependencias. | Medio. Se mantiene la imagen del contenedor, no el SO. | Bajo/Nulo. El proveedor se encarga de todo el mantenimiento de la infraestructura. |

**M9 AE 4: Diagnóstico y elección del tipo de base de datos cloud**

**Caso 1: Aplicación financiera:** Una institución desarrolla una aplicación para gestionar cuentas, transferencias y pagos en línea. En este escenario, la base de datos debe garantizar transacciones seguras, consistencia estricta y cumplimiento de normas financieras. Además, requiere alta disponibilidad y respaldo automático para evitar pérdidas de información. Por estas características, una base de datos relacional gestionada en la nube como Amazon RDS resulta la opción más adecuada, ya que ofrece soporte a transacciones ACID y mecanismos de seguridad integrados.

**Se recomienda el uso Amazon RDS debido a soporte ACID, seguridad, backup, seguridad y encriptación.**

**Caso 2: Sistema de análisis de marketing:** Una empresa necesita procesar grandes volúmenes de datos provenientes de campañas digitales, redes sociales y plataformas de ventas. El objetivo es realizar consultas complejas y obtener informes en tiempo real o casi real para orientar la toma de decisiones. En este caso, la prioridad es la capacidad de análisis, la escalabilidad y la velocidad en consultas masivas. La mejor alterna va es Amazon Redshift, que permite implementar un data warehouse optimizado para consultas analíticas de alto rendimiento.

**Se recomienda Amazon Redshift, ya que permite la carga de gran cantidad de datos y análisis de los datos en la nube con consultas complejas.**

**Caso 3: Plataforma de e-commerce:** Un negocio de comercio electrónico gestiona catálogos extensos de productos, carritos de compra, usuarios conectados simultáneamente y procesos de recomendación en tiempo real. El sistema debe ser flexible, escalable y tolerante a picos de tráfico como los que ocurren en fechas de alto consumo. Aquí, una base de datos NoSQL como Amazon DynamoDB es la opción más conveniente, ya que ofrece baja latencia, escalabilidad horizontal automática y soporte para estructuras de datos dinámicas que se adaptan mejor a este tipo de aplicaciones.

**Se recomienda Dynamo DB, ya que permite alta concurrencia y velocidad y adaptable a altos picos de tráfico y tiene una estructura flexible y dinámica.**

**Comparativa de servicios gestionados de bases de datos de AWS:**

|  |  |  |  |
| --- | --- | --- | --- |
| **Criterio** | **Amazon RDS** | **Amazon DynamoDB** | **Amazon Redshift** |
| Modelo de Datos | Relacional (SQL). Utiliza tablas, filas y columnas, con relaciones definidas por claves. SQL, Postgres. | NoSQL clave-valor y documento. Almacena datos como pares clave-valor o en formato de documento (JSON), sin un esquema fijo. | Almacén de datos (Data Warehouse). Utiliza un modelo relacional, pero con almacenamiento en columnas optimizado para análisis y consultas complejas. |
| Escalabilidad | Permite el escalado vertical y horizontal. El escalado puede requerir algo de tiempo de inactividad, a menos que se use Amazon Aurora. | Ofrece escalabilidad automática y fluida para manejar cargas de trabajo masivas. Se ajusta automáticamente al tráfico de lectura y escritura. | Se escala tanto en computación como en almacenamiento. El escalado de la capacidad de cómputo se realiza añadiendo nodos al clúster, pero puede requerir intervención manual. |
| Velocidad | Está optimizado para transacciones y consultas complejas en datos estructurados. El rendimiento varía según el tipo de instancia y el motor de base de datos. | Proporciona un rendimiento de latencia de milisegundos constante, incluso a gran escala. Ideal para cargas de trabajo con un alto volumen de lecturas y escrituras. | Diseñado para el análisis de grandes conjuntos de datos. Es extremadamente rápido para ejecutar consultas analíticas complejas que involucran grandes volúmenes de datos. |
| Costo | El costo depende del tipo de instancia (tamaño y motor), el almacenamiento, las copias de seguridad y la transferencia de datos. Se puede pagar por uso o con instancias reservadas para un menor costo. | El modelo de precios es flexible, basado en la capacidad de lectura/escritura (aprovisionada o bajo demanda) y el almacenamiento consumido. | Se basa en el tipo de nodo (instancia) y el tiempo que el clúster está en funcionamiento. Es el más costoso de los tres, pero ofrece un excelente rendimiento de precio para cargas de trabajo analíticas. |
| Flexibilidad | Alta flexibilidad para usar diferentes motores (MySQL, PostgreSQL, Oracle, etc.), pero requiere un esquema de datos rígido. | Muy flexible en cuanto a la estructura de los datos (sin esquema), lo que lo hace ideal para datos semiestructurados o no estructurados. | Menos flexible para transacciones. Su modelo de almacenamiento en columnas no es adecuado para la escritura frecuente de datos. |
| Casos de Uso | Aplicaciones web y móviles, sistemas de gestión de contenido (CMS), sistemas de planificación de recursos (ERP) y cualquier aplicación que requiera un modelo de datos relacional estricto. | Aplicaciones con picos de tráfico impredecibles, tablas de clasificación de juegos, plataformas de streaming de medios, seguimiento de eventos, y aplicaciones que necesitan un rendimiento constante a cualquier escala. | Análisis de grandes datos, informes de inteligencia de negocios (BI), cuadros de mando de rendimiento, y cualquier carga de trabajo que requiera ejecutar consultas analíticas complejas sobre petabytes de datos. |

**Recomendación y conclusión**

La elección del servicio de base de datos de AWS debe estar alineada con los requisitos de la aplicación y la arquitectura de la organización.

**Amazon RDS** se beneficia de las relaciones de datos y un esquema estricto. Es la opción estándar para la mayoría de las aplicaciones transaccionales y de propósito general.

**Amazon DynamoDB** cuando la velocidad, la escalabilidad masiva y la flexibilidad del esquema son cruciales. Es ideal para aplicaciones modernas que manejan grandes volúmenes de datos con picos de tráfico impredecibles, como videojuegos o IoT.

**Amazon Redshift** análisis de datos a gran escala y la inteligencia de negocios. Su arquitectura está optimizada para consultas complejas sobre petabytes de datos, lo que lo convierte en la mejor opción para data warehousing.

En resumen, no hay un servicio "mejor" que los demás, sino que cada uno está diseñado para resolver un problema específico. La clave es entender las características de cada uno para tomar la decisión correcta y alinear la base de datos con los objetivos del proyecto.

**M9 AE 5: Movimientos de Datos en la Nube con ETL y AWS Glue.**

**Diseñar un flujo ETL para resolver un caso real:**

**Fuentes de Datos:**

**Tiendas Físicas:** archivos CSV periódicos.

**Sitio de comercio electrónico:** base de datos en la nube con actualizaciones casi en tiempo real.

**Marketplace:** Hojas de cálculo en reportes semanales.

**Transformación de los Datos:** Se debe normalizar los datos, creando tablas de homologación. Se debe realizar limpieza y estandarización, eliminando duplicados, imputando datos, consolidar los datos y almacenarlos en una base datos para su análisis.

**Destino de los datos:** Se recomienda el uso de Amazon S3, debido a que los datos completos se obtienen semanalmente, además considera menor costo (respecto a Redshift).

**Representación del Flujo y Automatización**

* **Flujo:**
* **Automatización:**

Se usa **Triggers** en AWS Glue para programar la ejecución de los trabajos. Se puede usar uno **basado en el tiempo** (ej. Cada semana con todos los datos) y uno **basado en eventos** (ej. ejecutar el trabajo cada vez que un nuevo archivo CSV se cargue en una carpeta de S3, para la información que llega a diario).

**Diagnóstico de un proceso ETL con fallas**

**Caso 1: Job de integración de ventas**

**Problemas Detectados:**

* **Cifras incompletas:** hay días en que faltan registros y en otros aparecen duplicados. **ERROR EN LA INGESTA.**

**Solución: revisar y mejorar la ingesta y transformación de los datos. Determinando el porqué de la falta de datos, podría deberse a errores de red, o errores en la ingesta. Se debe revisar la transformación, evitando la duplicación de datos.**

* Los logs de Glue muestran mensajes de error relacionados con “**access denied**” en ciertos objetos. **ERROR DE PERMISOS.**

**Solución: Se deben verificar los accesos a los datos de acuerdo con la gobernanza de datos implementada.**

* Advertencias de “**mismatch”** en los tipos de datos de una de las columnas más usadas en las consultas. **ERROR DE TIPO DE DATOS.**

**Solución: Puede ser un error del CRAWLER o del proceso de transformación. Se debe verificar la Transformación de los datos.**

* En el panel de CloudWatch se observa que **el job tarda más de lo habitual** en completarse, y a veces se detiene antes de llegar a la carga final en Redshi. **ERROR DE RED o Ineficiencias en las consultas.**

**Solución: Se debe monitorear posibles cuellos de botella, o problemas de red. Además, se debe revisar las consultas pues podrían estar siendo ineficientes.**

**Preguntas a la clase**

**¿Por qué es clave automatizar procesos ETL en entornos cloud?**

Permiten automatizar procesos, la disminución de la intervención de los datos que podrían generar errores. Permite la escalabilidad en caso de ser requerido y permite mejorar eficiencia en tiempos y recursos, lo que genera ahorros.

**¿Qué ventajas tiene AWS Glue frente a herramientas ETL tradicionales?**

No requiere instalar recursos en forma local ni mantenimiento. Tiene un entorno gráfico de fácil comprensión. Permite manejo de Metadatos y es flexible.

**¿Qué riesgos puede traer un mal diseño de permisos en un pipeline de datos?**

Error en los datos, lo que dificultaría la toma de decisiones. Además, debe haber seguridad en el acceso a los datos para evitar pérdidas, fugas o deficiencia en el cumplimiento de las normas. Todo ello genera mayores tiempos, uso de recursos y costos adicionales.

**M9 AE 6 Diseño de un job de procesamiento distribuido:**

**Contexto: Las organizaciones suelen necesitar procesar grandes volúmenes de datos provenientes de múltiples fuentes. Amazon EMR permite escalar ese procesamiento con frameworks como Spark o Hadoop. Deberán decidir qué datos se procesan, con qué tecnología, y cómo organizarían el clúster.**

**Paso a paso:**

**Caso: Reportes Financieros, un banco necesita generar reportes mensuales combinando datos de distintas fuentes y consolidar los datos para su análisis**

**Definan qué framework usarían (Spark, Hadoop, Hive) y por qué.**

Se implementa el procesamiento con SPARK, debido a la facilidad de manejar gran cantidad de datos, con información ya disponible en Amazon S3

**Esbocen cómo se estructuraría el clúster: nodo maestro, principales y de tareas.**

Se considera el uso de nodo maestro, nodos principales y nodos de tareas, los cuales son flexibles y se pueden aumentar o disminuir de acuerdo con las necesidades.

**Indiquen cómo y desde dónde se consumirían los datos (ej.: S3, DynamoDB).**

Se extrae información desde Amazon S3 y de sistemas propios de la entidad financiera, los que serán ingestado mediante BATCH

**Elaboren un diagrama simple del flujo de datos y procesamiento.**

Se programa un flujo ETL en EMR que:

* Extrae datos desde múltiples orígenes bases locales y ya cargada en S3.
* Normaliza formatos.
* Calcula indicadores y estadísticas.

Los resultados se exportan en Amazon S3 y se disponibilizan para para reportes visuales.

**Diagnóstico de errores de red en un job EMR real**

**Analicen logs simulados, detecten errores, justifiquen el diagnóstico y propongan ajustes. Redacten una breve recomendación al equipo técnico de la empresa.**

|  |  |  |  |
| --- | --- | --- | --- |
| Errores |  |  |  |
| Mensaje | Detalle | Error | Solución |
| Permiso denegado | Ruta=hdfs:///user/data/input.csv, Usuario=spark | Error de acceso a los datos | Se requiere verificar problemas de accesos, problemas en IAM |
| Fallo en escritura HDFS | Ruta=hdfs:///tmp/output.csv, Motivo=No space left on device | No Hay espacio de almacenamiento o error en archivo temporal de salida | Eliminar archivos temporales, administrar el almacenamiento |
| Timeout en conexiÃ³n | Servicio=YARN, Espera=300s | Problema en servicio YARN | Revisar posibles Problemas de Red, congestión de datos ajustar parámetros de YARN |
| Error en cÃ³digo PySpark | ValueError: Cannot parse 'NaN' as integer | Error de datos nulos o no numéricos donde se esperaba un entero | Mejorar la transformación de los datos |
| Instancia caÃ­da | NodeID=i-0efgh456, Motivo=DiskFullException | Error de espacio en disco en el nodo | Ampliar el tamaño del disco contratado o eliminar información innecesaria en la nube |

|  |  |  |  |
| --- | --- | --- | --- |
| Warning |  |  |  |
| mensaje | detalle | Error | Solución |
| Cuello de botella en shuffle | Stage=4, Datos=10GB, Particiones=2 | cuello de botella, demasiados datos en solo 2 particiones | incrementar la cantidad de particiones |
| Memoria cercana al lÃ­mite | Executor=2, Memoria=7.8GB/8GB | se está completando la memoria | Aumentar la memoria y mejorar autoscaling en EMR |
| Tarea lenta detectada | Stage=2, Task=5, DuraciÃ³n=120s, Datos procesados=2MB | las tareas demoran más de lo habitual | Revisar las tareas lentas, revisar los datos que pueden estar desbalanceados |

**Recomendaciones:**

Se recomienda revisar los procesos de transformación de los datos, y aumentar los servicios contratados, con aumento de memoria y disco. Además, se debe mejorar los procesos de limpieza de archivos temporales. Por último, se debe verificar los accesos a los servicios, para evitar problemas de autentificación y posibles fugas de información.

**Clase M9 AE7 Procesamiento sin servidor (aws lambda)**

**Automatizar un flujo de datos con AWS Lambda**

**Caso:** Se procesará una base de datos de clientes para agruparlos por región y poder ofrecer servicios personalizados de acuerdo con el tipo de producto y de cada sucursal. Se capturan archivos capturados desde el marketplace, en el cual se identifica a los usuarios las compras o consultas realizadas en la página web.

Se realizarán procesos de normalización en el cual se eliminarán columnas innecesarias.

Se requieren permisos de lectura de la base de datos en la página web marketplace, se requieren permisos de escritura para el procesamiento de las funciones LAMBDA en los bucket de S3 y permisos de escritura de la base de datos en el Bucket de S3.

**Código**

Se realiza lectura desde el json de la página web del carro de compra y de los datos de cliente.

Se genera la limpieza de datos (normalización y estandarización) y se consolida sólo la información relevante, como compras, datos del cliente y región.

Se carga la información en S3.

Se disponibiliza la información para análisis y reportes en Amazon QuickSight.

**Diseña una API serverless para gestión de datos:**

**Caso: API serverless para control de inventario.**

Se necesita una API para gestionar el inventario de una tienda, permitiendo agregar, consultar y eliminar productos. Esta solución se implementará en una arquitectura serverless en AWS, lo que permite que sea escalable y rentable.

La API será construida con Amazon API Gateway y AWS Lambda, utilizando Amazon DynamoDB como base de datos NoSQL para almacenar los datos de los productos.

**Endpoints necesarios:**

Crear un producto: POST /productos

Consultar un producto: GET /productos/{id}

Eliminar un producto: DELETE /productos/{id}

Cada endpoint se asocia con una función Lambda distinta:

POST /productos se asocia con una función llamada CrearProductoLambda.

GET /productos/{id} se asocia con una función llamada ConsultarProductoLambda.

DELETE /productos/{id} se asocia con una función llamada EliminarProductoLambda.

**Conexión a DynamoDB:** Cada función Lambda tendrá un rol de **IAM** con los permisos necesarios para realizar operaciones de escritura (PutItem) y lectura (GetItem, DeleteItem) en la tabla de DynamoDB.

**Diagrama del flujo**:

Un usuario hace una solicitud HTTP a la API Gateway.

API Gateway invoca la función Lambda correspondiente al endpoint.

La función Lambda interactúa con la tabla de DynamoDB (crea, lee o elimina un producto).

La función Lambda devuelve una respuesta a API Gateway.

API Gateway envía la respuesta al usuario.

Diagrama

Descripción generada automáticamente

**Ventajas frente a una API tradicional**

Escalabilidad automática: La API serverless escala de forma automática para manejar picos de tráfico, a diferencia de una API tradicional donde es necesario aprovisionar y gestionar servidores.

Modelo de pago por uso: Solo se paga por el tiempo de ejecución del código de las funciones Lambda y el consumo de las solicitudes a API Gateway. No hay costos cuando la API no está siendo utilizada.

Mantenimiento mínimo: AWS se encarga del mantenimiento de la infraestructura subyacente, lo que permite a los desarrolladores centrarse únicamente en la lógica del negocio.

Reducción de costos operativos: Al no tener que gestionar servidores, se minimizan los costos de operación y la complejidad de la infraestructura.