POSTULACIÓN INGENIERO CLOUD

Tabla de contenido

POSTULACIÓN INGENIERO CLOUD 1
¿Cuál es la diferencia entre nube pública, privada e hibrida? 2
Describa tres prácticas de seguridad en la nube 2
¿Qué es la IaC, y cuáles son sus principales beneficios?, mencione 2 herramientas de iaC y sus principales características. 3
Beneficios Principales 3
Herramientas de IaC y sus Principales Características 3
¿Qué métricas considera esenciales para el monitoreo de soluciones en la nube? 4
¿Qué es Docker y cuáles son sus componentes principales? 5
Caso práctico 6
Diseño 8

¿Cuál es la diferencia entre nube pública, privada e hibrida?

La principal diferencia entre las nubes pública, privada e híbrida tiene que ver con el modelo de implementación y el nivel de control (propiedad) que se ejerce sobre la infraestructura y los servicios.

Nube Pública: Los servicios se ofrecen bajo un modelo multi-tenant en infraestructura compartida, con acceso vía Internet y facturación por consumo. Se reduce la carga operativa pero se limita el control sobre la capa física, lo que puede implicar retos de cumplimiento y localización de datos.

Nube Privada: Recursos exclusivos para una sola organización, ya sea on-premise o en hosting dedicado. Permite un control total y mayor personalización en seguridad, a costa de inversiones significativas en hardware, software y personal especializado.

Nube Híbrida: Combina la infraestructura privada con la pública para aprovechar la escalabilidad de esta última sin sacrificar el control y la seguridad de la primera. Exige un alto grado de integración, orquestación y gobernanza para administrar ambos entornos de forma eficiente.

Describa tres prácticas de seguridad en la nube.

1. **Gestión de identidades y accesos (IAM):**
    - Implementar el principio de privilegios mínimos (Least Privilege) y controles de acceso estrictos para limitar las acciones de usuarios y servicios.
    - Habilitar autenticación multifactor (MFA) siempre que sea posible.
    - Establecer políticas de rotación de credenciales y contraseñas, así como monitorear y revisar periódicamente los permisos asignados.
2. **Cifrado de datos (en tránsito y en reposo):**
    - Utilizar protocolos seguros como TLS/SSL para proteger la comunicación de datos en tránsito entre clientes y servidores.
    - Emplear cifrado a nivel de disco o cifrado de base de datos para datos en reposo, así como la gestión segura de claves (por ejemplo, usando KMS en AWS o Key Vault en Azure).
    - Asegurarse de cumplir con las regulaciones y estándares aplicables (ISO 27001, HIPAA, GDPR, etc.).
3. **Monitoreo, registro y auditoría continua:**
    - Configurar herramientas de monitoreo en tiempo real (CloudWatch, Azure Monitor, GCP Cloud Logging) y sistemas de alerta para detectar comportamientos anómalos.
    - Centralizar registros (logs) en un sistema de SIEM (Security Information and Event Management) para correlacionar eventos de seguridad y responder más rápidamente a incidentes.
    - Revisar periódicamente los informes de auditoría para identificar vulnerabilidades o brechas de seguridad.

¿Qué es la IaC, y cuáles son sus principales beneficios?, mencione 2 herramientas de iaC y sus principales características.

IaC (Infrastructure as Code) se refiere a la práctica de administrar y aprovisionar la infraestructura (servidores, redes, balanceadores, bases de datos, etc.) mediante ficheros de configuración o scripts de automatización, de forma similar a cómo se maneja el código de una aplicación. Esto permite que la infraestructura sea reproducible, versionable y fácil de escalar o modificar.

Beneficios Principales

1. **Consistencia y Repetibilidad**
    - Al describir la infraestructura en archivos de configuración, se garantiza que cada entorno (desarrollo, pruebas, producción) se aprovisione de manera consistente.
    - Se evitan configuraciones manuales que podrían introducir errores o discrepancias.
2. **Versionamiento y Control de Cambios**
    - Se pueden usar sistemas de control de versiones (como Git) para rastrear y auditar todas las modificaciones en la infraestructura.
    - Permite revertir cambios (rollbacks) en caso de problemas, de forma mucho más sencilla y confiable.
3. **Automatización y Escalabilidad**
    - Reduce el tiempo y el esfuerzo manual a la hora de crear o actualizar entornos.
    - Facilita la escalabilidad al poder replicar infraestructura con un simple cambio de parámetros.
4. **Documentación Viva**
    - Los archivos de configuración sirven como "documentación activa" de la infraestructura, que siempre está sincronizada con el estado real del sistema.

Herramientas de IaC y sus Principales Características

1. Terraform

1. **Proveedor Multicloud**: Admite múltiples proveedores (AWS, Azure, GCP, entre otros), lo que posibilita definir y aprovisionar recursos en diferentes nubes con una misma sintaxis (HCL - HashiCorp Configuration Language).
2. **Enfoque Declarativo**: El usuario describe el estado deseado de la infraestructura y Terraform se encarga de crearlo o actualizarlo para llegar a ese estado.

3. **Planificación y Ejecución**: Ofrece el comando terraform plan para previsualizar cambios antes de aplicarlos con terraform apply.
4. **Estado (State Management)**: Mantiene un archivo de estado con la información de la infraestructura aprovisionada, lo que ayuda a rastrear cambios y dependencias.

2. AWS CloudFormation

- **Integración Nativa con AWS**: Diseñado específicamente para servicios de Amazon Web Services. Permite aprovisionar y administrar recursos de AWS mediante plantillas YAML o JSON.
- **Plantillas Versionadas**: Se definen plantillas que describen la configuración de los recursos; estos archivos se pueden almacenar en repositorios Git.
- **Stacks**: CloudFormation maneja grupos de recursos como "stacks"; es decir, conjuntos de recursos que se despliegan y actualizan como una unidad lógica.
- **Automatización de Dependencias**: La herramienta resuelve automáticamente las dependencias entre servicios (por ejemplo, crear primero una VPC antes que las instancias EC2 que la usan).

En síntesis, **IaC** permite que la infraestructura sea tratada como software, con todos los beneficios asociados de automatización, trazabilidad y consistencia. Herramientas como **Terraform** y **AWS CloudFormation** facilitan la creación y gestión de entornos escalables, reduciendo la complejidad y los errores manuales.

¿Qué métricas considera esenciales para el monitoreo de soluciones en la nube?

Al monitorear soluciones en la nube es recomendable hacer un seguimiento tanto de métricas de **infraestructura** como de **aplicación**, además de incluir indicadores de **costos** y **seguridad**. A continuación se detallan las categorías y métricas más relevantes:

1. **Infraestructura**: Se supervisan el uso de CPU, memoria, disco (I/O, espacio, latencia) y el tráfico de red (throughput, latencia, pérdida de paquetes) para prevenir cuellos de botella o saturaciones.
2. **Aplicación y Experiencia de Usuario**: Se controla el tiempo de respuesta (latencia), la tasa de errores, el throughput (RPS) y la satisfacción de los usuarios (Apdex Score), lo que permite identificar problemas de rendimiento y necesidades de escalado.
3. **Seguridad**: Incluye el seguimiento de registros de acceso y autenticación (IAM), la detección de intrusiones o comportamientos anómalos y la revisión de eventos críticos en WAF, firewalls y logs de red.
4. **Contenedores y Orquestadores**: Se mide el consumo de recursos de cada contenedor/pod, la cantidad de servicios activos y la salud de los pods mediante liveness probes y health checks para detectar fallas tempranamente.

5. **Disponibilidad y SLA/SLO**: Se supervisan indicadores de uptime y se definen objetivos de nivel de servicio (SLI/SLO) para evaluar la continuidad y calidad del servicio.
6. **Costos**: Se monitorea el uso de recursos por servicio y los patrones de facturación (billing metrics), identificando picos de consumo que incrementen los gastos.
7. **Logs y Auditoría**: Centralizar y analizar logs permite diagnosticar errores y conductas anómalas, mientras que la auditoría de configuración (IaC) ayuda a rastrear cambios inesperados y mantener la coherencia de los entornos.

¿Qué es Docker y cuáles son sus componentes principales?

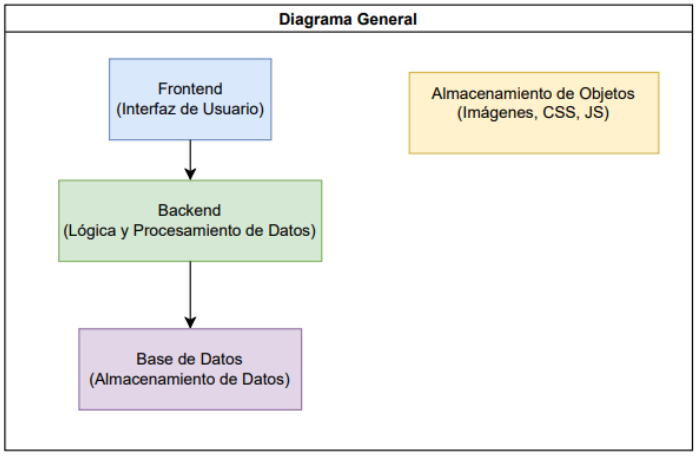
Docker es una plataforma de **contenedores** que permite empaquetar una aplicación con todas sus dependencias bibliotecas, configuraciones, etc, en una sola imagen, de modo que pueda ejecutarse de forma consistente en cualquier entorno que soporte Docker.
A continuación se describen sus principales componentes:

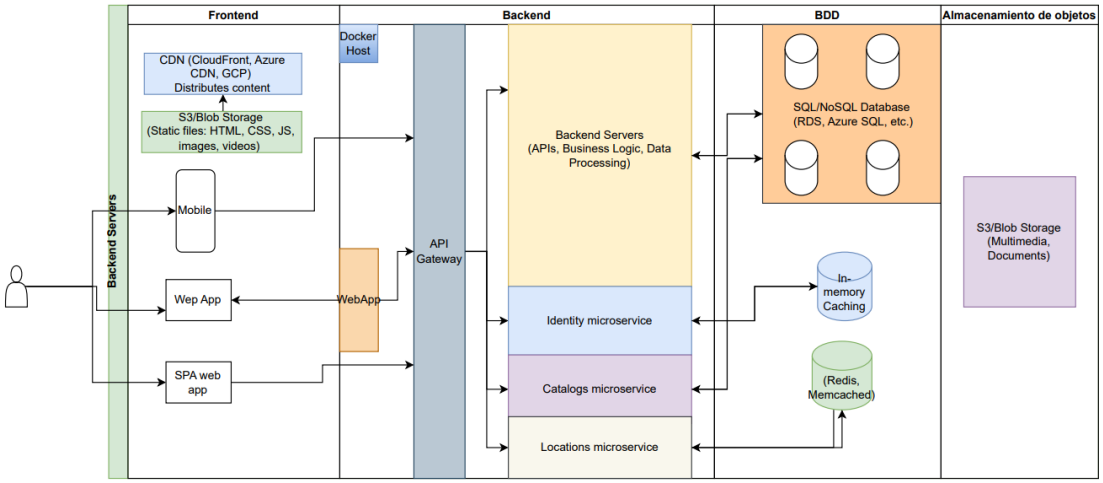
- Docker Engine: La parte central que incluye el Daemon, es encargado de la ejecución y administración de contenedores y la CLI para interactuar con él.
- Docker Daemon: Se ejecuta en segundo plano y gestiona la construcción y ejecución de contenedores a partir de las peticiones de la CLI o la API.
- Docker CLI: Interfaz de línea de comandos para acciones como crear imágenes (docker build), ejecutar contenedores (docker run) o gestionar imágenes en el registro.
- Imágenes Docker: Plantillas de solo lectura que contienen el sistema de archivos y las dependencias necesarias. Se construyen con un Dockerfile.
- Contenedores Docker: Instancias de una imagen que se ejecutan de forma aislada, compartiendo el kernel del sistema operativo anfitrión.
- Docker Registry: Servicio para almacenar y distribuir imágenes (p. ej., Docker Hub, ECR, GitHub Container Registry).
- Docker Compose (opcional): Permite definir y ejecutar aplicaciones compuestas de varios contenedores mediante un archivo docker-compose.yml.

En conjunto, Docker facilita el desarrollo, la distribución y el despliegue de aplicaciones de manera eficiente y reproducible, eliminando discrepancias entre entornos de desarrollo y producción.

Caso práctico

Cree un diseño de arquitectura para una aplicación nativa de nube considerando los siguientes componentes





Frontend

Componentes y Funciones:

- **CDN (Content Delivery Network) (CloudFront, Azure CDN, GCP)**: Sirve contenido estático globalmente, lo que incluye HTML, CSS, JS, imágenes y videos. Al utilizar un CDN, se minimiza la latencia, se mejora la velocidad de carga para los usuarios finales y se reduce la carga directa sobre los servidores de origen.
- **S3/Blob Storage**: Almacena los archivos estáticos necesarios para el frontend, proporcionando un acceso escalable y duradero a estos recursos. Al ser integrado con el CDN, asegura que los contenidos más demandados estén siempre disponibles de manera eficiente.
- **Web App / Mobile / SPA Web App**: Interfaces de usuario diseñadas para diferentes plataformas (web tradicional, móviles, single-page applications), todas sirviendo como

puntos de entrada para los usuarios finales, interactuando con el backend a través de llamadas API.

Justificación del Diseño: El uso del CDN y el almacenamiento en S3 garantiza un rendimiento óptimo del frontend, crucial para mantener una experiencia de usuario fluida y rápida, especialmente importante en aplicaciones que dependen de la velocidad y la eficiencia para la satisfacción del usuario y la retención.

Backend

Componentes y Funciones:

- **Backend Servers**: Encargados de procesar la lógica de negocios, manejar las solicitudes API y realizar operaciones de datos. Estos servidores son el núcleo de la capacidad operativa de la aplicación.
- **Microservices (Identity, Catalogs, Locations)**: Servicios desacoplados que gestionan diferentes aspectos de la aplicación, como autenticación y manejo de usuarios (Identity), gestión de catálogos (Catalogs) y servicios relacionados con ubicaciones (Locations). Estos microservicios permiten una escalabilidad y mantenimiento más eficientes.
- **API Gateway**: Funciona como el punto de entrada para todas las llamadas entrantes al backend, encaminándolas a los servicios internos correspondientes. Facilita la gestión de autenticaciones, autorizaciones y enrutamientos.

Justificación del Diseño: La arquitectura de microservicios junto con un API Gateway facilita el desarrollo, el despliegue independiente y la escala de diferentes partes del sistema sin afectar a otros componentes, lo que mejora significativamente la agilidad y la eficiencia operativa.

Base de Datos (BDD)

Componentes y Funciones:

- **SQL/NoSQL Database**: Sistemas de gestión de bases de datos que almacenan y recuperan toda la información necesaria para la aplicación. Puede incluir bases de datos relacionales (SQL) o no relacionales (NoSQL) dependiendo de los requisitos de estructura de datos y escalabilidad.
- **In-memory Caching (Redis, Memcached)**: Acelera el acceso a datos frecuentemente solicitados, reduciendo la carga en las bases de datos y mejorando el tiempo de respuesta de las aplicaciones.

Justificación del Diseño: La combinación de bases de datos administradas y sistemas de caché en memoria asegura que la aplicación pueda manejar tanto grandes volúmenes de datos como responder rápidamente a las solicitudes de los usuarios, crucial para el rendimiento y la escalabilidad.

Almacenamiento de Objetos

Componentes y Funciones:

- **S3/Blob Storage (Multimedia, Documents)**: Se utiliza para el almacenamiento de objetos grandes, como archivos multimedia o documentos, que requieren alta durabilidad y disponibilidad pero no acceso instantáneo como el contenido estático del frontend.

Justificación del Diseño: Este almacenamiento es esencial para manejar grandes volúmenes de datos no estructurados, ofreciendo escalabilidad y reduciendo los costos al permitir el almacenamiento eficiente de grandes cantidades de datos que no requieren el alto I/O de los sistemas de archivos tradicionales.

Este diseño proporciona una base sólida para una arquitectura de aplicación en la nube robusta, escalable y segura, utilizando las mejores prácticas y servicios disponibles en AWS para asegurar la eficiencia operativa y la mejor experiencia posible para el usuario.

Diseño

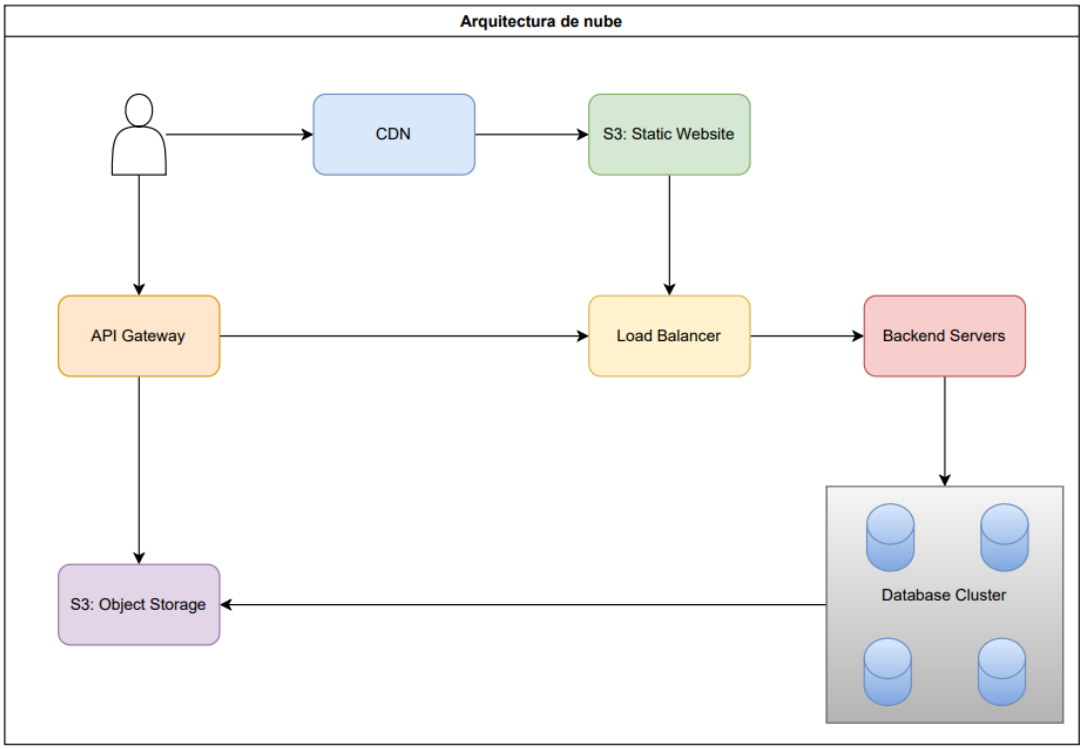
Elección del proveedor de nube: Amazon Web Services (AWS)

1. **Madurez y amplitud de servicios**: AWS ofrece un extenso catálogo de servicios (cómputo, bases de datos, almacenamiento, redes, etc.) con una gran adopción en la industria.
2. **Ecosistema de herramientas**: Servicios como **Amazon S3**, **Amazon CloudFront**, **Amazon RDS** y las diversas opciones de cómputo (EC2, ECS, EKS, Lambda) facilitan la construcción de arquitecturas escalables y seguras.
3. **Alto nivel de comunidad y soporte**: Al ser uno de los líderes de mercado, se cuenta con abundante documentación oficial, foros y soluciones de terceros.

Descripción General de la Arquitectura

La arquitectura propuesta se basa en **una aplicación web** que atiende solicitudes de los clientes a través de un **servicio de CDN y balanceo de carga**, conectada a un **capa de cómputo** (contenedores o instancias) que se comunican con un **sistema de base de datos administrada**. El contenido estático (imágenes, archivos, etc.) se aloja en **Amazon S3**, distribuido opcionalmente por **CloudFront** para mejorar el rendimiento global.

Diagrama de Alto Nivel



Justificación de Cada Componente

1. **Amazon CloudFront (CDN)**
    - Mejora el desempeño global al distribuir contenido estático (HTML, CSS, JS, imágenes) a través de edge locations, reduciendo la latencia para el usuario final.
    - Agrega una capa adicional de seguridad (integración con AWS WAF, protección DDoS).
2. **Amazon S3 (Simple Storage Service)**
    - Almacén de objetos de alta durabilidad (99.999999999%) y disponibilidad.
    - Ideal para alojar contenido estático como imágenes, archivos multimedia, descargas, etc.
    - Permite escalar de manera elástica sin necesidad de gestionar almacenamiento en servidores.
3. **Application Load Balancer (ALB)**
    - Distribuye el tráfico entrante (HTTP/HTTPS) hacia los recursos de cómputo en distintas zonas de disponibilidad (Multi-AZ).
    - Ofrece reglas de enrutamiento basadas en la ruta o cabeceras, facilitando arquitectura de microservicios.

4. **Capa de Cómputo**
    - **Amazon ECS/EKS** (Contenedores):
        - Facilita la ejecución de servicios en contenedores Docker, con escalado automático (ECS con Fargate no requiere gestionar servidores directamente).
    - **Amazon EC2** (Instancias virtuales):
        - Permite más control sobre el SO subyacente si se necesitan personalizaciones profundas o software que no esté contenedorizado.
    - La decisión dependerá de la estrategia interna y el nivel de modernización de la aplicación.
5. **Amazon RDS (Base de Datos Relacional)**
    - Ofrece un motor administrado (p.ej., MySQL, PostgreSQL, MariaDB, etc.) con backups automáticos, Multi-AZ para alta disponibilidad, réplicas de lectura, etc.
    - Reduce la carga operativa de administrar parches, seguridad y mantenimiento de la base de datos.
6. **(Opcional) Amazon ElastiCache**
    - Puede usarse para mejorar el rendimiento y reducir la latencia en operaciones de lectura frecuentes (Redis o Memcached).
    - Ideal si se necesita almacenar sesiones, caché de consultas, etc.
7. **Seguridad y Redes**
    - **Amazon VPC**: Red aislada lógicamente, con subnets públicas y privadas, seguridad por SG (Security Groups) y ACLs (Access Control Lists).
    - **AWS WAF (opcional)**: Firewall de aplicaciones web para filtrar tráfico malicioso en CloudFront/ALB.
    - **IAM (Identity & Access Management)**: Control de permisos y roles para cada servicio y usuario.

Decisiones de Diseño Clave

1. **CDN y S3 para contenido estático**: Minimiza la latencia y reduce la carga en la aplicación al servir archivos desde un almacenamiento altamente escalable.
2. **Balanceo de carga y autoscaling**: Garantiza alta disponibilidad, distribuye el tráfico y ajusta el número de instancias o contenedores según la demanda.
3. **Base de datos administrada**: Libera al equipo de la gestión de parches y backups, ofreciendo escalado vertical y Multi-AZ para alta disponibilidad.
4. **Separación de componentes en subnets**:

   - Subnets públicas para el ALB (recibe el tráfico de Internet).
   - Subnets privadas para la capa de cómputo (EC2/ECS/EKS) y la base de datos (RDS).
   - Reglas de seguridad específicas minimizan la superficie de ataque.

5. **Monitorización y Logs**: Integración con **Amazon CloudWatch** para recopilar métricas y logs de la aplicación, los contenedores/instancias y la base de datos.

Escalabilidad y Alta Disponibilidad

- **Multi-AZ**: Implementación de la aplicación y la base de datos en al menos dos zonas de disponibilidad para tolerar la caída de una de ellas.
- **Auto Scaling**: Configuración de escalado horizontal (agregar/eliminar instancias o contenedores) basándose en métricas de CPU, memoria o latencia.
- **Health Checks**: Tanto ALB como ECS/EKS realizan verificaciones de salud para asegurarse de que solo se enrute el tráfico a instancias/contendores saludables.

Conclusión

Esta **arquitectura nativa de nube en AWS** aprovecha servicios administrados y escalables para asegurar disponibilidad, rendimiento y simplicidad operativa. El uso de **contenedores** o **instancias** se elige según la estrategia de desarrollo y despliegue. Mientras tanto, la **separación de responsabilidades** (frontend distribuido con CloudFront, almacenamiento estático en S3, RDS administrado para la base de datos) promueve un diseño modular, seguro y fácil de evolucionar.