DOCUMENTO DE DISEÑO DE ARQUITECTURA 3-TIER

Fecha: 23 de diciembre de 2025

Autor: Brandon Concha

Prueba Técnica

Github: https://github.com/brandon2399/arquitectura-3-tier

Parte 1: Preguntas Teóricas

1. Diferencia entre nube pública, privada e híbrida

- **Nube Pública:** Los servicios (computación, almacenamiento) son propiedad de un proveedor externo (AWS, Azure, GCP) y se comparten con otras organizaciones a través de internet. Es altamente escalable y pagas solo por lo que usas.
- **Nube Privada:** Recursos informáticos utilizados exclusivamente por una sola organización. Puede situarse físicamente en el centro de datos de la empresa o ser gestionada por un tercero. Ofrece mayor control y seguridad personalizada.
- **Nube Híbrida:** Una combinación de ambas que permite que los datos y las aplicaciones se compartan entre ellas. Ideal para empresas que quieren mantener datos sensibles en local (privada) pero aprovechar la escalabilidad de la nube pública para cargas de trabajo variables.

2. Tres prácticas de seguridad en la nube

1. **Principio de Menor Privilegio (Least Privilege):** Configurar políticas de IAM (Identity and Access Management) para que los usuarios y servicios tengan solo los permisos estrictamente necesarios.
2. **Cifrado de datos:** Implementar cifrado tanto **en reposo** (discos, bases de datos) como **en tránsito** (uso de certificados SSL/TLS para HTTPS).
3. **Seguridad Perimetral y de Red:** Utilizar firewalls gestionados (como Grupos de Seguridad o WAF) y segmentar la red mediante subredes públicas y privadas para aislar componentes críticos.

3. ¿Qué es IaC y cuáles son sus beneficios?

La **Infraestructura como Código (IaC)** es la gestión y el aprovisionamiento de la infraestructura a través de archivos de configuración legibles por máquina, en lugar de procesos manuales.

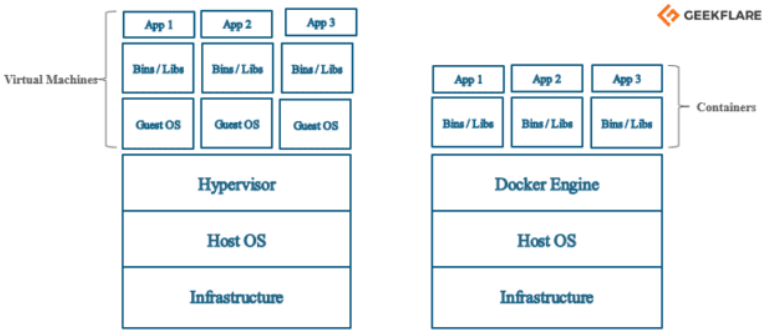
- **Beneficios:** Repetibilidad (evita el "error humano"), Velocidad de despliegue, Consistencia entre entornos (Dev, Test, Prod) y Control de versiones (Git).
- **Herramientas:**
    - **Terraform:** Agnóstica (funciona con cualquier proveedor), utiliza lenguaje HCL y mantiene un archivo de "estado" de la infraestructura.
    - **AWS CloudFormation:** Nativa de AWS, utiliza JSON/YAML y permite gestionar todos los recursos de AWS de forma integrada.

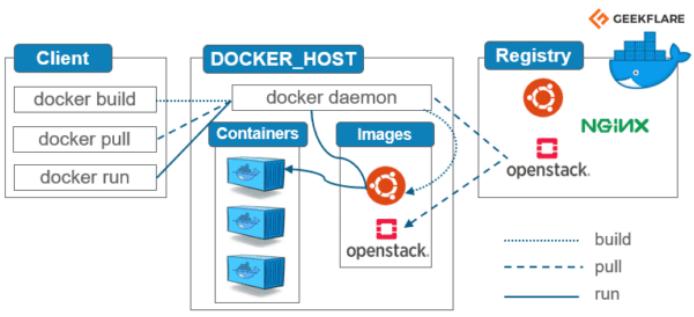
4. Métricas esenciales para el monitoreo

- **Rendimiento de recursos:** Uso de CPU, Memoria RAM y Latencia de red.
- **Disponibilidad:** Porcentaje de tiempo de actividad (Uptime) y tasas de error (ej. errores 5xx en HTTP).
- **Almacenamiento:** Espacio en disco disponible e IOPS (operaciones de entrada/salida).
- **Costos:** Monitoreo del presupuesto diario para evitar picos inesperados.

5. ¿Qué es Docker y sus componentes?

Docker es una plataforma de código abierto que automatiza el despliegue de aplicaciones dentro de contenedores de software, permitiendo que la app funcione igual en cualquier entorno.





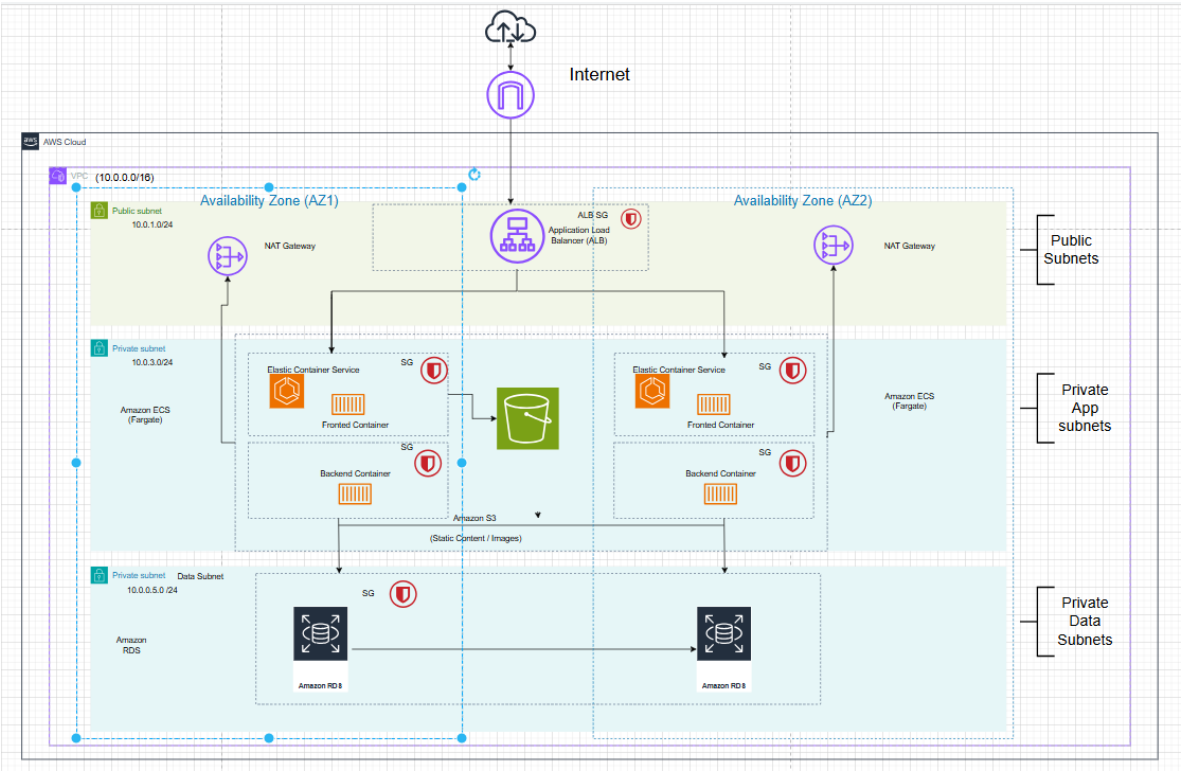
- **Componentes principales:**
    - **Dockerfile:** Archivo de texto con las instrucciones para construir la imagen.
    - **Imagen:** Plantilla de solo lectura que contiene el código, librerías y dependencias.
    - **Contenedor:** La instancia ejecutable y ligera de una imagen.
    - **Docker Hub/Registry:** Repositorio donde se almacenan y comparten las imágenes.

Parte 2: Caso Práctico - Diseño de Arquitectura 3-TIER - Justificación Técnica de Diseño

1. Resumen

El presente documento describe la arquitectura técnica para una aplicación web moderna desplegada en AWS. El diseño prioriza la **seguridad por aislamiento**, la **alta disponibilidad** y el **bajo costo operativo** mediante el uso de servicios gestionados.

2. Diagrama Conceptual



3. Decisiones de Diseño y Justificación

Cómputo: AWS Fargate (Serverless Containers)

- **Decisión:** Utilizar Amazon ECS con capacidad Fargate en lugar de instancias EC2 tradicionales.
- **Justificación:** * **Menor Carga Operativa:** No hay servidores que parchar ni sistemas operativos que asegurar.
    - **Escalabilidad:** Fargate escala horizontalmente de forma nativa según la demanda.
    - **Eficiencia de Costos:** Solo se paga por los recursos (CPU/RAM) que el contenedor consume mientras está encendido.

Red: VPC con Aislamiento de 3 Capas

- **Decisión:** Segmentación de recursos en subredes Públicas, Privadas de Aplicación y Privadas de Datos en 2 Zonas de Disponibilidad.
- **Justificación:**
    - **Seguridad (Defensa en Profundidad):** Los componentes más críticos (DB y Backend) están protegidos por múltiples capas de red.
    - **Alta Disponibilidad:** El diseño Multi-AZ garantiza que si un centro de datos físico de AWS falla, el demo permanecerá activo en la otra zona.

Almacenamiento: Amazon S3 y RDS

- **Decisión:** Desacoplar el almacenamiento de objetos (S3) y la base de datos relacional (RDS).
- **Justificación:**
    - **Persistencia:** Los contenedores son efímeros; los datos deben vivir en servicios diseñados para una durabilidad del 99.99%.
    - **Rendimiento:** RDS ofrece backups automáticos y optimización de consultas que sería costosa de mantener manualmente.

Seguridad: Principio de Menor Privilegio

- **Decisión:** Implementación de Security Groups específicos y Roles de IAM.
- **Justificación:**
    - **Control Granular:** Cada componente solo puede hablar con quien necesita. Por ejemplo, la base de datos solo escucha al backend, bloqueando cualquier otro intento de conexión incluso desde dentro de la misma red.

Flujo de Operación

1. **Entrada:** El tráfico es recibido por el ALB (Punto único de entrada).
2. **Procesamiento:** ECS Fargate ejecuta la lógica de negocio en subredes privadas.
3. **Almacenamiento:** Los activos estáticos se sirven/guardan en S3 y los datos transaccionales en RDS.
4. **Salida:** Los logs y métricas se envían a CloudWatch para monitoreo en tiempo real.

Esta arquitectura no solo cumple con los requisitos del demo, sino que establece las bases para una aplicación de grado de producción. Es segura, fácil de auditar y altamente resiliente.

Cabe mencionar que es posible realizar mejoras y optimaciones no hay una única solución posible pero esta es la que yo haría.

Reflexión propia

Todas las decisiones de arquitectura son una negociación entre "beneficios" y "costo"

cada decisión trae riesgos, los riesgos se mitigan, se trasladan o se asumen, en el mundo ideal basado en la probabilidad de ocurrencia, en el mundo real basado tal vez en cuánto me cuesta los trade-offs que llaman.

https://github.com/brandon2399/arquitectura-3-tier