PRUEBA TÉCNICA - DEVSU

1. **¿Cuál es la diferencia entre nube pública, privada e híbrida?**
    a. **Nube pública:** Proporcionada por terceros (como AWS, Azure, GCP) a través de Internet y compartida entre múltiples clientes. Ejemplos de servicios incluyen almacenamiento, máquinas virtuales y bases de datos.
        i. Ventajas: costo reducido, escalabilidad.
        ii. Desventajas: menos control directo sobre la infraestructura.
    b. **Nube privada:** Infraestructura dedicada para una sola organización, ya sea gestionada internamente o por un proveedor externo, pero en un entorno privado.
        i. Ventajas: mayor control y personalización.
        ii. Desventajas: costos más altos y menor escalabilidad inmediata.
    c. **Nube híbrida:** Combinación de nubes públicas y privadas, permitiendo que las organizaciones aprovechen lo mejor de ambos mundos. Se pueden mantener servicios sensibles en la nube privada y escalar recursos no críticos en la nube pública.
        i. Ventajas: flexibilidad, balance de costos y control.
        ii. Desventajas: complejidad en la integración.

2. **Describa tres prácticas de seguridad en la nube.**
    a. **Control de acceso basado en roles (RBAC):** Asegurarse de que los usuarios solo tengan acceso a los recursos que necesitan para su rol específico, limitando permisos excesivos.
    b. **Encriptación de datos:** Asegurar que los datos estén cifrados tanto en reposo como en tránsito para proteger información sensible ante accesos no autorizados.
    c. **Monitoreo y auditoría continua:** Implementar sistemas como AWS CloudTrail o Azure Monitor para rastrear las actividades y cambios en los recursos en la nube, permitiendo detectar posibles incidentes de seguridad.

3. **¿Qué es la IaC y cuáles son sus principales beneficios? Mencione 2 herramientas de IaC y sus principales características.**
    a. **IaC (Infraestructura como Código):** Es la práctica de gestionar y aprovisionar la infraestructura mediante archivos de configuración en lugar de hacerlo manualmente. Esto permite automatizar la creación, modificación y eliminación de recursos.

b. **Beneficios:**
    i. Automatización: Facilita la automatización de la infraestructura, reduciendo errores humanos.
    ii. Escalabilidad: Permite replicar entornos rápidamente, ayudando a escalar según sea necesario.
    iii. Versionamiento: Al tratarse de código, se puede versionar y auditar fácilmente.
c. **Herramientas:**
    i. Terraform: Una herramienta multiplataforma que permite gestionar infraestructura en múltiples proveedores de nube, usando archivos de configuración que pueden ser versionados.
    ii. AWS CloudFormation: Específico para AWS, permite definir la infraestructura como plantillas JSON o YAML y automatizar el aprovisionamiento de recursos dentro de la nube de AWS.

4. **¿Qué métricas considera esenciales para el monitoreo de soluciones en la nube?**
    d. **Uso de CPU y memoria:** Para medir la eficiencia y detectar posibles sobrecargas en servidores o servicios.
    e. **Latencia de red:** Para medir el rendimiento y la rapidez con la que las aplicaciones responden.
    f. **Tasa de errores:** Monitorear la cantidad de errores en las aplicaciones, como respuestas 500 en servicios web.
    g. **Disponibilidad/Uptime:** Verificar el tiempo que los servicios han estado en funcionamiento, ayudando a mantener acuerdos de nivel de servicio (SLA).

5. **¿Qué es Docker y cuáles son sus componentes principales?**
    a. **Docker:** Es una plataforma que permite la creación, ejecución y administración de contenedores, que son unidades empaquetadas de software que incluyen todo lo necesario para que una aplicación funcione, independientemente del entorno donde se ejecute.
    b. **Componentes principales:**
        i. Docker Engine: El motor que permite la creación y ejecución de contenedores.
        ii. Dockerfile: Un archivo que contiene instrucciones para crear una imagen de Docker, definiendo cómo se empaqueta una aplicación.

iii. Imágenes: Son snapshots que contienen el software, dependencias y configuraciones para ejecutar una aplicación.

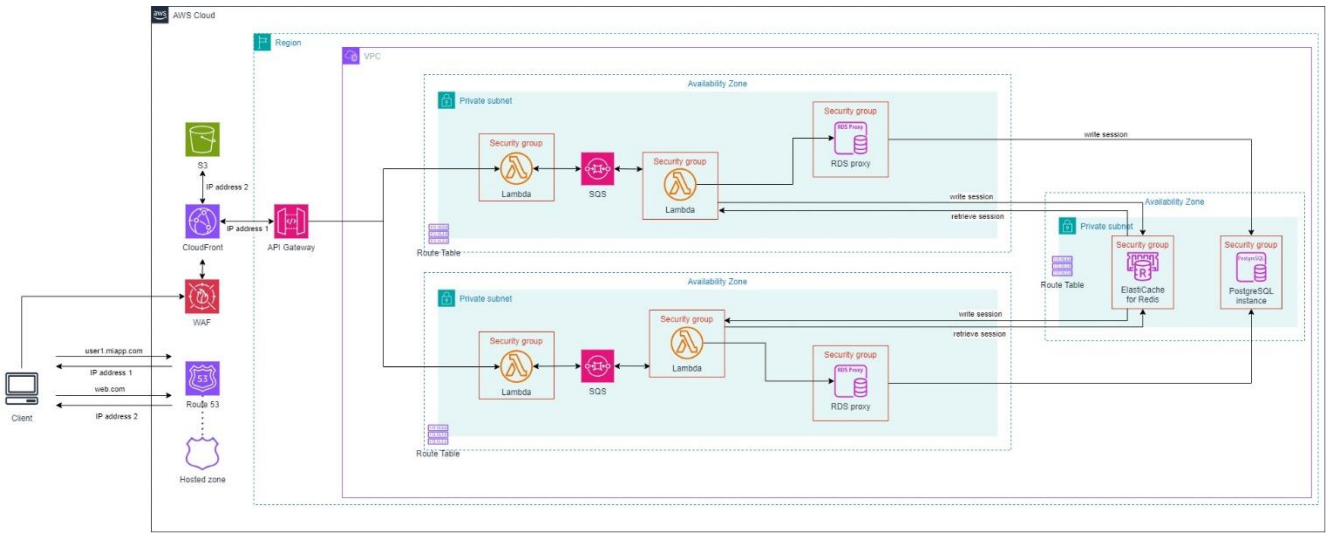
iv. Contenedores: Instancias de una imagen que se están ejecutando.

v. Docker Hub: Un repositorio público donde se almacenan imágenes de Docker listas para usar o compartir.

6. Caso práctico

a. Arquitectura

Elegí crear una arquitectura para un escenario de una Empresa (Startup) SaaS Multiinquilino



b. Explicación

Elegí AWS por mi expertis y tiempo de uso, ya que de las 3 nubes (AWS, AZURE, GCP), con AWS es con la que más he interactuado. A continuación, doy un detalle de cada parte de la arquitectura.

I. **Frontend**:

- En este escenario el sitio web estático de la aplicación está alojado en Amazon S3, utilizando Amazon CloudFront para la distribución global del contenido estático (HTML, CSS, JavaScript). Esto asegura que las solicitudes de los usuarios sean atendidas de manera rápida y eficiente, reduciendo la latencia.
- Elegí usar AWS WAF (Web Application Firewall) para proteger el sitio contra ataques DDoS y otros posibles riesgos de seguridad.

- En este escenario cada usuario tiene su tienda online con su propio subdominio (por ejemplo, user1.miapp.com), y los subdominios se gestionan utilizando Amazon Route 53. Esto facilita la creación de un entorno personalizado para cada cliente.

II. **Backend**:

- Para el backend elegí un enfoque serverless utilizando AWS Lambda. Aquí es donde se ejecuta la lógica principal de la aplicación en respuesta a las solicitudes API que llegan a través de API Gateway. Como ejemplo, en algunas de mis funciones Lambda podría estar ejecutando código en Node.js o Python, lo que da flexibilidad para ajustar las tecnologías según sea necesario.
- Entre el API Gateway y las funciones Lambda, elegí usar Amazon SQS FIFO para desacoplar las capas del frontend y backend. Esto asegura que las solicitudes se procesen de manera ordenada y que el sistema sea más resiliente ante picos de carga.

III. **Base de datos**:

- Para almacenar los datos de los usuarios, utilizo Amazon RDS con PostgreSQL. Es una base de datos multiinquilino (multi-tenant), lo que significa que todas las páginas (o tiendas) de los usuarios están gestionadas en una sola base de datos, pero cada "tienda" está aislada lógicamente mediante un tenantId.
- Para mejorar el rendimiento, elegí RDS Proxy, que ayuda a gestionar el agrupamiento de conexiones y evita problemas de sobrecarga cuando hay un aumento en las solicitudes.

IV. **Almacenamiento de objetos y caché**:

- Los archivos como imágenes y otros recursos multimedia subidos por los usuarios se almacenan en Amazon S3, lo que asegura una alta disponibilidad y durabilidad de los datos.
- Para mejorar la velocidad de acceso a la información crítica, elegí Amazon ElastiCache con Redis, que se utiliza para almacenar sesiones y para caching de datos que se necesitan rápidamente en las interfaces de administración y las tiendas online.

V. **Alta disponibilidad**:

- Para asegurar la alta disponibilidad de los servicios más críticos, decidí que todos los componentes clave estén en múltiples zonas de disponibilidad

(AZ). Esto garantiza que, en caso de una falla en una zona de disponibilidad, el sistema pueda continuar operando sin interrupciones.

- Elegí usar solo subredes privadas ya que, para este escenario, se consideran que no se necesita que los recursos tengan acceso a internet.