Reto Técnico

1. ¿Cuál es la diferencia entre nube pública, privada e híbrida?

En la **nube publica,** la infraestructura le pertenece a un proveedor como AWS, AZURE o GCP, y muchas empresas u organizaciones, la usan al mismo tiempo. En la **nube privada,** este tipo de nube es gestionado por una organización. Se utiliza para ofrecer servicios de tecnología de la información a los empleados internos, mediante la implementación de una infraestructura de nube en las instalaciones de la propia empresa, Y la **nube hibrida,** combina ambos entornos, lo que permite mover cargas entre sí.

Nube Publica	**Nube Privada**	**Nube Hibrida**
Proveedor (Aws, Azure, Gcp)	Infraestructura propia	Mezcla de pública y privada
Compartida	Control total	Flexibilidad
Escalable	Seguridad avanzada	Escalamiento dinámico
Bajo Costo inicial	Cumplimiento regulatorio	Migración progresiva

2. ¿Describa tres prácticas de seguridad en la nube?

Primer practica de seguridad, "**Permisos"** adopto esta práctica siempre. Porque es fundamental dar los permisos necesarios, y ni un privilegio extra. Tanto para rol, servicio, usuario o aplicación debe tener únicamente los permisos necesarios para funcionar.

Segunda practica de seguridad, **“Cifrado”,** tener muy claro que tener cifrado todo, es también muy fundamental, ya que no queremos que se vaya a filtrar ningún dato confidencial, en AWS uso kms para manejar las claves, en Azure uso Key vault que hace lo mismo, todo lo que sea tráfico entre servicios debe ir por Https/tls. El cifrado garantiza que, incluso si llegase ocurrir un acceso indebido, un error, o una exposición accidental de un bucket. Los datos siguen siendo ilegibles.

Tercer practica de seguridad, **“Monitoreo”** y es que después de haber tomado estas practicas necesitamos saber que esta ocurriendo en la infraestructura en cada momento, yo en especial en Azure uso Azure Monitor con log analytic y Microsoft Defender for Cloud, en Aws utilizo X-Ray, Cloudtrail y CloudWatch. en Gcp Cloud monitoring, SCC. En conclusión, cada proveedor cuenta con herramientas nativas de monitoreo, sin embargo, existen plataformas de monitoreo avanzada que nos ayuda a monitorear mejor una infraestructura.

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

La IaC, es una forma de gestionar recursos y aprovisionar infraestructura utilizando archivos en lugar de hacerlos manualmente. Basicamente en vez de entrar a Google cloud, Azure o AWS, a crear recursos uno por uno, se define todo en archivos que se pueden automatizar, gestionar y desplegar.

Beneficios:

- **Automatización completa del despliegue:** Se puede desplegar toda una arquitectura con un comando, o dentro de un pipeline CI/CD.

- **Entornos repetibles y consistentes:** Se puede crear o reconstruir entornos idénticos tantas veces como sea necesario. Esto evita las configuraciones manuales, y la reducción de tiempos operativos.

- **Control de cambios y auditoria:** Como todo está en Git, es posible ver qué se cambió, cuándo se cambió y quién lo cambió. Esto mejora la trazabilidad y permite hacer rollback rápido si algo falla.

- **Reducción de errores:** Menos acciones manuales significa menos posibilidades de mala configuración y mayor estabilidad en los entornos, esto ayuda mucho ya que no hay configuraciones diferente por accidente.

Dos herramientas de IaC y sus principales características

1. **Terraform:** es una de las herramientas de IaC más usadas a nivel empresarial. Permite definir y administrar infraestructura usando un lenguaje declarativo HCL y funciona con prácticamente cualquier proveedor de nube.

   Principales características:

   - **Multinube:** compatible con AWS, Azure, GCP y cientos de proveedores más (Kubernetes, VMware, Oracle, Cloudflare, GitHub entre otros). Esto permite manejar toda la infraestructura desde una sola herramienta sin importar donde esté desplegada.

   - **Lenguaje Declarativo (HCL):** describes qué quieres, y Terraform se encarga de cómo construirlo.

   - **Planificación de cambios (terraform plan):** registra qué recursos existen para aplicar cambios de forma precisa, aquí muestra exactamente que se va a crear, modificar o eliminar antes de aplicarlo.

   - **Automatizable y fácil de integrar con CI/CD:** permite tener despliegues consistentes y repetibles

   - **Compatibilidad con políticas**: se pueden aplicar reglas para controlar quién despliega, qué configuraciones están permitidas y asegurar que toda la infraestructura siga las mejores prácticas.

2. **Azure Bicep:** es una herramienta que permite crear y administrar toda la infraestructura en Azure de forma automática, usando un lenguaje limpio y fácil de leer diseñado específicamente para Azure. Es completamente integrado con Azure Resource Manager (ARM), entonces cuando salen nuevos servicios o features, Bicep suele soportarlos de inmediato.

Principales características:

- **Totalmente integrado con Azure:** No necesitas plugins, dependencias o configuraciones externas. Funciona directo con todo lo que ofrece Azure y se compila automáticamente a plantillas ARM, no necesitas herramientas extra, funciona directo con todos los servicios.
- **Compilación automática:** Bicep se convierte en ARM internamente, así que siempre es compatible con los recursos más nuevos.
- **Automatiza entornos completos:** Con un solo archivo Bicep puedes desplegar desde una red virtual hasta una aplicación completa con App Services, bases de datos, almacenamiento y monitoreo.Validación y autocompletado: ofrece ayudas en el editor para evitar errores y crear plantillas más confiables.

3. **AWS CloudFormation:** es una herramienta que te permite crear y administrar toda tu infraestructura en AWS de forma automática, usando plantillas escritas en JSON o YAML, es completamente integrado con AWS, entonces cuando salen nuevos servicios o features, CloudFormation suele soportarlos rápido.

   Principales características:

   - **Totalmente integrado con AWS:** No necesitas plugins, dependencias, configuraciones externas. Funciona directo con todo lo que ofrece AWS.

   - **Plantillas en YAML o JSON:** Puedes documentar y estandarizar tu infraestructura en archivos que cualquiera del equipo puede revisar, versionar y mejorar.

   - **Automatiza entornos completos:** con un solo archivo puedes desplegar desde una VPC hasta un clúster ECS entero.

- **Rollback automático:** Si un deployment falla, CloudFormation deshace los cambios sin que tengas que correr a arreglar nada manualmente.

- **Gestión de cambios:** Detecta si alguien modificó recursos desde la consola y te avisa antes de aplicar cambios. Eso evita muchas inconsistencias y fallos de users.

- **StackSets para múltiples cuentas y regiones**

Te permite desplegar la misma infraestructura en varias cuentas y regiones en un solo paso.

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

Cuando hablamos de monitoreo en la nube, no se trata solo de ver si un servidor está “encendido”, sino de entender cómo se está comportando la infraestructura, los servicios y las aplicaciones. Hay métricas que considero fundamentales porque permiten detectar problemas antes de que afecten al usuario final.

En lo personal lo primero que suelo configurar son las métricas de rendimiento, la latencia es crítica, y nadie quiere usar una aplicación que tarda 3 segundos en responder cada click, no exento de las demás métricas que también suelo usar o se suele usar por grandes compañías, aquí explicare varias métricas esenciales para el monitoreo.

- **Métricas de rendimiento:** Ayudan a saber si los recursos están funcionando con fluidez y si la capacidad asignada realmente coincide con lo que la aplicación necesita.

- **Métricas de red:** Permiten identificar cuellos de botella en la comunicación.

- **Métricas de disponibilidad:** confirman si el servicio realmente está accesible y garantizan que los componentes críticos continúen funcionando incluso cuando hay fallos.

- **Métricas de logs y eventos:** son clave para entender qué pasó y por qué pasó, porque dejan un rastro detallado del comportamiento de la aplicación y ayudan en cualquier análisis o investigación posterior.

- **Métricas de seguridad:** ayudan a detectar comportamientos inusuales o amenazas y permiten reaccionar a tiempo antes de que un incidente afecte datos.

- **Métricas de costos y uso:** sirven para evitar gastos innecesarios y encontrar oportunidades de optimización.

5. ¿Qué es Docker y cuáles son sus componentes principales?

Docker es una herramienta muy potente donde podemos crear, empaquetar y ejecutar aplicaciones mucho más sencillo y ordenado, podemos con estos ejecutar en unas unidades llamadas contenedores. La idea es que, en lugar de instalar todo a mano en cada servidor Docker, se mete la aplicación dentro de un contenedor, que es como una “caja” ligera donde va todo lo que la app necesita: sus dependencias, configuraciones, versiones exactas de librerías, etc, gracias a eso, la aplicación funciona igual sin importar dónde la ejecutes: en tu computador, en un servidor o en la nube.

Docker Engine

Es el “corazon” o “motor” de Docker, el programa que se encarga de crear y ejecutar los contenedores, sin esto nada funciona.

Docker Images

Son plantillas que contienen todo lo necesario para ejecutar una aplicación, aquí se define que debe llevar en la plantilla “imagen” (dependencias, sistema base, la aplicación), y a partir de la imagen se crean dichos contenedores.

Contenedores (Docker Containers)

Son instancias vivas de una imagen. Un contenedor es la aplicación corriendo, y con todo lo necesario para funcionar y cada contenedor puede iniciarse, detenerse, eliminarse o replicarse sin afectar a los demás.

Dockerfile

Es un archivo donde escribes las instrucciones para construir una imagen, donde le dices como construir una imagen paso a paso.

Ahí definimos todo: que sistema usar, que instalar, que copiar y que comando debe ejecutarse.

Docker Hub (o Registry)

Es un repositorio donde guardas tus imágenes, parecido a GitHub, pero para Docker. Hay podemos encontrar imágenes, descargarlas o usar imágenes oficiales de bases de datos, lenguajes, etc.

Volúmenes

Los contenedores por default no persisten datos cuando se destruyen

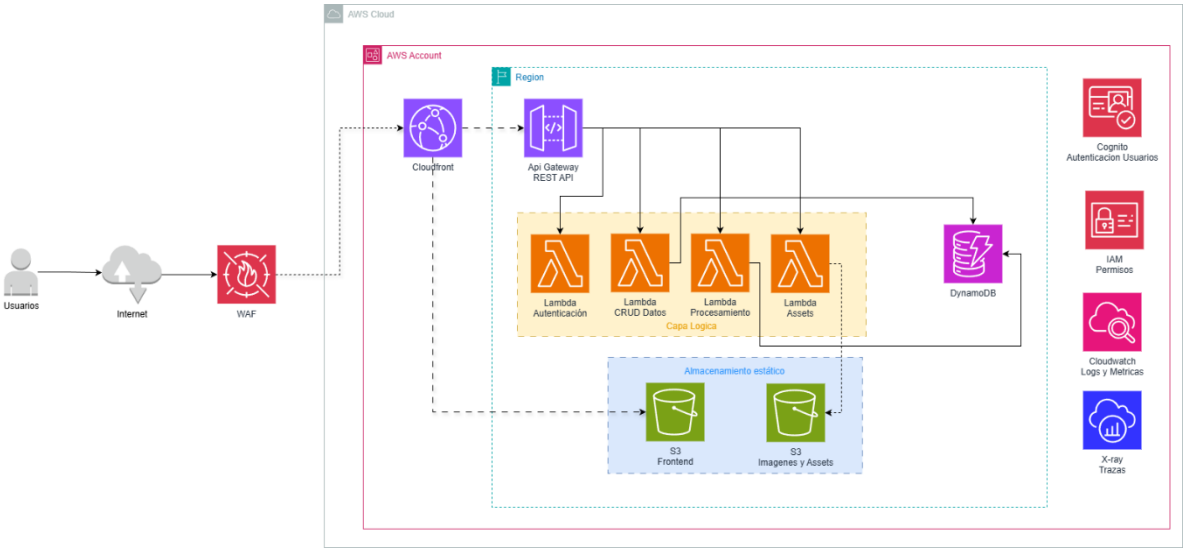
Si se destruyen perdemos todo lo que teníamos dentro de ello. Los volúmenes son la forma de mantener datos persistentes fuera del contenedor, útil para cosas como bases de datos, y esto hace que nos permita guardar datos que sobreviven al contenedor.

Docker Compose

Es un servicio que permite levantar varios contenedores al mismo tiempo con un solo archivo YAML, definimos todo en un archivo docker-compose.yml los servicios, redes, volúmenes, y con un "docker-compose up" levantas todo junto. Es muy útil para desarrollo cuando tenemos frontend, backend, base de datos, etc.

Caso Práctico – Diseño de Arquitectura en la Nube con (AWS)

Para este caso práctico diseñé una arquitectura totalmente serverless en AWS, enfocada en lograr una plataforma escalable, segura y de bajo costo. La solución contempla los tres componentes solicitados: frontend, backend y un sistema de almacenamiento de objetos.



Flujo general de la arquitectura

Los usuarios acceden a la aplicación desde Internet. El tráfico pasa primero por AWS WAF, que actúa como firewall filtrando peticiones maliciosas. Luego llega a CloudFront, que distribuye el contenido globalmente y lo entrega desde el punto más cercano al usuario. Desde ahí, las peticiones se dirigen al API Gateway, que orquesta las llamadas a las funciones Lambda correspondientes. Estas funciones interactúan con DynamoDB para datos estructurados y con S3 para almacenamiento de archivos.

Elección del Proveedor de Nube: AWS

Elegí AWS porque ofrece servicios serverless muy maduros, integración nativa entre componentes, un ecosistema amplio de seguridad y un modelo de costos que se ajusta muy bien a aplicaciones que crecen progresivamente.

Además, servicios como Lambda, API Gateway y S3 permiten construir una arquitectura flexible sin necesidad de administrar servidores.

Componentes principales

AWS WAF - Protección perimetral

Coloqué WAF como primera barrera de seguridad porque es fundamental proteger la aplicación antes de que el tráfico llegue a los componentes internos. WAF filtra automáticamente ataques conocidos como inyección SQL, cross-site scripting y bots maliciosos. También permite crear reglas personalizadas para bloquear patrones específicos de tráfico sospechoso. Esto reduce la carga en los servicios internos y mejora la seguridad general.

Amazon CloudFront - Distribución global de contenido

CloudFront es la CDN que elegí porque tiene más de 400 puntos de presencia en todo el mundo. Esto significa que un usuario en Asia accede al contenido desde un servidor cercano, no desde la región principal de AWS. El resultado es una latencia mucho menor. Además, CloudFront cachea el contenido estático (imágenes, CSS, JavaScript), lo que reduce las peticiones al origen y disminuye costos. También ofrece protección contra ataques DDoS de forma nativa.

Amazon S3 - Frontend y almacenamiento

Utilizo S3 para dos propósitos, primero, alojo el frontend de la aplicación (archivos HTML, CSS, JavaScript) en un bucket configurado para hosting estático. Esto es mucho más simple y económico que mantener servidores web. Segundo, tengo otro bucket para almacenar imágenes y anexos que los usuarios suben. S3 ofrece durabilidad del 99.999999999%, lo que significa que los archivos están prácticamente a prueba de pérdidas. También permite configurar políticas de ciclo de vida para mover archivos antiguos a clases de almacenamiento más baratas.

Amazon API Gateway - Puerta de entrada a las APIs

API Gateway actúa como punto de entrada único para todas las peticiones de la aplicación. Recibe las llamadas HTTP, valida los parámetros, aplica autenticación mediante tokens JWT, y enruta cada petición a la función Lambda correspondiente. Una ventaja importante es que maneja automáticamente el throttling, limitando el número de peticiones por segundo para evitar abusos. También genera documentación de la API automáticamente en formato OpenAPI, lo que facilita la integración con otros sistemas.

AWS Lambda - Lógica de negocio distribuida

Dividí la lógica de la aplicación en cuatro funciones Lambda independientes, cada una con una responsabilidad clara:

- **Lambda Autenticación:** Maneja todo lo relacionado con usuarios: registro, login, validación de tokens, y gestión de sesiones. Se integra directamente con Cognito para verificar credenciales.

- **Lambda CRUD Datos:** Ejecuta las operaciones básicas sobre DynamoDB (crear, leer, actualizar, eliminar). Esta función es la que más se invoca porque maneja todas las consultas de datos de la aplicación.

- **Lambda Procesamiento:** Contiene la lógica de negocio más compleja: cálculos, validaciones, transformaciones de datos, y cualquier proceso que requiera más tiempo de ejecución.

- **Lambda Anexos:** Gestiona la subida y descarga de archivos, esto genera urls firmadas temporales que permiten a los usuarios subir archivos directamente a S3 sin pasar por la función Lambda, lo que mejora el rendimiento y reduce costos.

Esta separación tiene varias ventajas. Si hay muchos usuarios autenticándose, solo la función de autenticación crea más instancias. Además, facilita el mantenimiento porque cada función es pequeña y enfocada en una tarea específica.

Amazon DynamoDB - Base de datos NoSQL

Elegí DynamoDB porque es una base de datos completamente administrada que no requiere configurar servidores, aplicar parches, o gestionar backups manualmente. Ofrece latencias consistentes de milisegundos incluso con millones de peticiones. Configuré el modo on-demand, que ajusta automáticamente la capacidad según el tráfico real, por lo que solo pago por las lecturas y escrituras que realmente ocurren. DynamoDB replica los datos automáticamente en tres zonas de disponibilidad, garantizando alta disponibilidad sin configuración adicional.

Organicé los datos en dos tablas: una para información transaccional (usuarios, sesiones, registros de actividad) y otra para metadatos de archivos (nombre, tamaño, fecha de subida, referencias a S3). Esta separación mejora el rendimiento porque cada tabla se optimiza para su patrón de acceso específico.

AWS Cognito - Autenticación de usuarios

Cognito gestiona todo el ciclo de vida de los usuarios, esto permite registro con verificación de email, login con contraseñas seguras, recuperación de contraseña, y autenticación multifactor (MFA). También soporta federación con proveedores externos como Google o Facebook. Cuando un usuario se autentica correctamente, Cognito genera un token JWT que la aplicación incluye en todas las peticiones subsiguientes. Las funciones Lambda validan este token antes de procesar

cualquier operación, asegurando que solo usuarios autorizados accedan a los datos.

Servicios de soporte y observabilidad

AWS IAM - Control de accesos

Cada función Lambda tiene un rol de IAM con permisos específicos y limitados. Por ejemplo, la función de autenticación solo puede leer de Cognito, pero no puede acceder a S3. La función de anexos puede escribir en el bucket de archivos, pero no puede modificar la base de datos. Este principio de mínimo privilegio reduce significativamente el riesgo de seguridad, porque si una función se ve comprometida, el daño potencial está limitado a sus permisos específicos.

Amazon CloudWatch - Monitoreo y logs

CloudWatch recopila automáticamente logs de todas las funciones Lambda y métricas de rendimiento. Puedo ver en tiempo real cuántas veces se ejecutó cada función, cuánto tiempo tardó cada invocación, cuánta memoria consumió, y si hubo errores. Configuré alarmas que me notifican cuando la tasa de errores supera el 5% o cuando la latencia excede los 3 segundos. Esto permite detectar y resolver problemas rápidamente antes de que afecten a muchos usuarios.

AWS X-Ray

X-Ray es fundamental para entender qué ocurre en una arquitectura distribuida. Cuando un usuario hace una petición, X-Ray rastrea el recorrido completo: API Gateway → Lambda → DynamoDB → S3. Genera un mapa visual que muestra cuánto tiempo tomó cada componente y dónde están los cuellos de botella. Si una petición falla, puedo ver exactamente en qué punto ocurrió el error y qué lo causó.

Conclusiones

La arquitectura serverless propuesta ofrece una solución moderna, escalable y economica para aplicaciones web. Al eliminar la gestión de servidores y aprovechar

servicios completamente administrados, el equipo puede enfocarse en crear valor para el negocio en lugar de mantener infraestructura.

AWS Lambda, API Gateway y DynamoDB forman la columna vertebral de una aplicación que puede escalar desde cero usuarios hasta millones sin cambios arquitectónicos significativos. La seguridad está incorporada en cada capa, desde WAF en el perímetro hasta cifrado de datos en reposo.

El modelo de costos basado en uso real hace que esta arquitectura sea especialmente atractiva para startups y proyectos nuevos, donde el tráfico inicial es bajo, pero puede crecer rápidamente. No hay inversión inicial en servidores que podrían quedar subutilizados.

Con las herramientas de monitoreo y observabilidad adecuadas, el equipo tiene visibilidad completa del sistema y puede detectar y resolver problemas rápidamente. El pipeline de CI/CD automatizado garantiza que los cambios se desplieguen de forma segura y se puedan revertir si es necesario. Esta arquitectura representa las mejores prácticas actuales para aplicaciones cloud nativas, balanceando rendimiento, seguridad, costos y facilidad de operación.