Respuesta al Reto Técnico Ingeniero Cloud.

Autor: Carlos Torres
email: carlosqba@gmail.com

Preguntas teóricas:

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

R/
En el contexto de la computación "nube" se refiere a la entrega bajo demanda de recursos de la tecnología de la información (TI) a través de una red. La nube ofrece servicios y aplicaciones de informática y para ello cuenta con hardware, software, redes y almacenamiento. La propiedad de los recursos, la manera en como se implementan y administran da lugar a tres modelos de nubes: públicas , privadas e híbridas. A continuación analizamos algunas de sus diferencias.

Nube pública
Está bajo la propiedad y administración de proveedores de servicios en la nube de terceros, como Amazon Web Services (AWS), Microsoft Azure, Google Cloud, IBM Cloud. Son altamente escalables, puedes aumentar o disminuir fácilmente los recursos que necesitan según sea necesario. Además se paga por lo que se utilice, con una buena configuración se puede ahorar bastante dinero. Las empresas que utilizan nubes publicas no tienen que invertir en hardware o software propios, lo que puede ahorrarles mucho dinero al eliminar la necesidad de invertir en infraestructura y personal de TI.

Nube privada
Está bajo la propiedad de una sola organización, los recursos informáticos se alojan en un centro de datos local o de un proveedor de servicios gestionado. Proporcionan mayor control, personalización y seguridad de los datos, pero son muy costosas y difícil de mantener.

Nube híbrida
Combina al menos un entorno de nube privado con al menos uno público. Permite aprovechar los recursos y servicios de ambos entornos y elegir el más apropiado según las necesidades.

2. Describa tres prácticas de seguridad en la nube.

R/
En la nube, la seguridad se aborda en el modelo de responsabilidad compartida, este modelo establece que el proveedor del servicio y el cliente tienen roles complementarios para garantizar la protección de los datos y sistemas. Los proveedores de servicios en la nube son responsables de que la infraestructura de la nube sea segura y esté disponible para su uso, esto incluye la protección de las localizaciones físicas de los centros de datos, mantenimiento de servidores y redes, actualizaciones del software base y corrección de vulnerabilidades en la infraestructura.

El cliente tiene la responsabilidad de proteger sus datos, aplicaciones y configuraciones dentro de la nube, esto incluye entre otras:

- La gestión de usuarios, roles y permisos para evitar accesos no autorizados.
- Implementar herramientas para detectar actividades sospechosas o vulnerabilidades.
- Realizar copias de seguridad periodicamente.
- Administrar correctamente la configuración de seguridad (firewalls, grupos de seguridad, etc.).
- Utilizar software actualizado y libre de vulnerabilidades.

A continuación tres prácticas de seguridad a aplicar por parte del cliente:

1. Aplicar el enfoque de privilegios mínimos.
   En este enfoque el administrador otorga a los usuarios solo los permisos que necesitan, pues al limitar los derechos de acceso, se minimiza el riesgo de una filtración de datos. En esencia, se busca lograr un equilibrio entre seguridad y funcionalidad. Si bien los usuarios no deben tener permisos excesivos, deben tener todos los permisos necesarios para realizar sus tareas y nada más. La revisión y actualización periódicas de las políticas de permisos garantizan este equilibrio, lo que convierte al enfoque de privilegios mínimos en un proceso dinámico que evoluciona según las necesidades de la organización. También implica el uso del control de acceso basado en roles, que asigna permisos según roles en lugar de usuarios individuales, esto simplifica la gestión del acceso y facilita el seguimiento de quién tiene acceso a qué recursos.

2. Gestionar los eventos de seguridad.
   Se debe simplificar la detección de amenazas con el registro y monitoreo centralizados. Con todos los registros y datos de monitoreo disponibles en un sistema centralizado, se pueden detectar comportamientos inusuales, como múltiples intentos fallidos de inicio de sesión o transferencias de datos inesperadas, que podrían indicar una posible brecha de seguridad. Los proveedores disponen de las herramientas adecuadas, ejemplo Amazon CloudWatch ofrece monitoreo en tiempo real y agrega registros de todos los recursos de AWS en un panel unificado.

3. Realizar copias de seguridad de los datos periódicamente.
   Las copias de seguridad periódicas garantizan la resiliencia operativa y minimizan el tiempo de inactividad. Y en caso de vulneraciones o riesgos de seguridad, aceleran la recuperación de datos. Las nubes ofrecen automatizados de copias de seguridad, las organizaciones pueden automatizar la duplicación y el respaldo de datos importantes, ejemplo AWS Backup permitiendo una recuperación rápida en caso de fallos del sistema, borrados accidentales o incidentes imprevistos.

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

Entendemos por IaC como La infraestructura como código (IaC del inglés Infrastructure as Code), es un método de gestión y aprovisionamiento de recursos en la nube, esta permite gestionar y preparar la infraestructura a través de código, en lugar de hacerlo manualmente. En este método se crean archivos de configuración que contienen las especificaciones que necesita la infraestructura. Podemos citar la reducción de costos, el aumento en la velocidad de implementación, la disminución de errores y la uniformidad de la infraestructura entre sus principales ventajas.

Dos de las herramientas más utilizadas son Terraform y AWS CloudFormation, a continuación sus principales características.

1. **Terraform** es una herramienta de infraestructura como código (IaC) que permite construir, modificar y versionar la infraestructura de forma segura y eficiente. Esto incluye instancias de cómputo, almacenamiento, redes, entradas DNS, grupos de autoescalado, etc. en un gran número de proveedores. Características principales de Terraform:

   - Infraestructura como código: utiliza el lenguaje de configuración de alto nivel

HashiCorp Configuration Language (HCL) para describir la infraestructura en archivos de configuración declarativos y legibles. Terraform permite crear una plantilla que se puede versionar, compartir y reutilizar.

- Planes de ejecución: una vez que el usuario describe la infraestructura, Terraform crea un plan de ejecución. Este plan describe lo que Terraform hará y solicita su aprobación antes de iniciar cualquier cambio en la infraestructura. Este paso le permite revisar los cambios antes de que Terraform realice cualquier modificación en la infraestructura, incluyendo crearla, actualizarla o eliminarla.
- Gráfico de recursos: Terraform genera un gráfico de recursos, ofreciendo a los usuarios una mayor comprensión de su infraestructura.
- Automatización de cambios: Terraform puede implementar conjuntos de cambios complejos en la infraestructura prácticamente sin intervención humana. Cuando los usuarios actualizan los archivos de configuración, Terraform detecta los cambios y crea un plan de ejecución incremental que respeta las dependencias.
- Terraform Registry: es un repositorio público de proveedores y módulos para desplegar de forma rápida configuraciones de infraestructuras comunes. Es soportado por HashiCorp, los diferentes proveedores (Microsoft, Google, AWS, etc.) y comunidades de desarrolladores.
- Plugins de proveedores: contienen todo el código necesario para autenticar y conectarse a un servicio específico, generalmente de un proveedor de nube pública, en nombre del usuario. Terraform es compatible con más de 200 proveedores de nube, entre ellos AWS, Azure , IBM , Google Cloud.
- Flujo de trabajo: la metodología para trabajar con Terraform es bastante simple.

    1. Escribir el código HCL en ficheros declarativos planos que definan nuestra estructura. Se ha de seleccionar un proveedor sobre el cual vamos a desplegar la infraestructura y a continuación definir los recursos.
    2. Inicializar (terraform init) el entorno de Terraform.
    3. Previsualizar (terraform plan) el estado final de nuestra infraestructura antes de aplicarse de forma definitiva. Permite anticiparnos a posibles errores, hacer comprobaciones, etc.
    4. Aplicar (terraform apply) los cambios de infraestructura de forma definitiva. Terraform analizará cómo alcanzar el estado de infraestructura deseado de la manera más óptima. El modo en el que gestiona su interacción con las APIs de cada proveedor (Azure, AWS, GCP, etc.) es transparente para nosotros.

2. **AWS CloudFormation** es la herramienta interna de IaC de Amazon Web Services (AWS) que permite a los usuarios crear, implementar y administrar infraestructuras mediante la definición de una plantilla de recursos. CloudFormation proporciona una forma de configurar y administrar recursos de forma segura y repetible. Podemos destacar las siguientes características:

    - Infraestructura como código: utiliza JSON o YAML para describir la infraestructura en archivos de configuración declarativos y legibles, estos archivos se conocen como plantillas. Puede crear plantillas mediante uno de los siguientes métodos:
      a) AWS Infrastructure Composer y AWS CloudFormation Designer: interfaz visual para diseñar plantillas.
      b) Editor de texto: escriba plantillas directamente en la sintaxis JSON o YAML.
      c) Generador de IaC: genere plantillas a partir de los recursos aprovisionados en su cuenta que CloudFormation no administra actualmente.
    - Planes de ejecución: una vez que el usuario describe la infraestructura, CloudFormation

crea un plan de ejecución. Con ChangeSets, puede ver antes de la ejecución, la versión preliminar de los cambios propuestos que CloudFormation pretende hacer en la infraestructura y los recursos de aplicación para que la implementación resulte tal como se la planificó. Este paso le permite revisar los cambios antes de que se realice cualquier modificación en la infraestructura, incluyendo crearla, actualizarla o eliminarla.

- Gráfico de recursos: El conjunto de cambios de AWS CloudFormation permite previsualizar de qué manera los cambios propuestos a una pila (colección de recursos de AWS, que puede administrar como una única unidad, definidos en un template) podrían afectar a los recursos en ejecución, por ejemplo, para verificar si los cambios eliminarán o reemplazarán un recurso crítico. CloudFormation implementa los cambios en la pila solo después de que usted decida ejecutar el conjunto de cambios.
- Automatización de cambios: AWS CloudFormation puede implementar conjuntos de cambios complejos en la infraestructura prácticamente sin intervención humana. Cuando los usuarios actualizan los archivos de configuración, AWS CloudFormation detecta los cambios y crea un plan de ejecución incremental que respeta las dependencias.
- CloudFormation Registry: permite a los usuarios administrar extenciones. Las extensiones incluyen tipos de recursos, módulos y hooks de AWS y de terceros, así como sus propias extensiones personalizadas.
- Flujo de trabajo: la metodología para trabajar con CloudFormation Command Line Interface es bastante simple.

1. Definir la plantilla de recursos, ya sea en JSON o en YAML.
2. Validar el template (aws cloudformation validate-template [parámetros]).
3. Se puede visualizar los cambios a aplicar utilizando AWS Infrastructure Composer y AWS CloudFormation Designer.
4. Crear la pila (aws cloudformation create-stack [parámetros]), esto es crear la infraestructura de forma definitiva. CloudFormation analizará cómo alcanzar el estado de infraestructura deseado de la manera más óptima.

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

Las métricas de la nube son mediciones cuantitativas que proporcionan visibilidad del comportamiento de los recursos y servicios en la nube. Estas métricas abarcan diversos aspectos de la nube, e incluyen métricas de rendimiento, disponibilidad, capacidad, seguridad, red, y costos, entre otras, a continuación aparecen las que considero esenciales.

- Utilización de la CPU: es la relación entre la utilización real de una CPU y la utilización total posible de una CPU. Su seguimiento revelará si una CPU está limitando el rendimiento debido a una sobreutilización o subutilización, podemos decidir si aumentar o disminuir las caracaterísticas del CPU.
- Utilización de memoria: ayuda a medir el uso de memoria, una utilización de memoria constantemente alta puede requerir ampliar la capacidad de memoria y disminuirla en caso contrario.
- Solicitudes por minuto: indica cuántas solicitudes recibe una aplicación en la nube por minuto. Es fundamental supervisar cómo y cuándo acceden los usuarios a la aplicación para poder escalar sus recursos en la nube y satisfacer la demanda, garantizando así un rendimiento óptimo.
- Utilización del disco: permite monitorear el volumen de disco, su capacidad de almacenamiento para determinar si es suficiente para sus cargas de trabajo.
- Latencia: mide el tiempo transcurrido entre el envío de una solicitud por parte del cliente y

la respuesta del proveedor de la nube. Una latencia alta puede afectar negativamente la experiencia del usuario.

- Métricas de solicitudes por minuto: no solo miden el rendimiento de la nube, sino también los riesgos. Un número elevado de solicitudes por minuto puede indicar una amenaza continua, como un ataque de denegación de servicio distribuido (DDOS).
- Pérdida de paquetes: mide el porcentaje de pérdida de paquetes de red entre el origen y el destino. La pérdida de paquetes puede causar latencia y congestión de la red cuando un protocolo de internet retransmite los datos.

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

Docker es una solución de virtualización, de código abierto que permite crear, implementar y gestionar aplicaciones en contenedores livianos para que las aplicaciones puedan funcionar de manera eficiente en diferentes entornos de forma aislada.

Los componentes principales de la arquitectura de Docker incluyen:

- Demonio de Docker : El demonio de Docker (o "motor") es el elemento central de la arquitectura de Docker. Es un proceso en segundo plano que gestiona, construye y ejecuta contenedores Docker.

- Cliente Docker : El cliente Docker es la interfaz que se utiliza para interactuar con el demonio Docker. Permite a los usuarios crear y gestionar imágenes, contenedores y redes Docker.

- Registro de Docker : es el lugar donde se guardan las imágenes de Docker. Puede ser un registro público o privado. Por defecto se utiliza Docker Hub.

- Objetos Docker:

    5. Imágenes: consiste en un conjunto de instrucciones y archivos que permiten crear un contenedor desde cero. Se construyen a partir del dockerfile, un archivo de texto con un conjunto de comandos o instrucciones.
    6. Contenedores: contenedores son las unidades estructurales de Docker, que se utilizan para contener todo el paquete que se necesita para ejecutar la aplicación. Un contenedor de Docker es un entorno en tiempo de ejecución de las imágenes de Docker.
    7. Volúmenes: permite la persistencia y el intercambio de datos del contenedor. El volumen de Docker es el directorio del host de Docker montado dentro del contenedor que permite al contenedor escribir datos en los volúmenes del host.
    8. Red: Las redes Docker son redes virtuales que se utilizan para conectar múltiples contenedores. Permiten que los contenedores se comuniquen entre sí y con el sistema host.

6. Caso práctico

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

- Frontend: Una aplicación web que los clientes utilizarán para navegación.
- Backend: Servicios que se comunican con la base de datos y el frontend.
- Base de datos: Un sistema de gestión de base de datos que almacene información.
- Almacenamiento de objetos: Para gestionar imágenes y contenido estático.

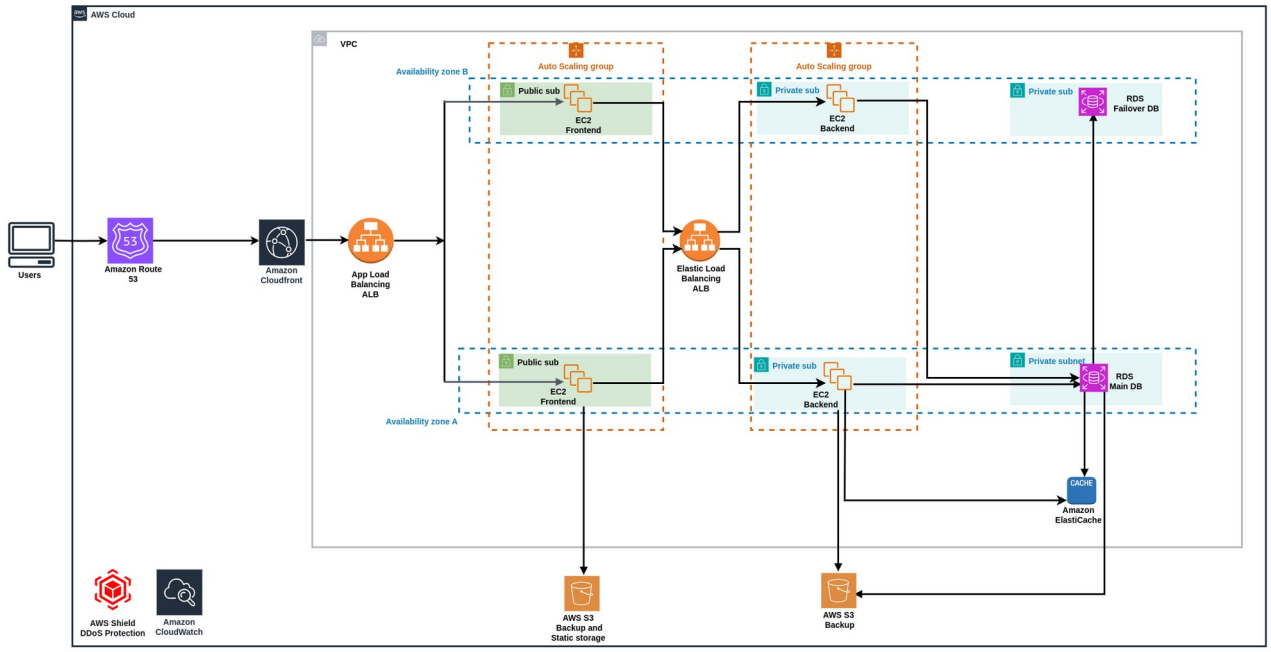
Diseño:

- Seleccione un proveedor de servicios de nube (Aws, Azure o GCP) y sustente su selección.
- Diseñe una arquitectura de nube. Incluya diagramas que representen la arquitectura y justifique sus decisiones de diseño (Utilice https://app.diagrams.net/).

R/

- Me decanto por AWS pues este proveedor ofrece una amplia variedad de servicios escalables que permiten crear y escalar aplicaciones de manera relativamente sencilla. Su red global de centros de datos le permite prestar servicios en cualquier parte del mundo, garantizando baja latencia y alta disponibilidad. Además, AWS ofrece un enfoque integral de seguridad incluyendo cifrado, firewall, autenticación multifactor copias de seguridad, protección contra ataque DdoS y un conjunto completo de herramientas para monitoreo y análisis. También se debe tener en cuenta que AWS puede resultar una opción mucho más económica pues permite a las empresas pagar solo por los servicios que utilizan. Una de las cosas que más ayudan a los especialistas en infraestructura es la abundante documentación y el soporte brindado por AWS

- La arquitectura propuesta aparece en la siguiente imagen, diseñada para ser implementada en AWS.



Las solicitudes de los usuarios se procesan en el servicio Amazon Route 53 (servicio de DNS) y son dirigidas hacia el servico de Amazon CloudFront..

Amazon CloudFront es un servicio web que agiliza la entrega de contenido web estático y dinámico a través de una red mundial de centros de datos que reciben el nombre de ubicaciones de borde. Cuando un usuario solicita contenido a CloudFront, la solicitud se redirige a la ubicación de borde que ofrece la mínima latencia. Si el contenido ya se encuentra en la ubicación de borde con menor latencia, CloudFront lo entrega inmediatamente. Si el contenido no se encuentra en dicha ubicación de borde, CloudFront lo recupera de un origen que haya definido, en nuestro caso sería el servidor de nuestro frontend. CloudFront llegará al servidor de origen a través del un balanceador de cargas [AWS Application Load Balancer (ALB)] el cual distribuye el tráfico entre los servidores de frontend desplegados en instancias del servicio EC2 (Elastic Compute Cloud). Para garantizar la alta disponibilidad todas las instancias EC2 están ubicadas en diferentes regiones de disponibilidad (AZ) y son desplegadas o eliminadas bajo demanda con la ayuda del servicio de autoescalado (Amazon EC2 Auto Scaling). Nuestro frontend delegará la entrega de contenido estático al servicio Amazon Simple Storage Service (Amazon S3). El frontend enviará solicitudes al backend, estas también serán gestionadas con un balanceador de carga.

El backend se comunica con la base de datos (BD) principal, esta puede ser gestionada por el servicio Amazon Relational Database Service (Amazon RDS) si la BD es relacional, existen otras soluciones (ejemplo Amazon DynamoDB para BD no relacionales). Por seguridad el backend y la BD se despliegan dentro de subredes privadas.

Se propone la implementación Multi-AZ de Amazon RDS pues ofrece mayor disponibilidad para las instancias de BD dentro de una misma región de AWS, pues los datos se replican sincrónicamente a una instancia en espera en una zona de disponibilidad diferente y en caso de una falla de la infraestructura, Amazon RDS realiza una conmutación por error automática al servidor en espera, minimizando la interrupción de sus aplicaciones sin intervención administrativa. Utilizamos ElastiCache para acelera el rendimiento de la BD, ofrece un tiempo de respuesta de microsegundos y una disponibilidad del 99,99 %.

Hacemos copias de seguridad de todo el proyecto en buckets de S3, este servicio de almacenamiento ofrece escalabilidad, disponibilidad de datos, seguridad y rendimiento.

Proponemos Amazon CloudWatch para recabar métricas y hacer un seguimiento de las mismas. Este servicio puede monitorear recursos de AWS tales como instancias Amazon EC2 e instancias de base de datos de Amazon RDS, se puede utilizar para obtener visibilidad de todo el sistema sobre la utilización de recursos, el rendimiento de las aplicaciones y el estado de funcionamiento.