Reto Técnico Ingeniero Cloud.

Preguntas teóricas.

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

La **nube pública** es un modelo en el cual los recursos como redes virtuales, máquinas virtuales y almacenamiento son compartidos por diferentes organizaciones, pero dichos recursos son administrados y propiedad del proveedor de nube, debemos tener en cuenta que aunque los recursos como compute y almacenamiento son compartidos, los datos se mantienen separados.

Mientras que, la **nube privada** brinda un ambiente dedicado a una organización en específico, es decir, que son accesibles para un solo cliente, sin embargo, los recursos deben estar en un datacenter propio o de un proveedor, por este aislamiento ofrece un mayor control de seguridad ya que en este modelo se protegen los datos confidenciales de maner más sólida.

Y finalmente, la **nube híbrida** es una combinación de los dos mundos de la nube pública y nube privada, lo cual conecta el data center en premisa de la organización con la nube pública y así tener un control de que la data sensitiva se quede en premisas para cumplimiento de PCI y que recursos o data no crítica pueda ser alojada en la nube pública para tener mejor escalabilidad.

2. **Describa tres prácticas de seguridad en la nube.**

Uso de IAM:

Esta práctica es crucial para mantener la integridad de los entornos en nube, ya que con la solución de IAM con RBAC es esencial para mantener un control sobre los accesos de usuarios a los recursos de nube. Esto nos permite definir roles, politicas de accesos condicionales como la obligación de tener MFA, autorización especifica basada en las funciones y responsabilidades del usuario, y con esto cumplimos con las buenas prácticas de least privileges.

Configuración segura de la Infraestructura y la red.

El despliegue de la infraestructura debe realizarse mediante templates seguros de IaC, asi evitando tareas manuales que puedan incurrir en errores que se materialicen con brechas de seguridad, puntos de auditoria y costos. Para esto se implementan políticas como código en el cloud, para poder establecer el cumplimiento de los estándares de seguridad y gobernanza. Para la segmentación de red, se debe usar subredes privadas, firewalls y

grupos de seguridad de red para tener un enfoque zero trust en la red, como también el uso de private endpoints para evitar la exposición de servicios importantes para la organización.

Protección de datos.

Es crucial encriptar los datos tanto cuando está guardado o en reposo y mientras está siendo transferida, esto con la misión de prevenir accesos no autorizados o que los datos sean interceptados y, mantener los datos de manera segura y confidencial para garantizar la integridad de los mismos.

3. **¿Qué es IaC, y cuales son principales beneficios?, mencione dos herramientas de IaC y sus principales caracteristicas.**

IaC es la práctica de administrar y aprovisionar infraestructura computacional en nube pública, híbrida y en premisas, como servidores, bases de datos, redes y más, a traves de plantillas escritas que definimos.

Beneficios de IaC:

- Reducción de errores humanos debido a la manualidad.
- Incrementar la productividad más rápido al desplegar la infraestructura.
- Mejor control al ser versionado con el source control e integración con CI pipelines y, con esto hacer rollback de forma más rápida y eficiente.
- Prevensión del drift en los ambientes y con esto lograr que los diferentes ambientes estén homologados.
- Entornos más fáciles de auditar.

Herramientas de IaC:

AWS CloudFormation. Herramienta nativa de AWS.

Caracteristicas de Cloud Formacion:

- Integración con todos los servicios de AWS al ser nativa de este proveedor de nube.
- Soporta el drift detection.
- Compatibilidad multi-cuenta y multi-región
- Gestión de stacks creados.
- Integración con CI/CD

Terraform. Es una herramienta open source que es tipo declarativa y con soporte multinube.

Caracteristicas de Terraform:

- Soporte multinube como Azure, AWS, GCP y más.
- Cuenta con un código modular y reusable.
- Permite planear y revisar los cambios en la infraestructura antes de aplicarlo en los ambientes.

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

Metricas de Rendimiento.

En este apartado contamos con métricas como disponibilidad para asegurar que las aplicaciones y servicios se encuentren disponibles para los clientes. Latencia para medir y detectar si hay algún tiempo de retraso en nuestra red la cual puede afectar la experiencia de usuario y materializarse como un incidente. Monitoreo de uso de recursos como memoria, CPU, disco y red para poder identificar los picos que puedan convertirse en cuellos de botella. Tiempo de respuesta para saber la velocidad de respuesta de la aplicación ante las solicitudes de los usuarios.

Métricas de Seguridad.

La detección de amanezas nos sirve para monitorear los eventos en tiempo real para identificar y mitigar ciertos ataques. Monitorear los controles de acceso para verificar la autorización.

Métricas de costos.

Esto nos permite monitorear la distribución de costos que tenemos en la nube en los diferentes recursos que tenemos desplegados nuestra cuenta o suscripción y con esto podemos medir la eficiencia con la optimización de los gastos para los recursos que apliquen.

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

Docker es una plataforma/herramienta de código abierto que nos brinda la facilidad de automatizar y empaquetar una aplicación como contenedores, estos contenedores ya vienen con el código de la aplicación, las librerias, runtime y dependencias que necesita para correr.

Componentes principales.

- **Docker Engine.**
- **Docker image.**
- **Docker Containers.**
- **Docker Registry.**
- **Dockerfiles.**

- **Volúmenes.**

6. **Caso práctico.**

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

- Frontend: Una aplicación web que los clientes utilizarán para su navegación.
- Backend: Servicios que se comuniquen 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:

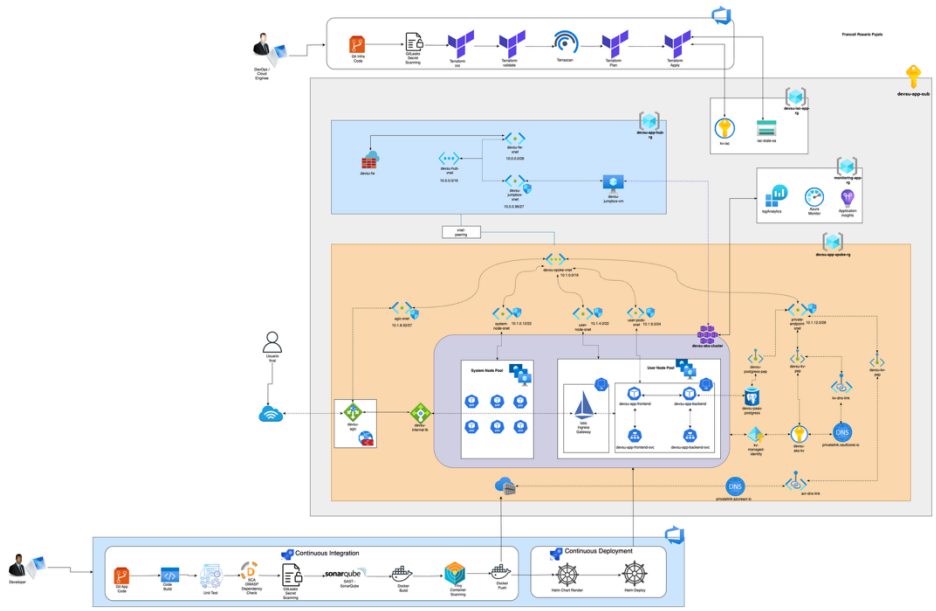
- Seleccione un proveedor de servicios de nube (AWS, Azure o GCP) y sustente su selección.
- Diseñe una arquitectura de nube. Incluya diagramas que representen la arquitectura y justifique su decisión.

Elección de Nube.

He elegido **Azure** debido a que ya el cliente contaba con recursos en dicha nube, como su arquitectura de redes virtuales y una conexión VPN con su data center en premisas, como también, su gestión de identidades y accesos internos IAM. Esto es una ventaja para tener una integración más sencilla y transparente, asi evitamos la fricción en el despliegue. Esta elección aprovecha de una mejor manera la integración de nuestro ecosistema.

Diseñe una arquitectura de nube. Incluya diagramas que representen la arquitectura y justifique su decisión.

Diagrama:



Justificación.

Despliegue de la Infrastructura con Terraform en un pipeline de Azure Devops.

Se contempla desplegar la infraestructura utilizada en este diseño mediante un pipelien, ya que nos permite orquestar la ejecución de Terraform de una forma auditable, ejecutando validaciones y escaneos de seguridad para detectar misconfigurations, como también vulnerabilidades.

Ciclo de vida del código con CI/CD pipelines de Azure Devops.

No hay mejor manera de controlar y contruir tu aplicación con flujo moderno, seguro y ágil que es lo que obtenemos de los pipelines. Por ende, nuestros contenedores son construidos de la siguiente manera; utilizamos pruebas unitarias para garantizar que el código cumple con lo propuesto, SCA con OWASP Dependency Check para analizar las dependencias del proyecto, para verificar si hay vulnerabilidades, GitLeaks para escanear el código en busca de secretos hadcoded, SAST con SonarQube para garantizar que el código cumple con los Quality Gates, detectar bugs y code smell, Docker build para construir nuestro contenedor, Trivy para que nos sirva de guardian al escanear nuestro contenedor antes construido, docker push para enviar nuestro contenedor encapsulado al un ACR. Para el despliegue continuo utilizamos Helm Chart Render para armar nuestro chart con los valores correspondientes por cada ambiente y Helm deploy para desplegar o actualizar los artefactos de nuestra aplicación.

Topología Hub & Spoke:

Utilicé este tipo de topología ya que aborda la necesidad de centralizar la seguridad, facilita la administración al tener los recursos segmentados y optimiza la conectividad de los recursos. En este diagrama en la división perteneciente al Hub, se encuentran Azure Firewall, las vnets y subnets de hub, como también el jumpbox que se utilizará para conectarse al clúster de kubernetes. En su contraparte, el Spoke, alojé las cargas de trabajo que en conjunto hacen que el aplicativo funcione de manera adecuada, aislando y facilitando la segmentación, como también limitando la exposición de la aplicación en cuestión.

Se implementó un vnet peering para comunicación entre ambas partes.

AKS con Istio, Application Gateway Ingress Controller, PostgreSQL y Keyvaul con endpoints privados.

Decidí utilizer un orquestador de contenedores como AKS para alojar el frontend como el backend, asi tener cubierta la escalabilidad que a medio-largo plazo se ve a requerir, ya que se irán desplegando nuevos aplicativos que requerirán un ambiente escalable. Se le añadió Istio para tener esa capa de control de tráfico, como también la visibilidad de los servicios y como interactúan entre si.

El frontend es expuesto al público por un AGIC con la capa de seguridad de un WAF, para garantizar la seguridad, la encriptación con TLS y reglas de capa 7.

Para almacenar los datos que vienen desde el Backend se está utilizando una base de datos PaaS con alta disponibilidad, para un mejor ciclo de vida del aplicativo, ya que garantiza un mantenimiento administrado y sobre todo algo muy importante, el uso de esta BD sin exponerla a internet, integrandola con private link, mediante un private endpoint.

En este caso el backend obtiene las credenciales y secretos inyectadas desde el keyvault, evitando secrets o variables de entorno hardcoded.

Observabilidad y monitoreo con Azure Monitor, Log Analytics y Application insights.

Algo sumamente importante en cualquier arquitectura es la observabilidad y monitoreo, lo cuál añadí en este diseño, teniendo azure monitor para recopilar las métricas y logs para tener una observabilidad centralizada de los recursos, Log analytics para almacenar los logs y application insight para tener esa capa de monitoreo a nivel de la aplicación.