Respuesta pregunta 01:

- La nube pública es una nube gestionado por algún proveedor de nube que brinda sus servicios de infraestructura, servicios y/o plataformas para cualquier usuario, en cualquier lugar del mundo y bajo sus tarifas; mientras que, la nube privada se diferencia en que es implementado, gestionado y manejado directamente por y para una empresa privada solamente, en el cual brinda los servicios de infraestructura, automatización y demás servicios de nube dentro de su misma organización, para sus partners y áreas dentro de su organización. La nube híbrida en cambio, es una combinación entre nube privada y pública, y se diferencia en que es un modelo de nube donde las empresas y/u organizaciones gestionan e implementan su propia nube privada, y lo interconectan hacia servicios de nubes públicas (por ejemplo; AWS, Azure, Google, etc) para aprovechar las tecnologías y manejo de costos de ambos mundos (Nubes privadas y públicas).

Respuesta pregunta 02:

- *Práctica 01: Implementar autenticación para acceso a los recursos*. Se refiere a implementar y gestionar el acceso a los recursos para usuarios y aplicaciones a través de diferentes métodos de autenticación, como, por ejemplo: MFA, Token dinámicos, políticas de contraseña robustas, etc.
- *Práctica 02: Cifrar los datos en tránsito y en reposo*. Para todos los tipos de datos que se manejen en nubes públicas y privadas, se debe implementar cifrado de datos en reposo como por ejemplo, las Bases de datos, los discos de máquinas virtuales, etc.; y protección en tránsito a través de protocolos seguros como SSH, TLS y HTTPS.
- *Práctica 03: Realizar análisis de vulnerabilidades para los servicios expuestos en nube*. Realizar análisis de vulnerabilidades para todos los servicios que son expuestos hacia internet y/o hacia redes privadas, para asegurar que todos los sistemas no cuenten con vulnerabilidades y/o huecos de seguridad que podrían ser explotados.

Respuesta pregunta 03:

- IaC es el acrónimo de “Infrastructure as Code” que significa Infraestructura como código. A través de esta tecnología se puede implementar, gestionar y mantener actualizada la infraestructura en nubes públicas y privadas a través de líneas código. Los principales beneficios son:
  - Reutilización de código, que sirve para volver a usar código en diferentes entornos para mantener la infraestructura semejante entre ellos; como por ejemplo; poder usar el mismo código de infraestructura para entornos de Desarrollo, Certificación o QA, Testing y Producción.
  - Rastrear y mantener los cambios en la infraestructura, para poder saber qué cambios se hicieron en diferentes instantes de tiempo a través del versionamiento de código con Git.
  - Evitar errores humanos, ya que al utilizar código, se puede crear infraestructura de manera más segura, a diferencia de hacerlo a través de “clicks” en las consolas de gestión vía Web o interfaz gráfica.
  - Rapidez en la implementación de la infraestructura. A través de IaC se puede implementar toda una infraestructura completa mucho más rápido que haciéndolo de manera gráfica.
- Ejemplos de herramientas:
  - Terraform:
    - ✓ Se caracteriza por ser una de las principales herramientas de IaC que existen en el mercado.
    - ✓ El uso de módulos es un beneficio para ahorrar la escritura de código.
    - ✓ No se necesita ser experto en programación ni programador para entender el lenguaje HCL, el cual es utilizado en Terraform. Es muy sencillo de interpretar

- Pulumi
  - ✓ Se caracteriza por ser utilizado con librerías de lenguajes de programación existentes, como .Net, Python, etc. Es decir, si es que algún desarrollador tiene experiencia con estos lenguajes, la adopción de IaC se hace más sencillo ya que se utilizan las mismas sintaxis y forma de desarrollo.

Respuesta pregunta 04:

Se pueden utilizar métricas sobre el rendimiento de los servicios de procesamiento como servidores, máquinas virtuales, contenedores, etc; como, por ejemplo: Consumo de CPU, Consumo de RAM, latencia de las conexiones, latencia en el tiempo de respuesta.

También podrían utilizarse métricas sobre los servicios publicados hacia el exterior como por ejemplo Balanceadores de carga, APIs, etc. Estas métricas pueden ser: Códigos o mensajes HTTP como 2XX, 4XX, 3XX, 5XX.

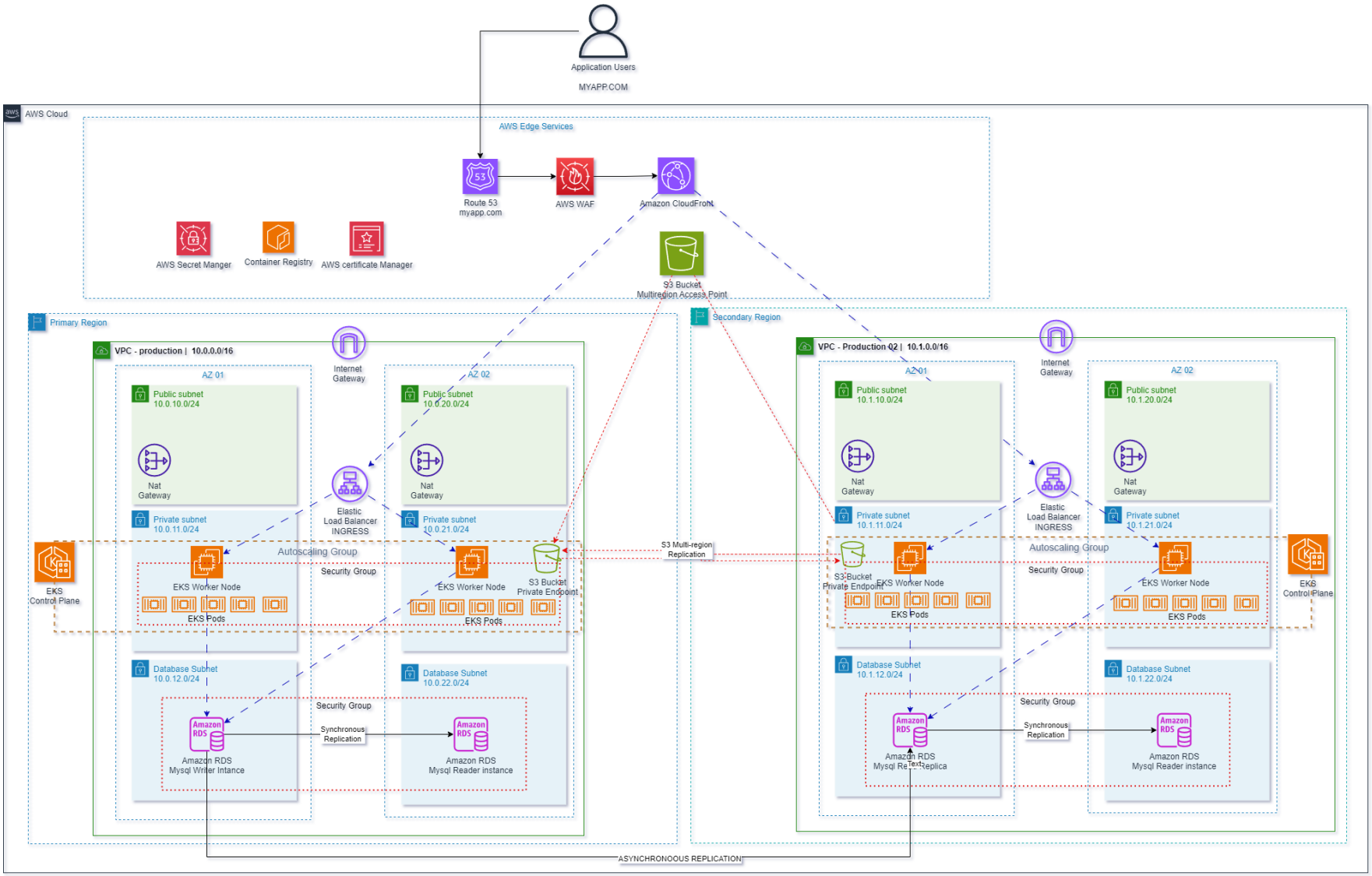
Podrían utilizarse también métricas y alertas de costo de servicios, para mantener los costos bajo el presupuesto asignado para los recursos en nube y no tener sorpresas al final del periodo de facturación.

Respuesta pregunta 05:

Docker es una tecnología que permite utilizar contenedores en sistemas operativos. Permite el aislamiento de procesos de contenedores para separar y gestionar los recursos de CPU, RAM, Disco y Red en éstos. Sus componentes principales son:

- *Docker Daemon (dockerd):* Es el demonio y/o proceso Docker que corre en cada servidor y se encarga de la gestión de los recursos como CPU, RAM, Disco y red para cada contenedor corriendo en el servidor. El demonio se encarga de administrar las imágenes, contenedores y volúmenes dentro del host.
- *Docker CLI:* Es la interfaz de líneas de comandos que permite interactuar con el demonio Docker.
- *Docker Registry*: Es el repositorio (privado y/o público) donde se gestionan las imágenes de contenedores para que puedan ser descargados y/o subidos.
- *Docker image*: Son las imágenes base que se utilizan para correr contenedores. Estas imágenes contienen la aplicación, librerías y dependencias necesarias para que puedan ser ejecutadas por el demonio de Docker. Las imágenes se obtienen de los repositorios (Docker registries) públicos y privados.

Respuesta pregunta 06:



Se presenta el diseño de la arquitectura para una aplicación nativa de nube utilizando el proveedor AWS.

Se seleccionó AWS debido a mi experiencia en arquitectura con los productos AWS y debido a la facilidad del uso de su documentación para todos sus servicios. AWS actualmente es el proveedor con mayor cantidad de puntos de presencia (PoP) en todo el mundo, capaz de entregar aplicaciones más rápido debido a la cercanía a los usuarios que se conectan desde cualquier país de cualquier continente. Al tener mayores PoP, se entregan las aplicaciones a los usuarios con baja latencia.

También se aprovecha la economía a escala de AWS para obtener los precios más bajos, ya que, al tener mayor cuota de mercado, los precios para adquirir sus servicios se hacen más económicos.

La arquitectura propuesta para este desafío presenta un diseño con alta disponibilidad entre 02 regiones en modo Active-Passive, es decir, la región primaria recibe todo el tráfico y la región secundaria está en stand-by, y será activada automáticamente cuando la región principal deje de responder o funcionar. Toda la redirección de tráfico se realiza a través de Route53. Para el ejemplo práctico y dependiendo las necesidades del cliente, las regiones podrían ser US, Europa o Asia.

Dentro de cada región se distribuyeron los servicios para la ejecución de aplicaciones y bases de datos en 02 zonas de disponibilidad como mínimo.

Se optó por utilizar el servicio EKS para la ejecución de las aplicaciones (frontend y backend) y Amazon RDS for Mysql para base de datos de la aplicación. Además de utilizar Amazon S3 para el almacenamiento de objetos.

El acceso a la aplicación de un usuario de la aplicación iniciará desde la resolución de nombre. En este caso, la aplicación es accedida a través de la url https://myapp.com; lo cual será resuelta a través de una zona DNS pública de Route53. Todo el tráfico que va hacia https://myapp.com irá de manera predeterminada hacia el servicio AWS WAF, en el cual se implementaron reglas de protección contra ataques basados en HTTP y HTTPS como por ejemplo: Cross site scripting, defacement, SQL injection, code injection, etc.

Luego de que el tráfico se inspecciona y se filtra, las solicitudes de objetos estáticos serán redirigidos hacia la red de PoP Amazon CloudFront, y basándose en la IP origen de la solicitud http, será entregado por el PoP más cercano al usuario. Si los objetos o recursos no son los más actualizados, Amazon CloudFront constantemente actualizará la versión de cada objeto para que pueda ser entregado con la mayor rapidez posible.

Para la ejecución de las capas Backend y FrondEnd, se utiliza un clúster EKS (Elastic Kubernetes Services) con 02 Worker Nodes en cada Zona de disponibilidad, con el fin de desplegar las aplicaciones y sus dependencias a través de contenedores. Los microservicios de Backend y Frontend serán aislados en el cluster EKS a través de NameSpaces. Las imágenes de los contenedores serán almacenadas en el servicio AWS Elastic Container Registry.

La capa frontend será publicada hacia internet a través de Ingress gestionados desde el cluster EKS, que, a través de la integración con AWS, genera Application Load Balancer para publicar las aplicaciones y las políticas de enrutamiento de Paths http y https. El ALB redistribuirá el tráfico entre los Pods de las 02 zonas de disponibilidad en cada región. Todo el tráfico HTTPS será asegurado a través de certificados digitales que serán gestionados y almacenados en el servicio AWS Certificate Manager.

Para el almacenamiento de objetos se optó por utilizar Amazon S3 con replicación multi-region, con el fin de mantener los objetos correctamente replicados cuando una de las regiones se vuelva indisponible. Se generaron también S3 buckts con private endpoint asociado a las subredes privadas donde se ejecutan las aplicaciones, con el fin de no exponer los buckets hacia internet.

Para el sistema de base de datos se utiliza Amazon RDS para Mysql, para generar un cluster de alta disponibilidad entre zonas y regiones, a través de instancias de Escritura que serán utilizadas como las instancias principales, y las instancias de Lectura que serán utilizadas solo para acceso de lectura y respaldo (failover) ante la falla de la instancia de Escritura.

Para que los servicios Backend puedan acceder a las bases de datos (lectura o escritura), se utiliza AWS Secret Manager, para gestionar y encriptar los secretos.

Como medida de seguridad, se generaron 03 subredes por cada zona de disponibilidad: Subnet pública, Subnet privada y DataBase subnet, para aislar el tráfico entre los mismos. Entre cada subnet, se generaron Security Groups para filtrar el tráfico solamente de los puertos necesarios.