**Prueba técnica Devops**

En este documento se describe la forma en la cual se puede deployar la Prueba técnica Devops

En AWS haciendo uso de buenas prácticas escalabilidad y alta disponibilidad, así como seguridad y resiliencia de la infraestructura.

Esta solución está dividida en 3 capas las cuales son las siguientes:

Layer 1 – Frontend

Layer 2 – App

Layer 3 – Data

Se describirá la solución de cada uno de los layers de la solución:

Nota: La solución esta segmentada en subredes distintas, donde hay 2 subredes públicas, solo para exponer servicios como HTTPS, 4 Subredes privadas donde en dos de ellas vivirá la BD y dos más donde vivirá el aplicativo.

**Layer 1 – Frontend**

**Route 53**: Se utilizará Amazon Route 53 como [servicio](https://aws.amazon.com/es/route53/what-is-dns/) DNS ya que es escalable y de alta disponibilidad en AWS.

**GuardDuty**: Servicio de detección de amenazas que supervisa de manera continua sus cargas de trabajo y la cuenta de AWS para detectar actividades maliciosas.

**Application Load Balancer**: Application Load Balancer actúa en la capa de aplicación, es decir, la séptima capa del modelo OSI. Una vez que el balanceador de carga ha recibido una solicitud, evalúa las reglas del listener por orden de prioridad con el fin de determinar qué regla se debe aplicar.

Se tendrán reglas para responder a 443 HTTPS donde se puede proporcionar un certificado SSL.

**Layer 2 – App**

**Aplicación Nodejs**: Con ejemplo para insertar datos en la base de datos, y obtener y mostrar los datos de dicha base de datos.

**Instancia EC2**: Se utilizará una instancia Ubuntu 20.04 LTS, para desplegar la aplicación Nodejs en contenedores. En esta instancia instalaremos Docker, sobre el que se crearán contenedores. También instalaremos el cliente MySQL en la instancia. El cliente MySQL es necesario para conectarse a la Instancia RDS de MyQSL para crear una tabla requerida.

**Nginx:** Se utilizará para habilitar HTTPS para la aplicación Nodejs de ejemplo y redirigir todas las solicitudes de los usuarios a la aplicación Nodejs. Actuará como un proxy inverso para redirigir las peticiones de los usuarios a la aplicación y ayudará a asegurar la conexión proporcionando la configuración para habilitar SSL/HTTPS.

**Crear aplicación NodeJs en Kubernetes.**

* Crear a Dockerfile
* Crear Kubernetes components

Abrir la aplicación Node.js en el navegador.

Una vez validado los pasos anteriores sacar una imagen de la EC2 para que sea la imagen semilla de nuestro Auto Scaling group.

**AWS Auto Scaling:** Ayuda a garantizar que cuenta con la cantidad correcta de instancias de Amazon EC2 disponibles para gestionar la carga de su aplicación. Crea colecciones de instancias EC2, denominadas grupos de Auto Scaling. Puede especificar el número mínimo de instancias en cada grupo de Auto Scaling y Amazon EC2 Auto Scaling garantizará que el grupo nunca tenga menos de esas instancias. Puede especificar el número máximo de instancias en cada grupo de Auto Scaling y Amazon EC2 Auto Scaling garantizará que el grupo nunca tenga más de esas instancias.

**IAM aws-auth:** El acceso al clúster mediante las entidades de AWS IAM está habilitado por el [Autenticador de IAM de AWS para Kubernetes](https://github.com/kubernetes-sigs/aws-iam-authenticator" \t "_blank), que se ejecuta en el control plane de Amazon EKS. El autenticador obtiene la información de la configuración del ConfigMap de aws-auth.

**ECR con GithubAcctions**

Configurar las acciones de GitHub para la construcción de la imagen y su envío a AWS ECR, esto con GitHub Actions.

**Layer 3 – Database**

**AWS RDS MyQSL**: Nuestros datos serán almacenados en AWS RDS MySQL. Donde este servicio vivirá en subredes destinadas a base de datos que serán privadas donde solo tendrá acceso el tráfico proveniente del Clúster de EKS.

Conclusión.

Para una mejor administración de la infraestructura podrá manejarse mediante Terraform.

Los principales beneficios que nos proporciona Terraform son:

* Admite crear infraestructuras en gran cantidad de proveedores.
* Nos permite administrar tanto infraestructuras grandes como una sola aplicación.
* Proporciona una sintaxis fácil y única que permite administrar casi cualquier recurso independientemente de la plataforma y servicio.
* Los ficheros de Terraform pueden ser compartidos y reutilizables para generar nuevos entornos y ser fácilmente exportables a otros clientes.
* Los modelos de datos pueden ser versionados, y de esta forma muy sencilla observar el progreso de nuestro servicio.
* También nos permite controlar los cambios en la infraestructura de manera simple y ágil.