Descripción General

Este proyecto tiene como objetivo principal facilitar la comunicación entre dos aplicaciones mediante el protocolo gRPC. Para lograr esto, se han desarrollado dos aplicaciones en Python y dos módulos de Terraform para gestionar la infraestructura en la nube de AWS.

Estructura del Proyecto

El proyecto está organizado en las siguientes carpetas y archivos:

- **cliente/:** Contiene el código del cliente.
- **mi_cliente.py:** Archivo principal del cliente.
- **proto/:** Contiene archivos protobuf.
- **mi_servicio.proto:** Define el servicio gRPC.
- **mi_servicio_pb2.py:** Archivo generado para mensajes protobuf.
- **mi_servicio_pb2_grpc.py:** Archivo generado para el servidor gRPC.
- **servidor/:** Contiene el código del servidor.
- **mi_servidor.py:** Archivo principal del servidor.
- **terraform/:** Contiene los módulos de Terraform.
- **networking/:** Módulo para gestionar la red.
- **eks/:** Módulo para desplegar en Amazon EKS.
- **main.tf:** Archivo principal de Terraform que utiliza los módulos anteriores.

Aplicaciones Python

Cliente (mi_cliente.py)

El cliente interactúa con el servidor gRPC para enviar solicitudes y recibir respuestas. Utiliza el archivo mi_servicio.proto para definir la estructura de los mensajes.

Servidor (mi_servidor.py)

El servidor proporciona servicios gRPC y responde a las solicitudes del cliente. También utiliza el archivo mi_servicio.proto para la definición del servicio.

Protocolo gRPC

Archivos Protobuf

mi_servicio.proto: Define el servicio gRPC y la estructura de mensajes.

mi_servicio_pb2.py: Archivo generado para mensajes protobuf.

mi_servicio_pb2_grpc.py: Archivo generado para el servidor gRPC.

Terraform

MODULO 1: Networking (terraform/networking/)

Este módulo gestiona la infraestructura de red, permitiendo el tráfico público y privado necesario para la comunicación entre las aplicaciones.

MODULO 2: EKS (terraform/eks/)

Este módulo despliega la aplicación en Amazon EKS, exponiéndola a través de un ingress y utilizando un balanceador de tipo application. Además, incluye integración con CI/CD en CodeBuild para facilitar la implementación continua.

Despliegue General (main.tf)

El archivo main.tf es el punto de entrada principal para Terraform. En este archivo, se llaman a los dos módulos anteriores y se muestra cómo utilizarlos en conjunto para desplegar la aplicación completa.

Diagrama de Arquitectura

Incluye un diagrama que representa la arquitectura completa del proyecto, mostrando la interacción entre las aplicaciones y la infraestructura en la nube.

Networking

El módulo de Networking tiene como objetivo establecer la infraestructura básica de red necesaria para el despliegue de aplicaciones en un entorno de Amazon Web Services (AWS). A continuación, se explica detalladamente cada recurso creado en este módulo.

Recursos Creados

VPC (Virtual Private Cloud):

Se crea una VPC con el bloque CIDR 10.0.0.0/16. Esta VPC actúa como un entorno aislado para alojar los recursos de red y las aplicaciones.

Subnet Pública:

Se crea una subnet pública (aws_subnet.public_subnet) con el bloque CIDR 10.0.0.0/24. Esta subnet se asocia a la VPC y está configurada para permitir el tráfico directo a Internet.

Subnet Privada:

Se crea una subnet privada (aws_subnet.private_subnet) con el bloque CIDR 10.0.1.0/24. Esta subnet se asocia a la VPC y se utiliza para alojar recursos que no necesitan acceso directo a Internet.

Tabla de Ruteo para Subnet Pública:

Se configura una tabla de ruteo pública (aws_route_table.public_route_table) asociada a la VPC. Esta tabla tiene una regla que dirige todo el tráfico (0.0.0.0/0) hacia una puerta de enlace de Internet (aws_internet_gateway.igw_eks).

Asociación de Subnet Pública con Tabla de Ruteo Pública:

La subnet pública se asocia a la tabla de ruteo pública para que herede las reglas de enrutamiento definidas en esa tabla (aws_route_table_association.public_subnet_association).

Puerta de Enlace de Internet:

Se crea una puerta de enlace de Internet (aws_internet_gateway.igw_eks) y se asocia a la VPC para permitir el acceso a Internet desde instancias en la subnet pública.

Tabla de Ruteo para Subnet Privada:

Se configura una tabla de ruteo privada (aws_route_table.private_route_table) asociada a la VPC. Esta tabla inicialmente no tiene reglas definidas.

Asociación de Subnet Privada con Tabla de Ruteo Privada:

La subnet privada se asocia a la tabla de ruteo privada (aws_route_table_association.private_subnet_association).

Despliegue en EKS

Objetivo del Despliegue

El despliegue en Amazon EKS tiene como objetivo implementar la aplicación en un clúster de Kubernetes gestionado por EKS. Aquí se detalla el proceso y las configuraciones realizadas.

Pasos de Despliegue

1. **Creación del Clúster de EKS:**
    - Se utiliza el recurso **aws_eks_cluster** para crear un clúster de EKS llamado "eks-cluster-proto". Se especifica un rol de IAM (**aws_iam_role.eks_cluster**) que tiene los permisos necesarios para interactuar con EKS.
2. **Configuración del Rol de IAM:**
    - Se crea un rol de IAM (**aws_iam_role.eks_cluster**) que permite a EKS asumir este rol. Este rol se adjunta con la política "AmazonEKSClusterPolicy".
3. **Configuración del Grupo de Seguridad:**
    - Se crea un grupo de seguridad (**aws_security_group.eks_cluster**) que define las reglas de tráfico permitido. En este caso, se permite todo el tráfico (ingreso y egreso) en todas las direcciones.
4. **Configuración de kubectl:**
    - Se utiliza el recurso **data "aws_eks_cluster_auth"** para obtener la configuración necesaria para kubectl. Esto configura la conexión a nuestro clúster de EKS en el archivo kubeconfig.
5. **Creación del ALB (Application Load Balancer):**
    - Se utiliza el recurso **aws_lb** para crear un ALB llamado "lb-eks". Este ALB se despliega en las subnets privadas (**aws_subnet.private_subnet**) y se asocia al grupo de seguridad creado anteriormente (**aws_security_group.app_lb**).
6. **Creación del Servicio en Kubernetes:**
    - Se utiliza el recurso **kubernetes_service** para crear un servicio en Kubernetes llamado "my-app-service". Este servicio expone la aplicación a través del ALB. Se especifican parámetros adicionales del ALB, como el tipo, la IP, etc.
7. **Despliegue de la Aplicación:**
    - Se utiliza el recurso **kubernetes_deployment** para implementar la aplicación en el clúster de EKS. En este caso, se implementa una imagen de contenedor de Nginx como ejemplo. Se especifica la replicación, puertos, y otros parámetros necesarios.

Gestión del Balanceo de Carga

El balanceo de carga se gestiona a través del servicio Kubernetes (**kubernetes_service.app_service**). Se configura como tipo "LoadBalancer", lo que permite que AWS cree automáticamente un Network Load Balancer (NLB). Este NLB dirige el tráfico al servicio de la aplicación.