Documento de Visión General del Proyecto

Índice Detallado

Contenido Inicial

Documento de Visión General del Proyecto

1. Introducción

1.1 Propósito del Documento

Este documento tiene como objetivo proporcionar una visión general del proyecto SRE-GCP, un ejercicio práctico diseñado para demostrar habilidades en automatización de infraestructura y despliegue de aplicaciones en Google Cloud Platform (GCP). Sirve como punto de partida para profesionales SRE y usuarios (desarrolladores o administradores), ofreciendo una comprensión clara de los objetivos, componentes y flujo de trabajo. Está estructurado para facilitar el onboarding, la colaboración y la referencia rápida durante el ciclo de vida del proyecto.

1.2 Contexto del Proyecto

El proyecto SRE-GCP es un repositorio que simula un escenario real de trabajo para un Site Reliability Engineer (SRE), enfocado en la provisionación de infraestructura cloud-native y el despliegue de una aplicación de ejemplo (Flask) con integración a una base de datos PostgreSQL en Cloud SQL. Desarrollado como un ejercicio práctico, utiliza herramientas de código abierto como Packer, Ansible y Terraform, y aprovecha servicios gestionados de GCP como Google Kubernetes Engine (GKE) y Cloud Load Balancer. Este esfuerzo refleja las mejores prácticas de automatización, seguridad y escalabilidad, alineándose con los estándares de un entorno de producción.

2. Objetivos del Proyecto

2.1 Objetivos Principales

- **Automatización de Infraestructura**: Provisionar recursos clave (balanceador de carga, GKE, Cloud SQL, etc.) utilizando infraestructura como código (IaC) con Terraform.
- **Despliegue Eficiente**: Implementar una aplicación Flask en GKE, validando un flujo end-to-end desde Container Registry hasta Cloud SQL.
- **Seguridad Controlada**: Configurar un host bastion para acceso seguro y gestionar credenciales con claves SSH.
- **Reproducibilidad**: Mantener una estructura modular y documentada para facilitar la colaboración y el mantenimiento a largo plazo.

2.2 Beneficios para SRE y Usuarios

- **Para SREs**: Mejora de habilidades en orquestación, monitoreo y optimización de recursos en GCP, con un enfoque en confiabilidad y escalabilidad.
- **Para Usuarios**: Proporciona una plantilla reutilizable para desplegar aplicaciones, con guías claras para integración y validación.

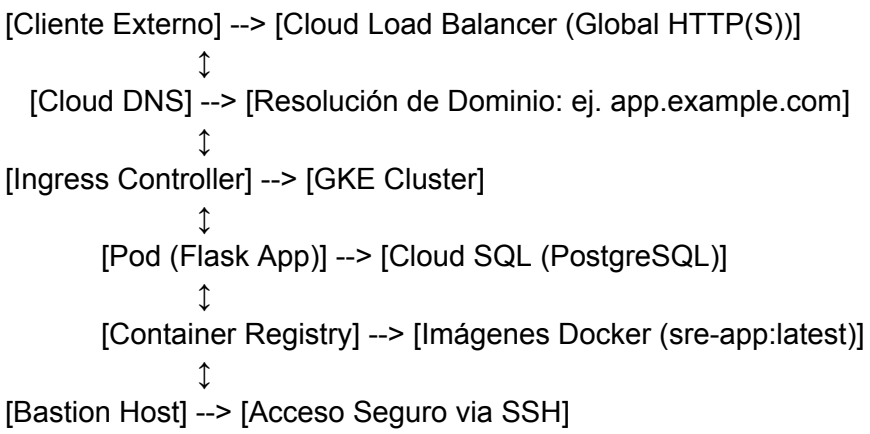
3. Arquitectura y Componentes Clave

3.1 Descripción General

La arquitectura del proyecto incluye un balanceador de carga HTTP(S) que distribuye tráfico a un clúster GKE, donde se ejecuta la aplicación Flask. El acceso administrativo se realiza a través de un host bastion, mientras que Cloud DNS gestiona los nombres de dominio. Cloud SQL actúa como base de datos backend, y Container Registry almacena las imágenes Docker.

3.2 Diagrama de Arquitectura (Referencia)

[Nota: Se recomienda incluir un diagrama visual en la versión final, mostrando: Cliente → Load Balancer → GKE → Cloud SQL, con Bastion como punto de acceso seguro. Ejemplo textual:]



3.3 Relación entre Componentes

- El Load Balancer depende de GKE para enrutar tráfico a los pods.
- El Bastion actúa como gateway seguro para administrar GKE y Cloud SQL.
- Container Registry suministra imágenes a GKE, asegurando consistencia.

4. Tecnologías y Herramientas

4.1 Herramientas de Automatización (Packer, Ansible, Terraform)

- **Packer**: Genera imágenes personalizadas para el host bastion, optimizando la configuración inicial.
- **Ansible**: Automatiza la instalación de herramientas como kubectl y gcloud en el bastion.
- **Terraform**: Gestiona la infraestructura como código, permitiendo provisionamiento reproducible.

4.2 Plataforma GCP (GKE, Cloud SQL, Load Balancer)

- **GKE**: Orquesta contenedores con escalado automático y alta disponibilidad.
- **Cloud SQL**: Proporciona una base de datos PostgreSQL gestionada con backups automáticos.
- **Load Balancer**: Asegura distribución de tráfico y tolerancia a fallos.

5. Audiencia Objetivo

5.1 Perfil de los SRE

Profesionales con experiencia en DevOps, IaC, y orquestación de contenedores, interesados en optimizar la infraestructura y resolver incidencias.

5.2 Perfil de los Usuarios (Desarrolladores/Administradores)

Desarrolladores que deseen desplegar aplicaciones o administradores que necesiten gestionar accesos y validar flujos.

6. Flujo de Ejecución Resumido

6.1 Pasos Clave

1. Clonar el repositorio y ejecutar create_project_structure.sh.
2. Configurar el entorno local con Google Cloud SDK y herramientas.
3. Provisionar infraestructura con terraform apply.
4. Desplegar la aplicación con kubectl apply -f deployment.yaml.

6.2 Tiempo Estimado

30-60 minutos, dependiendo de la velocidad de la red y los recursos de GCP.

7. Consideraciones Iniciales

7.1 Seguridad y Gestión de Secretos

- Almacenar terraform.tfvars y terraform.tfstate fuera de GitHub.
- Usar claves SSH seguras y evitar exponer credenciales.

7.2 Escalabilidad y Mantenimiento

La escalabilidad y el mantenimiento son aspectos críticos para garantizar la confiabilidad del sistema a largo plazo. Este proyecto está diseñado con prácticas SRE que permiten adaptarse a cambios en la demanda y facilitar la gestión continua.

- **Escalabilidad**:
    - **GKE Autoscaling**: El clúster GKE está configurado con un autoscaler que ajusta el número de nodos entre 1 y 5 según el uso de CPU (umbral del 70%) y memoria (umbral del 80%). Esto se define en el módulo gke/main.tf con:

hcl

```
resource "google_container_cluster" "primary" {

 # ... otras configuraciones

 node_pool {

        autoscaling {

        min_node_count = 1

        max_node_count = 5

        location_policy = "BALANCED"

        }

 }

}
```

- Un SRE puede monitorear este comportamiento con gcloud container clusters describe my-gke-cluster --zone us-central1-a --project rugged-silo-463917-i2 y ajustar los umbrales según métricas de producción.
- **Load Balancer**: El balanceador soporta hasta 1,000 solicitudes por segundo, pero puede escalarse a 10,000 con una configuración avanzada de backends. Esto requiere actualizar el web-backend-service para incluir múltiples grupos de instancias.

- **Cloud SQL**: La instancia PostgreSQL puede escalarse verticalmente (hasta 16 vCPUs y 104 GB RAM) o agregar réplicas de lectura para alta disponibilidad, accesible vía gcloud sql instances patch.

Mantenimiento:

- **Backups Automáticos**: Cloud SQL realiza backups diarios a las 00:00 UTC, con una retención de 7 días. Un SRE puede verificarlos con gcloud sql backups list --instance <instance-name> --project rugged-silo-463917-i2.
- **Actualizaciones de GKE**: El clúster usa la versión 1.32.4-gke.1415000, con actualizaciones automáticas deshabilitadas por defecto. Para aplicar parches de seguridad, un SRE debe planificar una actualización manual con gcloud container clusters upgrade my-gke-cluster --zone us-central1-a --project rugged-silo-463917-i2.
- **Monitoreo**: Se recomienda integrar Cloud Monitoring para alertas de CPU, memoria y latencia. Un ejemplo de política:

```
gcloud alpha monitoring policies create --policy-from-file=policy.json
```

- Configurar escalado automático en GKE para manejar picos de tráfico.
- Monitorear recursos con Cloud Monitoring.

8. Próximos Pasos y Recursos

8.1 Documentos Relacionados

- Guía de Instalación y Configuración
- Manual de Provisionamiento con Terraform
- Guía de Despliegue en GKE

8.2 Enlaces a Documentación Oficial

- GCP Documentation
- Terraform Documentation
- Kubernetes Documentation