**Guía de Implementación en GCC**

**1. Crear las instancias**

Usa el siguiente comando para crear las instancias. Cambia INSTANCE\_NAME por un nombre que elijas para tu instancia.

bash

Copiar código

gcloud compute instances create INSTANCE\_NAME \

--zone=us-central1-a \

--machine-type=e2-micro \

--image-family=ubuntu-2204-lts \

--image-project=ubuntu-os-cloud \

--tags=http-server \

--metadata=startup-script='#!/bin/bash

sudo apt-get update

sudo apt-get install -y python3-pip

sudo pip3 install django psycopg2-binary

cd /home/$(whoami)/repo-logs/django\_app

python3 manage.py migrate

'

**Repite este comando** para crear más instancias si es necesario, cambiando INSTANCE\_NAME por un nombre único para cada una.

**2. Configurar el firewall**

Crea una regla de firewall para permitir el tráfico HTTP. Puedes hacerlo con el siguiente comando:

bash

Copiar código

gcloud compute firewall-rules create allow-http \

--allow tcp:80 \

--source-ranges 0.0.0.0/0 \

--target-tags http-server \

--direction INGRESS \

--priority 1000

**3. Crear el balanceador de carga**

Para crear un balanceador de carga HTTP, sigue estos pasos:

**1. Crea un grupo de instancias:**

bash

Copiar código

gcloud compute instance-groups unmanaged create INSTANCE\_GROUP\_NAME --zone=us-central1-a

**2. Agrega las instancias al grupo:**

bash

Copiar código

gcloud compute instance-groups unmanaged add-instances INSTANCE\_GROUP\_NAME \

--instances=INSTANCE\_NAME\_1,INSTANCE\_NAME\_2 \

--zone=us-central1-a

**3. Crea un balanceador de carga:**

bash

Copiar código

gcloud compute backend-services create backend-service-name \

--protocol=HTTP \

--port-name=http \

--health-checks=health-check-name \

--global

**4. Crea una dirección IP para el balanceador de carga:**

bash

Copiar código

gcloud compute addresses create load-balancer-ip --global

**5. Crea un recurso de configuración de URL:**

bash

Copiar código

gcloud compute url-maps create url-map-name \

--default-service backend-service-name

**6. Crea un proxy de carga:**

bash

Copiar código

gcloud compute target-http-proxies create proxy-name \

--url-map=url-map-name

**7. Crea un recurso de reenvío:**

bash

Copiar código

gcloud compute forwarding-rules create forwarding-rule-name \

--global \

--target-http-proxy=proxy-name \

--ports=80 \

--address=load-balancer-ip

**4. Clonar el repositorio**

Accede a tu instancia y clona tu repositorio:

bash

Copiar código

cd /home/$(whoami)

git clone https://github.com/Diplomatictw0/repo-logs.git

**5. Desplegar la aplicación**

Asegúrate de estar en la carpeta correcta y ejecuta el servidor Django:

bash

Copiar código

cd repo-logs/django\_app

python3 manage.py runserver 0.0.0.0:8000

**Notas Finales**

* Asegúrate de ajustar los nombres de los recursos (instancias, grupos, balanceadores) según tus preferencias.
* Monitoriza el uso de recursos para evitar costos innecesarios. Utiliza instancias e2-micro para mantener costos bajos.
* Puedes configurar un servicio de monitoreo para asegurarte de que el rendimiento se mantenga dentro de los límites especificados en el ASR.

Con esta guía, deberías poder configurar y simular tu aplicación de manera efectiva en GCC. Si tienes más preguntas, no dudes en preguntar.

.