Configuración de un Balanceador de Cargas de Red (GSP007) en Google Cloud

Resumen del Proyecto

En este proyecto, implementé un **Balanceador de Cargas de Red (NLB) de transferencia (passthrough)** en Google Cloud Platform. El objetivo fue configurar un sistema de alta disponibilidad que distribuye el tráfico TCP entrante (en el puerto 80) de manera uniforme entre un grupo de tres servidores web Apache alojados en instancias de Compute Engine.

Este documento detalla el objetivo, la arquitectura implementada y un desglose de cada comando ejecutado para demostrar la comprensión de los componentes involucrados.

1. Objetivo del Laboratorio

El objetivo principal de este lab (GSP007) fue construir y verificar un balanceador de cargas de red L4 (Capa 4 - Transporte) funcional.

Esto implicó:

- Crear y configurar múltiples instancias de VM (servidores web).
- Configurar reglas de firewall para permitir el tráfico web.
- Reservar una dirección IP estática como punto de entrada único.
- Implementar una verificación de estado (Health Check) para monitorear los servidores.
- Configurar un "Grupo de Destino" (Target Pool) para agrupar las instancias.
- Crear una "Regla de Reenvío" (Forwarding Rule) para vincular la IP pública con el grupo de servidores.

2. El Problema a Solucionar

En un escenario real, alojar una aplicación en un solo servidor presenta dos riesgos críticos:

1. **Punto Único de Falla:** Si ese servidor falla (por hardware, software o mantenimiento), la aplicación se vuelve completamente inaccesible.

2. **Cuello de Botella:** Si la aplicación recibe un alto volumen de tráfico (muchos usuarios), el servidor puede saturarse, volviéndose lento o dejando de responder.

Un **Balanceador de Cargas** soluciona esto al actuar como un "director de tráfico". Proporciona una única dirección IP pública a los usuarios y distribuye de forma inteligente las solicitudes entrantes entre un grupo de servidores idénticos. Esto garantiza **alta disponibilidad** (si un servidor falla, el tráfico se redirige a los sanos) y **escalabilidad** (se pueden añadir más servidores al grupo para manejar más tráfico).

3. Arquitectura Creada

La solución final consistió en los siguientes componentes de GCP:

- **3 Instancias de VM (Compute Engine):** `www1`, `www2`, `www3`. Cada una ejecutando un servidor web Apache con una página de inicio única para identificarla.
- **1 Regla de Firewall:** `www-firewall-network-lb` que permite el tráfico TCP por el puerto 80 (HTTP) a cualquier instancia con la etiqueta `network-lb-tag`.
- **1 Dirección IP Externa Estática:** `network-lb-ip-1` (regional) como el punto de entrada público y fijo para todo el tráfico.
- **1 Verificación de Estado (Health Check):** `basic-check` (HTTP) que GCP usa para sondear constantemente los servidores y asegurarse de que estén respondiendo.
- **1 Grupo de Destino (Target Pool):** `www-pool` que agrupa las 3 VMs y está asociado a la verificación de estado.
- **1 Regla de Reenvío (Forwarding Rule):** `www-rule` que es el "pegamento" final. Vincula la IP pública (`network-lb-ip-1`) con el grupo de destino (`www-pool`), indicando que todo el tráfico que llegue a esa IP por el puerto 80 debe ser reenviado a dicho grupo.

4. Desglose de Comandos y Pasos

A continuación, se detalla cada comando ejecutado y una explicación de sus componentes.

Tarea 1: Configuración de la región y la zona

```
# 1. Configurar la región predeterminada
gcloud config set compute/region Region

# 2. Configurar la zona predeterminada
gcloud config set compute/zone Zone
```

- **Explicación:** Estos comandos le dicen a la herramienta de línea de comandos `gcloud` en qué región y zona geográfica queremos trabajar por defecto. Esto evita tener que especificarlo en cada comando posterior (`--region Region` , `--zone Zone`), simplificando la sintaxis.

Tarea 2: Creación de las instancias y regla de firewall

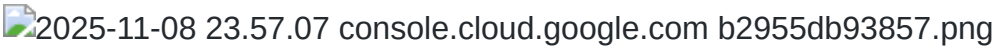
```
# 3. Crear la primera instancia (www1)
gcloud compute instances create www1 \
    --zone=Zone \
    --tags=network-lb-tag \
    --machine-type=e2-small \
    --image-family=debian-11 \
    --image-project=debian-cloud \
    --metadata=startup-script='#!/bin/bash
      apt-get update
      apt-get install apache2 -y
      service apache2 restart
      echo "
<h3>Web Server: www1</h3>" | tee /var/www/html/index.html'
```

- `gcloud compute instances create www1` : Comando principal para crear una nueva VM llamada `www1` .
- `--zone=Zone` : Especifica la zona de disponibilidad donde vivirá la VM (usando el valor que configuramos en la Tarea 1).
- `--tags=network-lb-tag` : **Componente clave**. Asigna una etiqueta ("tag") a la VM. Esta etiqueta será usada más adelante por la regla de firewall para identificar a qué VMs aplicar la regla.
- `--machine-type=e2-small` : Define el tipo de máquina (CPU y RAM).
- `--image-family` / `--image-project` : Especifica la imagen del sistema operativo (Debian 11).
- `--metadata=startup-script='...'` : Ejecuta un script la *primera vez* que la VM arranca. Este script actualiza el SO, instala el servidor web Apache y crea una página `index.html`

personalizada (en este caso, mostrando "Web Server: www1"). Esto es lo que nos permite identificar qué servidor responde en la prueba final.

(Los comandos para `www2` y `www3` son idénticos, solo cambia el nombre de la instancia y el contenido del `index.html`)

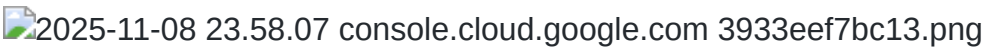
- `gcloud compute firewall-rules create ...`: Comando para crear una nueva regla de firewall.
- `--target-tags network-lb-tag`: **El vínculo más importante**. Esta regla se aplica *solo* a las instancias que tengan la etiqueta `network-lb-tag` (las 3 que acabamos de crear).
- `--allow tcp:80`: Permite el tráfico entrante (`ingress`) que use el protocolo TCP en el puerto 80 (el puerto estándar de HTTP).



```
# 5. Listar las instancias
gcloud compute instances list
```

- **Explicación:** Este comando lista todas las VMs, permitiéndonos ver su estado (`RUNNING`) y, lo más importante, su `EXTERNAL_IP` (IP externa).

```
# 6. Probar la conexión directa a una VM
curl http://[IP_ADDRESS]
```



- **Explicación:** Antes de configurar el balanceador, verificamos que cada servidor web funcione individualmente. Al ejecutar `curl` contra la IP externa de cada VM, confirmamos que Apache está sirviendo la página web correctamente.

Tarea 3: Configuración del servicio de balanceo de cargas

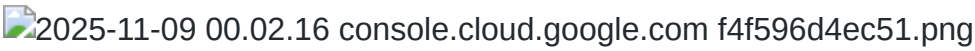
```
# 7. Crear una dirección IP estática para el balanceador
gcloud compute addresses create network-lb-ip-1 \
  --region Region
```

- **Explicación:** Reservamos una dirección IP externa estática (fija). Si no hiciéramos esto, al balanceador se le asignaría una IP *efímera* (temporal) que podría cambiar si se reinicia, lo

cual no es deseable para un servicio público (los registros DNS quedarían obsoletos).

```
# 8. Crear una verificación de estado (Health Check)
gcloud compute http-health-checks create basic-check
```

- **Explicación:** Crea un "verificador de salud" de tipo HTTP. GCP usará este servicio para enviar solicitudes HTTP (pings) a cada servidor en el puerto 80. Si el servidor responde con un "200 OK", se marca como `HEALTHY`. Si no responde, se marca como `UNHEALTHY` y el balanceador de cargas dejará de enviarle tráfico temporalmente. Esto es esencial para la alta disponibilidad.



Tarea 4: Creación del grupo de destino y la regla de reenvío

```
# 9. Crear el Grupo de Destino (Target Pool)
gcloud compute target-pools create www-pool \
  --region Region --http-health-check basic-check
```

- `gcloud compute target-pools create www-pool`: Crea el "grupo" que contendrá nuestros servidores backend.
- `--region`: El grupo debe existir en la misma región que las instancias.
- `--http-health-check basic-check`: **Vínculo clave**. Asocia la verificación de estado `basic-check` (creada en el paso anterior) a este grupo. Ahora, el *pool* sabe cómo determinar si sus miembros están sanos.

```
# 10. Añadir las instancias al grupo
gcloud compute target-pools add-instances www-pool \
    --instances www1,www2,www3
```

- **Explicación:** Este comando le dice al `www-pool` cuáles son sus miembros. Ahora el grupo `www-pool` gestiona las instancias `www1`, `www2` y `www3`, y las monitorea usando `basic-check`.

```
# 11. Crear la Regla de Reenvío (Forwarding Rule)
gcloud compute forwarding-rules create www-rule \
    --region  Region \
    --ports 80 \
    --address network-lb-ip-1 \
    --target-pool www-pool
```

- **Explicación:** Esta es la pieza central que une todo.
- `gcloud compute forwarding-rules create www-rule`: Crea la regla de reenvío.
- `--ports 80`: Especifica que esta regla "escucha" el tráfico en el puerto 80.
- `--address network-lb-ip-1`: Vincula la regla a la IP estática que reservamos (`network-lb-ip-1`).
- `--target-pool www-pool`: Indica que todo el tráfico que coincida (IP: `network-lb-ip-1`, Puerto: `80`) debe ser **reenviado** al `www-pool`.

5. Verificación y Pruebas (Tarea 5)

Ahora que todo está conectado, probamos la solución completa.

```
# 12. Obtener la IP del balanceador
gcloud compute forwarding-rules describe www-rule --region Region
```

- **Explicación:** Este comando nos da los detalles de nuestra regla de reenvío, incluyendo la IP pública (`IPAddress`) que se le asignó.

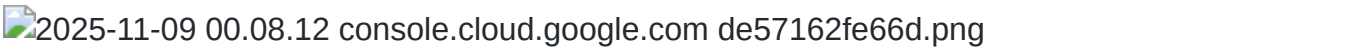
```
# 13. Guardar la IP del balanceador en una variable
IPADDRESS=$(gcloud compute forwarding-rules describe www-rule --region Region --format="json" | jq -r .IPAddress)
```

- **Explicación:** Este es un comando de shell más avanzado.
  - `--format="json"`: Pide la salida del comando anterior en formato JSON.
  - `| jq -r .IPAddress`: "Entuba" (pipe) esa salida JSON a la herramienta `jq`, que extrae (`-r`) el valor de la clave `IPAddress`.
  - `IPADDRESS=$(...)`: Guarda ese valor extraído en una variable de shell llamada `IPADDRESS`.

```
# 14. Imprimir la variable para verificar
echo $IPADDRESS
```

```
# 15. Enviar tráfico de prueba en bucle
while true; do curl -m1 $IPADDRESS; done
```

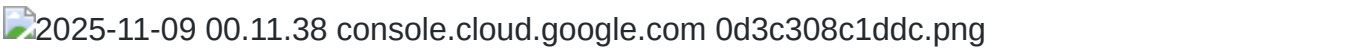
- **Explicación:** El comando final de prueba.
  - `while true; do ... done`: Inicia un bucle infinito.
  - `curl -m1 $IPADDRESS`: Realiza una solicitud HTTP (`curl`) a la IP pública del balanceador, con un tiempo límite (`-m1`) de 1 segundo.



Resultado

El resultado en la terminal demostró que el balanceador de cargas estaba funcionando, distribuyendo las solicitudes de manera aleatoria entre los tres servidores sanos:

```
<h3>Web Server: www2</h3>
<h3>Web Server: www1</h3>
<h3>Web Server: www3</h3>
<h3>Web Server: www2</h3>
<h3>Web Server: www1</h3>
<h3>Web Server: www1</h3>
<h3>Web Server: www3</h3>
...
(Presiona Ctrl+C para detener)
```



6. Conclusión y Aprendizajes Clave

Este laboratorio fue una demostración práctica y fundamental de los conceptos de **alta disponibilidad** y **distribución de tráfico** en la nube.

- **Comprensión de L4:** Entendí que un Balanceador de Cargas de Red opera en la Capa 4 (TCP/UDP). No "mira" el contenido del tráfico (como las cabeceras HTTP), simplemente reenvía los paquetes basándose en la IP y el puerto. Esto lo hace extremadamente rápido y eficiente.

- **Modularidad de GCP:** Comprendí cómo los servicios de GCP son modulares. No existe un solo "botón" para "crear balanceador". En su lugar, se deben construir y vincular varios recursos independientes (IPs, Health Checks, Target Pools, Forwarding Rules) para crear la solución completa.
- **Importancia de las Etiquetas (Tags):** Vi el poder de las etiquetas (`tags`) para aplicar políticas, como reglas de firewall, a grupos de recursos de manera dinámica y organizada.

Este proyecto solidifica mi comprensión de la infraestructura de red fundamental en Google Cloud, una habilidad esencial para construir aplicaciones resilientes y escalables.