

Reporte de Laboratorio: Desarrolla tu red de Google Cloud (GSP321) - Lab de Desafío

Christhian Rodriguez

Curso: Google Cloud Skills Boost - Cloud Engineering

1. Resumen Ejecutivo y Escenario

Situación

Jooli Inc. ha solicitado la configuración de una infraestructura de red robusta y segmentada para apoyar a su equipo de desarrollo (Griffin). El objetivo es desplegar un entorno de WordPress utilizando contenedores (Kubernetes) para el frontend y una base de datos administrada (Cloud SQL) para el backend.

Objetivo

El desafío consiste en construir manualmente una arquitectura de red que incluya:

- Segregación de Redes:** VPCs separadas para Desarrollo y Producción.
- Seguridad:** Un host de bastión para acceso seguro y reglas de firewall.
- Contenedores:** Un clúster de Google Kubernetes Engine (GKE).
- Persistencia de Datos:** Cloud SQL para MySQL.
- Observabilidad:** Monitoreo de tiempo de actividad.
- Gestión de Accesos:** Configuración de roles IAM para colaboradores.

2. Arquitectura de la Solución

La solución implementada utiliza una arquitectura de microservicios sobre una infraestructura de red personalizada.

- VPC de Desarrollo (griffin-dev-vpc):** Aloja el clúster de Kubernetes y la base de datos.
- VPC de Producción (griffin-prod-vpc):** Preparada para futuras cargas de trabajo.

- **Bastión:** Instancia puente con interfaces en ambas redes para gestión segura.

Team Griffin Infrastructure

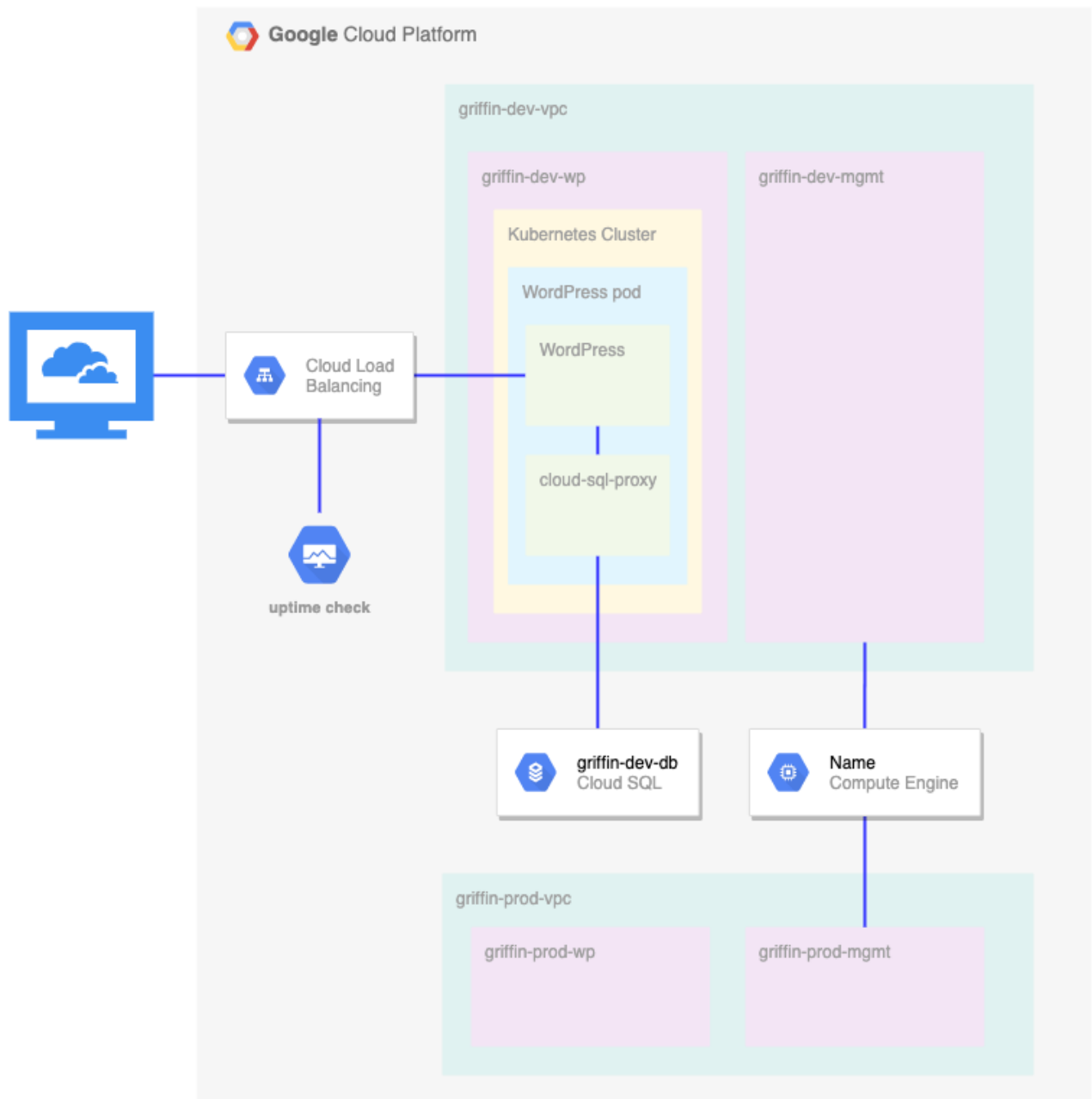


Figura 1: Diagrama de arquitectura propuesto para la infraestructura del equipo Griffin.

3. Desarrollo e Implementación

Para asegurar la reproducibilidad y eficiencia, la mayoría de las tareas se realizaron utilizando `gcloud CLI` en Cloud Shell, en lugar de la consola gráfica.

Fase 1: Configuración de Redes (VPC y Subredes)

Se crearon dos redes VPC en modo personalizado para tener control total sobre los rangos de IP.

Comandos utilizados:

```
# Variables de entorno para consistencia
export REGION=us-central1
export ZONE=us-central1-c

# Creación de VPCs y Subredes
gcloud compute networks create griffin-dev-vpc --subnet-mode=custom
gcloud compute networks subnets create griffin-dev-wp --network=griffin-dev-vpc
--region=$REGION --range=192.168.16.0/20
gcloud compute networks subnets create griffin-dev-mgmt --network=griffin-dev-
vpc --region=$REGION --range=192.168.32.0/20

# Repetición para Producción (Prod)
gcloud compute networks create griffin-prod-vpc --subnet-mode=custom
gcloud compute networks subnets create griffin-prod-wp --network=griffin-prod-
vpc --region=$REGION --range=192.168.48.0/20
gcloud compute networks subnets create griffin-prod-mgmt --network=griffin-
prod-vpc --region=$REGION --range=192.168.64.0/20
```

```
CLOUD SHELL
Terminal (qwiklabs-gcp-01-0ed45439799f) x + v
i Gemini CLI is available in Cloud Shell terminal! Type gemini to try it. Learn more

IPV6_ACCESS_TYPE:
INTERNAL_IPV6_PREFIX:
EXTERNAL_IPV6_PREFIX:
Created [https://www.googleapis.com/compute/v1/projects/qwiklabs-gcp-01-0ed45439799f/regions/us-central1/subnetworks/griffin-dev-mgmt].
NAME: griffin-dev-mgmt
REGION: us-central1
NETWORK: griffin-dev-vpc
RANGE: 192.168.32.0/20
STACK_TYPE: IPV4_ONLY
IPV6_ACCESS_TYPE:
INTERNAL_IPV6_PREFIX:
EXTERNAL_IPV6_PREFIX:
student_01_ccc1cd50fe47@cloudshell:~ (qwiklabs-gcp-01-0ed45439799f)$ gcloud compute networks create griffin-prod-vpc --subnet-mode=custom
Created [https://www.googleapis.com/compute/v1/projects/qwiklabs-gcp-01-0ed45439799f/global/networks/griffin-prod-vpc].
NAME: griffin-prod-vpc
SUBNET_MODE: CUSTOM
BGP_ROUTING_MODE: REGIONAL
IPV4_RANGE:
GATEWAY_IPV4:
INTERNAL_IPV6_RANGE:

Instances on this network will not be reachable until firewall rules
are created. As an example, you can allow all internal traffic between
instances as well as SSH, RDP, and ICMP by running:

$ gcloud compute firewall-rules create <FIREWALL_NAME> --network griffin-prod-vpc --allow tcp,udp,icmp --source-ranges <IP_RANGE>
$ gcloud compute firewall-rules create <FIREWALL_NAME> --network griffin-prod-vpc --allow tcp:22,tcp:3389,icmp

student_01_ccc1cd50fe47@cloudshell:~ (qwiklabs-gcp-01-0ed45439799f)$ gcloud compute networks subnets create griffin-prod-wp \
--network=griffin-prod-vpc \
--region=$REGION \
--range=192.168.48.0/20

gcloud compute networks subnets create griffin-prod-mgmt \
--network=griffin-prod-vpc \
--region=$REGION \
--range=192.168.64.0/20
Created [https://www.googleapis.com/compute/v1/projects/qwiklabs-gcp-01-0ed45439799f/regions/us-central1/subnetworks/griffin-prod-wp].
NAME: griffin-prod-wp
REGION: us-central1
NETWORK: griffin-prod-vpc
RANGE: 192.168.48.0/20
STACK_TYPE: IPV4_ONLY
IPV6_ACCESS_TYPE:
INTERNAL_IPV6_PREFIX:
EXTERNAL_IPV6_PREFIX:
Created [https://www.googleapis.com/compute/v1/projects/qwiklabs-gcp-01-0ed45439799f/regions/us-central1/subnetworks/griffin-prod-mgmt].
NAME: griffin-prod-mgmt
REGION: us-central1
NETWORK: griffin-prod-vpc
RANGE: 192.168.64.0/20
STACK_TYPE: IPV4_ONLY
IPV6_ACCESS_TYPE:
INTERNAL_IPV6_PREFIX:
EXTERNAL_IPV6_PREFIX:
student_01_ccc1cd50fe47@cloudshell:~ (qwiklabs-gcp-01-0ed45439799f)$
```

Figura 2: Creación exitosa de las VPCs y subredes mediante Cloud Shell.

Fase 2: Host de Bastión y Seguridad

Se configuró una instancia de Compute Engine con **dos interfaces de red** (nic0 y nic1) para tener presencia simultánea en la red de desarrollo y producción, permitiendo la administración centralizada. Se habilitaron reglas de firewall para permitir SSH (Puerto 22).

Comandos utilizados:

```
# Crear reglas de firewall para SSH en ambas VPCs
gcloud compute firewall-rules create allow-ssh-dev --network=griffin-dev-vpc --allow=tcp:22 --source-ranges=0.0.0.0/0
gcloud compute firewall-rules create allow-ssh-prod --network=griffin-prod-vpc --allow=tcp:22 --source-ranges=0.0.0.0/0

# Crear la instancia Bastión con dos interfaces
gcloud compute instances create griffin-bastion \
  --zone=$ZONE \
  --machine-type=e2-medium \
  --network-interface=network=griffin-dev-vpc,subnet=griffin-dev-mgmt \
  --network-interface=network=griffin-prod-vpc,subnet=griffin-prod-mgmt,no-address \
  --tags=bastion
```

```
student_01_cc1cd50fe47@cloudshell:~ (qwiklabs-gcp-01-0ed45439799f)$ # 1. Crear reglas de firewall para SSH en ambas VPCs
gcloud compute firewall-rules create allow-ssh-dev --network=griffin-dev-vpc --allow=tcp:22 --source-ranges=0.0.0.0/0
gcloud compute firewall-rules create allow-ssh-prod --network=griffin-prod-vpc --allow=tcp:22 --source-ranges=0.0.0.0/0

# 2. Crear la instancia Bastión (e2-medium) con dos interfaces
# La primera interfaz (dev) tendrá IP pública para que entres por SSH.
# La segunda (prod) será solo interna.
gcloud compute instances create griffin-bastion \
  --zone=$ZONE \
  --machine-type=e2-medium \
  --network-interface=network=griffin-dev-vpc,subnet=griffin-dev-mgmt \
  --network-interface=network=griffin-prod-vpc,subnet=griffin-prod-mgmt,no-address \
  --tags=bastion
Creating firewall...working..Created [https://www.googleapis.com/compute/v1/projects/qwiklabs-gcp-01-0ed45439799f/global/firewalls/allow-ssh-dev].
Creating firewall...done.
NAME: allow-ssh-dev
NETWORK: griffin-dev-vpc
DIRECTION: INGRESS
PRIORITY: 1000
ALLOW: tcp:22
DENY:
DISABLED: False
Creating firewall...working..Created [https://www.googleapis.com/compute/v1/projects/qwiklabs-gcp-01-0ed45439799f/global/firewalls/allow-ssh-prod].
Creating firewall...done.
NAME: allow-ssh-prod
NETWORK: griffin-prod-vpc
DIRECTION: INGRESS
PRIORITY: 1000
ALLOW: tcp:22
DENY:
DISABLED: False
Created [https://www.googleapis.com/compute/v1/projects/qwiklabs-gcp-01-0ed45439799f/zones/us-central1-c/instances/griffin-bastion].
NAME: griffin-bastion
ZONE: us-central1-c
MACHINE_TYPE: e2-medium
PREEMPTIBLE:
INTERNAL_IP: 192.168.32.2
192.168.64.2
EXTERNAL_IP: 34.60.234.60
STATUS: RUNNING
```

Figura 3: Despliegue del host de bastión con múltiples interfaces de red.

Fase 3: Base de Datos y Kubernetes

Se desplegó una instancia de Cloud SQL y un clúster de GKE.

1. **Cloud SQL:** Se creó una instancia MySQL y se configuró la base de datos `wordpress` y el usuario `wp_user` mediante comandos SQL directos.
2. **GKE:** Se aprovisionó un clúster regional de 2 nodos.

Comandos utilizados:

```
# Crear instancia Cloud SQL
gcloud sql instances create griffin-dev-db --region=$REGION --tier=db-f1-micro
--root-password=password123

# Configuración de base de datos (Ejecutado tras conectar con 'gcloud sql
connect')
# CREATE DATABASE wordpress;
# CREATE USER "wp_user"@"%" IDENTIFIED BY "stormwind_rules";
# GRANT ALL PRIVILEGES ON wordpress.* TO "wp_user"@"%";
# FLUSH PRIVILEGES;

# Crear clúster GKE
gcloud container clusters create griffin-dev \
  --zone=$ZONE \
  --network=griffin-dev-vpc \
  --subnetwork=griffin-dev-wp \
  --machine-type=e2-standard-4 \
  --num-nodes=2
```

```
CLOUD SHELL
Terminal (qwiklabs-gcp-01-0ed45439799f) x (qwiklabs-gcp-01-0ed45439799f) x + v Open Editor

Gemini CLI is available in Cloud Shell terminal! Type gemini to try it. Learn more

Welcome to Cloud Shell! Type "help" to get started, or type "gemini" to try prompting with Gemini CLI.
Your Cloud Platform project in this session is set to qwiklabs-gcp-01-0ed45439799f.
Use 'gcloud config set project [PROJECT_ID]' to change to a different project.
student_01_ccc1cd50fe47@cloudshell:~ (qwiklabs-gcp-01-0ed45439799f)$ AC
student_01_ccc1cd50fe47@cloudshell:~ (qwiklabs-gcp-01-0ed45439799f)$ AC
student_01_ccc1cd50fe47@cloudshell:~ (qwiklabs-gcp-01-0ed45439799f)$ export REGION=us-central1
export ZONE=us-central1-c
student_01_ccc1cd50fe47@cloudshell:~ (qwiklabs-gcp-01-0ed45439799f)$ gcloud container clusters create griffin-dev \
  --zone=$ZONE \
  --network=griffin-dev-vpc \
  --subnetwork=griffin-dev-wp \
  --machine-type=e2-standard-4 \
  --num-nodes=2
Note: Your Pod address range ('--cluster-ipv4-cidr') can accommodate at most 1008 node(s).
Creating cluster griffin-dev in us-central1-c... Cluster is being health-checked (Kubernetes Control Plane is healthy)...done.
Created [https://container.googleapis.com/v1/projects/qwiklabs-gcp-01-0ed45439799f/zones/us-central1-c/clusters/griffin-dev].
To inspect the contents of your cluster, go to: https://console.cloud.google.com/kubernetes/workload/_gcloud/us-central1-c/griffin-dev?project=qwiklabs-gcp-01-0ed45439799f
kubeconfig entry generated for griffin-dev.
NAME: griffin-dev
LOCATION: us-central1-c
MASTER_VERSION: 1.33.5-gke.1201000
MASTER_IP: 34.29.155.41
MACHINE_TYPE: e2-standard-4
NODE_VERSION: 1.33.5-gke.1201000
NUM_NODES: 2
STATUS: RUNNING
STACK_TYPE: IPV4
student_01_ccc1cd50fe47@cloudshell:~ (qwiklabs-gcp-01-0ed45439799f)$
```

Figura 4: Aprovisionamiento del clúster de Kubernetes 'griffin-dev'.

```

STATUS: RUNNING
student_01_ccc1cd50fe47@cloudshell:~ (qwiklabs-gcp-01-0ed45439799f)$ gcloud sql instances create griffin-dev-db \
--region=REGION \
--tier=db-f1-micro \
--root-password=password123
Creating Cloud SQL instance for MySQL 8.0...done.
Created [https://sqladmin.googleapis.com/sql/v1beta4/projects/qwiklabs-gcp-01-0ed45439799f/instances/griffin-dev-db].
NAME: griffin-dev-db
DATABASE_VERSION: MYSQL_8_0
LOCATION: us-central1-c
TIER: db-f1-micro
PRIMARY_ADDRESS: 34.44.246.179
PRIVATE_ADDRESS: -
STATUS: RUNNABLE
student_01_ccc1cd50fe47@cloudshell:~ (qwiklabs-gcp-01-0ed45439799f)$ # Conectarse a la base de datos (presiona Enter si te pide contraseña la primera vez, o usa la que definiste arriba)
gcloud sql connect griffin-dev-db --user=root --quiet

# DENTRO de la consola SQL, pega esto:
CREATE DATABASE wordpress;
CREATE USER "wp_user"@"%" IDENTIFIED BY "stormwind_rules";
GRANT ALL PRIVILEGES ON wordpress.* TO "wp_user"@"%";
FLUSH PRIVILEGES;
EXIT;
Allowlisting your IP for incoming connection for 5 minutes...done.
Connecting to database with SQL user [root].Enter password:
Welcome to the MySQL monitor.  Commands end with ; or \g.
Your MySQL connection id is 67
Server version: 8.0.41-google (Google)

Copyright (c) 2000, 2025, Oracle and/or its affiliates.

Oracle is a registered trademark of Oracle Corporation and/or its
affiliates. Other names may be trademarks of their respective
owners.

Type 'help;' or '\h' for help. Type '\c' to clear the current input statement.

mysql> CREATE DATABASE wordpress;
Query OK, 1 row affected (0.08 sec)

mysql> CREATE USER "wp_user"@"%" IDENTIFIED BY "stormwind_rules";
Query OK, 0 rows affected (0.05 sec)

mysql> GRANT ALL PRIVILEGES ON wordpress.* TO "wp_user"@"%";
Query OK, 0 rows affected (0.04 sec)

mysql> FLUSH PRIVILEGES;
Query OK, 0 rows affected (0.04 sec)

mysql> EXIT;

```

Figura 5: Creación de la base de datos y usuario SQL.

Fase 4: Despliegue de la Aplicación (WordPress)

Se utilizaron manifiestos de Kubernetes (`.yaml`) para desplegar WordPress, configurando secretos y volúmenes persistentes.

Comandos utilizados:

```
# Obtener credenciales del clúster y preparar archivos
gcloud container clusters get-credentials griffin-dev --zone=$ZONE
gsutil cp -r gs://spl/spls/gsp321/wp-k8s .

# Crear Secretos para la DB y la Service Account
kubectl create secret generic cloudsql-db-credentials --from-
literal=username=wp_user --from-literal=password=stormwind_rules
gcloud iam service-accounts keys create key.json --iam-account=cloud-sql-
proxy@$GOOGLE_CLOUD_PROJECT.iam.gserviceaccount.com
kubectl create secret generic cloudsql-instance-credentials --from-file
key.json

# Automatización: Inyectar nombre de conexión SQL en el YAML
export INSTANCE_CONNECTION_NAME=$(gcloud sql instances describe griffin-dev-db
--format='value(connectionName)')
sed -i "s/YOUR_SQL_INSTANCE/$INSTANCE_CONNECTION_NAME/g" wp-deployment.yaml

# Aplicar configuraciones y despliegue
kubectl apply -f wp-env.yaml
kubectl apply -f wp-deployment.yaml
kubectl apply -f wp-service.yaml
```

```
STACK_TYPE: IPV4
student_01_ccc1cd50fe47@cloudshell:~ (qwiklabs-gcp-01-0ed45439799f)$ gcloud container clusters get-credentials griffin-dev --zone=us-central1-c
Fetching cluster endpoint and auth data.
kubeconfig entry generated for griffin-dev.
student_01_ccc1cd50fe47@cloudshell:~ (qwiklabs-gcp-01-0ed45439799f)$ cd ~
gsutil cp -r gs://spl/spls/gsp321/wp-k8s .
cd wp-k8s
Copying gs://spl/spls/gsp321/wp-k8s/wp-deployment.yaml...
Copying gs://spl/spls/gsp321/wp-k8s/wp-env.yaml...
Copying gs://spl/spls/gsp321/wp-k8s/wp-service.yaml...
/ [3 files] 2.1 KiB/ 2.1 KiB
Operation completed over 3 objects/2.1 KiB.
student_01_ccc1cd50fe47@cloudshell:~/wp-k8s (qwiklabs-gcp-01-0ed45439799f)$ kubectl create secret generic cloudsql-db-credentials \
--from-literal=username=wp_user \
--from-literal=password=stormwind_rules
secret/cloudsql-db-credentials created
student_01_ccc1cd50fe47@cloudshell:~/wp-k8s (qwiklabs-gcp-01-0ed45439799f)$ gcloud iam service-accounts keys create key.json \
--iam-account=cloud-sql-proxy@$GOOGLE_CLOUD_PROJECT.iam.gserviceaccount.com
student_01_ccc1cd50fe47@cloudshell:~/wp-k8s (qwiklabs-gcp-01-0ed45439799f)$ kubectl create secret generic cloudsql-instance-credentials \
--from-file key.json
created key [d523f6d3c1a0d244ada52bc15a56901fe1f94007] of type [json] as [key.json] for [cloud-sql-proxy@qwiklabs-gcp-01-0ed45439799f.iam.gserviceaccount.com]
secret/cloudsql-instance-credentials created
student_01_ccc1cd50fe47@cloudshell:~/wp-k8s (qwiklabs-gcp-01-0ed45439799f)$ kubectl apply -f wp-env.yaml
persistentvolumeclaim/wordpress-volumeclaim created
secret/database created
student_01_ccc1cd50fe47@cloudshell:~/wp-k8s (qwiklabs-gcp-01-0ed45439799f)$
```

Figura 6: Generación de secretos en Kubernetes y aplicación de manifiestos.

4. Resolución de Problemas (Troubleshooting)

Durante el despliegue, se encontró un error crítico al intentar acceder a la aplicación web.

Problema:

Al acceder a la IP externa del balanceador de carga, WordPress mostró el mensaje: "Error establishing a database connection".

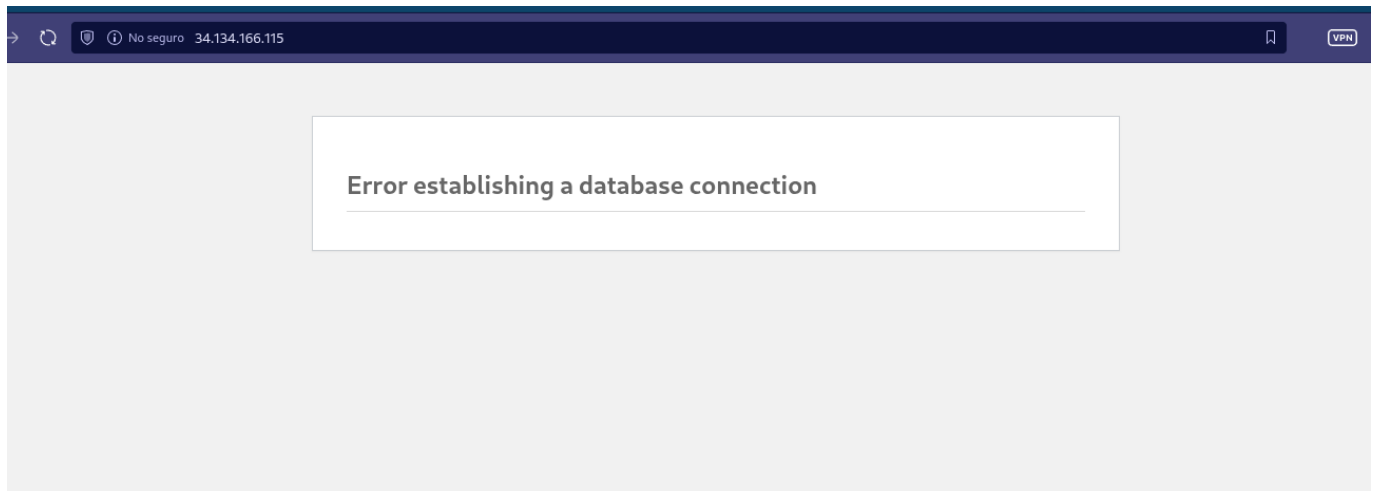


Figura 7: Error de conexión a la base de datos observado en el navegador.

Análisis y Solución:

El error indicaba que los Pods de WordPress no podían autenticarse con Cloud SQL.

1. **Verificación SQL:** Se confirmó que el usuario `wp_user` existía y tenía permisos (`GRANT ALL`).
2. **Recreación de Secretos:** Se sospechó de un error en el Secret de Kubernetes `cloudsql-db-credentials` . Se eliminó y recreó el secreto asegurando que no hubiera caracteres ocultos.
3. **Reinicio de Pods:** Se eliminaron los Pods existentes (`kubectl delete pods --all`) para forzar al `Deployment` a crear nuevos Pods que leyeran el secreto corregido.

Comandos de solución:

```
# Recrear el secreto de credenciales
kubectl delete secret cloudsql-db-credentials
kubectl create secret generic cloudsql-db-credentials --from-
literal=username=wp_user --from-literal=password=stormwind_rules

# Reiniciar pods para aplicar cambios
kubectl delete pods --all
```

Resultado: Tras el reinicio, la conexión se estableció correctamente y apareció el instalador de WordPress.

5. Monitoreo y IAM

Como pasos finales, se configuró una alerta de Uptime Check y se otorgaron permisos a un segundo ingeniero.

- **Nota Técnica:** Hubo un error inicial usando `gcloud monitoring uptime-check-configs` (comando deprecado). Se corrigió utilizando la sintaxis moderna `gcloud monitoring uptime create`.

Comandos utilizados:

```
# Crear Uptime Check (Comando corregido)
gcloud monitoring uptime create "WordPress Dev Check" \
  --resource-type=uptime-url \
  --resource-labels=host=[TU_IP_EXTERNA] \
  --path="/" \
  --period=1

# Asignar rol de Editor al segundo usuario
gcloud projects add-iam-policy-binding $GOOGLE_CLOUD_PROJECT \
  --member='user:[EMAIL_SEGUNDO_USUARIO]' \
  --role='roles/editor'
```

```
student_01_ccc1cd50fe47@cloudshell:~/wp-k8s (qwiklabs-gcp-01-0ed45439799f)$ # Crear el Deployment (los pods de WordPress)
student_01_ccc1cd50fe47@cloudshell:~/wp-k8s (qwiklabs-gcp-01-0ed45439799f)$ gcloud monitoring uptime-check-configs create "griffin-dev-uptime-check" \
  --display-name="WordPress Dev Check" \
  --resource-type=uptime-url \
  --resource-labels=host=34.134.166.115 \
  --path="/" \
  --period=1
ERROR: (gcloud.monitoring) Invalid choice: 'uptime-check-configs'.
Maybe you meant:
  gcloud monitoring uptime create
  gcloud monitoring uptime delete
  gcloud monitoring uptime describe
  gcloud monitoring uptime list-configs
  gcloud monitoring uptime list-ips
  gcloud monitoring uptime update

To search the help text of gcloud commands, run:
  gcloud help -- SEARCH_TERMS
student_01_ccc1cd50fe47@cloudshell:~/wp-k8s (qwiklabs-gcp-01-0ed45439799f)$ gcloud monitoring uptime create "WordPress Dev Check" \
  --resource-type=uptime-url \
  --resource-labels=host=34.134.166.115 \
  --path="/" \
  --period=1
Created uptime [projects/qwiklabs-gcp-01-0ed45439799f/uptimeCheckConfigs/wordpress-dev-check-DqJGK40qVEQ].
student_01_ccc1cd50fe47@cloudshell:~/wp-k8s (qwiklabs-gcp-01-0ed45439799f)$ gcloud projects add-iam-policy-binding $GOOGLE_CLOUD_PROJECT \
  --member='user:student-01-1333160d51d7@qwiklabs.net' \
  --role='roles/editor'
Updated IAM policy for project [qwiklabs-gcp-01-0ed45439799f].
bindings:
- members:
  - serviceAccount:qwiklabs-gcp-01-0ed45439799f@qwiklabs-gcp-01-0ed45439799f.iam.gserviceaccount.com
  role: roles/bigquery.admin
- members:
  - serviceAccount:257182518310@cloudbuild.gserviceaccount.com
  role: roles/cloudbuild.builds.builder
- members:
  - serviceAccount:service-257182518310@gcp-sa-cloudbuild.iam.gserviceaccount.com
  role: roles/cloudbuild.serviceAgent
- members:
```

Figura 8: Creación exitosa del Uptime Check y asignación de rol de Editor.

6. Cheat Sheet de Comandos Clave

Acción	Comando / Explicación
Crear VPC	<code>gcloud compute networks create [NOMBRE] --subnet-mode=custom</code>
Crear Subred	<code>gcloud compute networks subnets create ... --range=[CIDR]</code>
Instancia Multi-NIC	<code>gcloud compute instances create ... --network-interface=... --network-interface=...</code>
Conectar SQL	<code>gcloud sql connect [INSTANCIA] --user=root</code>
Credenciales GKE	<code>gcloud container clusters get-credentials [CLUSTER]</code>
Crear Secreto K8s	<code>kubectl create secret generic [NOMBRE] --from-literal=key=value</code>
Ver Servicios	<code>kubectl get services -w</code> (El flag <code>-w</code> observa cambios en tiempo real)
Edición rápida	<code>sed -i "s/viejo/nuevo/g" archivo</code> (Reemplazo de texto en línea)

7. Conclusiones

Este laboratorio demostró la importancia de la **Infraestructura como Código** al utilizar la línea de comandos para desplegar un entorno complejo.

Puntos clave aprendidos:

1. **VPCs Personalizadas:** Son esenciales para un control granular de la red, a diferencia de las redes automáticas.
2. **Troubleshooting en K8s:** Los errores de conexión suelen residir en la configuración de Secretos o en la necesidad de reiniciar los Pods para aplicar cambios.
3. **Interconexión de Servicios:** El uso del Proxy de Cloud SQL (sidecar) es fundamental para conectar GKE con Cloud SQL de forma segura sin exponer la base de datos a la red pública.
4. **Automatización:** Herramientas como `sed` y variables de entorno (`$REGION`) reducen el error humano significativamente.

Resultado Final:

El laboratorio fue completado con éxito obteniendo una puntuación de 100/100.

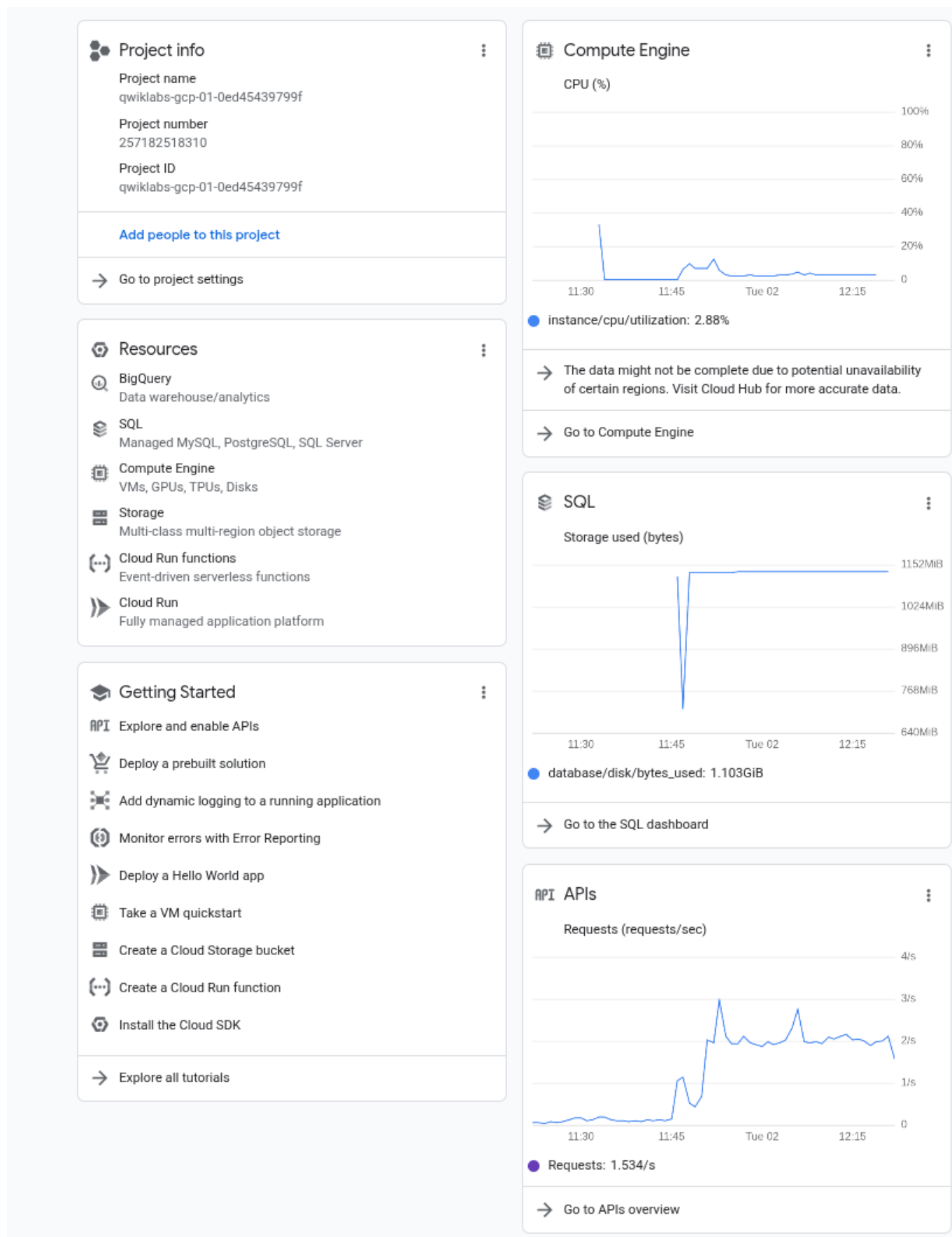


Figura 9: Estado final de los recursos en la consola de Google Cloud.