Conectar una app en GKE a Google Cloud SQL

Esta guía explica, en español y paso a paso, cómo conectar una aplicación que corre en Google Kubernetes Engine (GKE) a una instancia de Google Cloud SQL (Postgres o MySQL). Incluye opciones seguras (recomendación: *Workload Identity* y *Cloud SQL Proxy*), ejemplos de comandos gcloud/kubectl, un ejemplo de Deployment con sidecar del Cloud SQL Proxy y buenas prácticas.

Índice

1) Objetivo rápido

Conectar una aplicación en GKE a una base de datos administrada en Cloud SQL.
La forma recomendada y más segura es: **usar Cloud SQL Auth Proxy como sidecar + Workload Identity** (para evitar claves locales y reducir riesgos).

2) Arquitecturas posibles

- **Cloud SQL Proxy (sidecar)** — Recomendado: el proxy gestiona autent., cifrado y resolución del endpoint; la app se conecta a localhost dentro del pod.
- **Conexión por Private IP** — Recomendado si tu GKE y Cloud SQL están en la misma VPC; evita paso por internet.
- **Public IP + redes autorizadas** — Menos seguro; suele requerir abrir la IP pública del pod/cluster.

3) Requisitos previos

- Cuenta y proyecto de GCP con facturación activa.
- gcloud y kubectl instalados y autenticados (gcloud auth login, gcloud config set project PROJECT_ID).
- GKE creado o permisos para crear uno. Se recomienda crear el clúster con **Workload Identity** habilitado:

gcloud container clusters create CLUSTER --zone ZONE \

--workload-pool=${PROJECT_ID}.svc.id.goog

- Cloud SQL Admin API activada: gcloud services enable sqladmin.googleapis.com
- Tener la instancia Cloud SQL creada (Postgres o MySQL). Necesitarás su *instance connection name* con formato:

PROJECT_ID:REGION:INSTANCE_NAME

4) Crear la instancia Cloud SQL (resumen)

Puedes crear la instancia por Cloud Console o gcloud sql instances create. Por ejemplo (Postgres):

gcloud sql instances create my-postgres --database-version=POSTGRES_14 --region=us-central1

Crea la base de datos y el usuario dentro de la instancia (o usa el usuario postgres):

gcloud sql databases create gmemegen_db --instance=my-postgres

gcloud sql users set-password postgres --instance=my-postgres --password='TuPasswordFuerte'

Nota: si vas a usar *Cloud SQL Proxy* como sidecar NO necesitas añadir la IP pública de tu clúster a las redes autorizadas.

5) Autenticación recomendada: Workload Identity (paso a paso)

Por qué: evita crear y distribuir claves JSON; GKE hace el enlace entre una *Kubernetes Service Account* (KSA) y una *GCP Service Account* (GSA).

Asumiendo variables:

PROJECT_ID=mi-proyecto

GSA_NAME=cloudsql-client-gsa

KSA_NAME=cloudsql-ksa

NAMESPACE=default

INSTANCE_CONNECTION_NAME="${PROJECT_ID}:REGION:INSTANCE_NAME"

1) Crear la GCP Service Account (GSA)

gcloud iam service-accounts create ${GSA_NAME} \

--project=${PROJECT_ID} \

```
 --display-name="Cloud SQL Client SA"

GSA_EMAIL=${GSA_NAME}@${PROJECT_ID}.iam.gserviceaccount.com
```

2) Darle permisos mínimos: roles/cloudsql.client

```
gcloud projects add-iam-policy-binding ${PROJECT_ID} \
 --member="serviceAccount:${GSA_EMAIL}" \
 --role="roles/cloudsql.client"
```

roles/cloudsql.client permite que el proxy inicie conexiones hacia instancias Cloud SQL.

3) Crear la Kubernetes Service Account (KSA)

```
kubectl create serviceaccount ${KSA_NAME} --namespace ${NAMESPACE}
```

4) Permitir que la KSA use la GSA (binding Workload Identity)

```
gcloud iam service-accounts add-iam-policy-binding ${GSA_EMAIL} \
 --project=${PROJECT_ID} \
 --role roles/iam.workloadIdentityUser \
 --member "serviceAccount:${PROJECT_ID}.svc.id.goog[${NAMESPACE}/${KSA_NAME}]"


kubectl annotate serviceaccount --namespace ${NAMESPACE} ${KSA_NAME} \
 iam.gke.io/gcp-service-account=${GSA_EMAIL}
```

Con esto, cualquier pod que use serviceAccountName: ${KSA_NAME} podrá obtener credenciales temporales y el Cloud SQL Proxy podrá autenticar sin fichero JSON.

6) Alternativa (menos recomendable): usar la key JSON en un Secret

Si no puedes usar Workload Identity, puedes crear una GSA, descargar la key y montarla como Secret en el pod. **No lo recomiendo** para producción.

Comandos (si lo eliges):

```
gcloud iam service-accounts keys create cloudsql-sa-key.json --iam-account=${GSA_EMAIL}

kubectl create secret generic cloudsql-instance-credentials \
 --from-file=credentials.json=cloudsql-sa-key.json
```

En ese caso el sidecar del proxy **sí** necesita montar /secrets/cloudsql/credentials.json y usar -credential_file en su comando.

7) Crear un Secret Kubernetes con las credenciales de la base de datos

Las credenciales de la DB (usuario/contraseña/nombre BD) suelen guardarse en un Secret K8s:

```
kubectl create secret generic cloudsql-db-credentials \
```

```
  --from-literal=username=postgres \
  --from-literal=password='TuPasswordFuerte' \
  --from-literal=dbname=gmemegen_db
```

Para más seguridad considera usar **Secret Manager** y el *Secret Manager CSI Driver* en lugar de Secrets de K8s.

8) YAML ejemplo: Deployment con sidecar Cloud SQL Proxy (Workload Identity)

A continuación un Deployment mínimo; **sustituye** los placeholders PROJECT, REGION, INSTANCE_NAME, REPO/IMAGE, etc.

```
apiVersion: apps/v1
kind: Deployment
metadata:
  name: gmemegen
  labels:
    app: gmemegen
spec:
  replicas: 1
  selector:
    matchLabels:
      app: gmemegen
  template:
    metadata:
      labels:
        app: gmemegen
    spec:
      # Usa la KSA que tiene la anotación de Workload Identity
      serviceAccountName: cloudsql-ksa
      containers:
        - name: gmemegen
          image: REGION-docker.pkg.dev/PROJECT/REPO/gmemegen-app:v1
          imagePullPolicy: Always
          ports:
            - containerPort: 8080
```

```
    env:
     - name: DB_HOST
       value: 127.0.0.1       # el proxy escucha en localhost dentro del pod
     - name: DB_PORT
       value: "5432"        # Postgres -> 5432, MySQL -> 3306
     - name: DB_USER
       valueFrom:
         secretKeyRef:
           name: cloudsql-db-credentials
           key: username
     - name: DB_PASS
       valueFrom:
         secretKeyRef:
           name: cloudsql-db-credentials
           key: password
     - name: DB_NAME
       valueFrom:
         secretKeyRef:
           name: cloudsql-db-credentials
           key: dbname
   - name: cloudsql-proxy
     image: gcr.io/cloudsql-docker/gce-proxy:latest
     # Si usas Workload Identity NO pongas -credential_file
     command:
       - "/cloud_sql_proxy"
       - "-instances=PROJECT:REGION:INSTANCE_NAME=tcp:5432"
       - "-port=5432"
     # (opcional) expón el puerto para depuración; la app debe usar 127.0.0.1:5432
     ports:
       - containerPort: 5432
```

Explicación (línea a línea importante)

- serviceAccountName: cloudsql-ksa: permite al pod obtener credenciales GCP (Workload Identity).
- DB_HOST=127.0.0.1 y DB_PORT=5432: la app se conecta al proxy que corre en el mismo pod.
- cloudsql-proxy sin -credential_file: proxy usará ADC (Application Default Credentials) provistas por Workload Identity.
- Reemplaza PROJECT:REGION:INSTANCE_NAME por el *instance connection name* real.

9) Construir y subir la imagen de la app (resumen)

Ejemplo con Artifact Registry (puedes usar Docker Hub si prefieres):

```
# Variables

REGION=us-east1

PROJECT_ID=mi-proyecto

REPO=gmemegen

IMAGE=${REGION}-docker.pkg.dev/${PROJECT_ID}/${REPO}/gmemegen-app:v1


# Crear repo (una sola vez)

gcloud artifacts repositories create ${REPO} --repository-format=docker --
location=${REGION}


docker build -t ${IMAGE} .

docker push ${IMAGE}
```

Asegúrate de que el clúster pueda leer de Artifact Registry (permiso de lectura al servicio de nodo o usar Workload Identity para pulls).

10) Desplegar y comprobar

1. Aplicar la configuración:

```
kubectl apply -f deployment.yaml
```

2. Ver pods:

```
kubectl get pods -l app=gmemegen -w
```

3. Ver logs (ej. del proxy o de la app):

```
POD=$(kubectl get pods -l app=gmemegen -o jsonpath='{.items[0].metadata.name}')

kubectl logs $POD -c cloudsql-proxy

kubectl logs $POD -c gmemegen
```

Probar conexión a la BD desde tu máquina (opción 1: port-forward del proxy)

Redirige el puerto 5432 del pod (proxy) a tu máquina

kubectl port-forward $POD 5432:5432 -c cloudsql-proxy

En otra terminal, usa psql o mysql según corresponda

psql "host=127.0.0.1 port=5432 user=postgres dbname=gmemegen_db"

Probar ejecutando cliente dentro del pod (opción 2)

kubectl exec -it $POD -c gmemegen -- bash

dentro del contenedor puedes ejecutar psql/mysql si están instalados

11) Buenas prácticas de seguridad

- **Usar Workload Identity** en lugar de claves JSON.
- Dar permisos mínimos al GSA: roles/cloudsql.client (no cloudsql.admin salvo que lo necesites para administrar instancias).
- Preferir **Private IP** para Cloud SQL si tu GKE está en la misma VPC.
- Guardar credenciales de BD en **Secret Manager** o al menos en Secrets de K8s (no en ConfigMap ni en variables en texto plano).
- Rotar contraseñas y monitorizar accesos.

12) Problemas comunes y soluciones

- **Proxy no arranca**: revisa kubectl logs -c cloudsql-proxy. Verifica INSTANCE_CONNECTION_NAME y permisos del GSA.
- **Permiso denegado 403**: la GSA no tiene roles/cloudsql.client o el Workload Identity binding no está aplicado.
- **Conexión rechazada**: puerto equivocado (MySQL=3306, Postgres=5432) o la app intenta conectar a la IP pública en vez de 127.0.0.1.
- **Timeouts**: si usas Private IP, confirma que el clúster y la instancia están en la misma VPC o que hay peering.

13) Limpiar recursos (ejemplos)

kubectl delete deployment gmemegen

kubectl delete serviceaccount ${KSA_NAME}

gcloud iam service-accounts delete ${GSA_EMAIL}

Borra la instancia Cloud SQL solo si no la necesitas

#gcloud sql instances delete my-postgres

14) Resumen rápido de comandos (copiar/pegar)

```
# Variables (ajusta)
PROJECT_ID=mi-proyecto
REGION=us-east1
INSTANCE_CONNECTION_NAME=${PROJECT_ID}:us-east1:postgres-gmemegen
GSA_NAME=cloudsql-client-gsa
KSA_NAME=cloudsql-ksa
NAMESPACE=default

# 1) Crear GSA y dar rol mínimo
gcloud iam service-accounts create ${GSA_NAME} --project=${PROJECT_ID}
GSA_EMAIL=${GSA_NAME}@${PROJECT_ID}.iam.gserviceaccount.com
gcloud projects add-iam-policy-binding ${PROJECT_ID} --
member="serviceAccount:${GSA_EMAIL}" --role="roles/cloudsql.client"

# 2) Crear KSA y enlazar (Workload Identity)
kubectl create serviceaccount ${KSA_NAME} --namespace ${NAMESPACE}
gcloud iam service-accounts add-iam-policy-binding ${GSA_EMAIL} --role
roles/iam.workloadIdentityUser --member
"serviceAccount:${PROJECT_ID}.svc.id.goog[${NAMESPACE}/${KSA_NAME}]"
kubectl annotate serviceaccount --namespace ${NAMESPACE} ${KSA_NAME}
iam.gke.io/gcp-service-account=${GSA_EMAIL}

# 3) Secret con credenciales de BD
kubectl create secret generic cloudsql-db-credentials --from-literal=username=postgres -
-from-literal=password='TuPasswordFuerte' --from-literal=dbname=gmemegen_db

# 4) Desplegar
kubectl apply -f deployment.yaml
kubectl get pods -l app=gmemegen

# 5) Probar
POD=$(kubectl get pods -l app=gmemegen -o jsonpath='{.items[0].metadata.name}')
kubectl logs $POD -c cloudsql-proxy
```

```
kubectl port-forward $POD 5432:5432 -c cloudsql-proxy
psql "host=127.0.0.1 port=5432 user=postgres dbname=gmemegen_db"
```