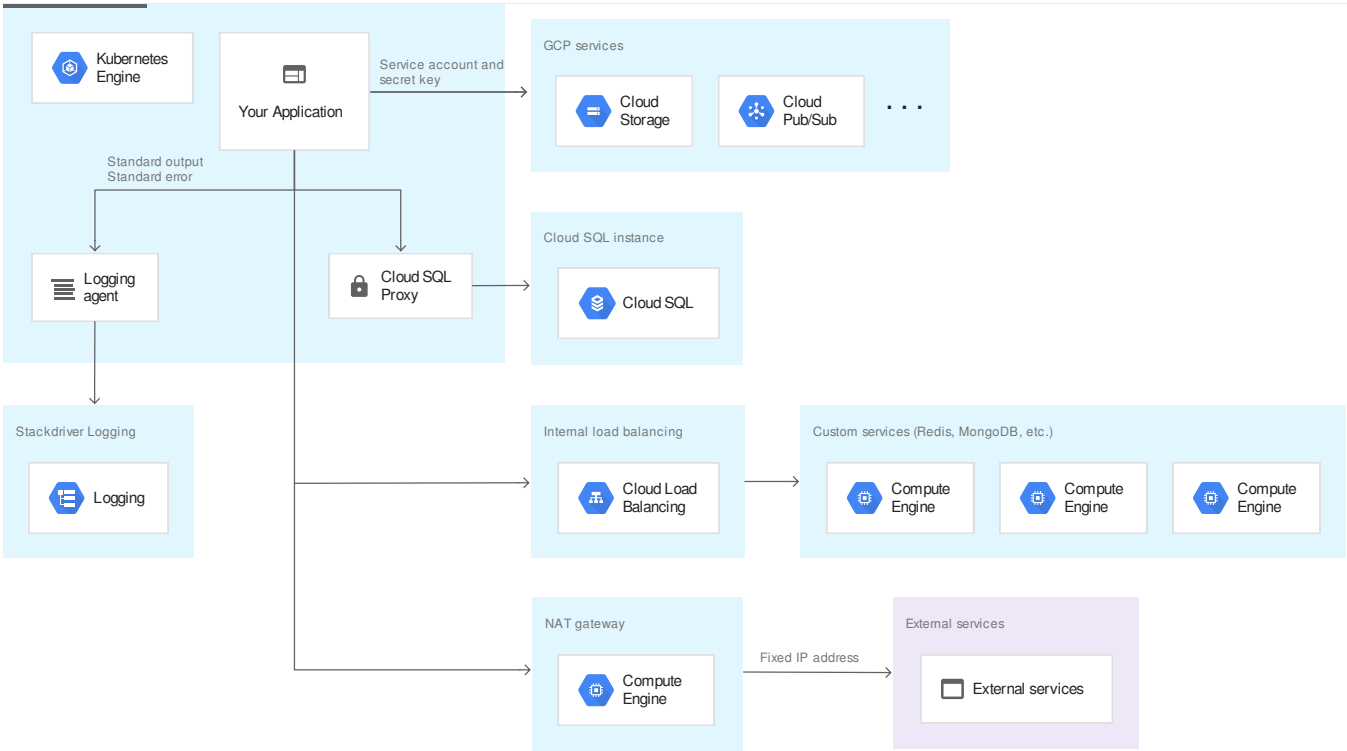
This document shows you how to use Google Cloud services from Google Kubernetes Engine (GKE). When you use Google Cloud services such as Cloud Storage or Cloud SQL from apps that run in GKE, you must configure your environment for the services that you use. This document explains common architectural patterns and their associated tasks, and provides links to documentation that explains example configurations.

- Configure a service account and a secret key to use Google Cloud services.
- Configure the Cloud SQL Proxy Docker image to use a Cloud SQL database.
- Configure internal load balancing to use custom services that run on Compute Engine VMs.
- Configure a NAT gateway to use external services that require a fixed IP address.
- Use Stackdriver Logging to record application logs.

The following diagram shows common architectural patterns of using other services from GKE.

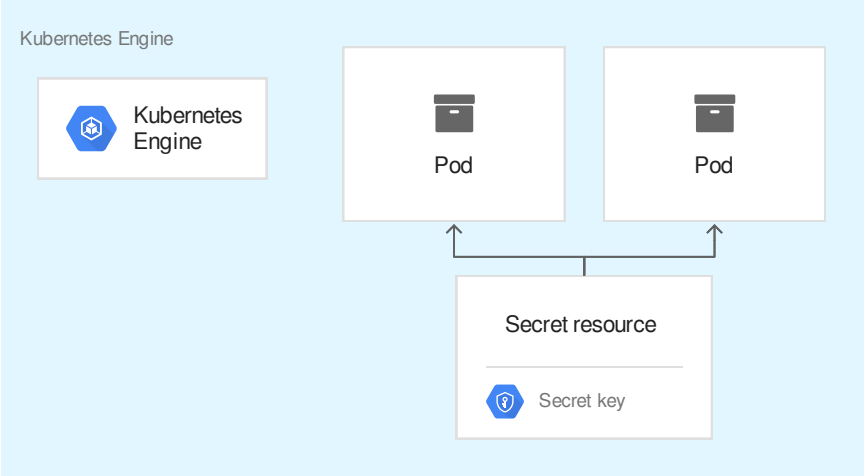


You can configure them with the following tasks:

- To use Google Cloud services such as Cloud Storage through the Cloud APIs, you assign an appropriate role (/iam/docs/understanding-roles) to the service account and provide the associated secret key to your app by using the Kubernetes `secret` object.
- To use Cloud SQL, you assign an appropriate role (/iam/docs/understanding-roles) to the service account, and add the Cloud SQL Proxy to your pod by using the sidecar pod pattern (https://kubernetes.io/blog/2015/06/the-distributed-system-toolkit-patterns/#example-1-sidecar-containers)
  .
- To use custom services that run on Compute Engine VMs in a scalable manner, you configure internal load balancing.
- To use external services that require a fixed IP address, you configure a NAT gateway.
- To record application logs in Logging, you have your app write log messages to standard output (`stdout`) and standard error (`stderr`).

The following sections have links to configuration steps.

You can use Google Cloud services through Cloud APIs by using a service account and a secret key. Kubernetes offers the `secret` resource type to store credentials inside the cluster and attach them to application pods, as shown in the following diagram:

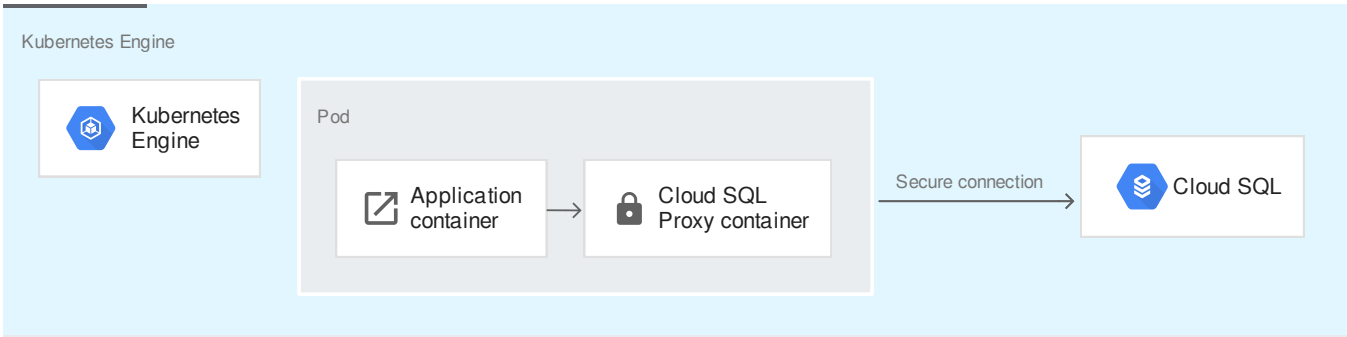


You must create a new service account for this purpose. You cannot reuse the service account that you created for G

For an example that shows how to use Pub/Sub from apps that run in GKE, see Authenticating to Cloud Platform with Service Accounts (/kubernetes-engine/docs/tutorials/authenticating-to-cloud-platform). You can apply the same steps for other Google Cloud services such as Cloud Storage, BigQuery, Datastore, and Spanner. However, you must choose an appropriate role for the service account and service, and you might need to perform steps that are specific to each service.

One exception to this approach is Cloud SQL. For that service, you need a different approach, because Cloud SQL requires Cloud SQL Proxy client in order to securely access a database, as explained in the next section.

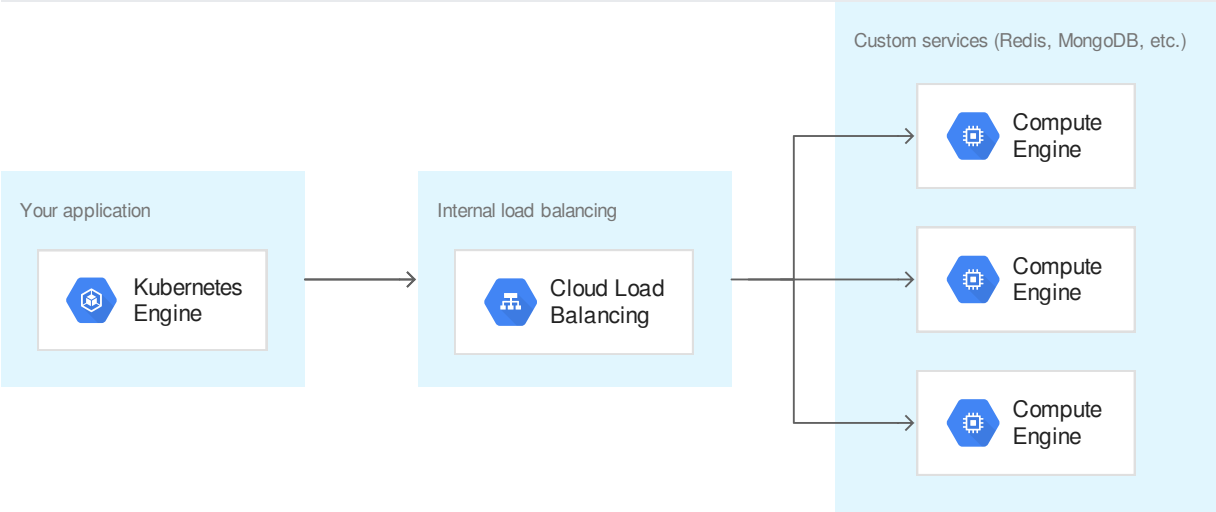
To access a Cloud SQL instance from an app that runs in GKE, you use the Cloud SQL Proxy Docker image. You attach the image to your application pod so that your app can use the Cloud SQL Proxy client in the same pod. The Cloud SQL Proxy client securely transfers the data between your app and the Cloud SQL instance, as shown in the following diagram:



For information about how to attach the Cloud SQL Proxy image to your application pod, see Cloud SQL: Connecting from GKE (/sql/docs/mysql/connect-kubernetes-engine).

To access external services from an app that runs in GKE, you use internal or external name services so that the app can discover the service endpoint. For an explanation of three ways to configure name services, see Connecting from inside a cluster to external services (/solutions/prep-kubernetes-engine-for-prod#connecting_from_inside_a_cluster_to_external_services).

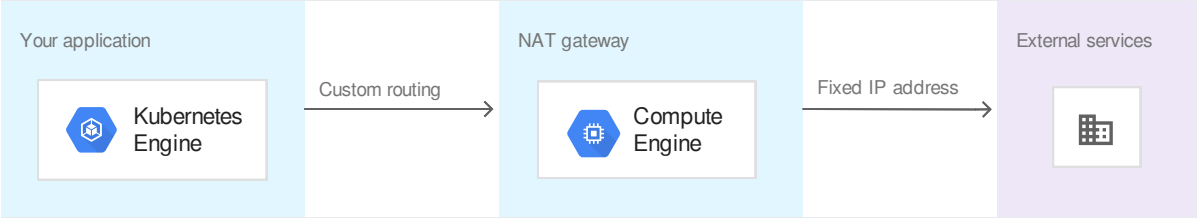
If the external service runs on Compute Engine instances, you might want to use internal load balancing to make the external service redundant and scalable. The following diagram illustrates this approach.



For information on how to set up internal load balancing for backend services that run Compute Engine instances, see Setting Up Internal Load Balancing (/compute/docs/load-balancing/internal/).

VM nodes that host the application pods send egress packets from apps that run in GKE. The VM nodes have ephemeral IP addresses that are used as the source IP address of egress packets. Because of this, the source IP address from the app might change depending on the VM node that sends the packets. As a result, external services receive packets from multiple source IP addresses even though the packets are sent from the same app. Under normal circumstances, this is not a problem. However, you might want to send packets from a fixed IP address, because some external services are configured to accept packets from only a single source.

In this scenario, you can use a Compute Engine instance that works as a NAT gateway. By creating custom routing rules for the NAT gateway, you can send packets to external services with a fixed IP address, as shown in the following diagram:



For more information, see Using a NAT Gateway with GKE (/solutions/using-a-nat-gateway-with-kubernetes-engine) and Build high availability and high bandwidth NAT gateways (/vpc/docs/special-configurations#multiple-natgateways). These articles explain how to deploy the NAT gateway instance and create custom routing rules.

You can apply the same architecture when you use a private cluster where VM nodes have only private IP addresses. In that case, the NAT gateway receives packets from a private subnet and transfers them to external services using a single public IP address.

Stackdriver Kubernetes Engine Monitoring is designed to monitor GKE clusters. It manages Stackdriver services together, including both Stackdriver Monitoring and Stackdriver Logging. It also features a console that provides a dashboard customized to GKE clusters.

For more information about Stackdriver Kubernetes Engine Monitoring, go to Overview of Stackdriver support for GKE (/monitoring/kubernetes-engine/).

Stackdriver supports two services to capture monitoring and logging data for a GKE cluster: Stackdriver Monitoring
itoring/kubernetes-engine/legacy-stackdriver/monitoring) and Stackdriver Kubernetes Engine Monitoring. If you're
y using Stackdriver Monitoring for an existing cluster, you can continue to do so, but be aware that this product is

cated. As a result, we recommend migrating to Stackdriver Kubernetes Engine Monitoring
itoring/kubernetes-engine/migration). If you're setting up a new configuration for monitoring a GKE cluster, we
mend Stackdriver Kubernetes Engine Monitoring, because it's the newer product and provides more features.

- Find more details in the GKE documentation (/kubernetes-engine/docs/).
- Read the solution Preparing a GKE Environment for Production (/solutions/prep-kubernetes-engine-for-prod).
- Read the solution Customizing Stackdriver Logs for GKE with Fluentd (/solutions/customizing-stackdriver-logs-fluentd).
- Try out the tutorials (/kubernetes-engine/docs/tutorials/) for GKE.
- Try out other Google Cloud features for yourself. Have a look at our tutorials (/docs/tutorials).