

Blog

Set up multi-cluster Anthos Service Mesh using managed control plane

Dave Cavaletto(<https://www.doit.com/author/dave-cavaletto/>)Date: July 7, 2022(<https://www.doit.com/2022/07/07/>)

This is a simple step-by-step guide to set up a multi-cluster Anthos Service Mesh (ASM) using the managed control plane.

The accompanying code can be found here:

<https://github.com/caddac/anthos-examples/tree/main/asm>

(<https://github.com/caddac/anthos-examples/tree/main/asm>)

What is a service mesh and why would you need one?

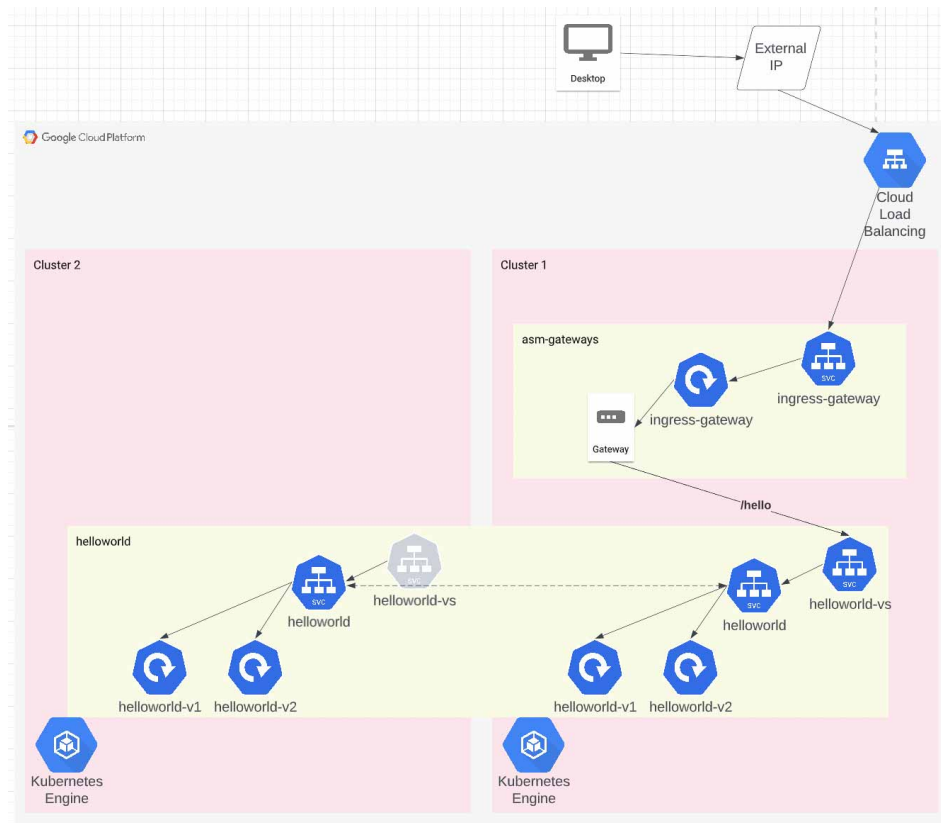
A service mesh is an abstraction built on top of various compute platforms – in this case clusters – to simplify inter-service communication, observability and security. It moves these responsibilities to the platform, facilitating a simpler application. In this blog post, we'll deploy Anthos Service Mesh, which is a managed Istio installation. Istio is a service mesh for kubernetes, a powerful tool to gain consistent insight into service-to-service observability and security. Using Istio, this is achieved by proxying all interservice communication through proxies run as sidecars to your pods.

Services can use Istio's consistent and transparent patterns to communicate with other services and don't need to concern themselves with how to reach another service, security from incoming requests, or publishing metrics about inter-service traffic. All of this is handled in Istio. These features are even more important when working with a multi-cluster configuration and operational overhead is even higher.

Using a service mesh does come with some drawbacks, including setup complexity, resource overhead and cost. Istio has many features and several ways to configure it, including `istioctl`, `IstioOperator` and `helm`. However, Anthos Service Mesh with managed control plane does not support the `IstioOperator` API or `helm`.

OK, here we go!

In this tutorial, we'll build two GKE clusters and deploy the istio hello-world application to them. Then we'll install a single ingress (on just one cluster) and show that traffic is load-balanced across the 4 hello-world pods running on two GKE clusters.



Deploy the terraform

In the `asm/infra` directory, update the values in the `var_inputs.auto.tfvars` file. Plan and apply it:


```
$ terraform init
$ terraform apply
```

Note: If the apply fails, try applying again. Sometimes terraform gets ahead of itself when applying so many dependent resources all at once. I had to apply it three times to get all the resources created.

There is a lot going on here, so let's cover it briefly.

- `infra.tf` (<https://github.com/caddac/anthos-examples/blob/main/asm/infra/infra.tf>)

This deploys a new GCP project and enables several required APIs [1].

Quick note here, the official docs seem to be missing the sts and anthos APIs, so make sure you enable those.  (https://www.doit.com/)

It also deploys a VPC, two subnetworks with secondary ranges and a firewall rule to let everything communicate. Finally, it enables the hub mesh feature for the project.

- cluster1.tf (https://github.com/caddac/anthos-examples/blob/main/asm/infra/cluster1.tf) and cluster2.tf (https://github.com/caddac/anthos-examples/blob/main/asm/infra/cluster2.tf)

Each file deploys a cluster. These differ only in the names of the clusters, regions and subnet references. Here are two things to note on the cluster config. First, we've set the `mesh_id` label so GCP knows which mesh this cluster belongs to. Second, we've enabled Workload Identity, which allows Kubernetes service accounts to act as GCP service accounts [2] without needing to download or even create JSON keys.

We register the cluster to the fleet, using the cluster name as the membership id.

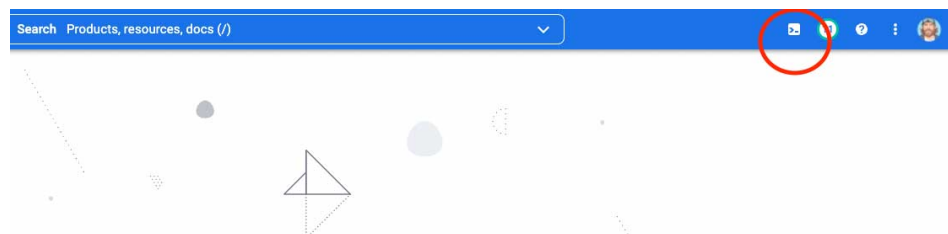
Finally, we install the managed control plane. We're using the automatic control-plane (which means "managed"), but which revision? Check the revision label on the namespace to see which revision of the control plane is being used [5]. Yes, it's that easy!

At this point, we have two clusters running in the same fleet, both with ASM installed using the managed control plane. So, although we can start installing Istio components, the clusters don't have cross-cluster service discovery working yet.

Configure cross cluster service discovery

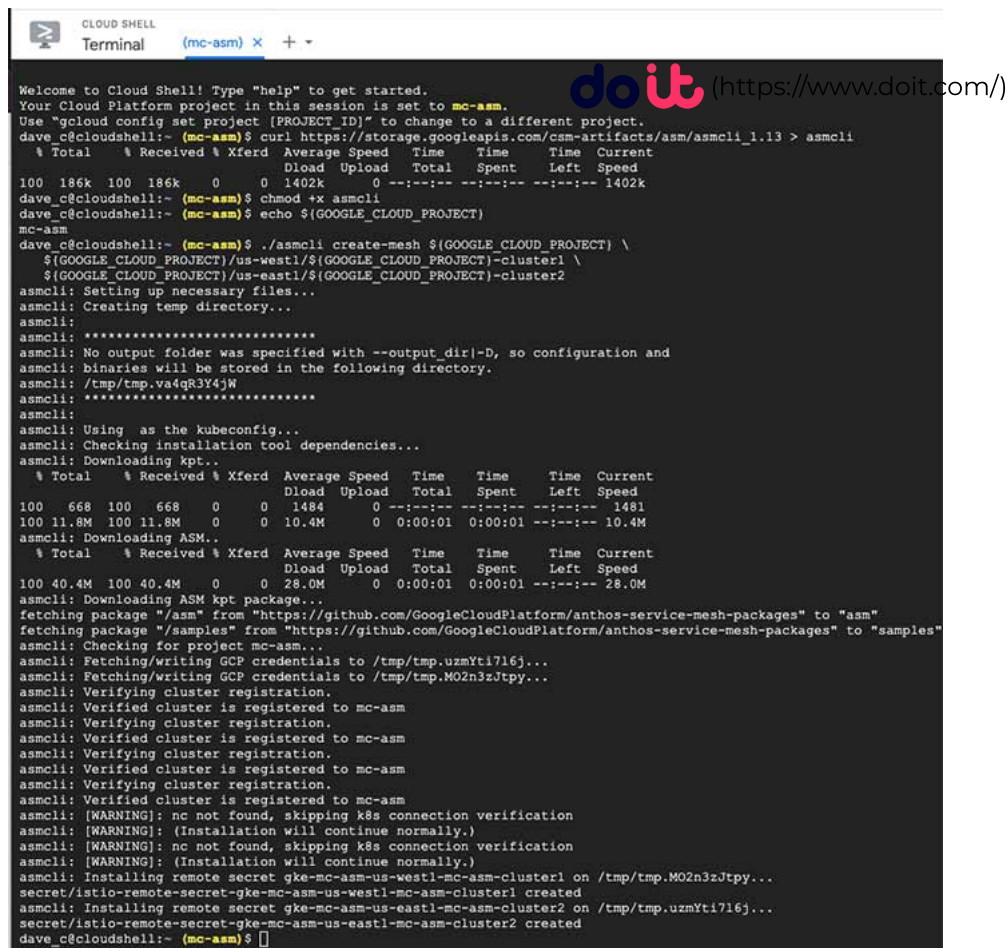
We need to install secrets for each cluster onto every other cluster in our fleet. Luckily, there is a cli tool for this – `asmcli`. Unfortunately, it doesn't work on macOS yet, so it's easiest to run it in the cloud console.

Open the cloud console in your project by browsing to the GCP console and clicking the cloud console button



After the console pops up at the bottom of the screen, run the following commands:

```
$ curl https://storage.googleapis.com/csm-artifacts/asm/asmcli_1.13 > asmcli
$ chmod +x asmcli
$ ./asmcli create-mesh ${GOOGLE_CLOUD_PROJECT} \
  ${GOOGLE_CLOUD_PROJECT}/us-west1/${GOOGLE_CLOUD_PROJECT}-cluster1 \
  ${GOOGLE_CLOUD_PROJECT}/us-east1/${GOOGLE_CLOUD_PROJECT}-cluster2
```



```

Welcome to Cloud Shell! Type "help" to get started.
Your Cloud Platform project in this session is set to mc-asm.
Use "gcloud config set project [PROJECT_ID]" to change to a different project.
dave c@cloudshell:~ (mc-asm)$ curl https://storage.googleapis.com/csm-artifacts/asm/asmcli_1.13 > asmcli
% Total % Received % Xferd Average Speed Time Time Time Current
Dload Upload Total Spent Left Speed
100 186k 100 186k 0 0 1402k 0 --:--:-- --:--:-- 1402k
dave c@cloudshell:~ (mc-asm)$ chmod +x asmcli
dave c@cloudshell:~ (mc-asm)$ echo $(GOOGLE_CLOUD_PROJECT)
mc-asm
dave c@cloudshell:~ (mc-asm)$ ./asmcli create-mesh $(GOOGLE_CLOUD_PROJECT) \
$(GOOGLE_CLOUD_PROJECT)/us-west1/$(GOOGLE_CLOUD_PROJECT)-cluster1 \
$(GOOGLE_CLOUD_PROJECT)/us-east1/$(GOOGLE_CLOUD_PROJECT)-cluster2
asmcli: Setting up necessary files...
asmcli: Creating temp directory...
asmcli:
asmcli: *****
asmcli: No output folder was specified with --output_dir|-D, so configuration and
asmcli: binaries will be stored in the following directory.
asmcli: /tmp/tmp.va4qR3Y4jW
asmcli: *****
asmcli:
asmcli: Using as the kubeconfig...
asmcli: Checking installation tool dependencies...
asmcli: Downloading kpt...
% Total % Received % Xferd Average Speed Time Time Time Current
Dload Upload Total Spent Left Speed
100 668 100 668 0 0 1484 0 --:--:-- --:--:-- 1481
100 11.8M 100 11.8M 0 0 10.4M 0 0:00:01 0:00:01 --:--:-- 10.4M
asmcli: Downloading ASM...
% Total % Received % Xferd Average Speed Time Time Time Current
Dload Upload Total Spent Left Speed
100 40.4M 100 40.4M 0 0 28.0M 0 0:00:01 0:00:01 --:--:-- 28.0M
asmcli: Downloading ASM kpt package...
fetching package "/asm" from "https://github.com/GoogleCloudPlatform/anthos-service-mesh-packages" to "asm"
fetching package "/samples" from "https://github.com/GoogleCloudPlatform/anthos-service-mesh-packages" to "samples"
asmcli: Checking for project mc-asm...
asmcli: Fetching/writing GCP credentials to /tmp/tmp.uzmYti716j...
asmcli: Fetching/writing GCP credentials to /tmp/tmp.M02n3zJtpy...
asmcli: Verifying cluster registration.
asmcli: Verified cluster is registered to mc-asm
asmcli: Verifying cluster registration.
asmcli: Verified cluster is registered to mc-asm
asmcli: Verifying cluster registration.
asmcli: Verified cluster is registered to mc-asm
asmcli: Verifying cluster registration.
asmcli: Verified cluster is registered to mc-asm
asmcli: [WARNING]: nc not found, skipping k8s connection verification
asmcli: [WARNING]: (Installation will continue normally.)
asmcli: [WARNING]: nc not found, skipping k8s connection verification
asmcli: [WARNING]: (Installation will continue normally.)
asmcli: Installing remote secret gke-mc-asm-us-west1-mc-asm-cluster1 on /tmp/tmp.M02n3zJtpy...
secret/istio-remote-secret-gke-mc-asm-us-west1-mc-asm-cluster1 created
asmcli: Installing remote secret gke-mc-asm-us-east1-mc-asm-cluster2 on /tmp/tmp.uzmYti716j...
secret/istio-remote-secret-gke-mc-asm-us-east1-mc-asm-cluster2 created
dave c@cloudshell:~ (mc-asm)$

```

Quick note here: If you are using private clusters, there are additional steps to enable cross-cluster service discovery [6][7].

Deploy the apps and Istio components

From the `asm/manifests` directory, we can install apps to test our mesh. One cluster 1, we're going to install the helloworld app and the Istio components for ingress.

Start by getting the contexts for the clusters:

```

export PROJECT_ID=<gcp_project_id>
gcloud container clusters get-credentials ${PROJECT_ID}-cluster1 --region us-west1 --
project ${PROJECT_ID}
gcloud container clusters get-credentials ${PROJECT_ID}-cluster2 --region us-east1 --
project ${PROJECT_ID}

```

Install cluster 1 by running:

```
kubectl apply -k manifests/ --context=gke_${PROJECT_ID}_us-west1_${PROJECT_ID}-cluster1
```

And cluster 2 by running:

```

kubectl apply -k manifests/app/ --context=gke_${PROJECT_ID}_us-east1_${PROJECT_ID}-
cluster2

```

```
~/src/anthos-examples/asm > on asm !3 79
kubectl apply -k manifests/ --context=gke_${PROJECT_ID}_us-west1_${PROJECT_ID}-cluster1
namespace/asm-gateways created
namespace/helloworld created
serviceaccount/istio-ingressgateway created
role.rbac.authorization.k8s.io/istio-ingressgateway created
rolebinding.rbac.authorization.k8s.io/istio-ingressgateway created
configmap/istio-asm-managed created
service/helloworld created
deployment.apps/istio-ingressgateway created
deployment.apps/helloworld-v1 created
deployment.apps/helloworld-v2 created
Warning: policy/v1beta1 PodDisruptionBudget is deprecated in v1.21+, unavailable in v1.25+; use policy/v1 PodDisruptionBudget
poddisruptionbudget.policy/istio-ingressgateway created
Warning: autoscaling/v2beta1 HorizontalPodAutoscaler is deprecated in v1.22+, unavailable in v1.25+; use autoscaling/v2beta2 HorizontalPodAutoscaler
horizontalpodautoscaler.autoscaling/istio-ingressgateway created
gateway.networking.istio.io/helloworld-gateway created
virtualservice.networking.istio.io/helloworld created
```



Review your work

After a few minutes, everything should be up, and you can inspect your work.

Verify all the helloworld pods are running both clusters:

```
$ export PROJECT_ID=gcp_project_id

$ kubectl get pods -n helloworld

--context=gke_${PROJECT_ID}_us-west1_${PROJECT_ID}-cluster1

$ kubectl get pods -n helloworld

--context=gke_${PROJECT_ID}_us-east1_${PROJECT_ID}-cluster2
```

```
~/src/anthos-examples/asm > on asm !3 79
kubectl get pods -n helloworld --context=gke_${PROJECT_ID}_us-west1_${PROJECT_ID}-cluster1
NAME                                READY   STATUS    RESTARTS   AGE
helloworld-v1-776f57d5f6-pr5tt      2/2     Running   0           111s
helloworld-v2-54df5f84b-w8vbk      2/2     Running   0           111s

~/src/anthos-examples/asm > on asm !3 79
kubectl get pods -n helloworld --context=gke_${PROJECT_ID}_us-east1_${PROJECT_ID}-cluster2
NAME                                READY   STATUS    RESTARTS   AGE
helloworld-v1-776f57d5f6-wv4lq      2/2     Running   0           70s
helloworld-v2-54df5f84b-lxcx9      2/2     Running   0           70s
```

Verify the ingress-gateway deployment is running on cluster 1

```
$ kubectl get deploy -n asm-gateways --context=gke_${PROJECT_ID}_us-
west1_${PROJECT_ID}-cluster1
```

```
~/src/anthos-examples/asm > on asm !3 79
kubectl get deploy -n asm-gateways --context=gke_${PROJECT_ID}_us-west1_${PROJECT_ID}-cluster1
NAME                                READY   UP-TO-DATE   AVAILABLE   AGE
istio-ingressgateway              3/3      3             3           2m51s
```

Verify the ingress-gateway service has an external IP on cluster 1

```
$ kubectl get svc -n asm-gateways --context=gke_${PROJECT_ID}_us-west1_${PROJECT_ID}-
cluster1
```

```
~/src/anthos-examples/asm > on asm !4 79
kubectl get svc -n asm-gateways --context=gke_${PROJECT_ID}_us-west1_${PROJECT_ID}-cluster1
NAME                                TYPE        CLUSTER-IP   EXTERNAL-IP   PORT(S)          AGE
istio-ingressgateway              LoadBalancer  10.10.13.245  34.145.41.204  15021:32354/TCP,80:31180/TCP,443:31197/TCP  50s
```

Finally, verify cross-cluster load balancing works by querying the public endpoint several times. You'll notice the version and pod id changes as different pods are hit. You should get 4 different pod ids and 2 different versions.

```
~/src/anthos-examples/asm > on asm !3 79
kubectl apply -k manifests/ --context=gke_${PROJECT_ID}_us-west1_${PROJECT_ID}-cluster1
namespace/asm-gateways created
namespace/helloworld created
serviceaccount/istio-ingressgateway created
role.rbac.authorization.k8s.io/istio-ingressgateway created
rolebinding.rbac.authorization.k8s.io/istio-ingressgateway created
configmap/istio-asm-managed created
service/helloworld created
deployment.apps/istio-ingressgateway created
deployment.apps/helloworld-v1 created
deployment.apps/helloworld-v2 created
Warning: policy/v1beta1 PodDisruptionBudget is deprecated in v1.21+, unavailable in v1.25+; use policy/v1 PodDisruptionBudget
poddisruptionbudget.policy/istio-ingressgateway created
Warning: autoscaling/v2beta1 HorizontalPodAutoscaler is deprecated in v1.22+, unavailable in v1.25+; use autoscaling/v2beta2 HorizontalPodAutoscaler
horizontalpodautoscaler.autoscaling/istio-ingressgateway created
gateway.networking.istio.io/helloworld-gateway created
virtualservice.networking.istio.io/helloworld created

$ curl <your external ip here>/hello
```




```

└─ curl 34.145.40.31/hello
Hello version: v1, instance: helloworld-v1-776f57d5f6-x8k7g
└─ ~ /src/anthos-examples/asm > on asm !4 79
└─ curl 34.145.41.204/hello
Hello version: v1, instance: helloworld-v1-776f57d5f6-wv4lq
└─ ~ /src/anthos-examples/asm > on asm !4 79
└─ curl 34.145.41.204/hello
Hello version: v1, instance: helloworld-v1-776f57d5f6-zv1v8
└─ ~ /src/anthos-examples/asm > on asm !4 79
└─ curl 34.145.41.204/hello
Hello version: v1, instance: helloworld-v1-776f57d5f6-wv4lq
└─ ~ /src/anthos-examples/asm > on asm !4 79
└─ curl 34.145.41.204/hello
Hello version: v1, instance: helloworld-v1-776f57d5f6-wv4lq
└─ ~ /src/anthos-examples/asm > on asm !4 79
└─ curl 34.145.41.204/hello
Hello version: v2, instance: helloworld-v2-54df5f84b-lxcx9
└─ ~ /src/anthos-examples/asm > on asm !4 79
└─ curl 34.145.41.204/hello
Hello version: v2, instance: helloworld-v2-54df5f84b-m48g6

```

Customizing, viewing logs

Enable envoy debug logs


We did this back when we deployed the asm configuration. Check out the file `asm/manifests/gateways/asm-config.yaml`. The ConfigMap we deployed configured the envoy proxies to log their access logs to stdout. There are other options you can enable in similar ways [3].

Control plane logs

You can still view control plane logs when using the managed control plane. They are in Logs Explorer under the resource “Istio Control Plane”

References

1. https://cloud.google.com/service-mesh/docs/managed/auto-control-plane-with-fleet#before_you_begin (https://cloud.google.com/service-mesh/docs/managed/auto-control-plane-with-fleet#before_you_begin)
2. <https://cloud.google.com/kubernetes-engine/docs/how-to/workload-identity> (<https://cloud.google.com/kubernetes-engine/docs/how-to/workload-identity>)
3. <https://cloud.google.com/service-mesh/docs/managed/enable-managed-anthos-service-mesh-optional-features> (<https://cloud.google.com/service-mesh/docs/managed/enable-managed-anthos-service-mesh-optional-features>)
4. <https://cloud.google.com/service-mesh/docs/managed/auto-control-plane-with-fleet> (<https://cloud.google.com/service-mesh/docs/managed/auto-control-plane-with-fleet>)
5. https://cloud.google.com/service-mesh/docs/managed/select-a-release-channel#how_to_select_a_release_channel (https://cloud.google.com/service-mesh/docs/managed/select-a-release-channel#how_to_select_a_release_channel)
6. <https://cloud.google.com/service-mesh/docs/unified-install/gke-install-multi-cluster#private-clusters-endpoint> (<https://cloud.google.com/service-mesh/docs/unified-install/gke-install-multi-cluster#private-clusters-endpoint>)
7. <https://cloud.google.com/service-mesh/docs/unified-install/gke-install-multi-cluster#private-clusters-authorized-network>

(<https://cloud.google.com/service-mesh/docs/unified-install/gke-install-multi-cluster#private-clusters-authorized-network>)
Leave a Reply  (<https://www.doit.com/>)

Your email address will not be published. Required fields are marked *

Comment *

Name

Email

Website

☐

Save my name, email, and website in this browser for the next time I comment.

☐

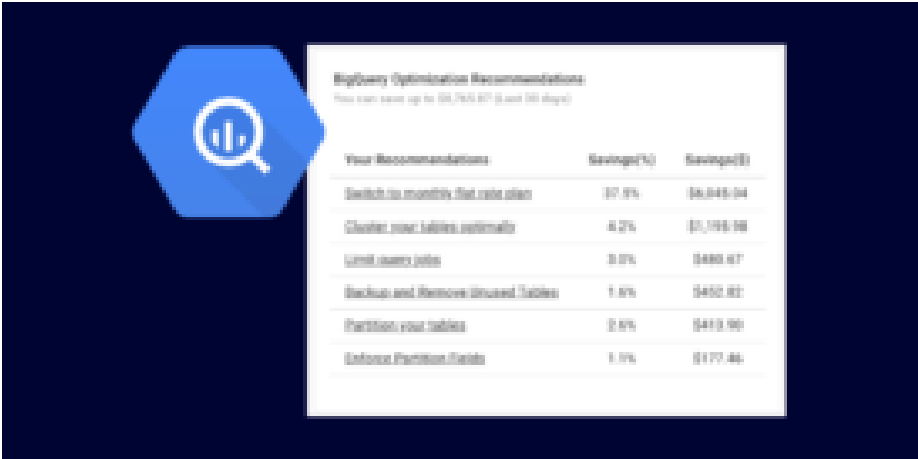
I'm not a robot

reCAPTCHA

[Privacy](#) - [Terms](#)

Post Comment

Related blogs



(<https://www.doit.com/putting-bigquery-cost-optimization-on-easy-mode-with-the-bigquery-lens/>)

Putting BigQuery cost optimization on “easy mode” with the BigQuery Lens (<https://www.doit.com/putting-bigquery-cost-optimization-on-easy-mode-with-the-bigquery-lens/>)

Keeping up with BigQuery cost optimization best practices is daunting – but what if you had a tool that

Like 1

Keep reading → (<https://www.doit.com/putting-bigquery-cost-optimization-on-easy-mode-with-the-bigquery-lens/>)



(<https://www.doit.com/google-unifies-its-data-analytics-programs-under-the-looker-brand/>)

Google unifies its data analytics programs under the Looker brand
(<https://www.doit.com/google-unifies-its-data-analytics-programs-under-the-looker-brand/>)

Google is unifying its business intelligence products by merging Looker and Data Studio to form Looker Studio. We explain the changes.

Like

Keep reading → (<https://www.doit.com/google-unifies-its-data-analytics-programs-under-the-looker-brand/>)



(<https://www.doit.com/design-options-for-google-cloud-certificate-authority-service/>)

Design options for Google Cloud Certificate Authority Service
(<https://www.doit.com/design-options-for-google-cloud-certificate-authority-service/>)

The Certificate Authority Service streamlines, automates and customizes the deployment, management and security of private certificate authorities (CA). We show you three different design options for your Google Cloud Certificate Authority Service.

Like

Keep reading → (<https://www.doit.com/design-options-for-google-cloud-certificate-authority-service/>)

[View all blogs](#)

([https://blog.doit-intl.com/?](https://blog.doit-intl.com/?__hstc=67060785.a04c53cc7e96f2179d2b49801973413f.1674565955037.1674565955037.1674724503608.2&__hssc=67060785.1)

[__hstc=67060785.a04c53cc7e96f2179d2b49801973413f.1674565955037.1674565955037.1674724503608.2&__hssc=67060785.1](https://blog.doit-intl.com/?__hstc=67060785.a04c53cc7e96f2179d2b49801973413f.1674565955037.1674565955037.1674724503608.2&__hssc=67060785.1)



Ready to get started?



[Contact us\(/contact/\)](#)

Subscribe to our newsletter

 (https://www.doit.com/)

in (https://www.linkedin.com/company/doitintl)

f (https://www.facebook.com/DoIT.International/)

▶ (https://www.youtube.com/@doitintl)

🐦 (https://twitter.com/doitintl)