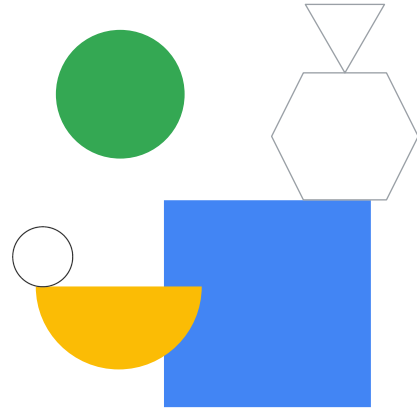


Serverless on Anthos



Welcome to Serverless on Anthos.

Learning objectives



Understand

Understand the benefits of serverless on both Google Cloud and Anthos clusters, so that you can speed up innovation everywhere.



Learn

Learn to deploy Cloud Run applications on Google Cloud and Anthos clusters and learn about the open-source technologies behind them.



Configure

Configure Cloud Run applications so that you can get all the flexibility in your service configuration with the highest levels of infrastructure automation.




Communicate

Communicate with applications, cloud services, protocols, and event sources without lock-in with Google's Eventarc service.

Google Cloud

In this module, you:

- Understand the benefits of serverless on both Google Cloud and Anthos clusters so that you can accelerate innovation everywhere.
- Learn to deploy Cloud Run applications on Google Cloud and Anthos clusters, and learn about the open-source technologies behind them.
- Configure Cloud Run applications so that you can get all the flexibility in your service configuration with the highest levels of infrastructure automation.
- Communicate with applications, cloud services, protocols, and event sources without lock-in with Google's Eventarc service.




Today's agenda

- 01 The need for serverless
- 02 The Cloud Run and Knative model
- 03 Cloud Run on Google Cloud
- 04 Cloud Run for Anthos
- 05 Serverless eventing



Google Cloud

Here is our agenda for the module.



Today's agenda

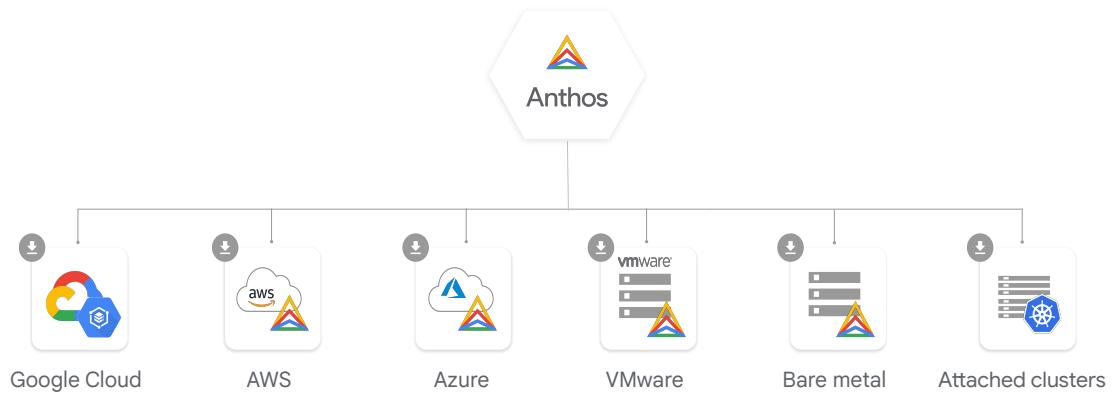
- 01 [The need for serverless](#)
- 02 The Cloud Run and Knative model
- 03 Cloud Run on Google Cloud
- 04 Cloud Run for Anthos
- 05 Serverless eventing



Google Cloud

Let's talk about the need for serverless.

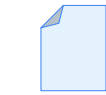
Anthos manages Kubernetes clusters anywhere



Google Cloud

As we saw, Anthos allows you to run Kubernetes on multiple environments as a managed service—installing, configuring, and updating your cluster—so you can focus on building your business logic.

Running production-grade Kubernetes applications is complex



Deployment.yaml



HorizontalPodAutoscaler.yaml



LivenessReadinessProbes.yaml



Service.yaml



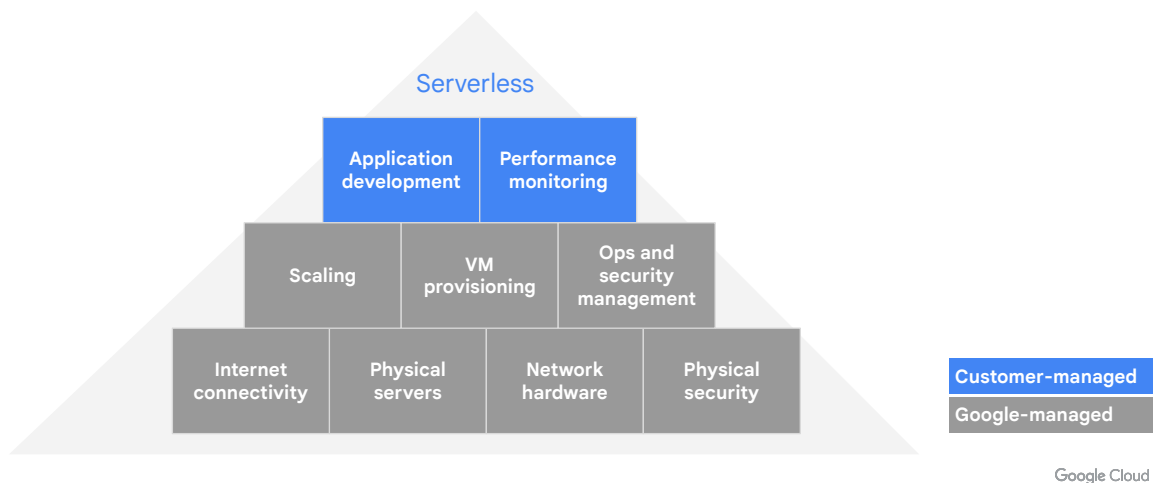
Ingress.yaml

Google Cloud

However, different companies build applications on Kubernetes in different ways, and there is a steep learning curve to understand and run production-ready Kubernetes applications.

After containerizing your application, you must decide how it will be deployed and upgraded, how to set up automatic scaling with horizontal or vertical autoscalers, and how to configure self-healing applications with liveness and readiness probes. Additionally, you must configure the networking layer with services and ingress resources and add complicated workflows in case you want to use advanced deployment strategies, such as canary deployments or perform A/B testing.

Developers want to focus on application development instead of infrastructure

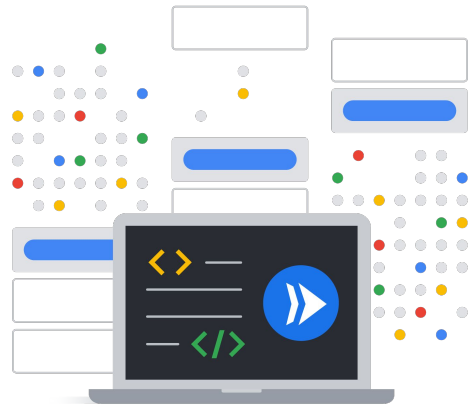


Developers want to focus on application development instead of infrastructure. When developers are working with serverless solutions in Google Cloud such as Cloud Functions, they can focus on developing their code and monitoring their applications after they are deployed, while Google takes care of the rest, such as scaling, networking, security, software patches, and upgrades.

Wouldn't it be great if we could have a solution that allows you to deploy applications on Kubernetes but manage them as if they were serverless applications?


Cloud Run is a serverless offering for deploying containerized apps on Google Cloud and Anthos clusters

- 1 **Automatic scaling** to the number of containers to handle all incoming requests
- 2 **Incremental application updates**, switching traffic gradually with easy rollback
- 3 **Load balancing** across zones and regions



Google Cloud

That solution is Cloud Run, a serverless offering for deploying containerized applications on Google Cloud and Anthos clusters. Cloud Run simplifies application management by taking care of your containerized solution and scaling the number of instances automatically based on incoming requests. You don't need to move traffic from one version to another when you deploy a new version to production because Cloud Run manages that for you, and you can specify how traffic should be split between versions. You can send a small part of the traffic to the new version to see whether any problems arise and, therefore, minimize customer disruption. With minimum effort, you can balance the load from Cloud Run applications that run on different regions and zones, which makes scaling easy while maintaining low latencies for your customers.



Today's agenda

- 01 The need for serverless
- 02 [The Cloud Run and Knative model](#)
- 03 Cloud Run on Google Cloud
- 04 Cloud Run for Anthos
- 05 Serverless eventing



Google Cloud

Let's talk about the Cloud Run and Knative model.

Cloud Run is based on Knative Serving, which makes apps portable



Cloud Run is based on open-source Knative Serving, which makes apps more portable.

Portability is important for application developers because:

- Data sovereignty might require you to run the application in a geographical region where Google Cloud has no physical presence.
- The developer might want to avoid vendor lock-in.

Applications on Cloud Run are portable in two ways:

- Cloud Run uses containers. (You already learned that containers can run anywhere, which makes your application inherently portable.)
- The *platform* Cloud Run is API-compatible with Knative, an open source project; it is an entirely different implementation of the same API (which means that Cloud Run is *not* managed Knative).

Knative is built on top of Kubernetes and leverages Istio for networking.

Cloud Run and Knative model

Overview

Service

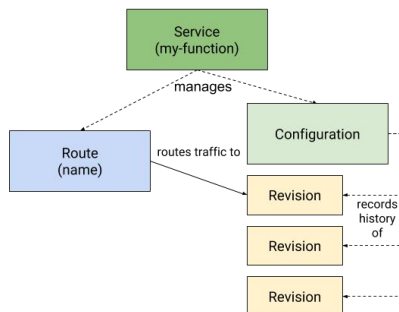
Route

Configuration

Revision

The modular architecture provides:

- Rapid deployment of serverless apps
- Automatic scaling up and down (even to zero)
- Routing and network programming for applications
- Point-in-time snapshots of deployed code and configurations



Google Cloud

Cloud Run and Knative provide modular architecture that allows:

- Rapid deployment of serverless applications
- Automatic scaling up to support millions of requests and down to zero instances to reduce consumption when it's not needed
- Routing and network programming for applications
- Point-in-time snapshots of deployed code and configurations that simplify managing rollouts and rollbacks.

The service manages the lifecycle of the workload

Overview

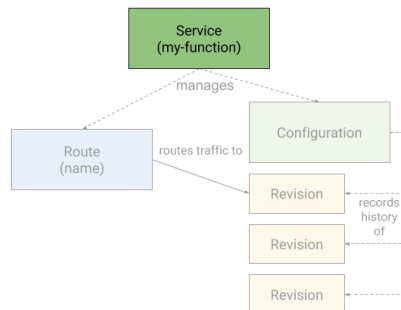
Service

Route

Configuration

Revision

- The service controls the creation of other objects, such as routes and revisions.
- Users (or other services) typically access services via HTTP/S connections to their Service URL; that is, `http://{service}.{namespace}.example.com`



Google Cloud

The service component in the Knative model manages the lifecycle of the workload and controls the creation of routes and revisions. With a service, you can configure applications that are accessible both internally and externally.

Knative Service example definition

Overview

Service

Route

Configuration

Revision

```
apiVersion: serving.knative.dev/v1
kind: Service
metadata:
  name: helloworld-go
  namespace: default
spec:
  template:
    spec:
      containers:
        - image: gcr.io/knative-samples/helloworld-go
          env:
            - name: TARGET
              value: "Go Sample v1"
```

Google Cloud

This is an example of Knative Service. It's a custom resource definition that extends the Kubernetes API. This looks very similar to a Kubernetes deployment, with similar fields like the name, namespace, or container image. However, it provides a static network identity, configures autoscaling, and is ready to serve traffic.

The route maps the Service URL to a specific revision

Overview

Service

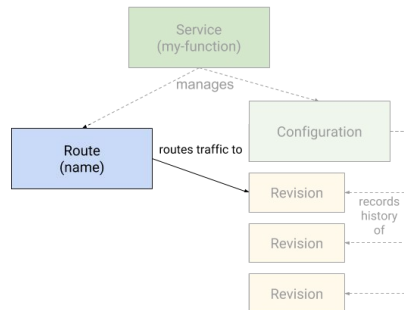
Route

Configuration

Revision

Traffic can be steered via:

- Named routes, which can be pinned to a specific revision.
- Fractional traffic rules to divide traffic between revisions.



Google Cloud

The route component maps the Service URL to a specific revision. Traffic can be steered based on the name routes, which point to a specific revision, or based on weights, which define the percentage of requests going to a specific destination.

Knative Route example definition

Overview

Service

Route

Configuration

Revision

```
apiVersion: serving.knative.dev/v1
kind: Service
metadata:
  name: helloworld-go
  namespace: default
spec:
  ...
  traffic:
    - percent: 0
      revisionName: example-service-1
      tag: staging
    - percent: 40
      revisionName: example-service-2
    - percent: 60
      revisionName: example-service-3
```

Google Cloud

In this example, a Knative Route extends the Service to provide traffic configuration based on the revision name and the traffic percent.

The configuration specifies the desired state of the service

Overview

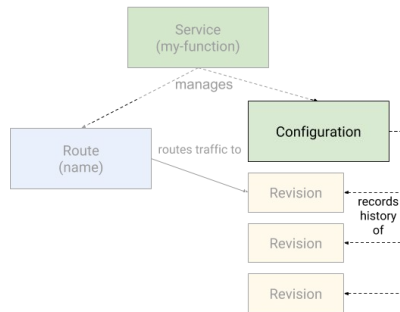
Service

Route

Configuration

Revision

- The configuration provides separation between the code and configuration data.
- Modifying the configuration data will automatically create a new revision.



Google Cloud

The configuration specifies the desired state of the Service. It provides separation between the code and configuration data. Modifying the configuration data will automatically create a new revision.

Revision is a point-in-time snapshot of the Service

Overview

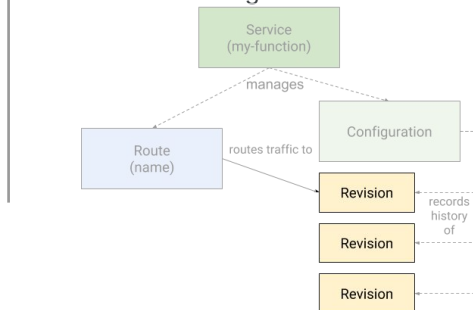
Service

Route

Configuration

Revision

- Revisions represent the state of the code and the corresponding configurations.
- Changes to the code or the configuration will generate a new revision.
- Revisions are immutable but might be removed.
- Each revision might have its own autoscaling configuration.



Google Cloud

A revision is a point-in-time snapshot of the Service.

- Revisions represent the state of the code and the corresponding configurations.
- Changes to the code or the configuration will generate a new revision.
- Revisions are immutable but might be removed.
- Each revision might have its own autoscaling configuration.

Revision is a point-in-time snapshot of the Service

Overview
Service
Route
Configuration
Revision


```
apiVersion: serving.knative.dev/v1
kind: Service
metadata:
  name: helloworld-go
spec:
  ...
  metadata:
    annotations:
      autoscaling.knative.dev/class: "kpa.autoscaling.knative.dev"
      autoscaling.knative.dev/target: "70"

apiVersion: v1
kind: ConfigMap
metadata:
  name: config-autoscaler
  namespace: knative-serving
data:
  enable-scale-to-zero: "true"
```

Google Cloud

This is an example of the autoscaling configuration for a given revision. You can either use the Knative Horizontal Autoscaler, which is enabled by default and scales to zero, or install the Kubernetes Horizontal Autoscaler, which does not scale down to zero but provides CPU-based autoscaling. The autoscaler can be configured per service, per revision, or per cluster, and it provides several options such as:

- The ability to scale down to zero
- The amount of concurrent requests per revision
- The time to wait before performing a downscaling event



Today's agenda

- 01 The need for serverless
- 02 The Cloud Run and Knative model
- 03 [Cloud Run on Google Cloud](#)
- 04 Cloud Run for Anthos
- 05 Serverless eventing



Google Cloud

Let's discuss Cloud Run in the context of Google Cloud.

Cloud Run: Serverless development platform for hybrid and multi-cloud environments



Deploy in seconds



Simplify development
complexity



Automatic HTTPS and
custom domains



No cluster management

Google Cloud

Cloud Run is a serverless development platform for hybrid and multi-cloud environments.

- Write code your way by deploying any code or container that listens for requests or events. Build applications in your favorite language, with your favorite dependencies and tools, and deploy them in seconds.
- Cloud Run abstracts away the complexity of underlying infrastructure so that you can easily develop locally and build and deploy apps anywhere.
- Cloud Run offers a stable HTTPS endpoint with TLS termination handled for you, and the option to map your services to your own domains.
- Cloud Run abstracts away all infrastructure management by automatically scaling up and down from zero almost instantaneously, depending on traffic. Cloud Run only charges you for the exact resources you use.

Local development and deployment with Cloud Run

- Run Cloud Run services in a local emulator.
- Local emulator available in gcloud, Cloud Code for VS Code and IntelliJ, and Cloud Shell Editor.
- Use a single command to build, push, and deploy local sources.
- Uses buildpacks, Cloud Build, and Artifact Registry.

```
$ gcloud beta code dev
Starting development environment
'gcloud-local-dev' ...

Building...
Starting deploy...
Press Ctrl+C to exit
Watching for changes...

Service URL: http://localhost:8080/
```

```
$ gcloud run deploy
Deploying from source.
Building using Buildpacks and deploying
container to Cloud Run service [my-service] in
project [my-project] region [europe-west1]

.: Building and deploying...
  ✓ Uploading sources...
  .: Building Container...
  . Creating Revision...
  . Routing traffic...
```

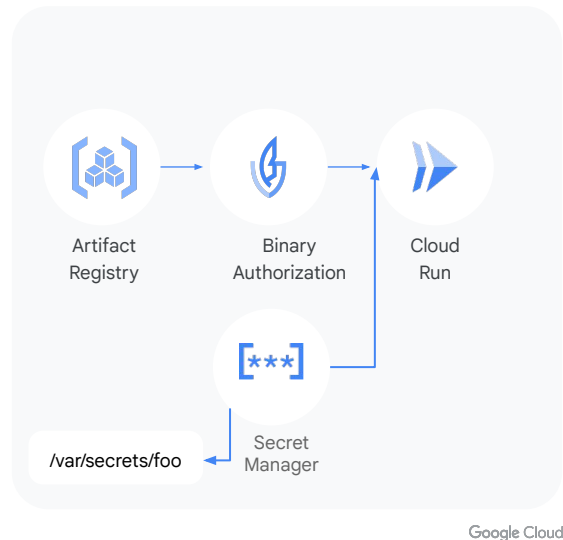
Google Cloud

With Cloud Run, you have an integrated end-to-end experience, from developing and testing your application locally to deploying it in the cloud. Cloud Run provides an emulator that is available via gcloud, Cloud Code for VS Code and IntelliJ, and Cloud Shell Editor, so that you can run your application locally without an internet connection.

Additionally, gcloud provides a single command to build, push, and deploy your application. The command “gcloud run deploy” uses either your Dockerfile or buildpacks as the source to build your container, Cloud Build to perform the actual build execution, and Artifact Registry to store the container image. Then, it deploys the container image into Cloud Run.

Secure CI/CD pipelines when deploying to Cloud Run

- Use Cloud Build to create a Binary Authorization attestation.
- Binary Authorization asserts the provenance of the image through signature validation, thus allowing only verified images to be deployed.
- Store secrets in Secret Manager and provide them to the application that is running directly in Cloud Run.



Securing your CI/CD pipelines when deploying to Cloud Run is easy because all Google products are deeply integrated.

You can use Cloud Build to build the application and create a Binary Authorization attestation to prove that the container was built here.

Binary Authorization asserts the provenance of the image through signature validation, thus allowing only verified images to be deployed.

Store secrets in Secret Manager and provide them to the application that is running directly in Cloud Run.

Serving web traffic with custom domains

- Services respond to HTTP and HTTPS requests sent to the Service URL.
- Cloud Run automatically generates an SSL certificate.
- Default domain example:
`https://gateway-12345-uc.a.run.app`
- For custom domains use:
 - A global external HTTP(S) load balancer
 - Firebase Hosting
 - Cloud Run domain mapping (limited availability)

Add mapping

You can map domains and subdomains to the selected Cloud Run service. [Learn more](#)

✓ Select or enter domain — ② Verify — ③ Update DNS records

Select a service to map to *
helloworld (us-central1) ▼

❗ Domain verification is in progress. It might take a few minutes for changes to propagate. Refresh once you finish the verification steps, or go back to [Webmaster Central](#) to review the instructions.

[REFRESH](#)

Select a verified domain
Verify a new domain... ▼

Base domain to verify
api.haggconsulting.com

e.g. to map "api.example.com", you need to verify "example.com".

[CONTINUE VERIFICATION AND CLOSE](#)

Google Cloud

To serve web traffic, Cloud Run offers a generated domain that you can access over HTTP and HTTPS. This domain automatically generates the SSL certificate. However, if you want to serve traffic on the internet, you probably want to set up a custom domain.

You have three options for setting up custom domains:

- Use a global external HTTP(S) load balancer if you plan to expand globally or want to configure Cloud CDN for caching and Google Cloud Armor for additional security.
- Use Firebase Hosting if you plan to stay in Google Cloud and have a more integrated development experience.
- Use Cloud Run domain mapping, which is the most straightforward option. However, it also has limited availability in terms of the regions that are available, and it only supports subdomains and not path domains.

Cloud Run service access controls

- By default, your Cloud Run services are inaccessible. You must grant users or services access via Identity and Access Management (IAM).
- The Cloud Run Invoker role determines which members can access the service.
- For additional security when using Cloud Run internally, use VPC Service Controls.
- For anonymous access, add the allUsers member to the Cloud Run Invoker role.
- Define allowed ingress source access to your Cloud Run services.

Authentication ?

- ☐ Allow unauthenticated invocations
Check this if you are creating a public API or website.
- ☒ Require authentication
Manage authorized users with Cloud IAM.

Ingress ?

- ☐ Allow all traffic
- ☐ Allow internal traffic and traffic from Cloud Load Balancing
- ☒ Allow internal traffic only

Google Cloud

By default, your Cloud Run services are inaccessible. You must grant users or services access via Identity and Access Management.

The Cloud Run Invoker role determines which members can access the service.

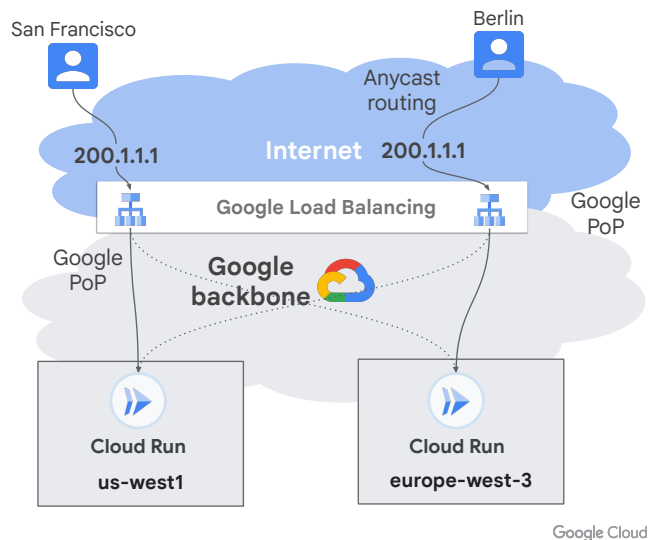
For additional security when using Cloud Run internally, use VPC Service Controls. Add your Cloud Run instances to a VPC Service Control, a Google Cloud feature that allows you to set up a secure perimeter to guard against data exfiltration.

For anonymous access, add the allUsers member to the Cloud Run Invoker role.

Additionally, you can define allowed ingress source access to your Cloud Run services; define whether you want to access all traffic, only traffic from the Cloud load balancer, or only internal traffic from the same VPC.

Cloud Run enables you to deploy globally

- Cloud Run is a regional resource, replicated across zones.
- Projects might contain multiple Services tied to different regions.
- Google Load Balancing is used to scale globally using a serverless network endpoint group.
- Google Load Balancer integrates with other services, such as Google Cloud Armor, Cloud DNS, and Cloud CDN.



Cloud Run enables you to deploy globally.

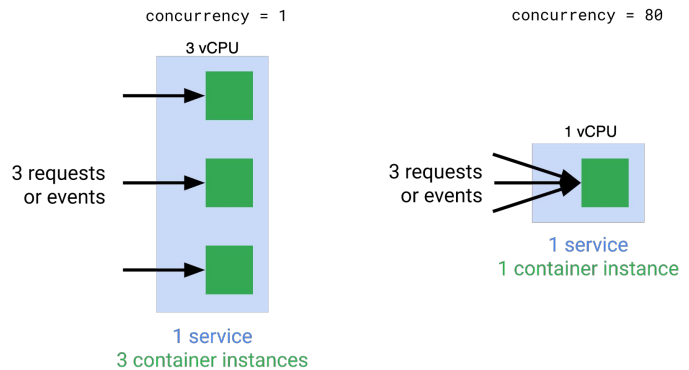
When you launch a service, it's deployed regionally and replicated across zones.

If you want to scale to multiple regions, you can deploy multiple Cloud Run services and use Google Load Balancing with serverless NEGs that point to the Cloud Run services.

Google Load Balancing integrates with other services, such as Google Cloud Armor, Cloud DNS, and Cloud CDN

Cloud Run enables you to scale

- Cloud Run automatically scales instances based on vCPU load and the concurrency configuration.
- Cloud Run provides concurrency features:
 - One instance can serve up to 1000 requests simultaneously.
 - Reduce concurrency to 1 for non-thread-safe code.
 - Lower concurrency might affect performance and increase costs.



Google Cloud

Cloud Run automatically scales the number of container instances in a region based on the vCPU load and the concurrency value in the Service's configuration data. This helps save costs because you only pay for the time that your instances run.

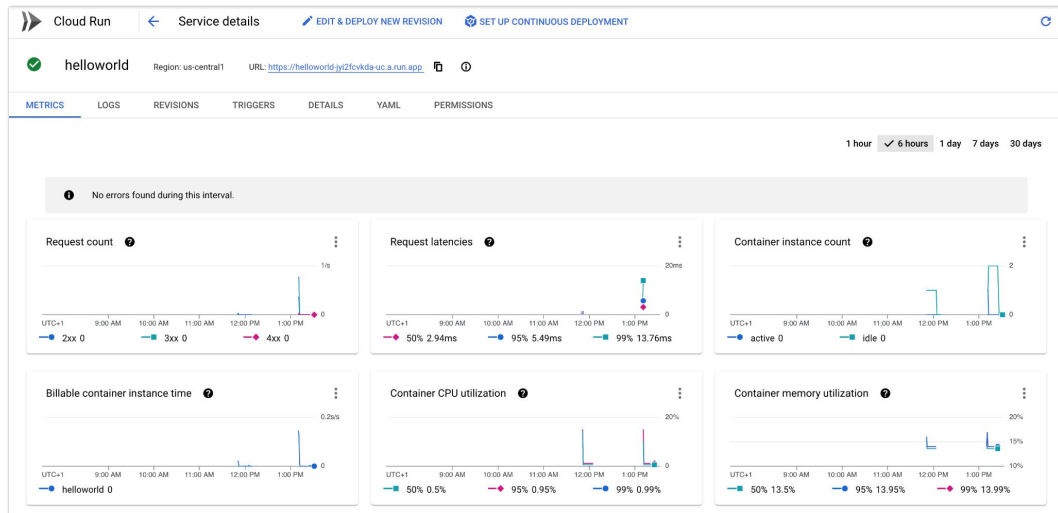
One instance can serve up to 1000 requests simultaneously, which is also the default concurrency for the service.

If the container's vCPU is under heavy load, Cloud Run might scale up the service regardless of the concurrency value.


Workloads that are not thread safe may require a value of 1. However, limiting concurrency to 1 will negatively affect performance and increase costs due to the number of containers required. A container can handle a new thread more easily than a new container can be started, handle a thread, and then be terminated.

Only load testing will help you determine your optimal concurrency setting. Note that you can also configure the amount of vCPUs and memory for your Cloud Run instances.


Integrated monitoring, logging, tracing, and error reporting



With integrated monitoring, logging, tracing, and error reporting, you can manage and monitor your Cloud Run applications directly from the Cloud Run dashboard with all the information that is already filtered and organized for you.



Today's agenda

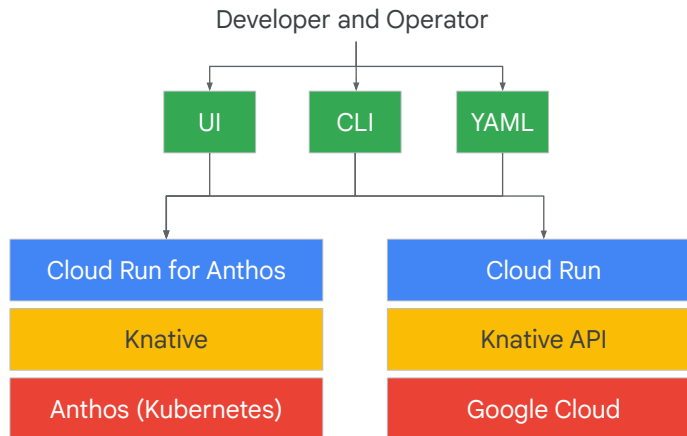


- 01 The need for serverless
- 02 The Cloud Run and Knative model
- 03 Cloud Run on Google Cloud
- 04 [Cloud Run for Anthos](#)
- 05 Serverless eventing

Google Cloud

Now that we saw how Cloud Run works on Google Cloud, let's explore how Cloud Run for Anthos works on your own infrastructure.



Cloud Run for Anthos offers a serverless experience



Google Cloud

Cloud Run for Anthos offers the same serverless experience that you get with Cloud Run. The way you deploy and manage serverless, container-based workloads on Google infrastructure can also be done on your own infrastructure, whether in Google cloud, hybrid, or other cloud environments. The difference is that you are now in a shared responsibility model, because—depending on the Anthos platform you choose—you or a partner owns the underlying hardware infrastructure, which you must ensure is available and running.

Differences between Cloud Run and Cloud Run for Anthos

| |  Cloud Run |  Cloud Run for Anthos |
|-------------------------------|---|--|
| Pricing | Pay per use | Included as part of Anthos |
| Machine types | Limited CPU and memory | Standard or custom machine types including GPUs |
| Autoscaling | Up to 1000 container instances by default; quotas can be increased | Limited by the capacity of your Anthos clusters |
| Identity and policy | IAM-based; VCP SC | RBAC controls; internet, cluster, or VPC accessibility |
| Networking | Access to VPC via Serverless VPC Access | Access to VPC; Services participate in the Anthos Service Mesh |
| URLs | Automatic service URLs and SSL certificates | Custom domains only with manual SSL certificates |
| Container isolation | Strict container isolation with gVisor sandbox | Default Kubernetes container isolation |
| Execution environments | Fully managed on Google infrastructure | Anthos clusters |

Google Cloud

There are some differences between Cloud Run and Cloud Run for Anthos.

In Cloud Run, because it runs on Google Cloud infrastructure and you pay per use, you are limited to the CPU and memory configurations offered, and the number of instances that you can run is limited. In Cloud Run for Anthos, there are no extra costs because it runs on your own infrastructure, you can leverage your hardware and benefit from more flexibility, such as using machines with higher CPU and memory or even using GPUs if those are available, and the number of instances will depend on the size of your Anthos cluster.

For security and networking, Cloud Run is governed by Identity and Access Management, and other Google Cloud services, such as VPC Service Controls or Serverless VPC Access. Cloud Run for Anthos works with RBAC controls and Istio Network Policies, and access can be configured to allow communication with the internet, in the cluster, or in the VPC. Additionally, because Cloud Run services are part of Anthos Service Mesh, communication with other mesh services can be controlled in a similar way.

Finally, other features that Cloud Run provides, such as custom domains, SSL certificates, or strict container isolation with gVisor sandbox, must be configured manually in your Kubernetes-based Anthos clusters.

Cloud Run for Anthos installation

1. Configure a cluster to have a machine type with at least four vCPUs, such as e2-standard-4.
2. Register the Anthos cluster.
3. Install Anthos Service Mesh.
4. (Additional configuration step for clusters outside Google Cloud.)
5. Enable Cloud Run for Anthos in your cluster.

```
gcloud container hub cloudrun enable --project=$PROJECT_ID
```

Cluster features

| | |
|----------------------|------------------|
| Feature Authorizer | ✓ Enabled |
| Binary Authorization | Anthos Feature ? |
| Cloud Run for Anthos | ✓ Enabled |

Google Cloud

To install Cloud Run for Anthos, follow these steps.

1. Configure a Kubernetes cluster to have a machine type with at least 4 vCPUs, such as e2-standard-4.
2. Register the Anthos cluster and ensure that it is part of a Fleet.
3. Install Anthos Service Mesh on your Anthos cluster.
4. If you are running your cluster outside of Google Cloud, you must perform additional steps that we see in the next slide.
5. Finally, enable Cloud Run for Anthos in your cluster by running the “gcloud container hub cloudrun enable” command or via the Google Cloud Console.

Cloud Run for Anthos outside Google Cloud installation

1. Create the Knative Serving namespace and a secret with a service account that contains monitoring.metricsWriter permissions.

```
kubectl apply \
  -f knative-serving-ns.yaml \
  -f monitoring.yaml
```

2. Deploy the Cloud Run custom resource.

```
gcloud container hub cloudrun \
  apply --config=cr.yaml
```

cr.yaml

```
apiVersion: operator.run.cloud.google.com/v1alpha1
kind: CloudRun
metadata:
  name: cloud-run
spec:
  metricscollector:
    stackdriver:
      projectId: my-gcp-project-id
      gcpzone: us-central1-c
      clustername: my-anthos-cluster-name
      secretname: my-gcp-logging-secret
      secretkey: key.json
```

Google Cloud

To install Cloud Run for Anthos outside of Google Cloud, you must perform some additional steps.

1. Ensure that Cloud Monitoring is enabled in your Anthos cluster.
2. Create a service account that has the required Monitoring Metric Writer role (monitoring.metricsWriter).
3. Create the Knative Serving namespace and install the service account in it.
4. Deploy your Cloud Run custom resource that points to your project ID, the zone where the cluster is located, the name of the cluster, and the secret name and key of the service account with the Cloud Monitoring permissions you created earlier.

Serving web traffic with custom domains

- By default, the services are set to the nip.io base domain, which allows you to immediately test your services at a URL like:

```
http://{SERVICE_NAME}.{NS}.kuberun.{EXTERNAL_IP}.nip.io
```

- Configure one or more custom domains:
 1. Optional: Reserve the IP address of your load balancer.
 2. Map to a custom domain in Cloud Run for Anthos.
 - Your services:

```
gcloud run domain-mappings create --service SERVICE --domain DOMAIN
```

- Your cluster:

```
kubectl patch configmap config-domain --namespace knative-serving --patch \
'{"data": {"nip.io": null, "DOMAIN": ""}}'
```

3. Update your DNS records in your domain registrar.

Google Cloud

You can easily serve web traffic from your Cloud Run for Anthos services.

By default, the services are set to the nip.io base domain, which allows you to immediately test your services at a URL like the one on the slide.

Alternatively, you can configure one or more custom domains. There are three simple steps.

Optionally, reserve the IP address of your load balancer. The implementation will differ depending on the Anthos platform you use.

Second, decide whether you want to map your services or your cluster to a custom domain.

If you want to map a specific service, use the “gcloud run domain-mappings create” command.

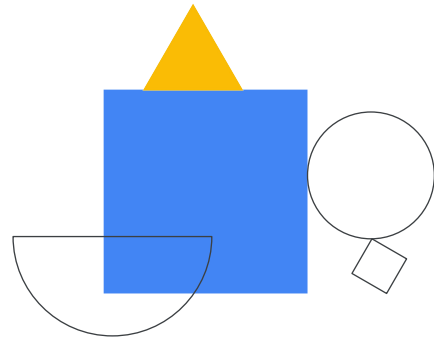
Otherwise, use the “kubectl patch configmap” command to change the default nip.io domain configuration to point to your domain.

Finally, update your DNS records in your domain registrar so that the requests are correctly delivered to your Cloud Run for Anthos service.

Lab intro


🕒 30 min

Deploy to Cloud Run



Google Cloud

In this lab, you build a node.js application, containerize it, and deploy it to Cloud Run on Google Cloud and Cloud Run for Anthos.



Today's agenda

- 01 The need for serverless
- 02 The Cloud Run and Knative model
- 03 Cloud Run on Google Cloud
- 04 Cloud Run for Anthos
- 05 [Serverless eventing](#)



Google Cloud

Breaking the monolith into microservices allows you to use new event-driven architectures that might be more suitable than traditional request-response architectures.

Knative Eventing offers agnostic messaging



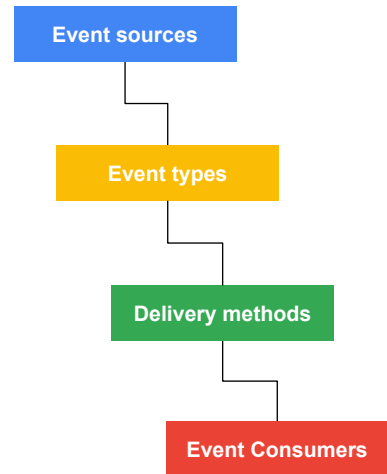
What is it?

For loosely coupled, event-driven services

Many different delivery methods and event sources

Scales from just a few events to live streams

Consistent with the CloudEvents specification



Google Cloud

Knative Eventing is another infrastructure-agnostic, open source, Kubernetes-based solution that is focused on message delivery for loosely coupled, event-driven services. It offers different delivery methods and event sources, scales from just a few events to live streams, and is consistent with CloudEvents, a specification for describing event data in a common way.

Knative Eventing sends event messages between services

Overview

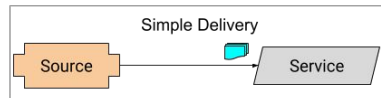
Event consumers

Brokers and triggers

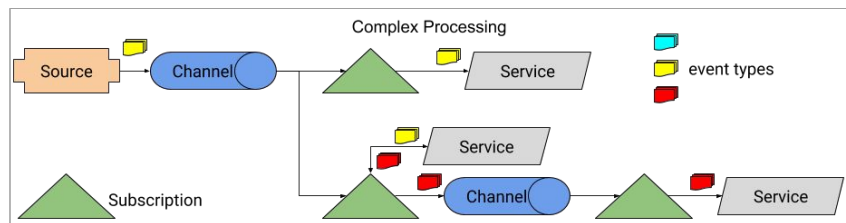
Channels and subscriptions

Hybrid implications

- It can operate in a simple, direct form:



- Or a more complex fan-out model similar to a message queue:



Google Cloud

Knative Eventing sends event messages between services and can operate in a simple direct form or in more complex fan-out scenarios similar to messaging queues.

Knative Eventing defines two types of consumers

Overview

Event consumers

Brokers and
triggers

Channels and
subscriptions

Hybrid implications

- **Addressable** consumers can only acknowledge the receipt of an event sent to them.
- **Callable** consumers, in addition to acknowledging the receipt of an event, can respond with a new event to process.

In both options, the event message is delivered via HTTP.

Google Cloud

Knative Eventing defines two types of event consumers, which are the actors receiving the events.

- Addressable consumers can only acknowledge the receipt of an event sent to them.
- Callable consumers, in addition to acknowledging the receipt of an event, can respond with a new event to process.

In both options, the event message is delivered via HTTP.

Native Eventing processing with brokers and triggers

Overview

Event consumers

**Brokers and
triggers**

Channels and
subscriptions

Hybrid implications

- A **broker** ensures that every event produced finds its way to the correct consumer.
- A **trigger** is used by the broker to decide which consumer (service) gets a particular event.

Google Cloud

Processing is done via brokers and triggers.

The broker ensures that every event produced finds its way to the correct consumer.

The trigger is used by the broker to decide which consumer service gets a particular event.

Knative Eventing works with channels

Overview

Event consumers

Brokers and
triggers

**Channels and
subscriptions**

Hybrid implications

- A channel is an event-forwarding and persistence layer.
- Events that arrive at a channel can be delivered to services or other channels using subscriptions.
- Channels are similar to Cloud Pub/Sub topics.
- Channels and subscriptions control how event messages are delivered; for example, some events go to services, and others go to Apache Kafka.

Google Cloud

Channel is an event-forwarding and persistence layer. When an event arrives to a channel, it can be delivered to a service or to other channels. The delivery takes place using subscriptions. When events arrive at a channel, subscriptions can be used to deliver them to Services or other Channels.

Channels are similar to Pub/Sub Topics.

Channels and Subscriptions control how event messages are delivered.

Examples of subscriptions include other Services or other messaging platforms such as Apache Kafka.

Knative Eventing examples in a hybrid environment

Overview

Event consumers

Brokers and
triggers

Channels and
subscriptions

**Hybrid
implications**

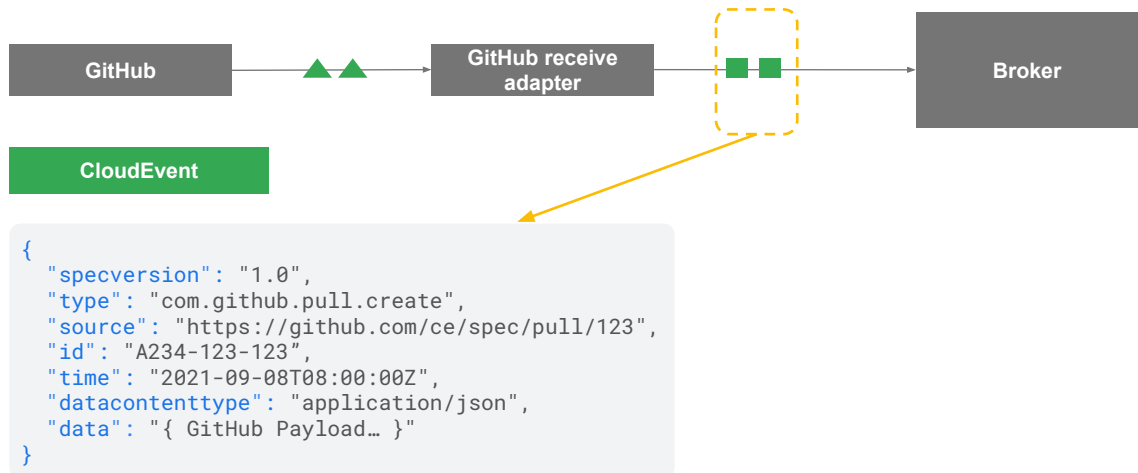
- Request to a service in cluster1 triggers an event pipeline that involves services in other clusters.
- User uploads a new document to a Cloud Storage bucket and an event is generated and processed in a on-premises service.

| Other event sources | Description |
|--------------------------|---|
| Apache Camel | Pushes events from Apache Camel into Knative. |
| Apache Kafka | Brings Kafka messages into Knative. |
| AWS SQS | Brings AWS SQS messages into Knative. |
| Pub/Sub Cloud Storage | Brings messages and notifications into Knative. |
| GitHub GitLab | Brings repository events into Knative. |

Google Cloud

Knative eventing allows you to design workloads that work with cloud services but are abstracted away in the delivery and consumption. That way, you can set up alerts to trigger an execution when a message is sent to Pub/Sub running on Google Cloud or AWS SQS running on AWS. Knative integrates with many third party services. If the integration you are looking for is not available, you can extend the service and create your own.

Knative Eventing has the CloudEvents format

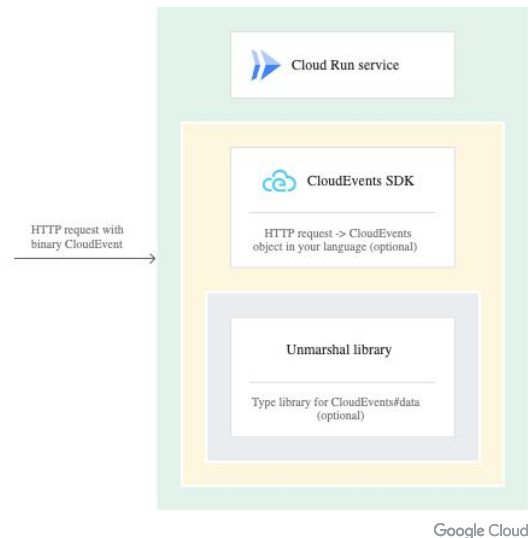


Google Cloud

Knative Eventing uses the CloudEvents format, a specification for describing event data in a common way, which is under the Cloud Native Computing Foundation (CNCF). It defines the same JSON fields for every message so that consumers can read messages from different providers in the same way.

Eventarc is a managed service for Knative Eventing

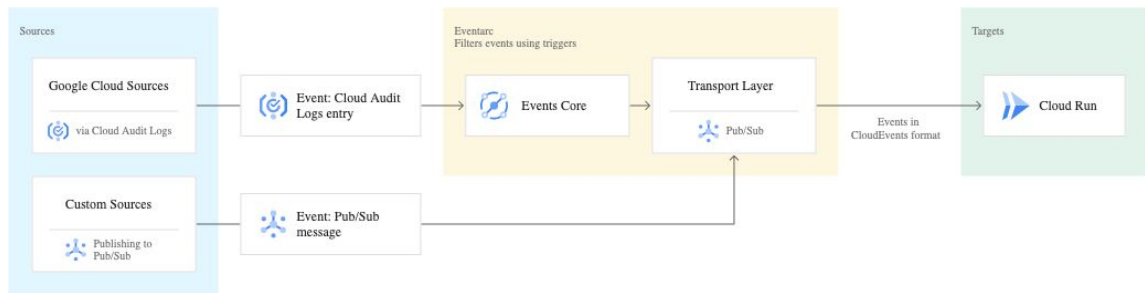
- Supports more than 90 Google Cloud sources.
- Delivers events, regardless of source, to Cloud Run and Cloud Run on Anthos.
- Uses the CloudEvents format in the HTTP requests.



Eventarc is a managed service for Knative Eventing. Eventarc integrates with over 90 Google Cloud services and delivers messages over HTTP to Cloud Run and Cloud Run on Anthos in the same consistent CloudEvents format.

Cloud Run events with Eventarc

Eventarc allows you to build [event-driven architectures](#) without having to implement, customize, or maintain the underlying infrastructure.



Google Cloud

Eventarc allows you to build event-driven architectures without having to implement, customize, or maintain the underlying infrastructure.

Set up an Eventarc trigger for Cloud Run

1. Enable and grant IAM permissions to the service account used by Cloud Run:

```
gcloud projects add-iam-policy-binding $PROJECT_ID \
  --member=serviceAccount:$PROJECT-compute@developer.gserviceaccount.com \
  --role='roles/eventarc.eventReceiver'
```

2. Create a trigger:

```
gcloud eventarc triggers create my-gke-trigger \
  --destination-run-service=helloworld-events \
  --destination-run-region=us-central1 \
  --event-filters="type=google.cloud.audit.log.v1.written" \
  --event-filters="serviceName=storage.googleapis.com" \
  --event-filters="methodName=storage.objects.create" \
  --service-account=$TRIGGER_GSA@$PROJECT_ID.iam.gserviceaccount.com
```

Google Cloud

For each trigger that targets a Cloud Run service, Eventarc creates an event-forwarder component that pulls events from Pub/Sub and forwards them to the target. To receive triggers from Eventarc, grant the eventarc.eventreceiver permissions to the Cloud Run service account.

Then, create a trigger that specifies the source of the event. In this case, we are triggering an event to analyze audit logs generated when Cloud Storage objects are created.

Notice that the destination is a Cloud Run service called helloworld-events located in us-central1.

Set up an Eventarc trigger for Cloud Run on Anthos

- Enable and grant IAM permissions to Cloud Run on Anthos:

```
gcloud eventarc gke-destinations init
```

- Create a trigger:

```
gcloud eventarc triggers create my-gke-trigger \
  --location=$TRIGGER_LOCATION \
  --destination-gke-cluster="events-cluster" \
  --destination-gke-location="us-central1" \
  --destination-gke-namespace="default" \
  --destination-gke-service="hello" \
  --destination-gke-path="/" \
  --event-filters="type=google.cloud.audit.log.v1.written" \
  --event-filters="serviceName=storage.googleapis.com" \
  --event-filters="methodName=storage.objects.create" \
  --service-account=$TRIGGER_GSA@$PROJECT_ID.iam.gserviceaccount.com
```

Google Cloud

Similarly, let's now set up a trigger for Cloud Run for Anthos as a target. To create the Eventarc component and manage resources in a GKE cluster, run the “gcloud eventarc gke-destinations init” command to enable and grant permissions to an Eventarc service account.

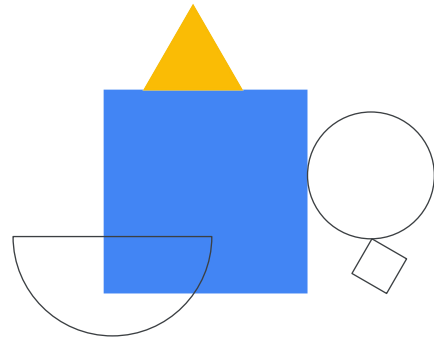
Next, create a trigger that specifies the source of the event. In this case, we are again triggering an event to analyze audit logs generated when Cloud Storage objects are created.

Notice that this time we are setting the destination to be a gke cluster, and the Cloud Run service, called “hello,” lives in the default namespace.

Lab intro

🕒 40 min

Receive Pub/Sub events with
Eventarc and Cloud Run



Google Cloud

In this lab, you deploy a Cloud Run service and use Eventarc to receive events from Pub/Sub. In the second part of the lab, you do the same on an Anthos GKE cluster with the Cloud Run for Anthos offering.

Questions and answers



1. Why would you send traffic to multiple revisions of your Cloud Run application?

1

Blue/green deployments

2

Canary deployments

3

Shadow deployments

4

All the above

1. Why would you send traffic to multiple revisions of your Cloud Run application?

1. Why would you send traffic to multiple revisions of your Cloud Run application?

1 Blue/green deployments

2 Canary deployments

3 Shadow deployments

4 All the above

Google Cloud

In canary deployments you send a small percentage of your traffic to a version of your application to test whether it's working as expected. If so, you then slowly shift more traffic to that new version.

In blue/green and shadow deployments, traffic is served from one version at a time.

2. What message format does Eventarc use?

1 CloudEvents

2 Proprietary format

3 Knative Eventing

4 All options are the same

2. What message format does Eventarc use?

2. What message format does Eventarc use?

1

CloudEvents

2

Proprietary format

3

Knative Eventing

4

All options are the same

Eventarc uses the CloudEvents standard to structure messages.

3. Does Cloud Run for Anthos run any additional resources in your Kubernetes cluster?

1

No, only your Cloud Run app

2

Yes, the operator to run Cloud Run in the Knative Serving namespace

3

Cloud Run for Anthos works with Anthos clusters but it does not run on the cluster

4

All options are the same

2. What message format does Eventarc use?

3. Does Cloud Run for Anthos run any additional resources in your Kubernetes cluster?

1

No, only your Cloud Run app

2

Yes, the operator to run Cloud Run in the Knative Serving namespace

3

Cloud Run for Anthos works with Anthos clusters but it does not run on the cluster

4

All options are the same

Google Cloud

Cloud Run for Anthos creates a Knative Serving namespace, configures the monitoring agent to gather metrics from your service and installs other resources such as the Activator, a Kubernetes resource that forwards traffic to the correct version of your service.