

will delete data that is stored on it. A persistent volume must be attached, or other data transfer methods used while the container is running, to prevent any stored data from being deleted.

Afterwards, Cloud Run automatically adds or removes servers to meet user demand, only charging for what is used to the nearest tenth of a second. Cloud Run provides the user the http endpoint, container usage information, and control over the container's billing.

The Cloud Run usage can be monitored easily in the GUI Metrics, along with logs from the container. Container deployment also has a built-in versioning feature, so a rollback to a previous container version is easy.

For those familiar with the AWS environment, Google Cloud run is not unlike [AWS Lambda](#), though there are [plenty of differences](#).

Cloud Run pros and cons

A developer gets a lot of freedom to create when [using a container](#). They can specify the app requirements in a docker file, using any library and any coding language. So, by and large, developers like containers.

Cloud Run makes container deployment even easier. It's particularly good for:

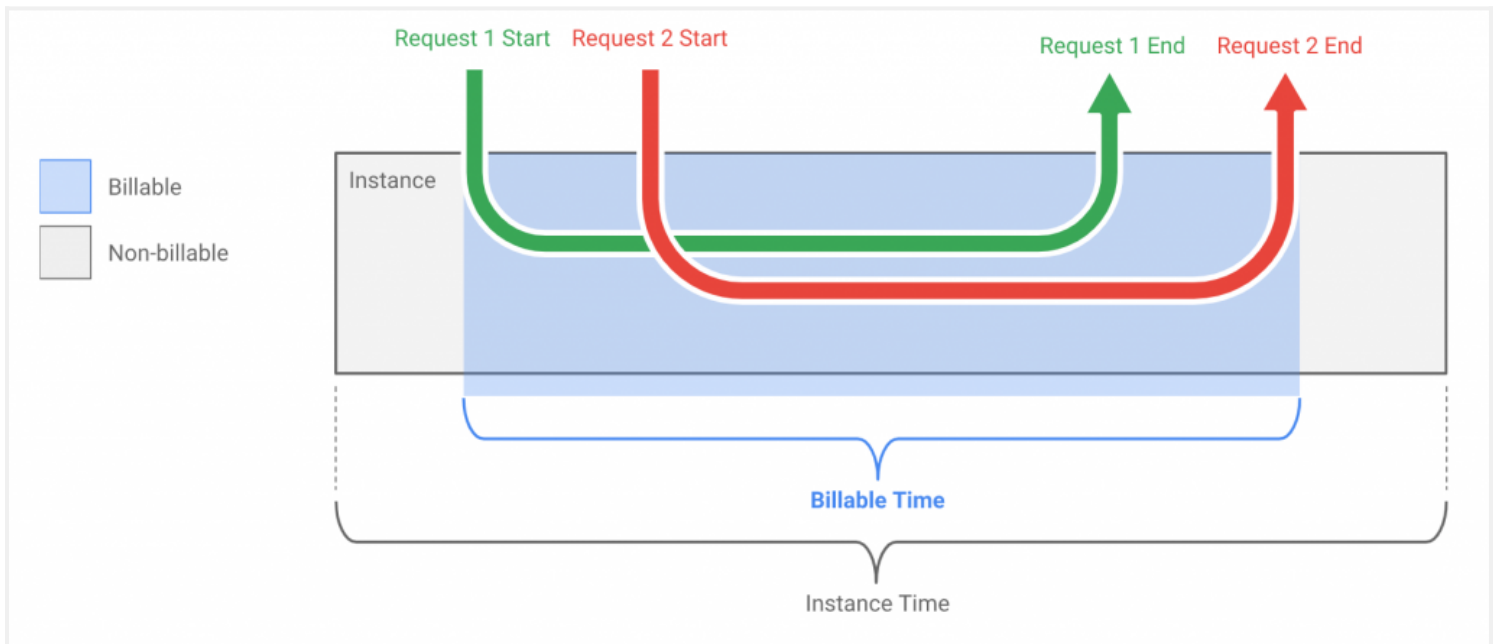
- Developing software in cloud applications
- Delivering web apps, APIs, background jobs

As for what's missing, Cloud Run does come with the limitation that comes with using containers. Containers handle all aspects of the service within a single container. [Serverless design](#) and pay-as-you-go pricing benefits by splitting tasks into different pieces of logic. That design cannot occur when one container handles everything. Google services like Cloud Functions can be used to split a container's functions into their own services (FaaS).

Cloud Run pricing

With Cloud Run, you only pay for what you use. Billed time is broken down to the nearest 0.1 of a second.

Google provides a nice chart that shows how billing works when two requests are made and their timing overlaps. The billing period will begin when the first request is made and when the last request is returned.

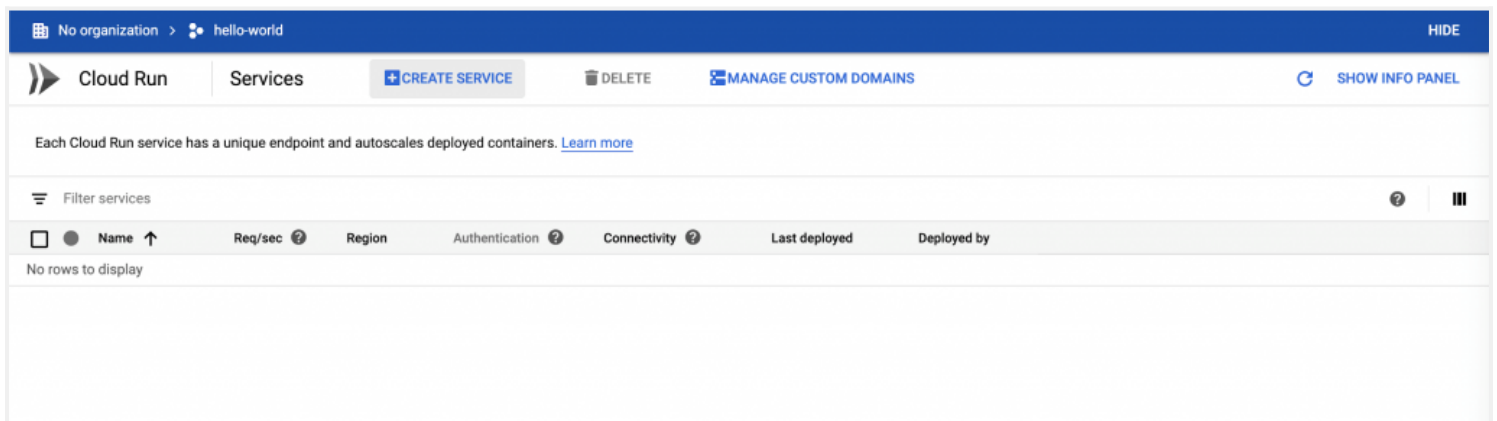


The free-tier allows for 180,000 vCPU-Seconds free, the first 360,000 GB-seconds of memory free, 2 million requests free, and 1 GB free egress within North America. Remember, Google gives [\\$300 in GCP credit](#) when you sign up.

How to start Google Cloud Run

Getting started with Google Cloud Run is straightforward. They include a demo container on their site, so you can go through the steps and try it yourself. Remember to delete the Cloud Run Service after testing.

1. Go to your Cloud Run.
2. Click "Create Service"



3. Choose your region and select a service name.



1 Service settings

Deployment platform and service name are the identifier of a service; they can't be changed once deployed.

Deployment platform ?

Cloud Run (fully managed)

Region *
us-central1 (Iowa) ▼

[How to pick a region?](#)

Cloud Run for Anthos

Service name *

hello-world

Authentication *

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

NEXT

2 Configure the service's first revision

CANCEL

4. Add the container URL and hit "Create".



Cloud Run



Create service

✓ **Service settings**

2 **Configure the service's first revision**

A service can have multiple revisions. The configurations of each revision are immutable.

Container image URL *

gcr.io/cloudrun/hello

SELECT

E.g. gcr.io/cloudrun/hello

Should listen for HTTP requests on \$PORT and not rely on local state. [How to build a container?](#)

✓ SHOW ADVANCED SETTINGS

CREATE

CANCEL

5. See your container deploy and access its URL.

hello-world Region: us-central1 URL: https://hello-world-z7y54x5pia-uc.a.run.app

METRICS REVISIONS LOGS DETAILS YAML PERMISSIONS

Revisions **MANAGE TRAFFIC**

Filter revisions

Name	Traffic	Deployed	Actions
hello-world-00001-vop	100%	Just now	

DETAILS YAML

Container image URL gcr.io/cloudrun/hello@sha256:010310eb93de75ec51...

Container port 8080

Container command and args (container entrypoint)

Autoscaling Up to 1,000 container instances

CPU allocated 1

Memory allocated 256Mi

Concurrency 80

Request timeout 300 seconds

Service account 637846218415-compute@developer.gserviceaccount.com

Build (no build information available)

Source (no source information available)

Environment variables None

Cloud SQL connections None

Additional resources

For more on using containers in the DevOps lifecycle, browse our [BMC DevOps Blog](#) or check out these BMC Blogs:

- [Container Management Platforms: Which Are Most Popular?](#)
- [Container Security Best Practices](#)
- [How Containers Fit in a DevOps Delivery Pipeline](#)
- [Containers in the Multi-Cloud](#)
- [Containers Aren't Always the Solution](#)