

SERVERLESS OPTIONS (#)

Serverless computing is a paradigm shift in application development that enables developers to focus on writing code without worrying about infrastructure. It offers a variety of benefits over traditional computing, including zero server management, no up-front provisioning, auto-scaling, and paying only for the resources used. These advantages make serverless ideal for use cases like stateless HTTP applications, web, mobile, IoT backends, batch and stream data processing, chatbots, and more.

GCP serverless compute portfolio



SERVERLESS FUNCTIONS & EVENTS

Cloud Functions

An event-driven compute platform to easily connect and extend Google and third-party cloud services and build applications that scale from zero to planet scale.

[Learn more](https://cloud.google.com/functions/) → (<https://cloud.google.com/functions/>)



SERVERLESS HTTP APPLICATIONS

App Engine standard environment

A fully managed serverless application platform for web and API backends. Use popular development languages without worrying about infrastructure management.

SERVERLESS OPTIONS (#)



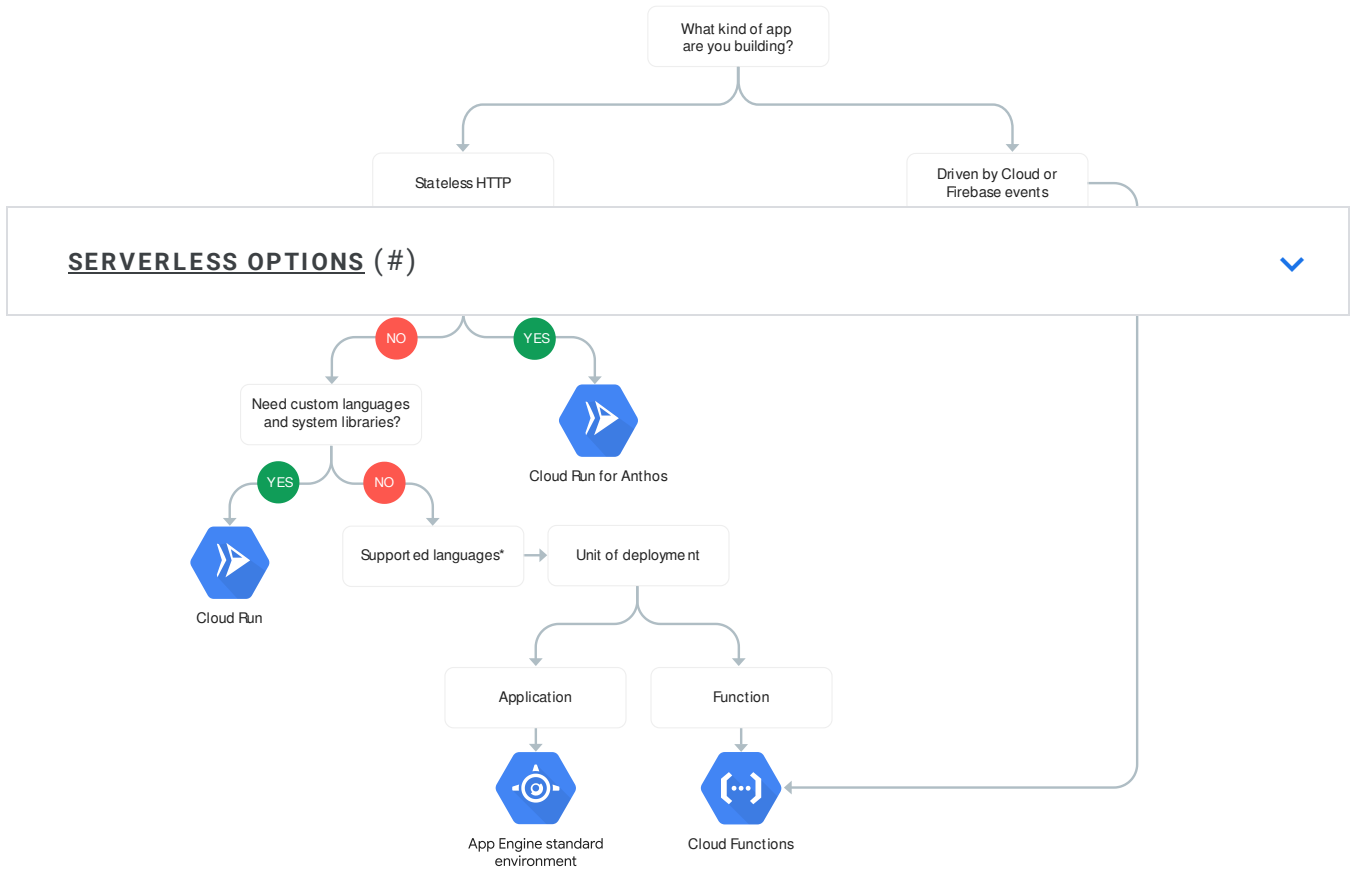
SERVERLESS CONTAINERS

Cloud Run

A serverless compute platform that enables you to run stateless containers invocable via HTTP requests. Cloud Run is available as a fully managed, pay-only-for-what-you-use platform and also as part of Anthos.

[Learn more](https://cloud.google.com/run) → (<https://cloud.google.com/run>)

Which serverless compute platform is right for you?



* App Engine standard environment supports Node.js, Python, Java, Go, PHP

* Cloud Function supports Node.js, Python, Go

Use cases



SERVERLESS OPTIONS (#) ▼

App Engine standard environment (<https://cloud.google.com/appengine/docs/standard/>) is good for minimal-ops web apps running in Node.js, Python, PHP, Java, and Go. Write your applications in a standard, idiomatic way using any language library. Fast deployment times and scaling responsiveness make the App Engine standard environment well suited for spiky workloads.



Asynchronous backend processing

Cloud Functions (<https://cloud.google.com/functions/>) is for responding to data events in the cloud and lightweight processing like resizing an image uploaded to Cloud Storage or validating data when a value is modified in the Firestore database.



Mobile backends

For traditional REST API backends for mobile applications, **App Engine standard environment** (<https://cloud.google.com/appengine/docs/about-the-standard-environment>) is an app platform that monitors, updates, and scales the hosting environment; all you need to do is write your mobile backend service code. **Firebase** (<https://firebase.google.com/>) provides a suite of powerful backend services that integrate directly into your mobile application: real-time NoSQL databases, authentication, hosting, file storage, and more. Firebase integrates with Cloud

Functions so you can easily connect with the rest of your Google Cloud Platform services.



SERVERLESS OPTIONS (#)



If you're building a simple API (a small set of functions to be accessed via HTTP or Cloud Pub/Sub), we recommend using [Cloud Functions](https://cloud.google.com/functions/docs/) (<https://cloud.google.com/functions/docs/>). It is designed for bursty workloads, and its programming paradigm (functions) helps keep small-scale backend code well organized. For a more complex API (such as a REST API with many routes), we recommend using App Engine standard environment, as it may be easier to organize your many functions. If you depend on [Cloud Endpoints](https://cloud.google.com/endpoints/) (<https://cloud.google.com/endpoints/>) for API management, we recommend using App Engine standard environment with Python 2.7 and Java 8 as it supports Cloud Endpoints.



Periodic operations

[Cloud Scheduler](https://cloud.google.com/scheduler/) (<https://cloud.google.com/scheduler/>) can send scheduled HTTP requests to trigger operations on a defined schedule. It can also target App Engine specifically or HTTP endpoints like Cloud Functions and [Cloud Run](https://cloud.google.com/run/) (<https://cloud.google.com/run/>).



Rapid prototyping and API stitching

For small-scale or “hackathon” projects that involve rapid prototyping and/or stitching together multiple APIs and services, we recommend using Cloud Functions. Its programming paradigm allows you to quickly develop both small-scale apps and/or “glue code” that stitches together existing APIs and services.

SERVERLESS OPTIONS (#)



Running provider-agnostic containers


Docker containers are an industry standard and can run in any cloud or on-premises. [Cloud Run \(https://cloud.google.com/run\)](https://cloud.google.com/run) can run containers in a serverless request-response fashion. We recommend using Cloud Run, unless you need custom hardware like GPUs or require a Kubernetes cluster, in which case you can run Cloud Run on GKE in your Google Kubernetes Engine cluster.



Combine serverless and stateful workloads

Cloud Run for Anthos allows you to easily run your serverless and stateful workloads together. For example, you can deploy MongoDB from the Marketplace into your Anthos GKE cluster to use as a document store for your serverless workloads. Anthos gives you the flexibility to run anything in your Kubernetes cluster, and you can use Cloud Run for Anthos to deploy serverless workloads alongside.

Product comparison

<u>SERVERLESS OPTIONS</u> (#) 	
Deployment artifact	App
Scale to zero	✓
Free tier	✓
Websockets	
Languages	Java, Node.js, Python, Go, PHP
Access controls	Oauth 2.0, CICP, Firebase Authentication, Google Sign-In, (https://cloud.google.com/appengine/docs/standard/python3/auth/)
HTTP/2 and gRPC	
Custom domain	✓
Request timeout	1 minute ³
GPUs and TPUs	
VPC connectivity (https://cloud.google.com/vpc/)	

1. Beta software [has no SLA \(https://cloud.google.com/terms/launch-stages\)](https://cloud.google.com/terms/launch-stages).

2. Cloud Run on GKE scales the number of **pods** to zero. The number of **nodes per cluster** cannot scale to zero, and these nodes are billed in the absence of requests.

3. [Automatic scaling](#)

(https://cloud.google.com/appengine/docs/standard/python/how-instances-are-managed#instance_scaling)

: 60-second deadline for HTTP requests.

SERVERLESS OPTIONS (#)



Advanced tips and best practices

Here are some additional factors you may want to consider.

EXPAND ALL

Partially-autoscaling architectures



Free usage + billing limits



Concurrent requests



Workload portability

