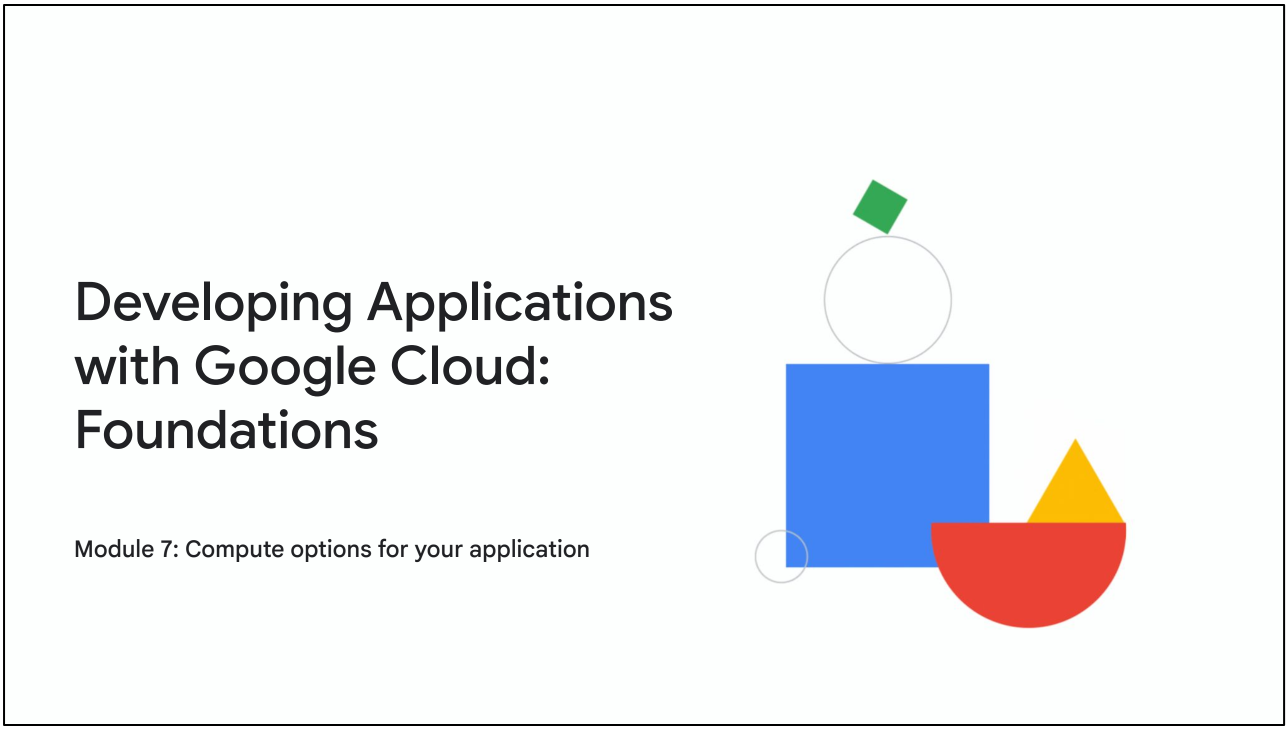
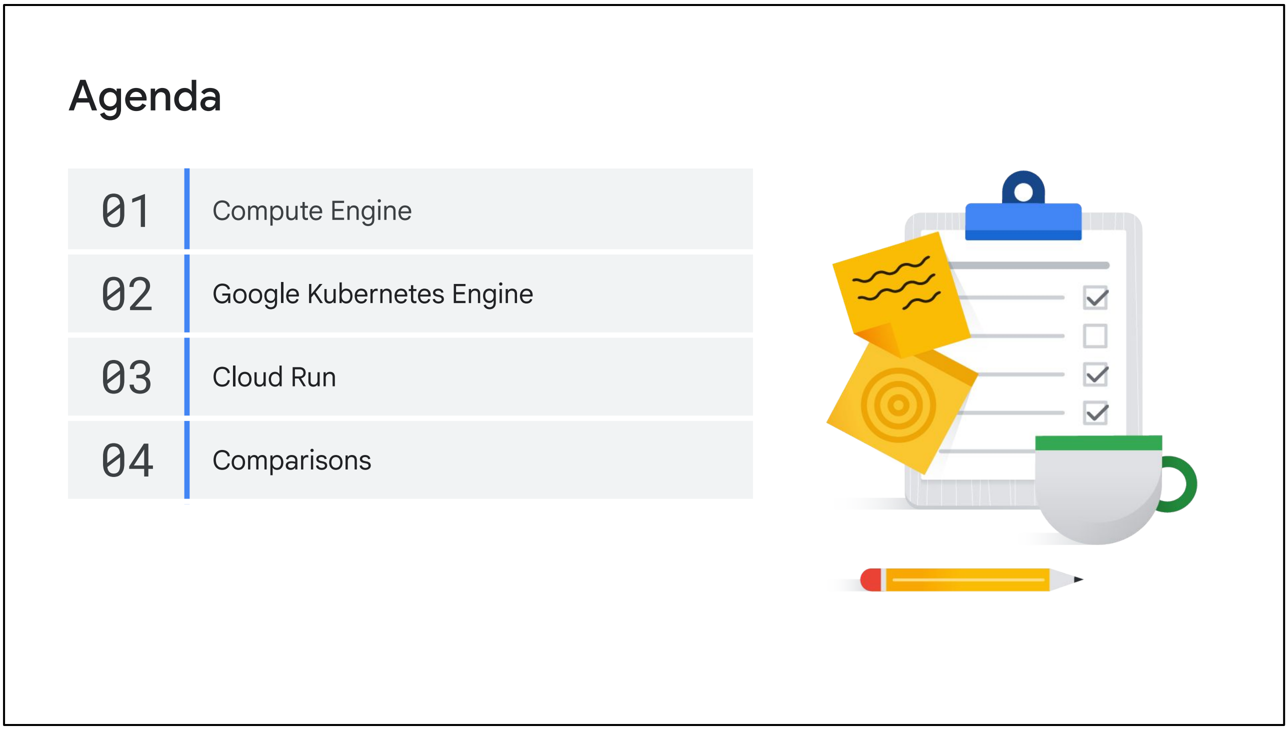


Developing Applications with Google Cloud: Foundations

Module 7: Compute options for your application

Welcome to Developing Applications with Google Cloud: Foundations, module 7: Compute options for your application.

Google Cloud has a range of compute options that you can use to run your applications. You can choose a platform that matches the needs of your application, including the level of control that you need for the infrastructure. Having more control over the infrastructure usually leads to a greater operational burden. If you use Cloud Client Libraries in your application to work with Google Cloud services, you can usually move to another platform without reworking your application.



Agenda

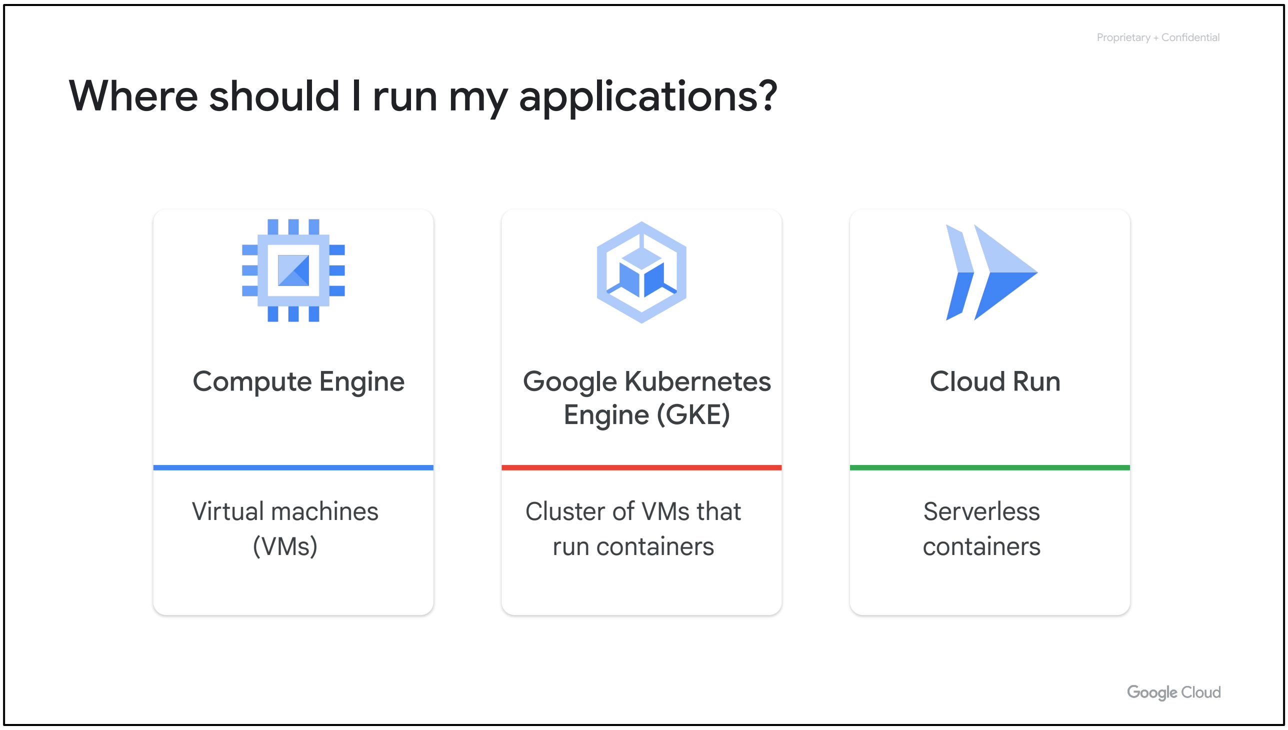
01 Compute Engine

02 Google Kubernetes Engine

03 Cloud Run

04 Comparisons

In this module, you learn about which use cases are most appropriate for each compute option and how to decide between them.



Where should I run my applications?

Compute Engine	Google Kubernetes Engine (GKE)	Cloud Run
Virtual machines (VMs)	Cluster of VMs that run containers	Serverless containers

Google Cloud

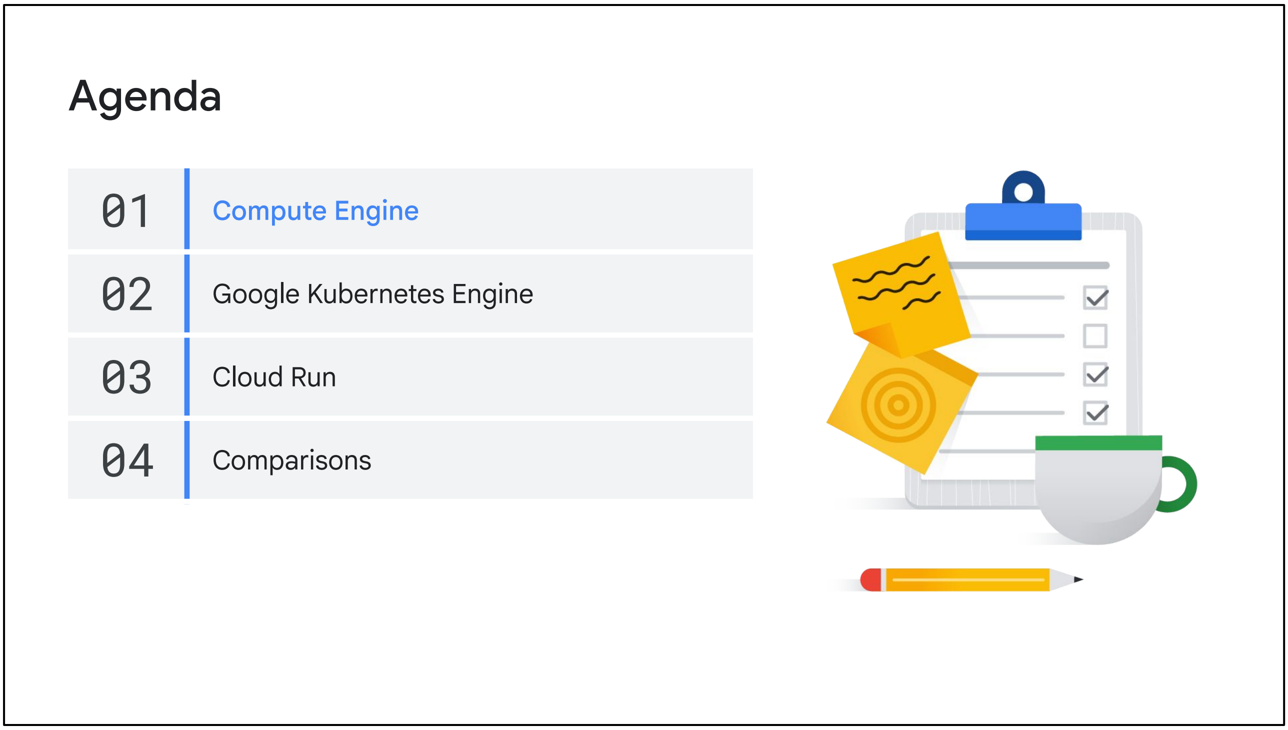
Google Cloud provides a range of platforms on which to run your applications.

- Compute Engine lets you create virtual machines, or VMs, that mimic the servers you may have used in a traditional data center. Virtual machines are highly flexible: they let you run the same wide range of applications you can run on physical hardware, but now on Google's infrastructure.

- Google Kubernetes Engine, or GKE, is Google Cloud's managed service for running containers and managing the virtual machines used to run them. With GKE, a cluster of virtual machines is created for running your containerized applications. When you deploy containerized applications to the cluster, GKE manages the scaling and security for the cluster and applications, reducing the operational costs of running your applications.

- Cloud Run is a fully managed serverless platform that also runs containerized applications. With Cloud Run, all infrastructure management is abstracted away. Cloud Run automatically scales up and down from zero almost instantaneously, depending on traffic. You only pay when your code is running. You can also have Cloud Run create and manage the containers, so you only need to supply the source code.

So, which compute option should you use to run your applications? Well, it depends.



Agenda

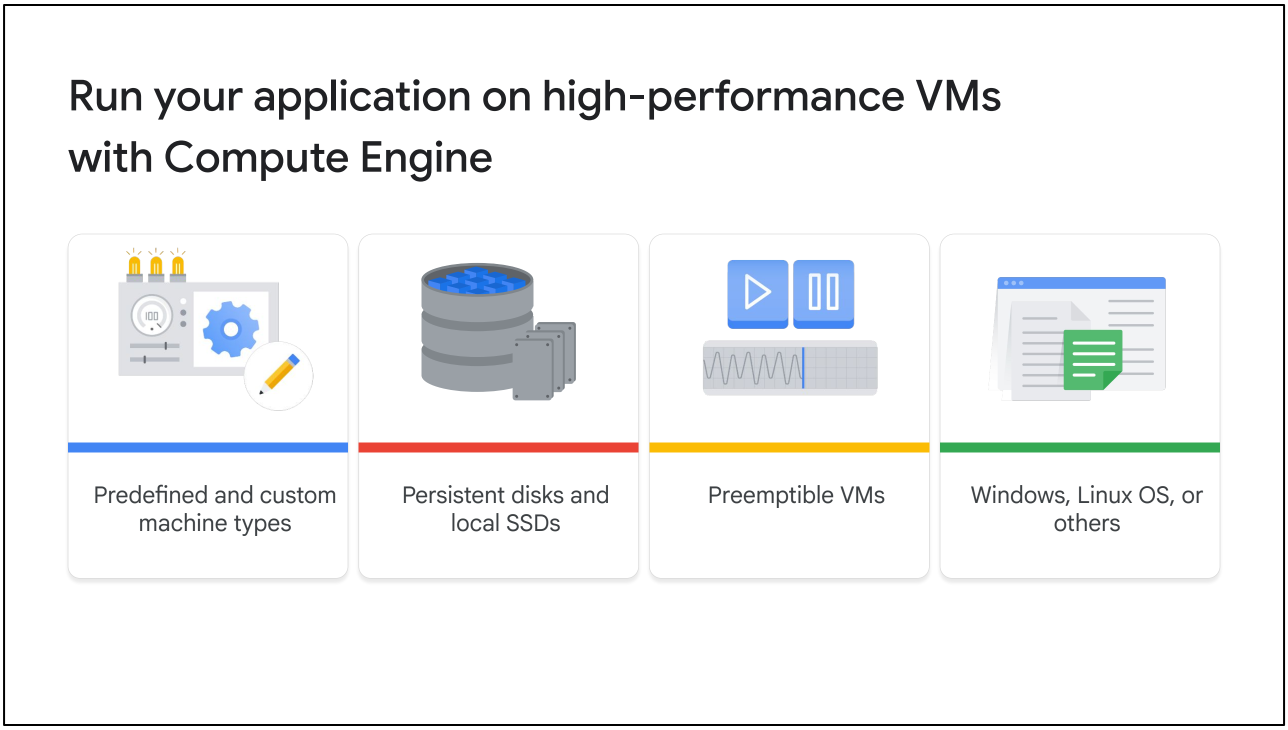
01 Compute Engine

02 Google Kubernetes Engine

03 Cloud Run

04 Comparisons

First, we start with Compute Engine. Compute Engine is the most flexible option for running your applications, but it requires the most operational effort to manage.



Run your application on high-performance VMs with Compute Engine

Predefined and custom machine types

Persistent disks and local SSDs

Preemptible VMs

Windows, Linux OS, or others

With Compute Engine, you create high-performance virtual machines and then install and run your applications on them.

- Compute Engine supports predefined machine types for popular configurations but also lets you create custom machine types to customize CPU and memory for your VMs.

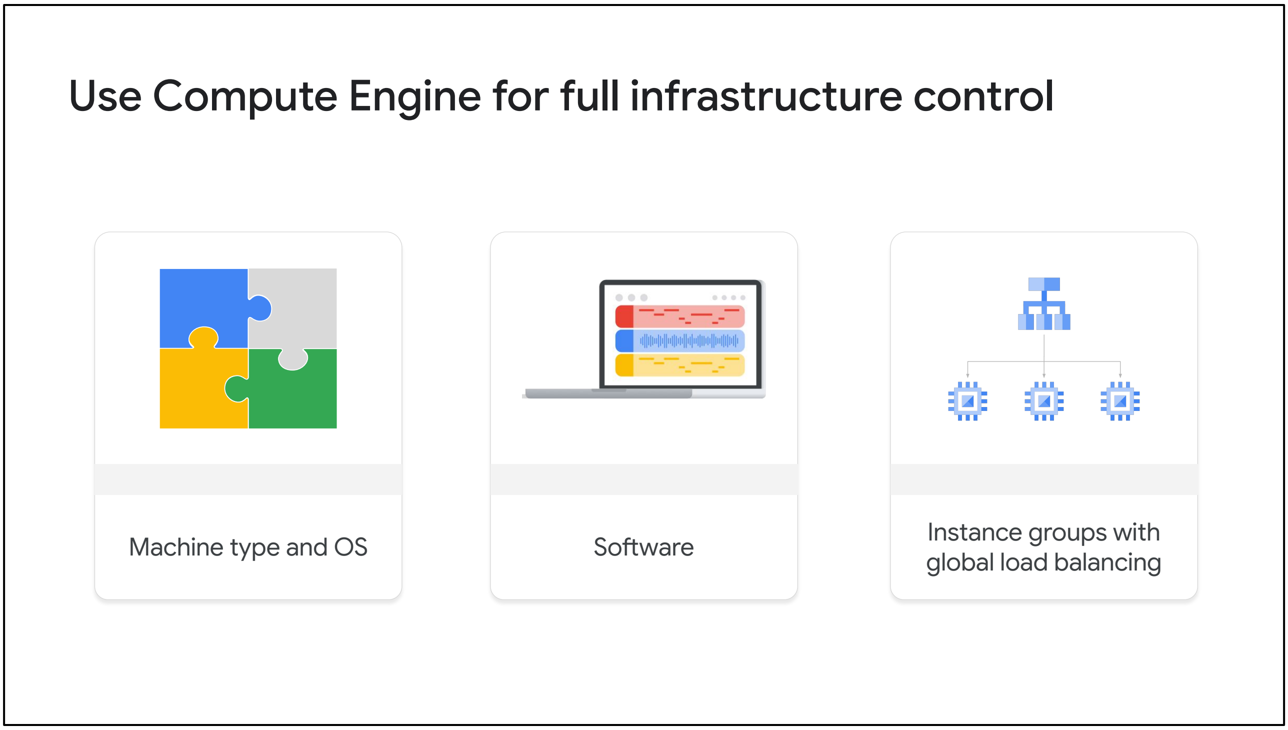
- Compute Engine lets you create and attach persistent disks and local SSDs. These disks can be accessed like physical disks in a desktop or server. Unlike typical physical disks, Compute Engine disks can be increased in size while they are running. The performance and throughput of persistent disks increase when they increase in size.

- Compute Engine provides preemptible VMs that are ideal for large compute and batch jobs. If capacity must be reclaimed, Google Cloud can terminate preemptible VMs. For applications that can handle these interruptions, preemptible VMs are available at a discount of at least 60% compared to standard VMs.

- You can run your choice of operating system on your VMs, including Debian, CentOS, Ubuntu, and various other flavors of Linux or Windows. You can also

- use a shared image from the Google Cloud community or bring your own operating system.

[Compute Engine: https://cloud.google.com/compute]



Use Compute Engine for full infrastructure control

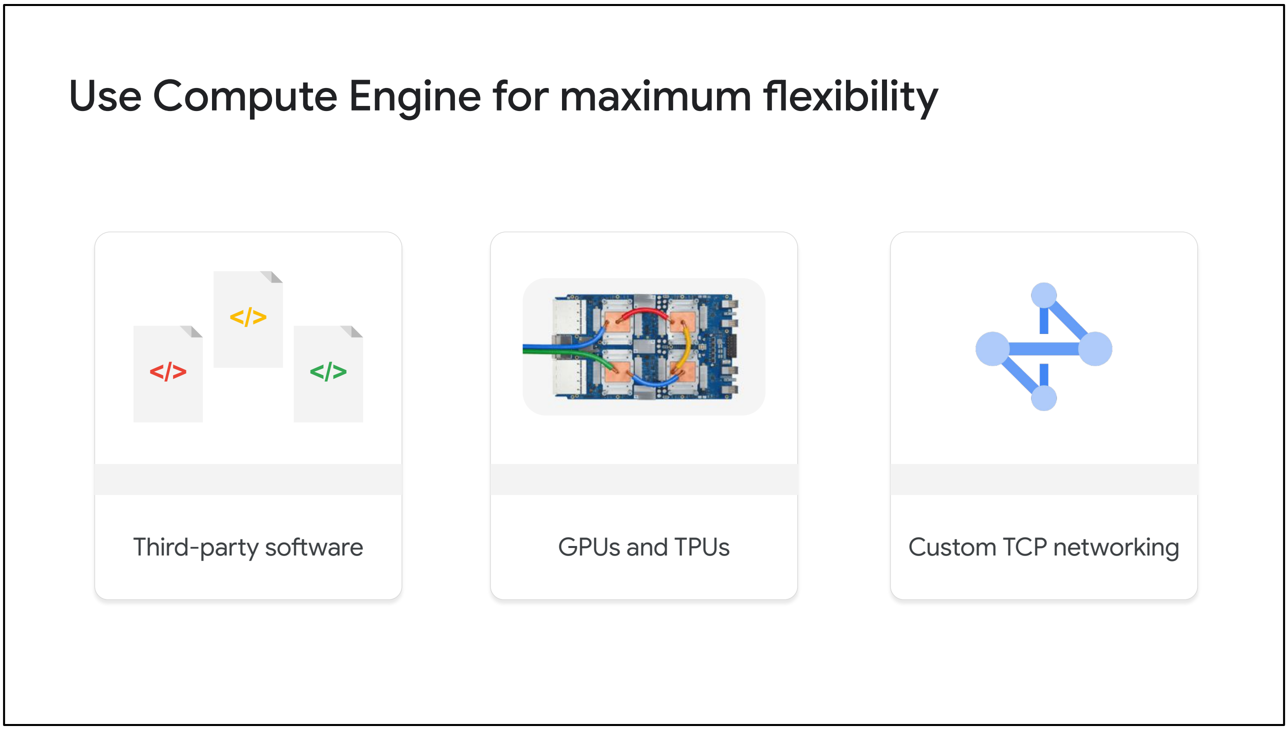
Machine type and OS

Software

Instance groups with global load balancing

Use Compute Engine when you want full control of your infrastructure.

- Compute Engine enables you to create highly customized VMs for specialized applications that have unique compute or operating system requirements.
- You can install and patch software that runs on a VM.
- You can create managed instance groups of VMs based on an instance template and configure global load balancing and autoscaling of the managed instance groups. Compute Engine can perform health checks and replace unhealthy instances in an instance group. Compute Engine can also autoscale the number of instances based on the traffic volume in specific regions.



Use Compute Engine for maximum flexibility

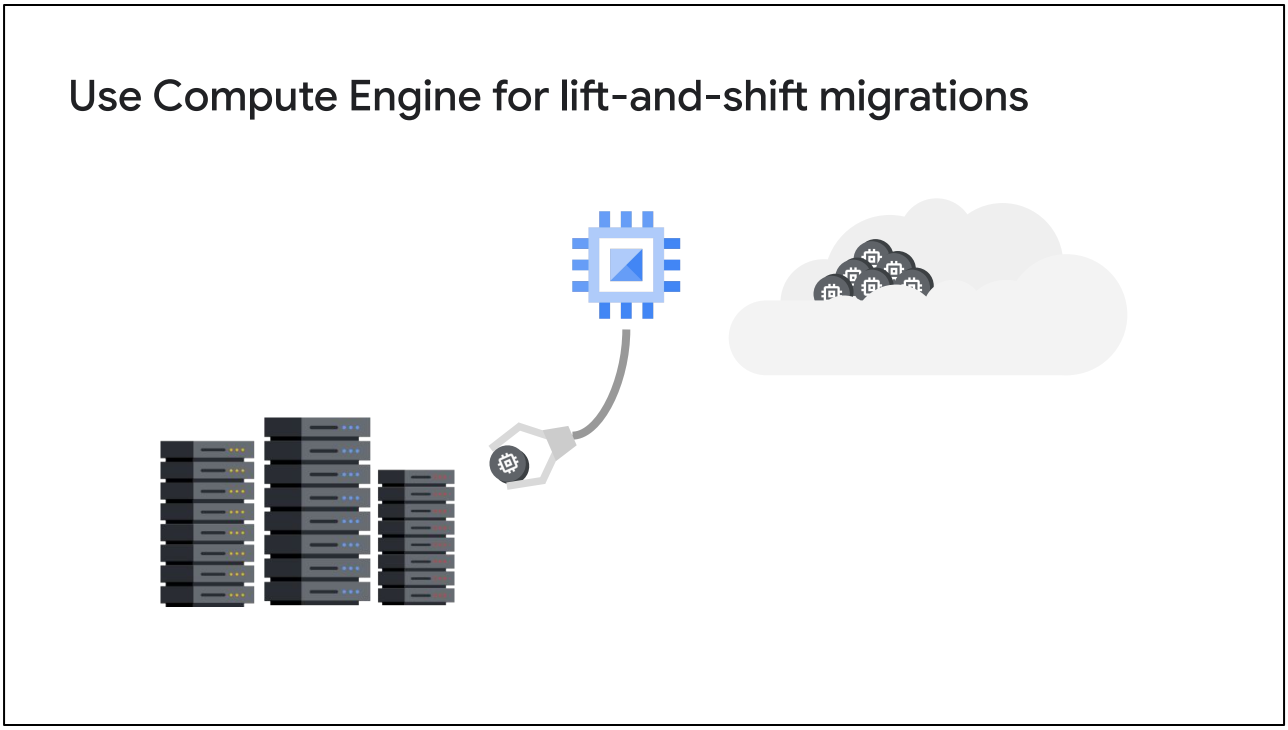
Third-party software

GPUs and TPUs

Custom TCP networking

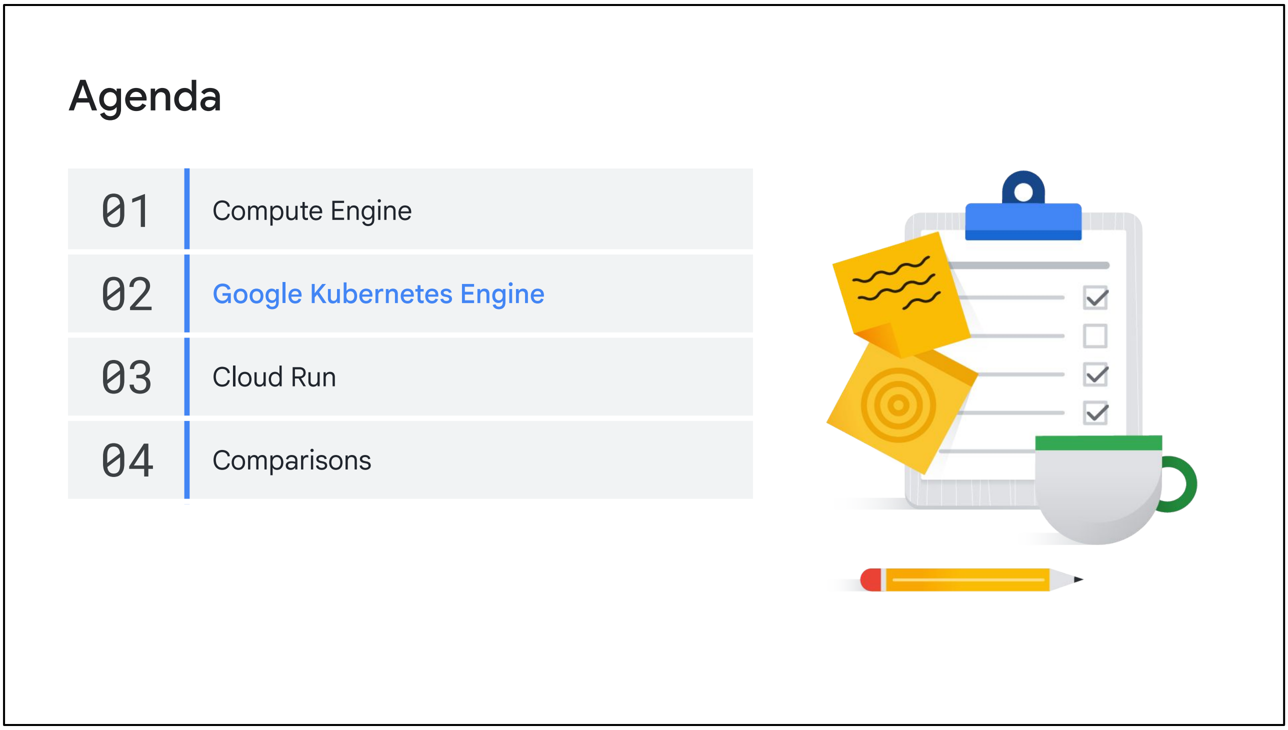
Compute Engine offers you the most flexibility to configure your resources for the specific type of application that you need to run.

- You can install and run any third-party licensed software on Compute Engine.
- You can attach graphics processing units (GPUs) and Tensor Processing Units (TPUs) to Compute Engine VMs to accelerate parallel processing and machine learning workloads.
- You can use Compute Engine for applications that require TCP network protocols other than HTTP or HTTPS.



Use Compute Engine for lift-and-shift migrations

Compute Engine is ideal for lift-and-shift migrations. You can move virtual machines from your on-premises data center, or another cloud provider, to Google Cloud without changing your application.



Agenda

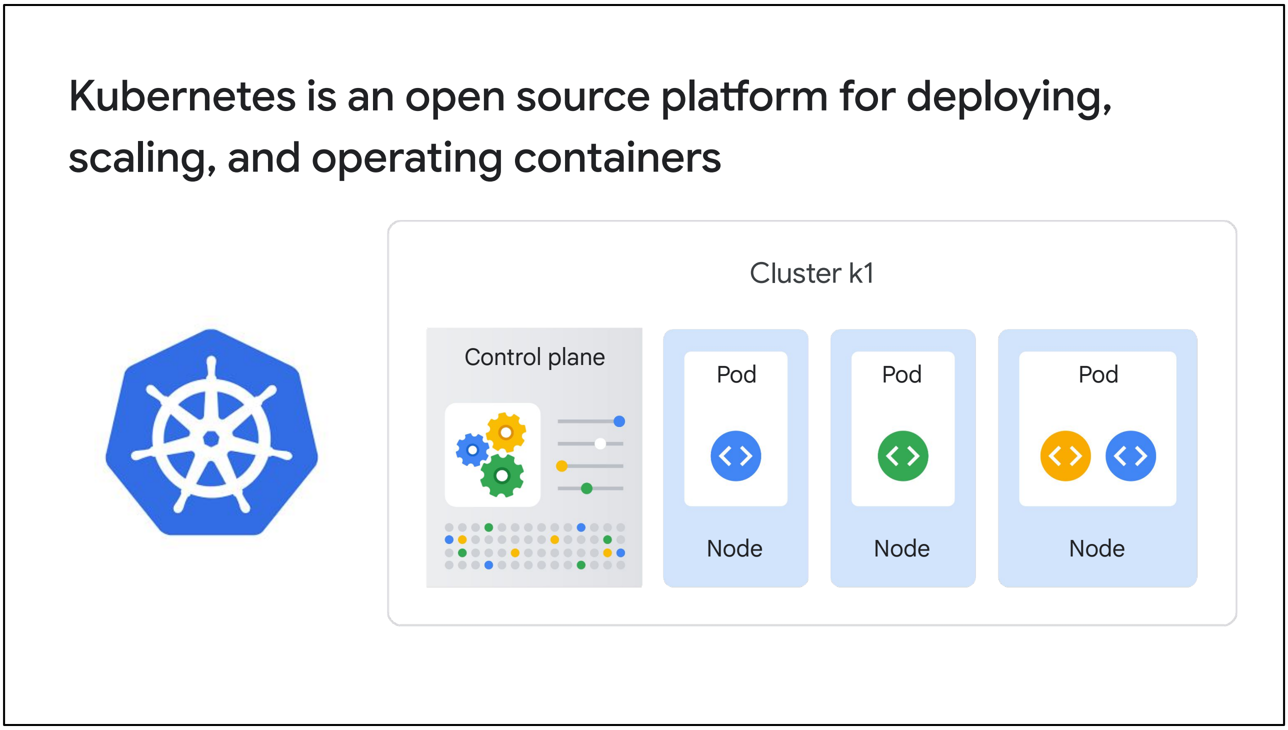
01 Compute Engine

02 Google Kubernetes Engine

03 Cloud Run

04 Comparisons

Next is Google Kubernetes Engine. Kubernetes is a leading open source platform for deploying, scaling, and operating containers. Kubernetes, first developed at Google, is now a Cloud Native Computing Foundation project with a large and active community.



Kubernetes is an open source platform for deploying, scaling, and operating containers

Cluster k1

Control plane

Pod

Node

Pod

Node

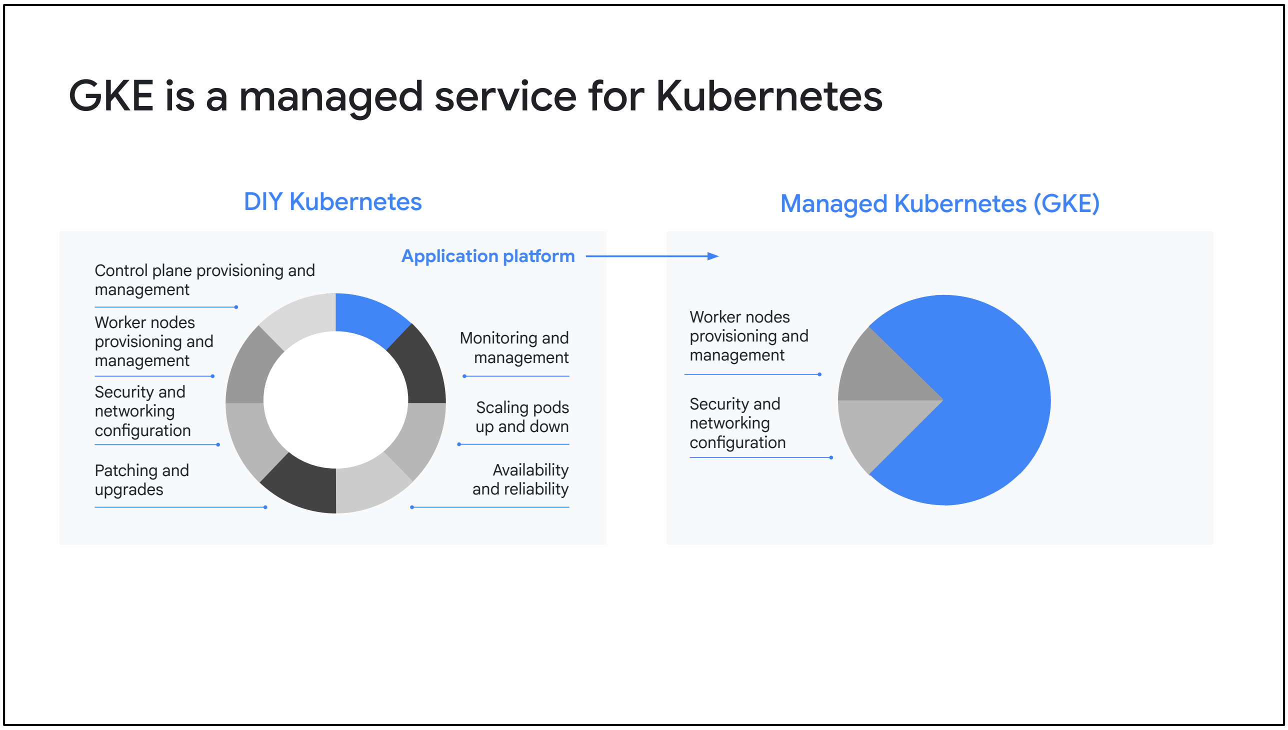
Pod

Node

Kubernetes provides you with a framework to run distributed containerized systems resiliently and at scale. It manages many operational tasks, such as scaling application components, providing network abstractions, orchestrating failovers, rolling out deployments, storage orchestration, and management of secrets and configurations.

A Kubernetes cluster contains control plane and worker nodes. The nodes in a cluster are the machines—virtual or physical—that run your applications. The Kubernetes control plane manages the worker nodes and the Pods in the cluster. A Pod is a group of containers that share networking and storage resources on the node.

[Cloud Native Computing Foundation: https://www.cncf.io/
Kubernetes documentation: https://kubernetes.io/docs/home/]



GKE is a managed service for Kubernetes

DIY Kubernetes

Application platform

Control plane provisioning and management

Worker nodes provisioning and management

Security and networking configuration

Patching and upgrades

Monitoring and management

Scaling pods up and down

Availability and reliability

Managed Kubernetes (GKE)

Worker nodes provisioning and management

Security and networking configuration

Google Kubernetes Engine, or GKE, is a managed Kubernetes service on Google infrastructure. GKE helps you deploy, manage, and scale Kubernetes environments for your containerized applications on Google Cloud.

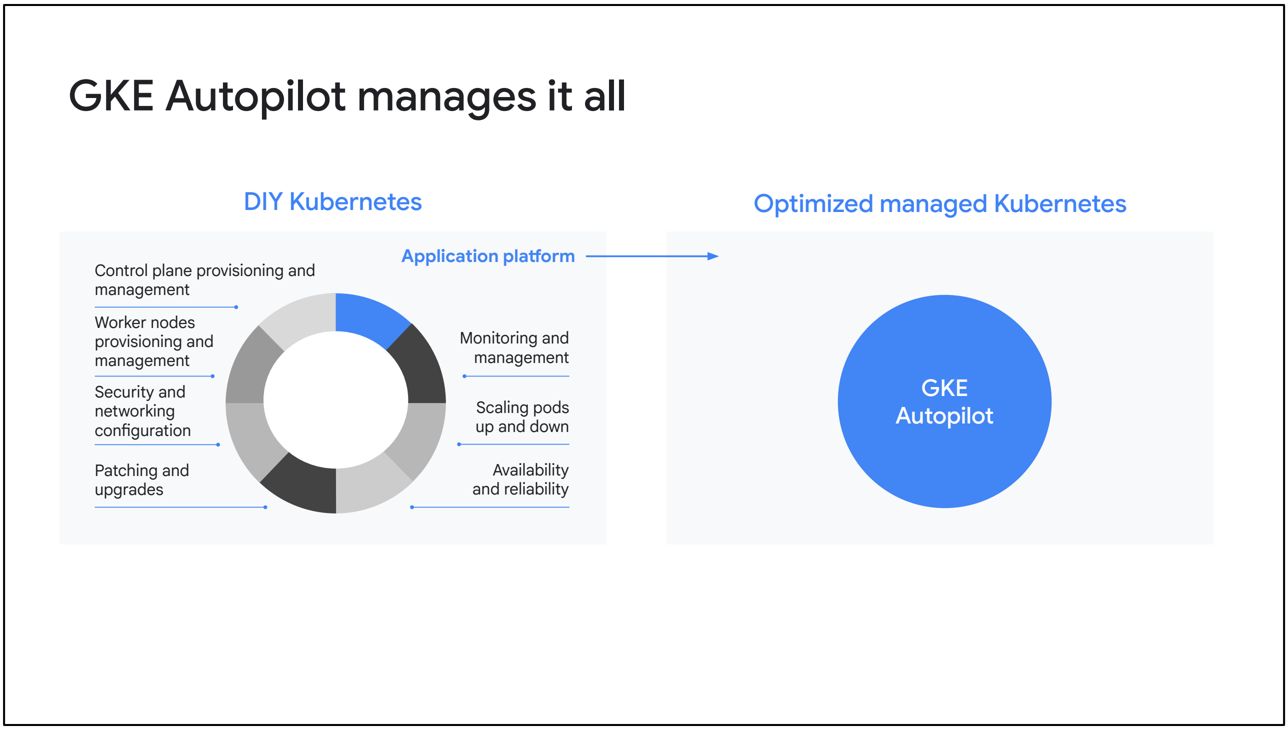
More specifically, GKE is a component of the Google Cloud compute offerings that facilitates bringing your Kubernetes workloads into the cloud.

For an unmanaged cluster, you need to manage most of the operational aspects of the cluster yourself.

GKE handles much of this operational effort for you automatically by eliminating many of the infrastructure tasks required to create and manage a Kubernetes cluster. With GKE, Google manages most of your cluster tasks. Google manages the control plane, scaling of Pods, node patching and upgrades, and the monitoring, availability, and reliability of the cluster.

By default, you manage the underlying nodes and node pools, including provisioning, maintenance, and lifecycle management. You're also responsible for selecting the security and networking configuration for your cluster. This level of management is the standard mode for GKE.

[Google Kubernetes Engine: https://cloud.google.com/kubernetes-engine/
Google Kubernetes Engine Overview:
https://cloud.google.com/kubernetes-engine/docs/concepts/kubernetes-engine-overview]



GKE Autopilot manages it all

DIY Kubernetes

Application platform

Control plane provisioning and management

Worker nodes provisioning and management

Security and networking configuration

Patching and upgrades

Monitoring and management

Scaling pods up and down

Availability and reliability

Optimized managed Kubernetes

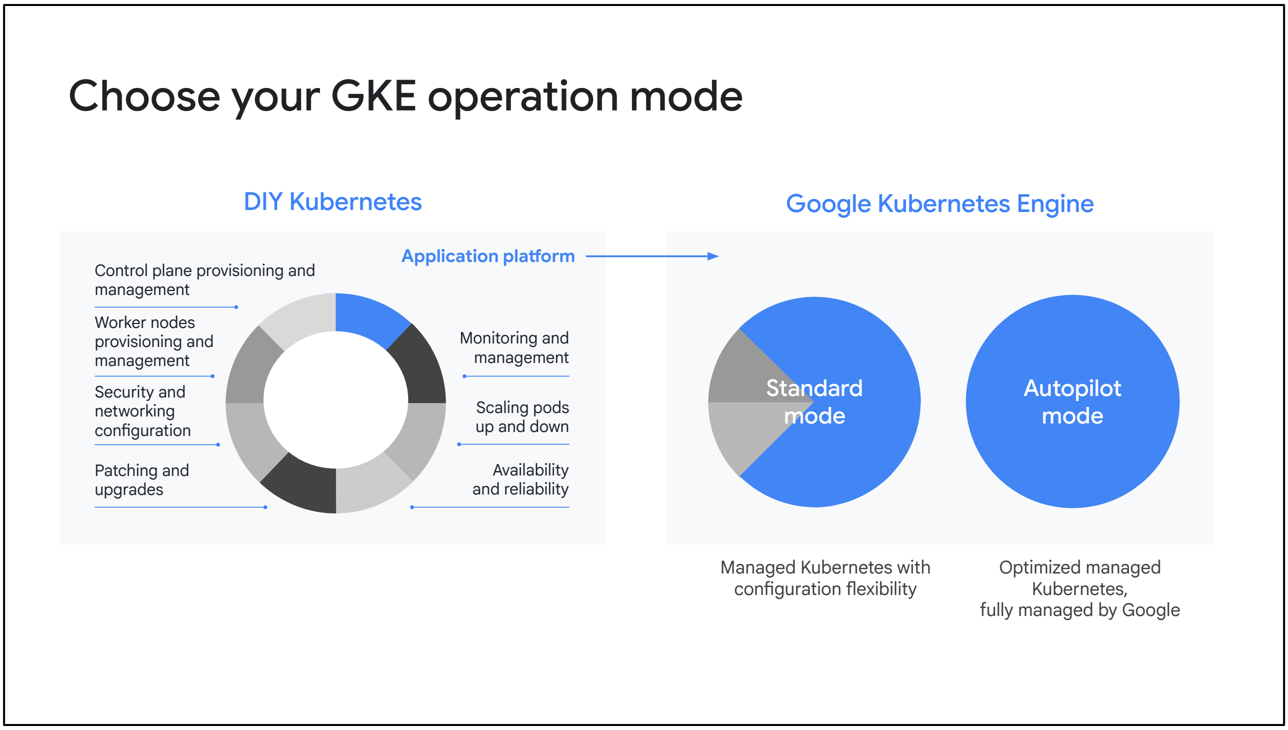
GKE Autopilot

GKE Autopilot is a mode of operation in which the entire cluster's infrastructure is managed for you, including control plane, node pools, and nodes.

By managing the cluster infrastructure, Autopilot helps reduce operational and maintenance costs while improving resource utilization. Autopilot is a fully managed Kubernetes experience that lets you focus on your workloads instead of the management of the cluster's infrastructure.

Autopilot automatically implements GKE hardening guidelines and security and networking best practices and blocks less safe practices.

[Introducing GKE Autopilot (blog):
https://cloud.google.com/blog/products/containers-kubernetes/introducing-gke-autopilot
Autopilot overview:
https://cloud.google.com/kubernetes-engine/docs/concepts/autopilot-overview]



Choose your GKE operation mode

DIY Kubernetes

Control plane provisioning and management

Worker nodes provisioning and management

Security and networking configuration

Patching and upgrades

Application platform

Monitoring and management

Scaling pods up and down

Availability and reliability

Google Kubernetes Engine

Standard mode

Autopilot mode

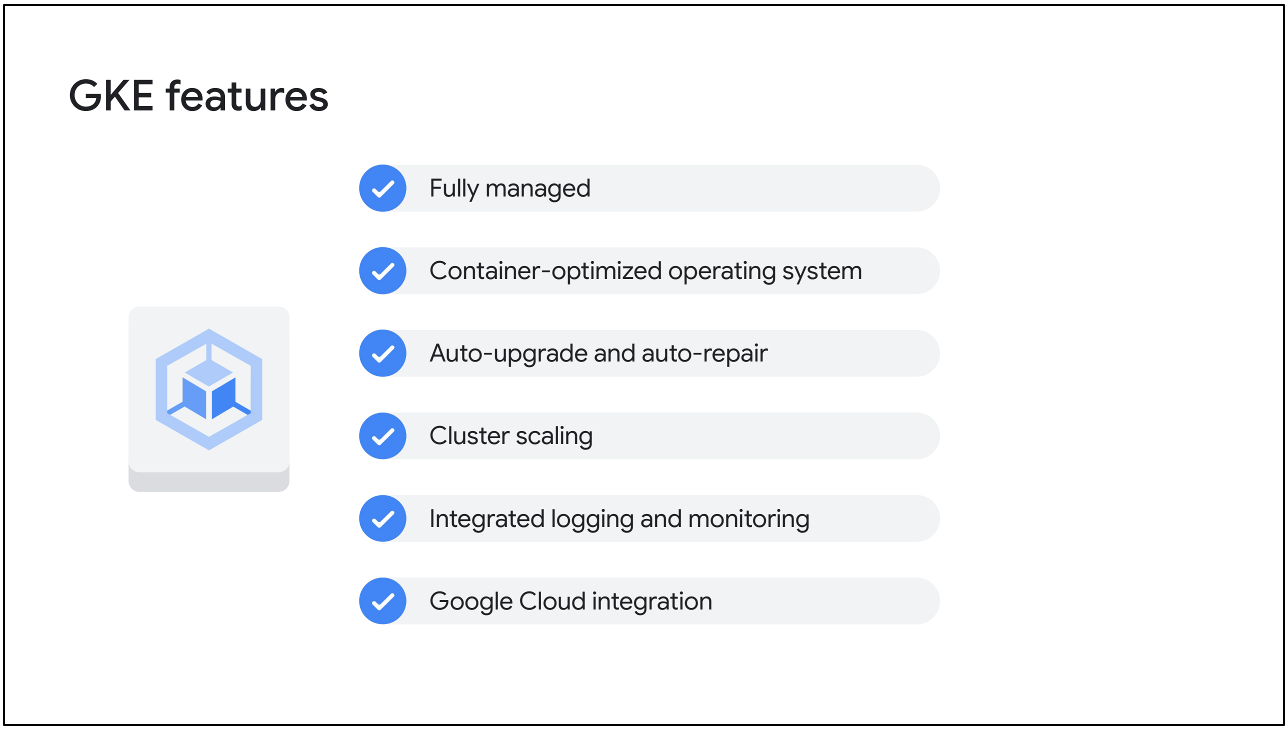
Managed Kubernetes with configuration flexibility

Optimized managed Kubernetes, fully managed by Google

GKE Standard mode provides customers with advanced configuration flexibility over the cluster infrastructure. GKE Autopilot mode lets Google provision and manage the entire cluster and underlying infrastructure.

You can use different modes for different clusters, depending on how much infrastructure control you need.

[Comparing Autopilot and Standard modes: https://cloud.google.com/kubernetes-engine/docs/concepts/autopilot-overview#comparison]



GKE features

- ✓ Fully managed
- ✓ Container-optimized operating system
- ✓ Auto-upgrade and auto-repair
- ✓ Cluster scaling
- ✓ Integrated logging and monitoring
- ✓ Google Cloud integration

- GKE is fully managed, which means that you don’t have to provision the underlying resources.

- GKE uses a container-optimized operating system to run your workloads. Google maintains this operating system, which is optimized to scale quickly with a minimal resource footprint.

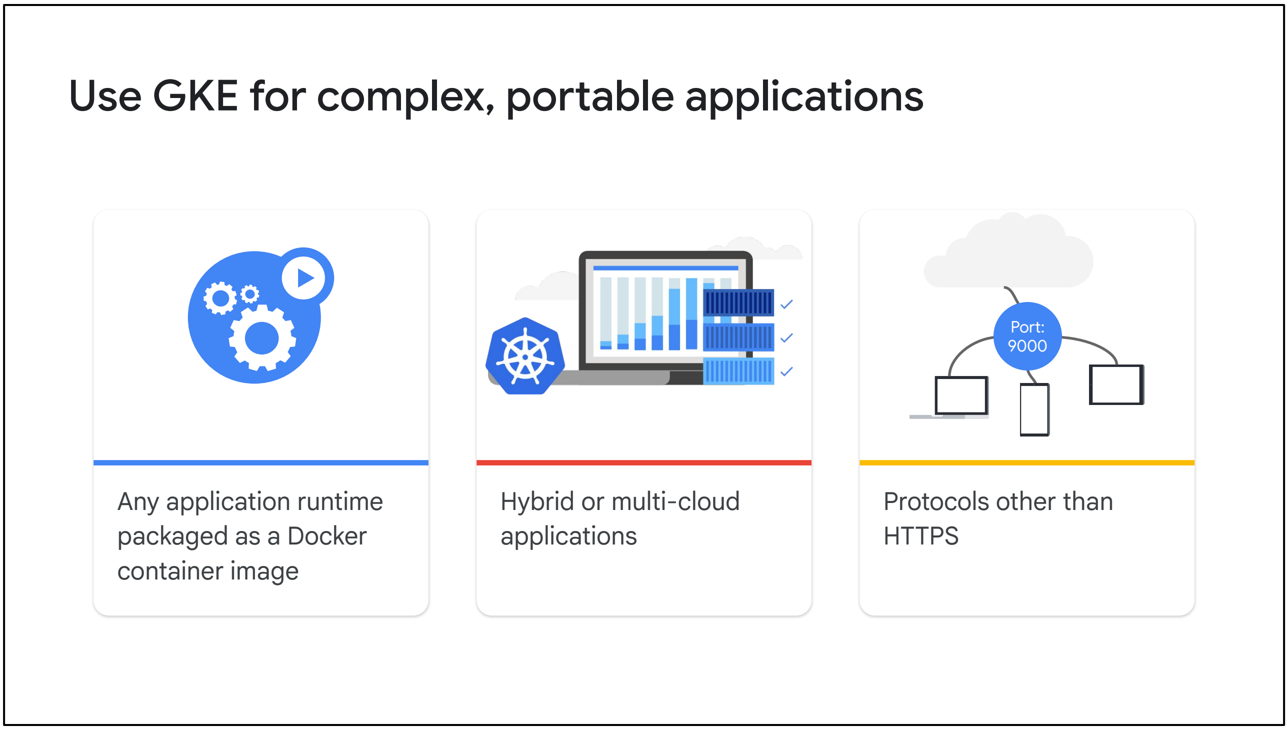
- When you use GKE, you start by directing the service to instantiate a Kubernetes cluster for you. The GKE auto-upgrade feature, when enabled, ensures that your clusters are always automatically upgraded with the latest stable version of Kubernetes.
  The virtual machines that host your containers in a GKE cluster are called nodes. Auto-repair can automatically repair unhealthy nodes for you. It performs periodic health checks on each node of the cluster. If a node is determined to be unhealthy and requires repair, GKE will drain the node, thus allowing workloads to gracefully exit. It will then recreate the node.

- GKE and Kubernetes both support the scaling of workloads within a cluster. GKE also supports scaling of the cluster itself.

- GKE uses Cloud Monitoring and Cloud Logging to help you monitor and

- understand your applications’ performance and behavior.

- GKE seamlessly integrates with many parts of Google Cloud. With Cloud Build, you can use private container images that you have securely stored in Artifact Registry to automate the deployment of your workloads. Identity and Access Management lets you control access by using accounts and role permissions. GKE is integrated with Virtual Private Clouds, or VPCs, which lets you use Google Cloud's networking features. And finally, the Google Cloud Console provides insights into GKE clusters and their resources, thus letting you view, inspect, and delete resources in those clusters.



Use GKE for complex, portable applications

Any application runtime packaged as a Docker container image

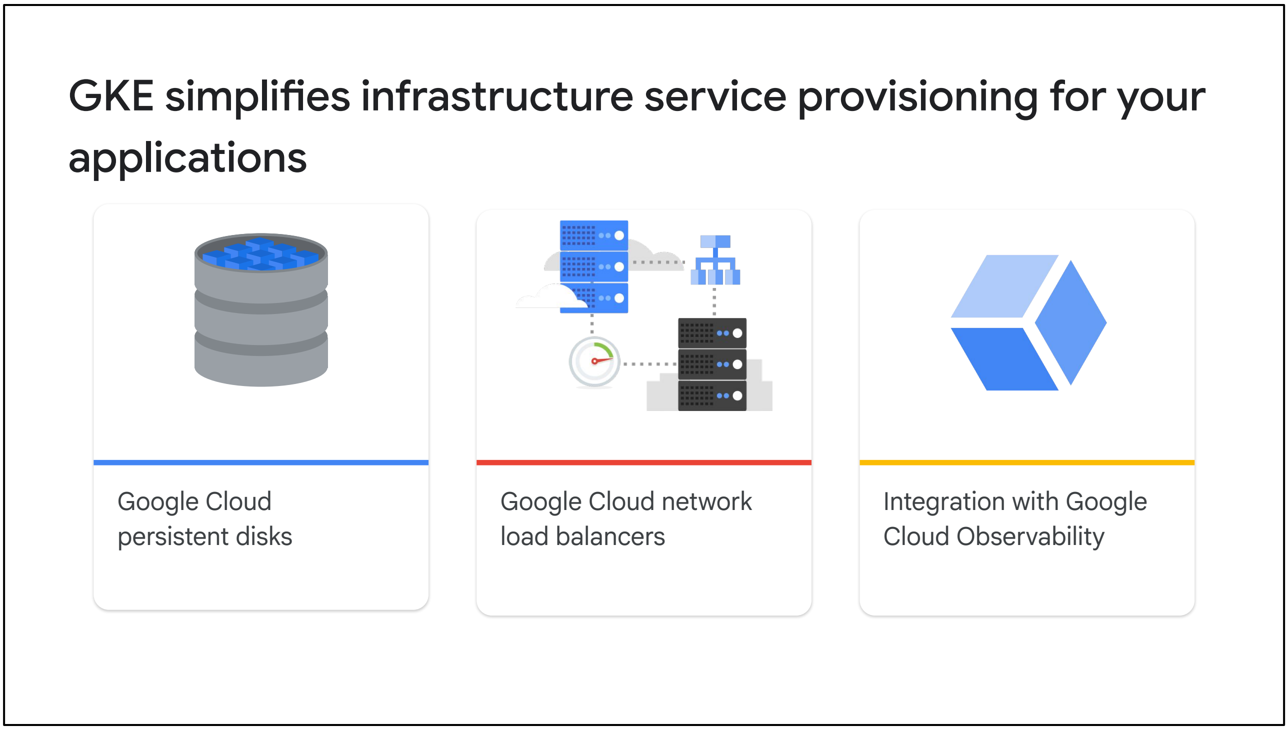
Hybrid or multi-cloud applications

Protocols other than HTTPS

- GKE supports any application runtime that you can package as a Docker image. GKE is ideally suited for containerized applications, including third-party containerized software.

- You can run your container image on Kubernetes in a hybrid or multi cloud environment. This feature is especially helpful when some parts of your application run on-premises and other parts run in the cloud.

- You can use GKE to run containerized applications that use network protocols other than HTTP and HTTPS.



GKE simplifies infrastructure service provisioning for your applications

Google Cloud persistent disks

Google Cloud network load balancers

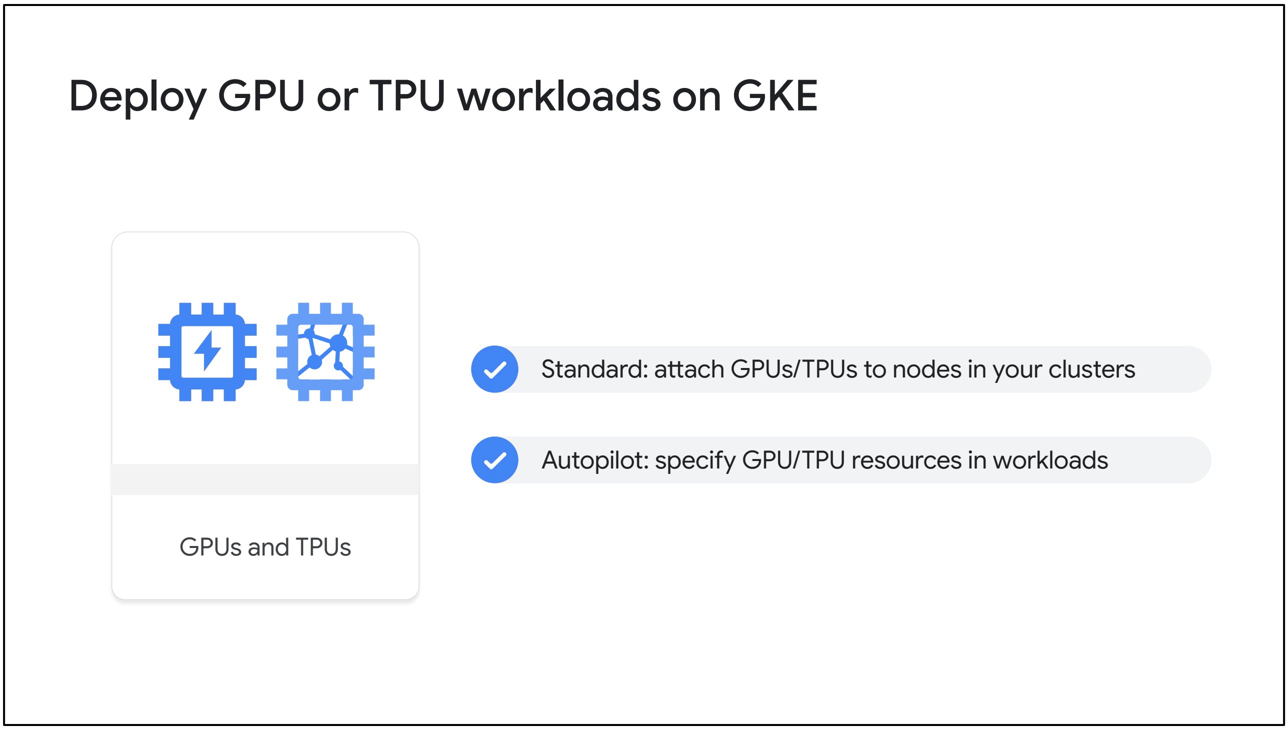
Integration with Google Cloud Observability

Managing the infrastructure for a Kubernetes environment can be complex. GKE simplifies many of the operational tasks associated with provisioning and managing the infrastructure.

- With GKE, Google Cloud persistent disks are automatically provisioned by default when you create Kubernetes persistent volumes to provide storage for stateful applications.

- GKE automatically provisions Google Cloud network load balancers when you deploy Kubernetes network load balancer services, and provisions Google Cloud HTTP and HTTP(S) Load Balancing when you configure Kubernetes Ingress resources. This auto-provisioning feature eliminates the need to configure and manage these resources manually.

- GKE has support for Google Cloud Observability, which provides integration with tools for troubleshooting and application and service monitoring.



Deploy GPU or TPU workloads on GKE

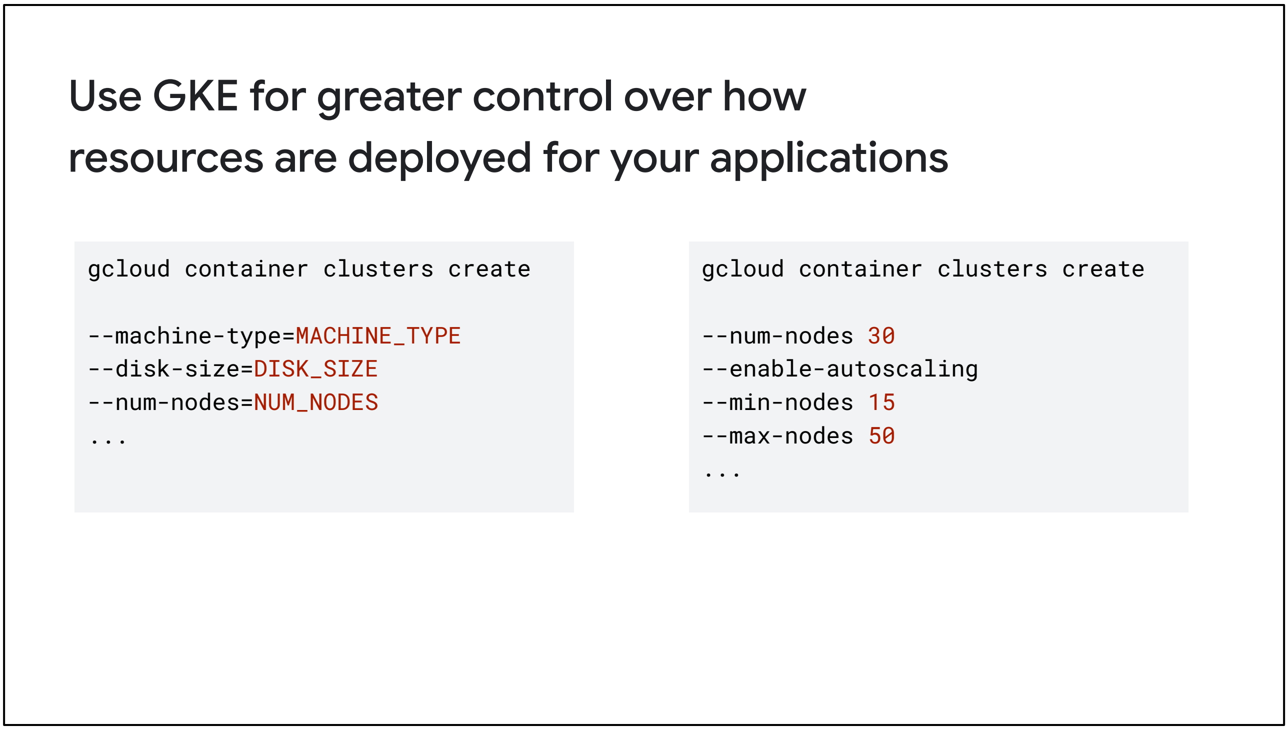
GPUs and TPUs

- ✓ Standard: attach GPUs/TPUs to nodes in your clusters
- ✓ Autopilot: specify GPU/TPU resources in workloads

With GKE, you can implement a robust, production-ready AI/ML platform with all the benefits of managed Kubernetes. GKE provides infrastructure orchestration that supports GPUs and TPUs for training and service of AI/ML workloads at scale.

With GKE Standard mode, you create node pools of VMs with attached GPUs or TPUs, and then allocate GPU or TPU resources to containerized workloads running on those nodes.

With GKE Autopilot mode, you specify the GPU or TPU resources you need for your workloads, and GKE can automatically manage nodes that provide those resources.



Use GKE for greater control over how resources are deployed for your applications

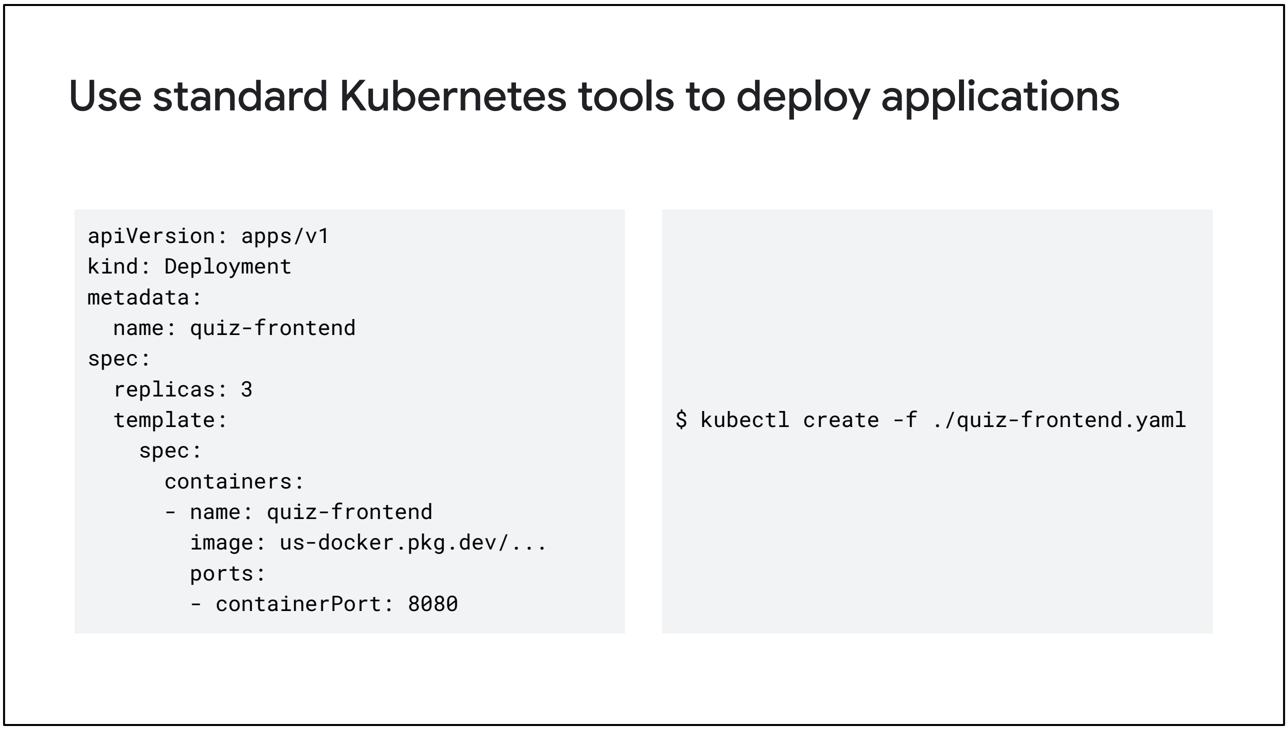
```
gcloud container clusters create

--machine-type=MACHINE_TYPE
--disk-size=DISK_SIZE
--num-nodes=NUM_NODES
...
```

```
gcloud container clusters create

--num-nodes 30
--enable-autoscaling
--min-nodes 15
--max-nodes 50
...
```

GKE simplifies cluster deployment and scaling. You can describe the compute, memory, network, and storage resources that you want to make available across all the containers required by your applications. GKE will provision and manage the underlying Google Cloud resources automatically. You can either deploy fixed-size clusters or configure your clusters to automatically scale. Autoscaling adds or removes compute instances in response to changes in the resource requirements of the containers that run inside the cluster.



Use standard Kubernetes tools to deploy applications

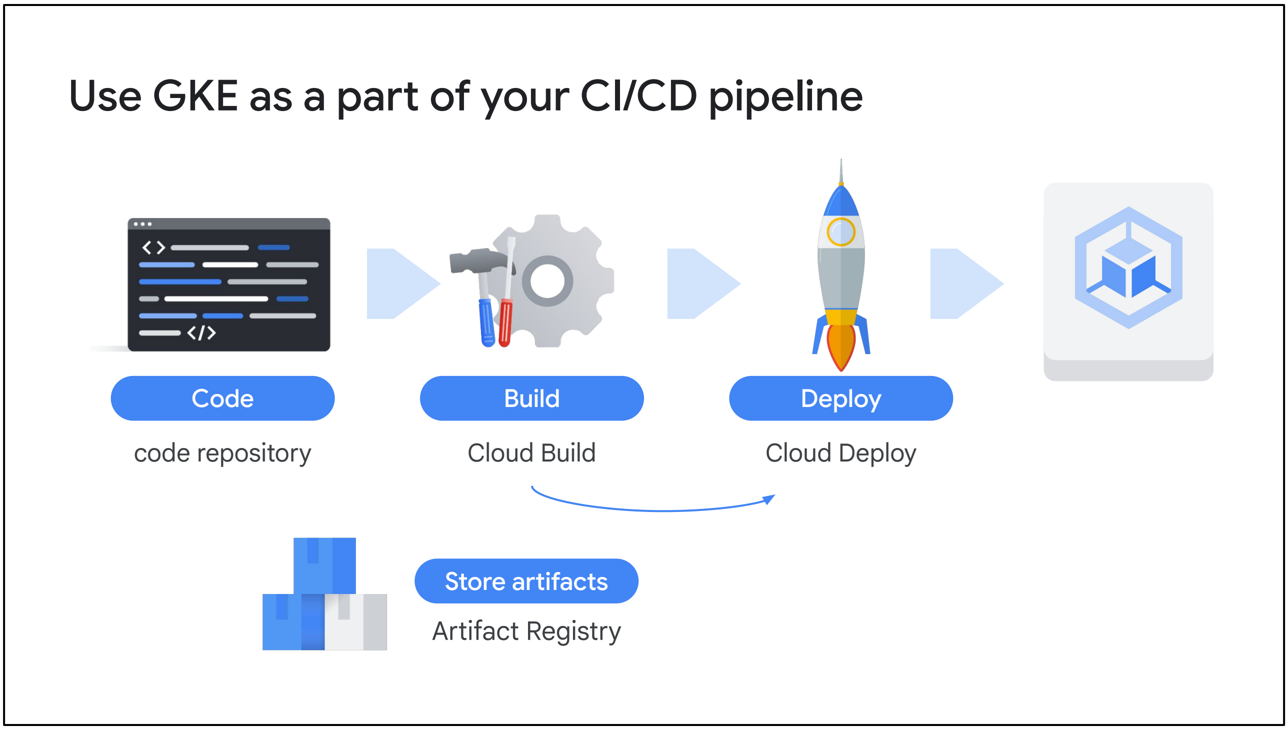
```yaml
apiVersion: apps/v1
kind: Deployment
metadata:
  name: quiz-frontend
spec:
  replicas: 3
  template:
    spec:
      containers:
      - name: quiz-frontend
        image: us-docker.pkg.dev/...
        ports:
        - containerPort: 8080
```

```
$ kubectl create -f ./quiz-frontend.yaml
```

You deploy and manage your containerized application for GKE the same way you would for any other Kubernetes environment. You can use the “kubectl” command to perform most operational tasks.

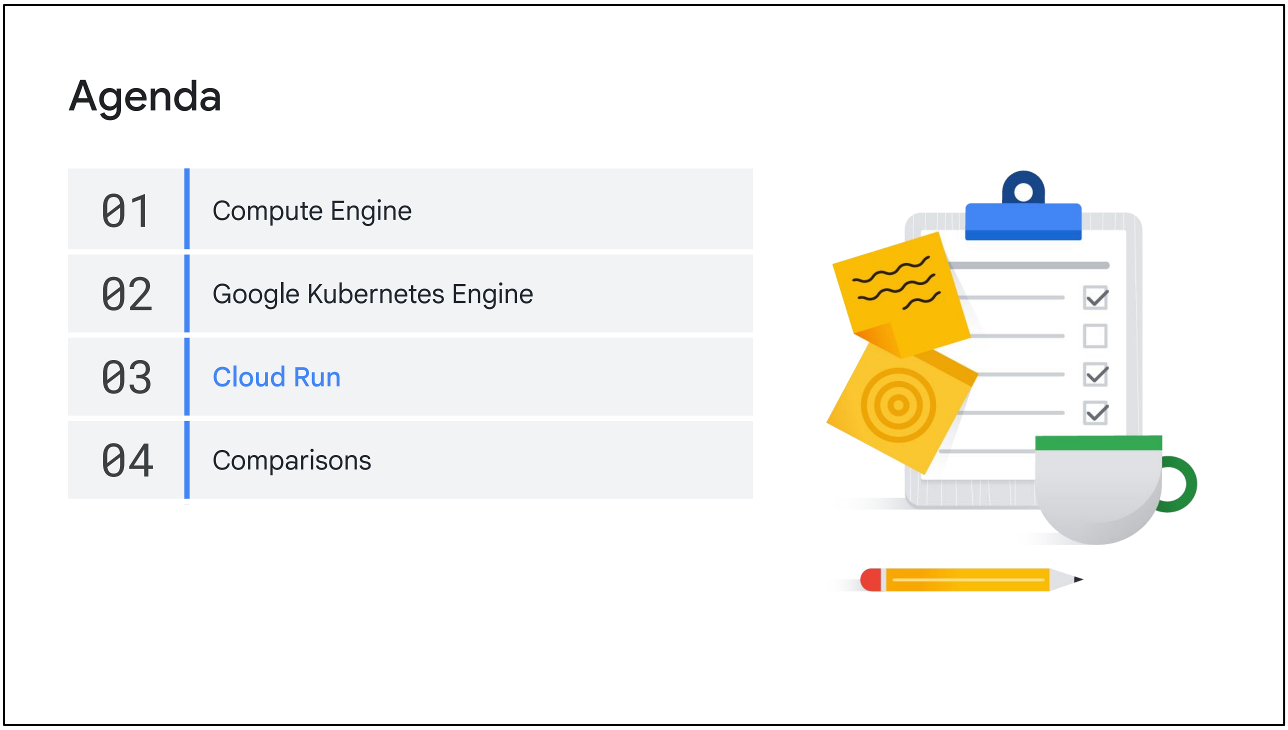
Although you can deploy ad hoc resources directly by using kubectl commands, the recommended best practice is to use YAML manifest files to define configurations. These files define the properties of the containers that are used for the components in your applications. Manifest files can also define the network services, security policies, and other Kubernetes objects that are used to deliver resilient, scalable, containerized applications.

Applications can be deployed by using Deployments, where Kubernetes continually ensures that a specified number of replicas for a Pod or set of Pods is running. The deployment shown here is for stateless components. You can also use StatefulSets for applications where you need persistent storage. You can also use YAML manifests to define a range of other resource types.



Use GKE as a part of your CI/CD pipeline

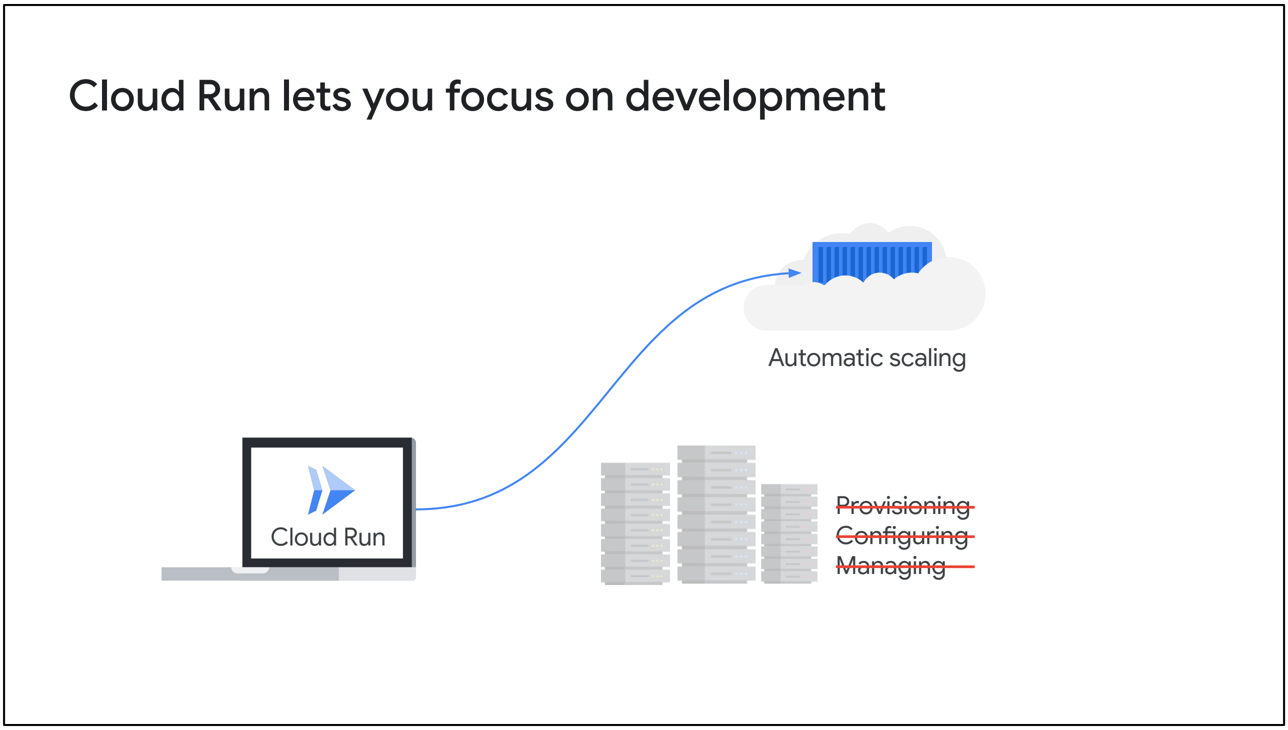
As a part of a continuous integration and delivery (CI/CD) pipeline, you can generate a new Docker image for each code commit. The CI/CD pipeline can automatically deploy the image to development, test, and production environments. Cloud Build, Artifact Registry, Cloud Deploy, and GKE can be used to create a strong CI/CD system.



Agenda

01	Compute Engine
02	Google Kubernetes Engine
03	Cloud Run
04	Comparisons

Next is Cloud Run. Cloud Run is a fully managed compute platform that allows you to run request or event-driven stateless workloads without having to worry about servers.



Cloud Run lets you focus on development

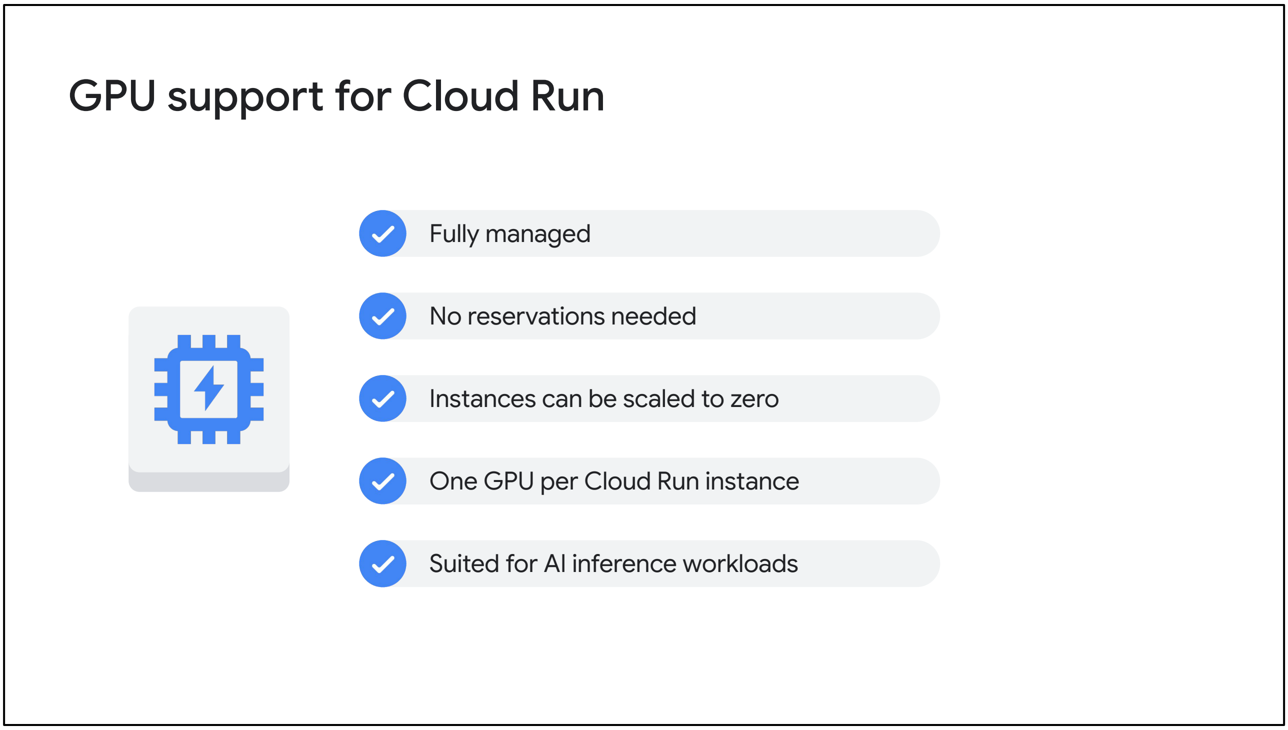
Automatic scaling

Cloud Run

~~Provisioning~~
~~Configuring~~
~~Managing~~

It abstracts away all infrastructure management such as provisioning, configuring, and managing servers, so you can focus on writing code. It automatically scales up and down from zero, almost instantaneously, depending on traffic, so you never have to worry about scale configuration. Cloud Run also charges you only for the resources you use, rounded up to the nearest 100 milliseconds, so you never pay for overprovisioned resources.

[Cloud Run: https://cloud.google.com/run/]



GPU support for Cloud Run

- ✓ Fully managed
- ✓ No reservations needed
- ✓ Instances can be scaled to zero
- ✓ One GPU per Cloud Run instance
- ✓ Suited for AI inference workloads

Cloud Run services and functions can also use GPUs.

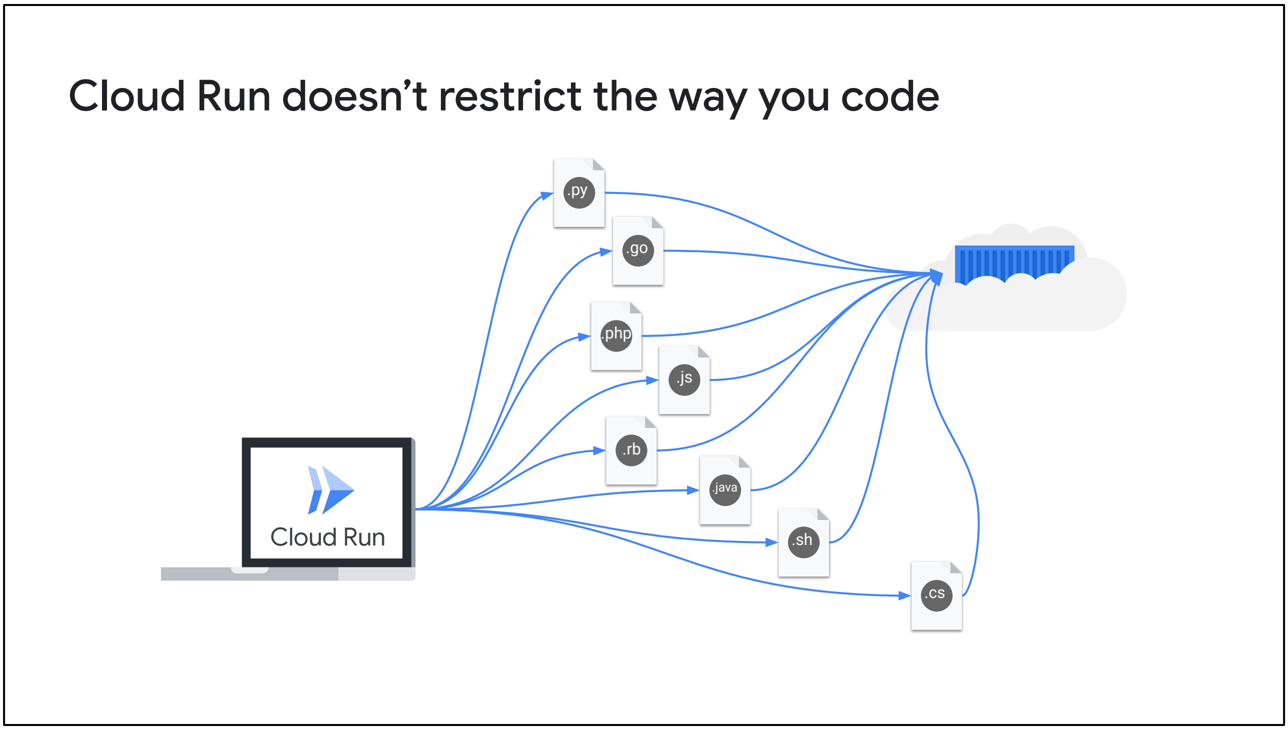
GPU on Cloud Run is fully managed, with no extra drivers or libraries needed.

The GPU feature offers on-demand availability without requiring reservations.

Instances of a Cloud Run service that has been configured to use GPU can scale down to zero for cost savings when not in use.

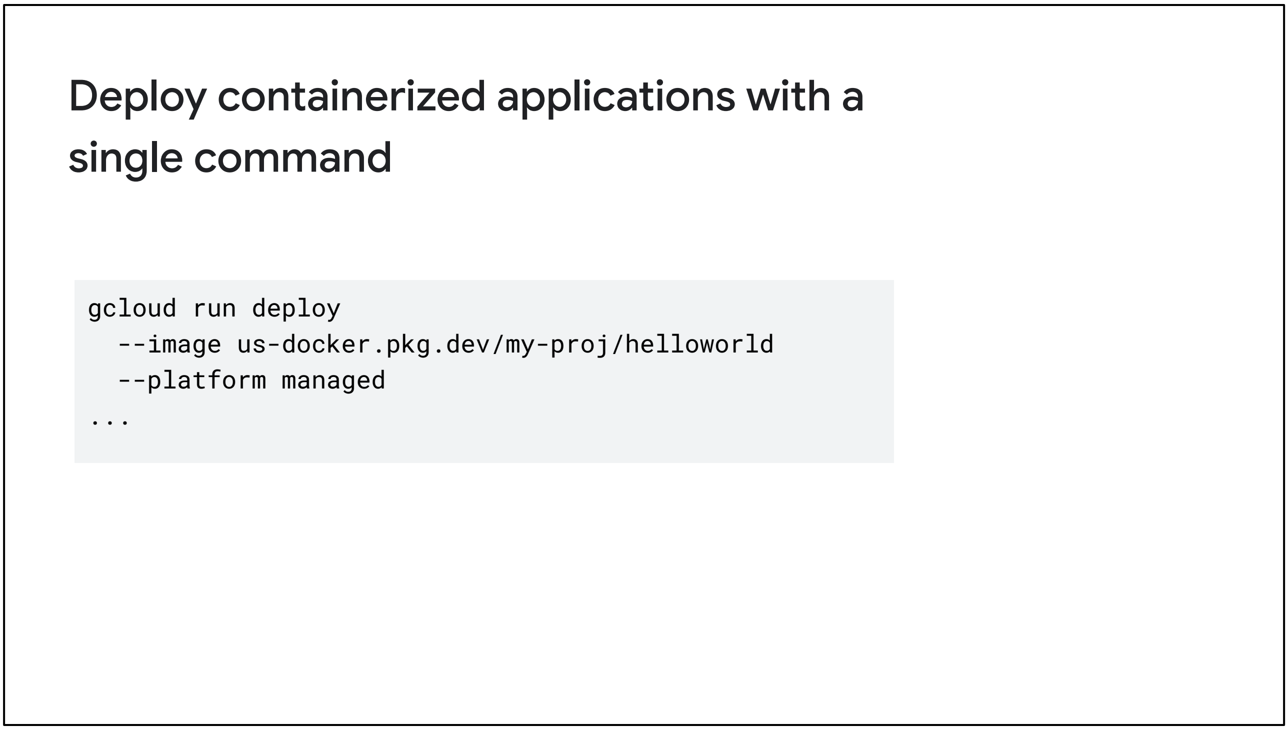
You can configure one GPU per Cloud Run instance.

Cloud Run services and functions with configured GPUs are well suited for running AI inference workloads with large language models. They are also appropriate for compute intensive non-AI use cases like video transcoding and 3D rendering.



Cloud Run doesn't restrict the way you code

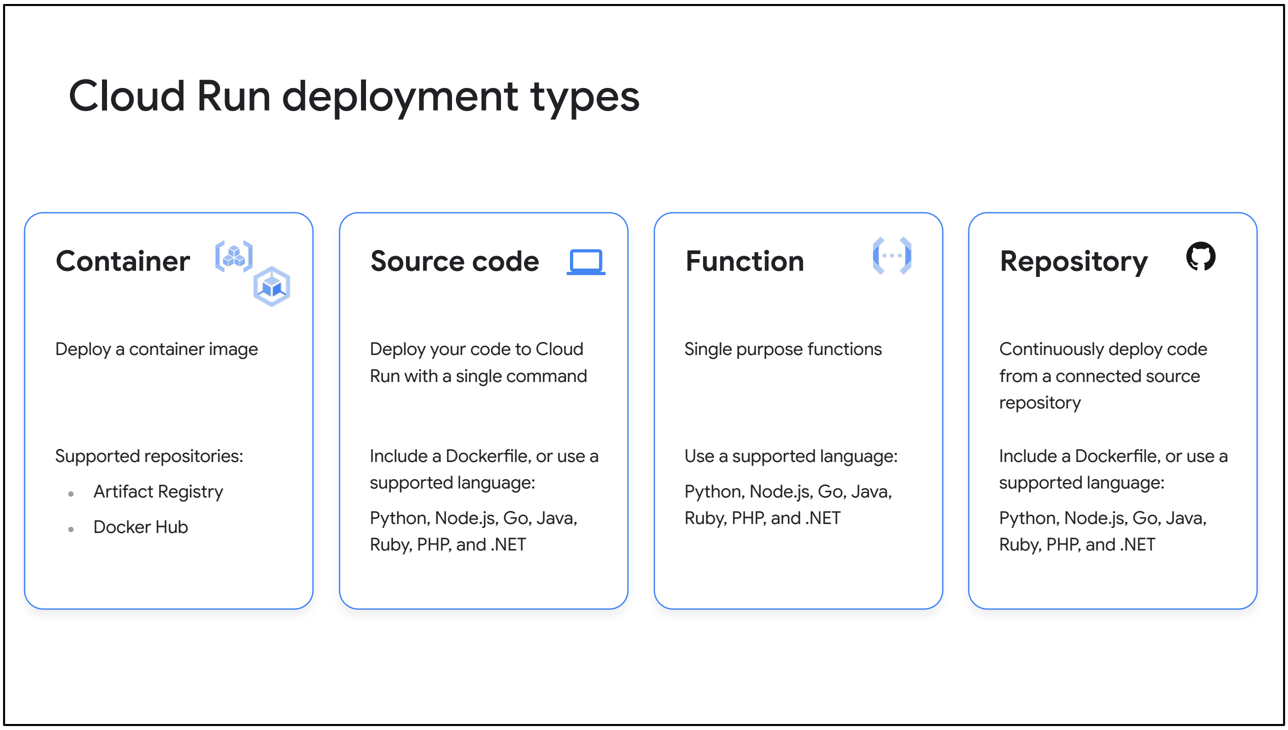
Many serverless platforms add constraints around support for languages and libraries, or even restrict the way you code. Cloud Run enables you to write code your way by letting you easily deploy any stateless containers that listen for requests or events delivered over HTTP or gRPC. Containers offer flexibility and portability of workloads. With Cloud Run, you can build your applications in any language using whatever frameworks and tools you want and deploy them in seconds.



Deploy containerized applications with a single command

```
gcloud run deploy
  --image us-docker.pkg.dev/my-proj/helloworld
  --platform managed
...
```

You can use a single command to deploy containerized applications. After that, Cloud Run horizontally scales your container image automatically in order to handle received requests, then scales it down when demand decreases. You only pay for the CPU, memory, and networking that are consumed during request handling.



Cloud Run deployment types

Container

Deploy a container image

Supported repositories:

- Artifact Registry
- Docker Hub

Source code

Deploy your code to Cloud Run with a single command

Include a Dockerfile, or use a supported language:

Python, Node.js, Go, Java, Ruby, PHP, and .NET

Function

Single purpose functions

Use a supported language:

Python, Node.js, Go, Java, Ruby, PHP, and .NET

Repository

Continuously deploy code from a connected source repository

Include a Dockerfile, or use a supported language:

Python, Node.js, Go, Java, Ruby, PHP, and .NET

Cloud Run offers several deployment options. All deployment options result in a container image that runs on Cloud Run's fully managed and highly scalable infrastructure.

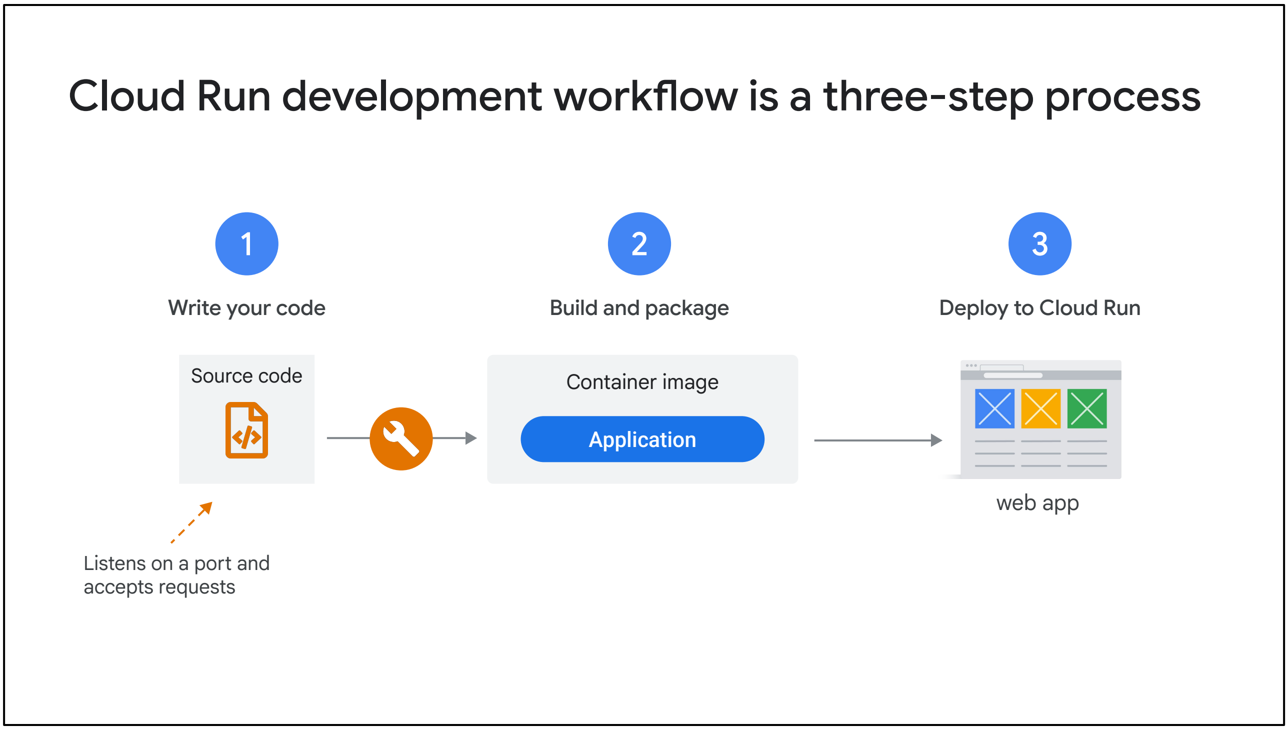
The first deployment type is a container. You can supply any container that uses any base image and runs code written in the programming language of your choice, provided that the container and app it runs follow specific rules. For example, the container and app should listen for requests on a port at 0.0.0.0, and the container should use HTTP without transport layer security (TLS), because TLS is terminated by Cloud Run, and requests are proxied to the container without TLS. You can directly deploy container images stored in Artifact Registry or Docker Hub.

The second deployment type is source code. Cloud Run lets you build and deploy source code from a single **gcloud run deploy** command. When deploying source code, Cloud Build transforms the code into a container image stored in Artifact Registry. You can deploy sources that include a Dockerfile, or, if no Dockerfile is present in the source code directory, source code must be written in one of the supported languages: Python, Node.js, Go, Java, Ruby, PHP, or .NET. The deploy command uses Google Cloud buildpacks and Cloud Build to automatically build container images from your source code. If a Dockerfile is present in the source code directory, the uploaded source code is built using that Dockerfile. If no Dockerfile is present, the language you are using is automatically detected, dependencies of the code are fetched, and a production-ready container image using a secure base image is built. The built container images are stored in Artifact Registry.

Third, you can deploy single-purpose functions that trigger when events are emitted from your cloud infrastructure and services, or from supported third-party providers. For functions, you provide only the function code in one of the supported languages, and a buildpack creates the container. Functions are deployed as Cloud Run services.

Finally, you can use continuous source deployment from a GitHub repository to deploy a Cloud Run service or function. Cloud Build can automate builds and deployments to Cloud Run when new commits are pushed to a specific branch of a Git repository. You can deploy source code stored in the repository that includes a Dockerfile or is written in one of the programming languages supported by Google Cloud buildpacks. For other repository providers, you can use Cloud Build and Cloud Deploy to set up your own CI/CD continuous deployment to Cloud Run.

[Deployment options and resource model:
https://cloud.google.com/run/docs/resource-model]
[Cloud Run container runtime contract:
https://cloud.google.com/run/docs/container-contract]



Cloud Run development workflow is a three-step process

1 Write your code

2 Build and package

3 Deploy to Cloud Run

Source code

Container image

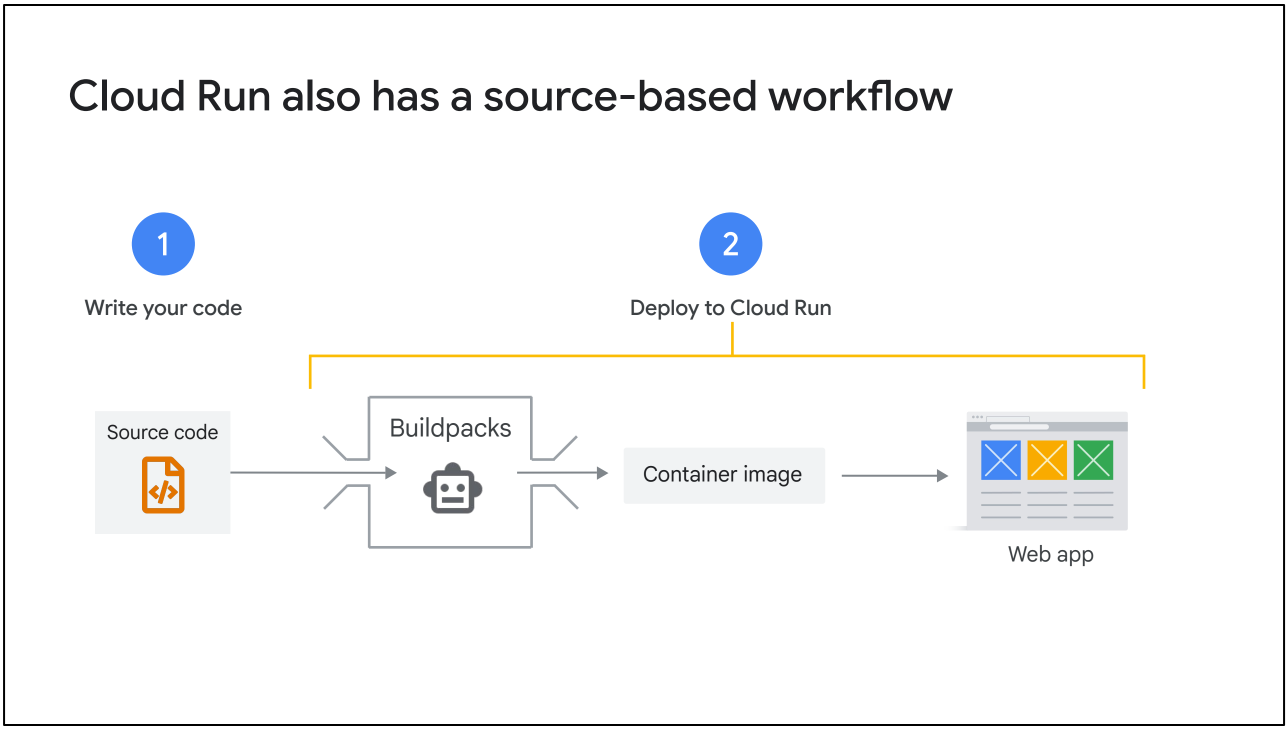
Application

web app

Listens on a port and accepts requests

The Cloud Run developer workflow is a straightforward three-step process.

- First, use your favorite programming language to write your application. This application should listen for web requests.
- Second, build and package your application into a container image.
- Finally, deploy the container image to Cloud Run.
  When you deploy your container image, you get a unique HTTP(S) URL. Cloud Run starts your container on demand to handle requests and dynamically adds and removes containers to ensure capacity to handle all incoming requests.



Cloud Run also has a source-based workflow

1

Write your code

2

Deploy to Cloud Run

Source code

Buildpacks

Container image

Web app

When you build a container image, you have full control over how the software is built and how every file is added. Sometimes you need this control.

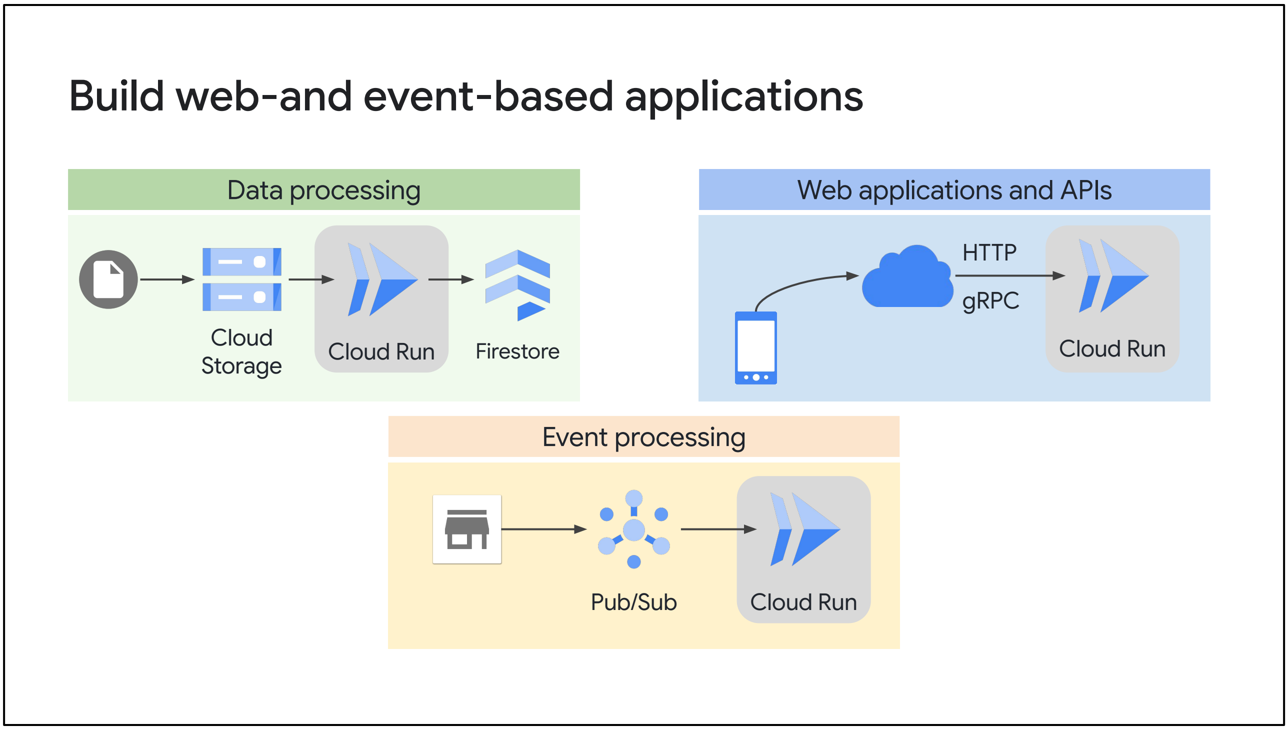
However, building an application is difficult enough already. And sometimes, you just want to turn source code into an HTTPS endpoint. You want to ensure that your container image is secure, well configured, and built in a consistent way.

With Cloud Run, you can choose either approach. You can use a container-based workflow *or* a source-based workflow.

When you use the source-based approach, you deploy your source code instead of a container image. Cloud Run then builds your source and packages the application into a secure container image for you automatically.

Cloud Run uses buildpacks to automatically build container images. Buildpacks transform your application source code into container images that can run on any cloud.

[Cloud Native Buildpacks: https://buildpacks.io/
Google Cloud buildpacks repository:
https://github.com/GoogleCloudPlatform/buildpacks]



Build web-and event-based applications

Data processing

Cloud Storage

Cloud Run

Firestore

Web applications and APIs

HTTP

gRPC

Cloud Run

Event processing

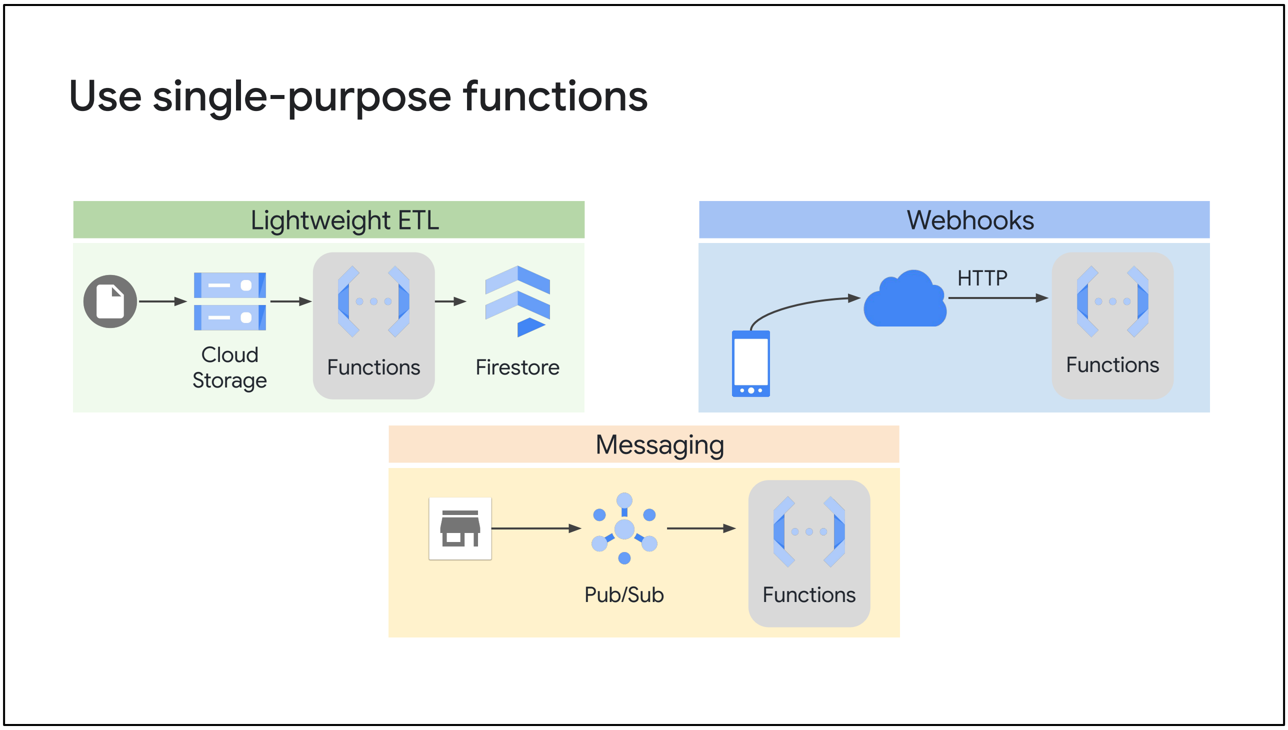
Pub/Sub

Cloud Run

Cloud Run can be used for many use cases.

- Cloud Run is an excellent choice for data processing applications. Because you pay for usage—not provisioned resources—you only pay when data is being processed.
- Cloud Run can also host small or large web applications and web APIs. Cloud Run scales up when necessary and scales down to zero when it's not in use.
- Jobs that must be run in response to Pub/Sub or Eventarc events are good candidates for Cloud Run.

[Can Cloud Run handle these 9 workloads: https://www.youtu.be/R2b7aZTRf-c]



Use single-purpose functions

Lightweight ETL

Cloud Storage

Functions

Firestore

Webhooks

HTTP

Functions

Messaging

Pub/Sub

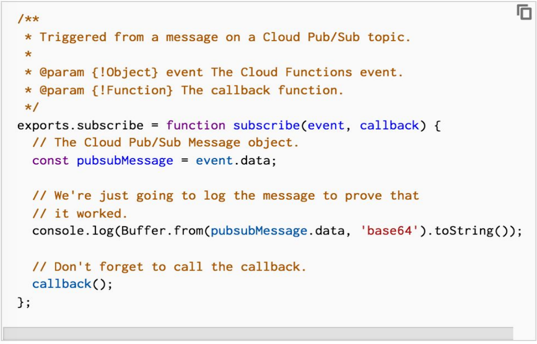
Functions

Functions deployed to Cloud Run are single-purpose, and are appropriate for many use cases.

- You can use functions for lightweight extract-transform-load, or ETL, operations, or for processing messages that are published to a Pub/Sub topic.
- Functions can also serve as a target for webhooks, which allow applications or services to make direct HTTP calls to invoke microservices.
- Functions are ideal for microservices that require a small piece of code to quickly process data in response to an event.

Focus on code: Cloud Client Libraries

index.js

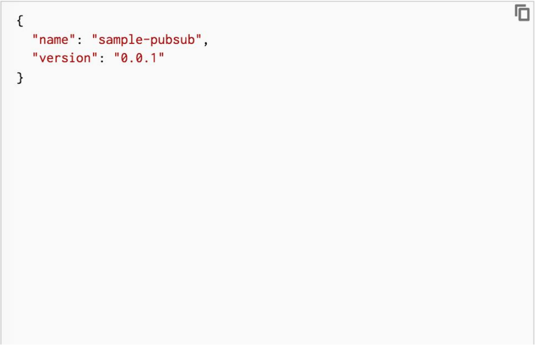


```
/**
 * Triggered from a message on a Cloud Pub/Sub topic.
 *
 * @param {!Object} event The Cloud Functions event.
 * @param {!Function} The callback function.
 */
exports.subscribe = function subscribe(event, callback) {
  // The Cloud Pub/Sub Message object.
  const pubsubMessage = event.data;

  // We're just going to log the message to prove that
  // it worked.
  console.log(Buffer.from(pubsubMessage.data, 'base64').toString());

  // Don't forget to call the callback.
  callback();
};
```

package.json

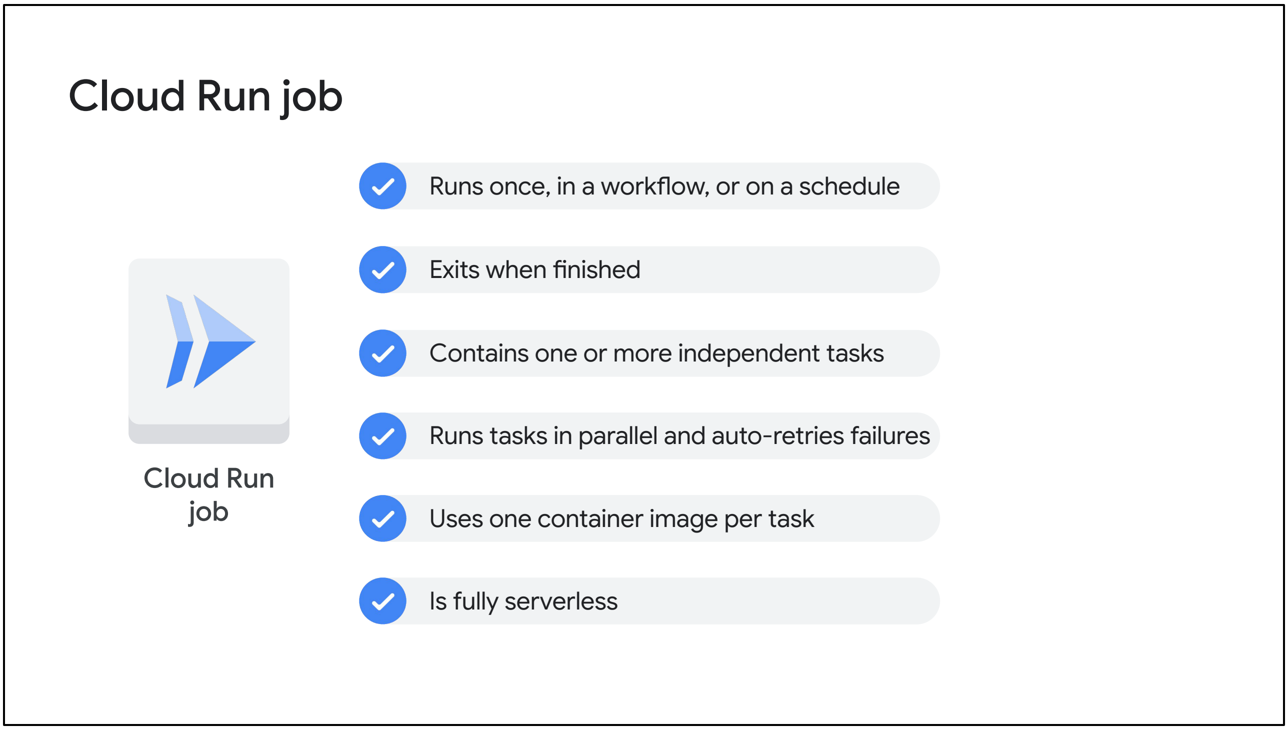


```
{
  "name": "sample-pubsub",
  "version": "0.0.1"
}
```

Functions let you focus on code. Let’s look at Node.js as an example. When you use the Node.js runtime, your function's source code must be exported in a Node.js module.

You do not need to upload ZIP files with packaged dependencies. You can specify any dependencies for your Node.js function in a package.json file. Cloud Run automatically installs all dependencies before running your code.

You can use Cloud Client Libraries to programmatically interact with other Google Cloud services.



Cloud Run job

Cloud Run job

- ✓ Runs once, in a workflow, or on a schedule
- ✓ Exits when finished
- ✓ Contains one or more independent tasks
- ✓ Runs tasks in parallel and auto-retries failures
- ✓ Uses one container image per task
- ✓ Is fully serverless

Cloud Run jobs work differently from HTTP Cloud Run services.

A Cloud Run job doesn't listen for and serve HTTP requests. There is no need to listen on a port or start a web server. Instead, the job is executed as a one-off task or as part of a workflow. You can also use Cloud Scheduler to run a job on a regular schedule.

When a Cloud Run job finishes, the job exits.

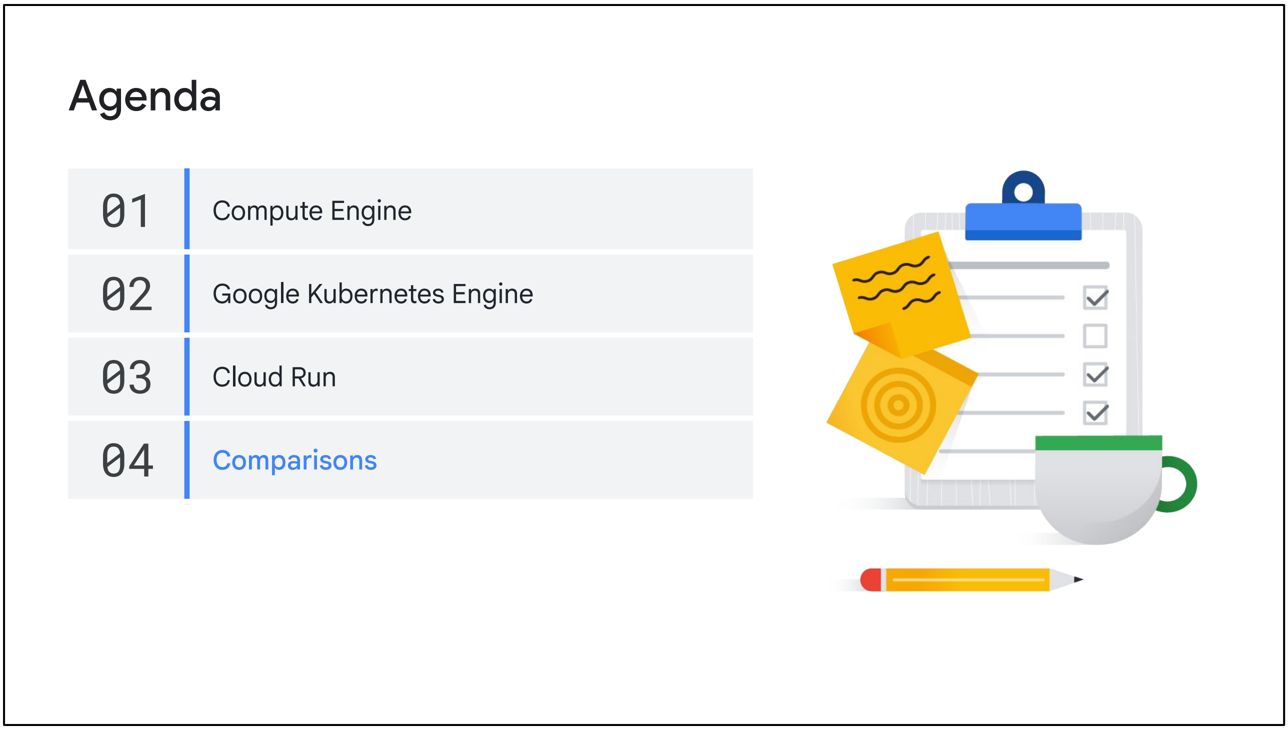
Each job can be composed of a single task or multiple independent tasks.

Because multiple tasks within a job are independent, the tasks can be run in parallel. Each task execution is aware of its task index, which is provided to the task in an environment variable. In addition, tasks that fail can be automatically retried. The maximum number of parallel tasks can be specified so that you don't overwhelm backend services with too many concurrent tasks.

Each task within a job runs a single container image. This container runs to completion.

Like Cloud Run services, Cloud Run jobs run on a fully serverless platform. You don't need to manage any infrastructure to run your jobs.

[Create jobs: https://cloud.google.com/run/docs/create-jobs]



Agenda

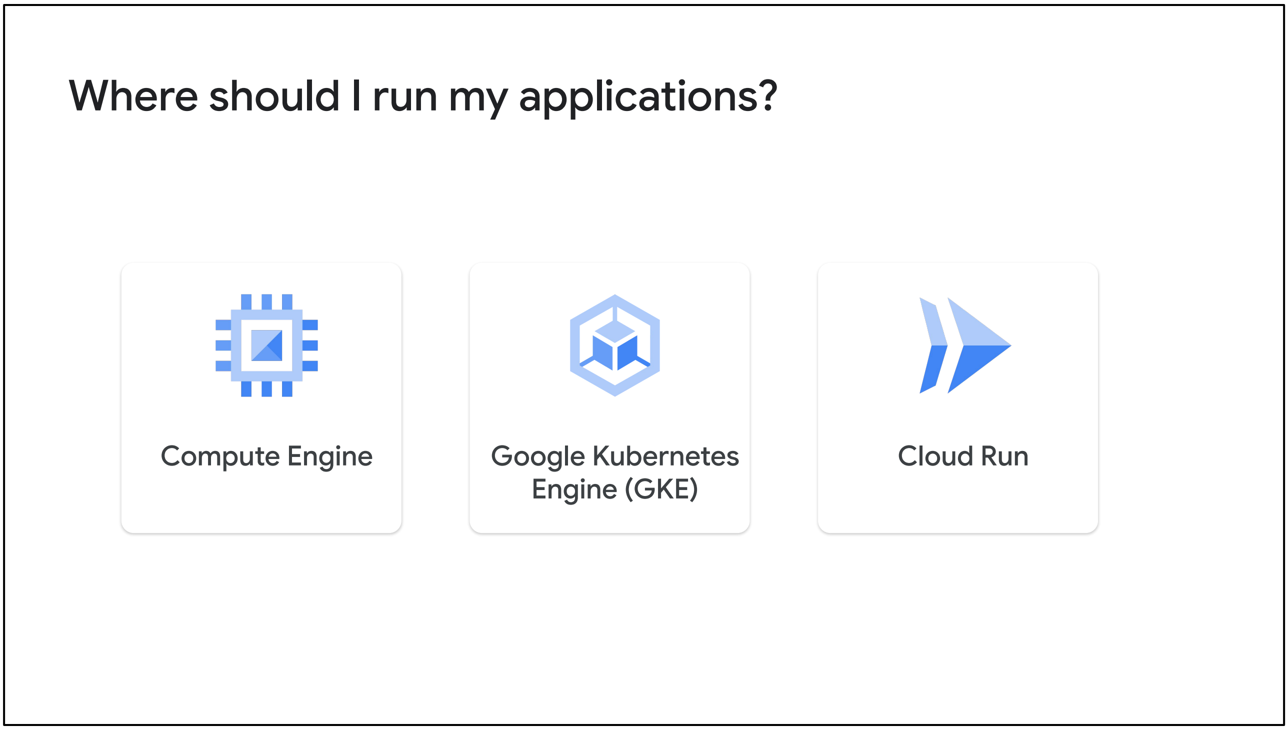
01 Compute Engine

02 Google Kubernetes Engine

03 Cloud Run

04 Comparisons

Now we compare the platforms to understand their relative strengths.



Where should I run my applications?

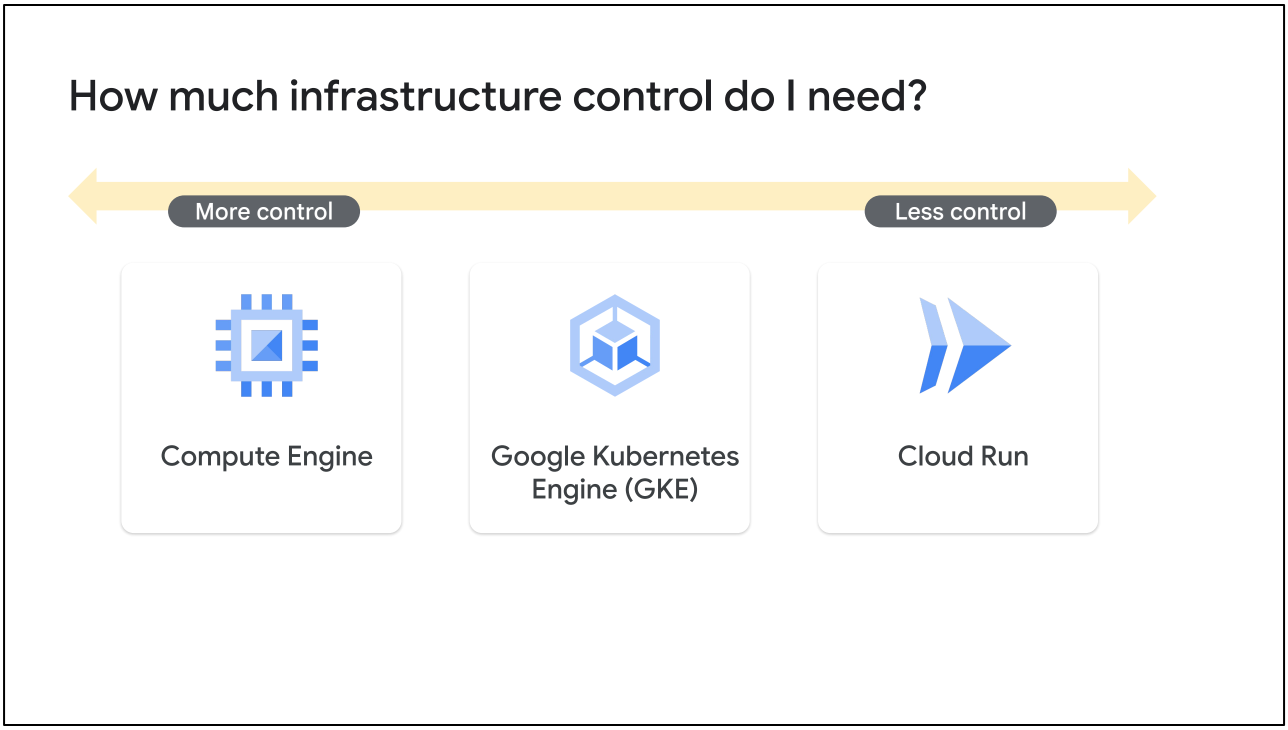
Compute Engine

Google Kubernetes Engine (GKE)

Cloud Run

After we’ve talked about those platforms, the question remains, "Where should I run my applications?"

And the answer is still “It depends.”



How much infrastructure control do I need?

More control

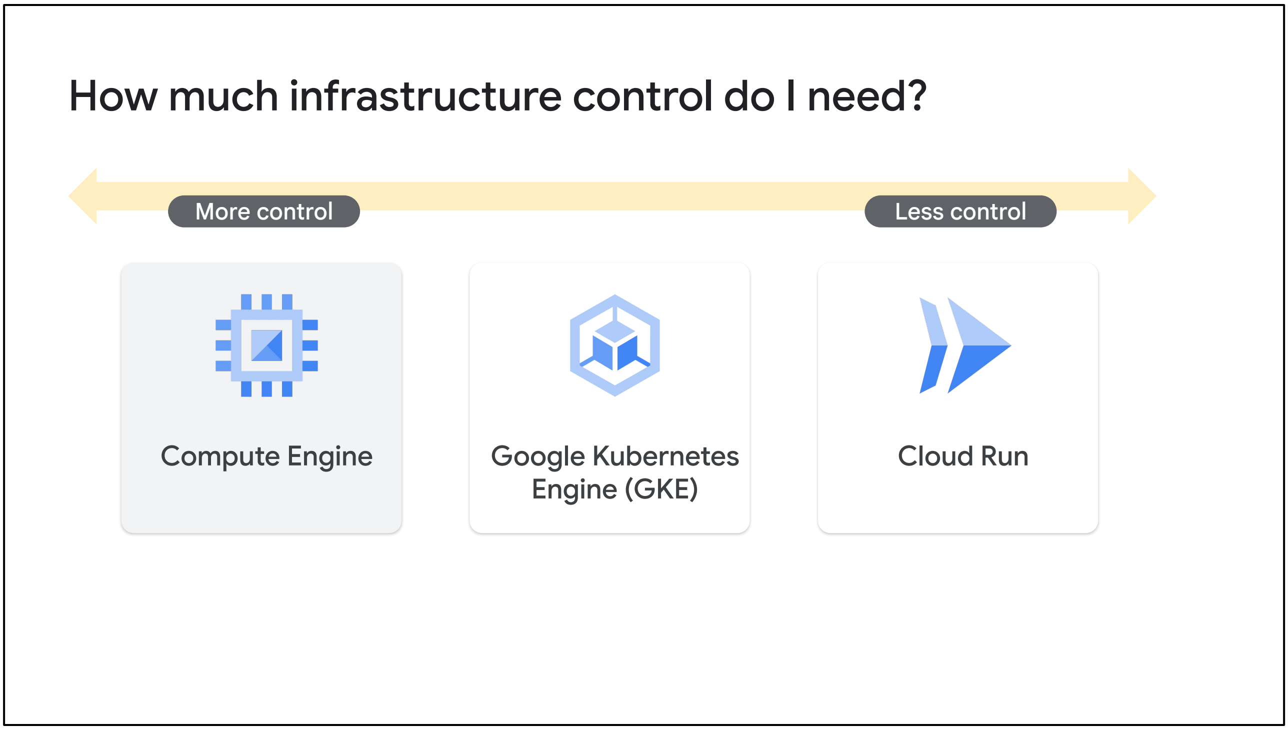
Less control

Compute Engine

Google Kubernetes Engine (GKE)

Cloud Run

The first question that you might ask is, “How much infrastructure control do I need?”



How much infrastructure control do I need?

More control

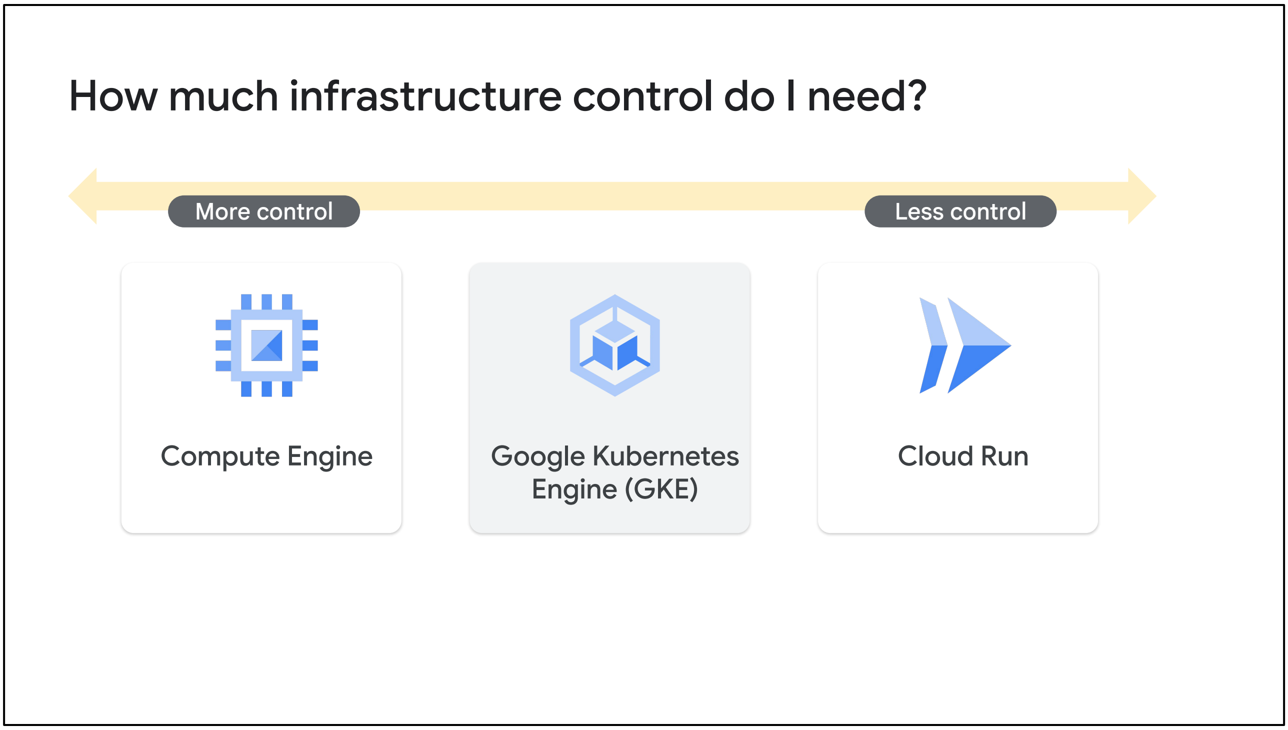
Less control

Compute Engine

Google Kubernetes Engine (GKE)

Cloud Run

If you want to lift-and-shift legacy systems to the cloud, or you have specific licensing requirements that depend on specific hardware, you might need to use Compute Engine.



How much infrastructure control do I need?

More control

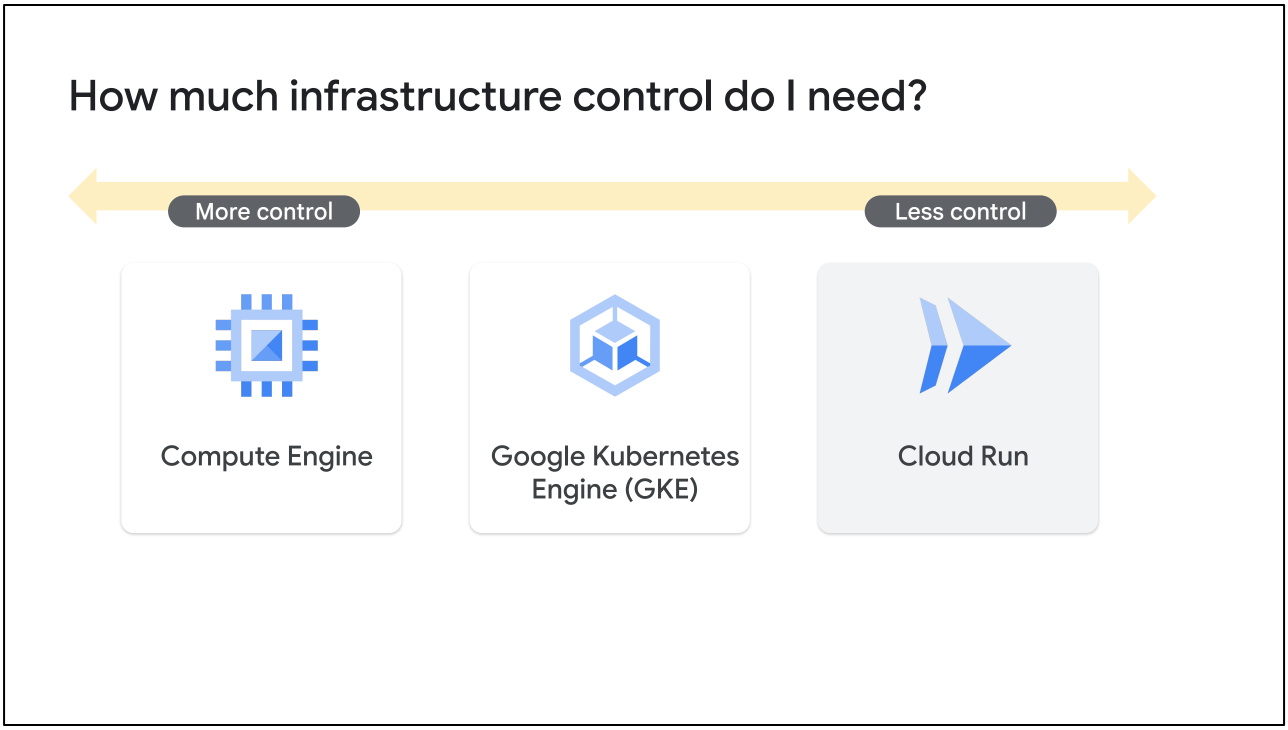
Less control

Compute Engine

Google Kubernetes Engine (GKE)

Cloud Run

If you can run containers and have hybrid systems on multiple clouds or data centers, or if you have applications that are not HTTP-based, Google Kubernetes Engine might be the right choice.



How much infrastructure control do I need?

More control

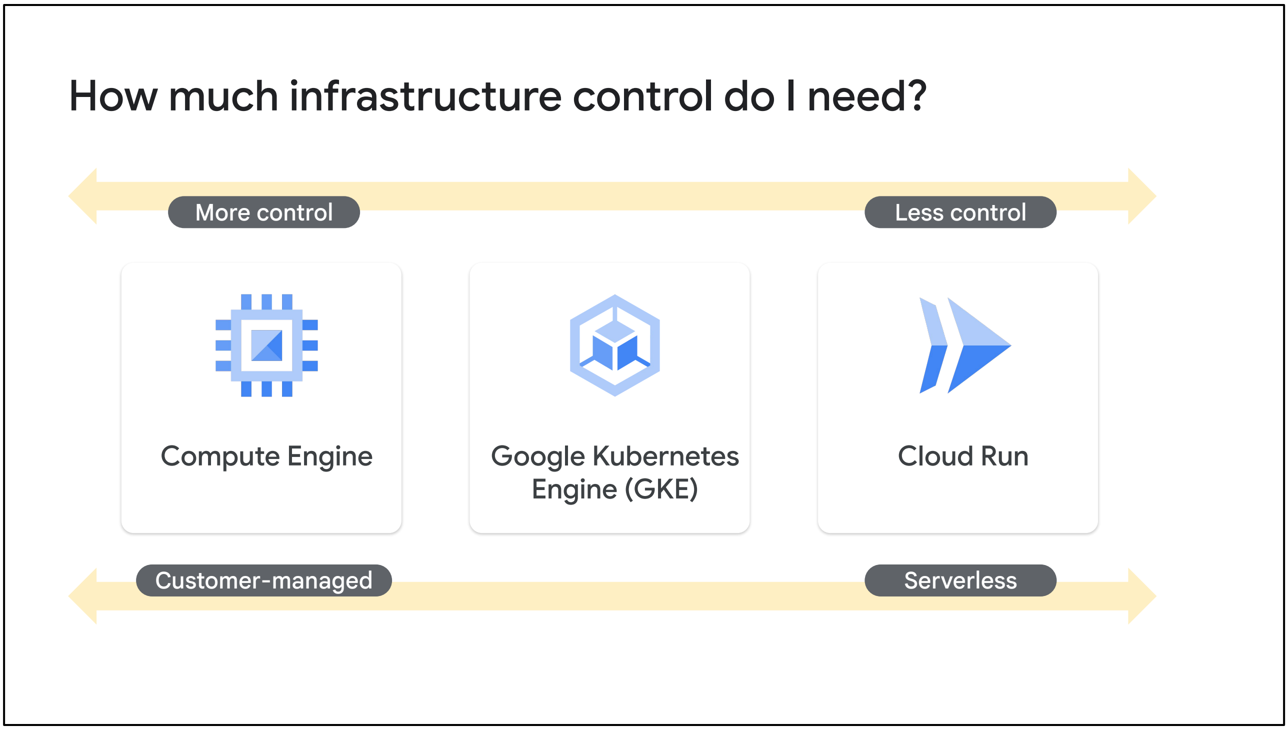
Less control

Compute Engine

Google Kubernetes Engine (GKE)

Cloud Run

If you want to run stateless containers but not manage the infrastructure at all, or you just need to write event-driven functions to connect cloud services, Cloud Run is probably the best choice.



How much infrastructure control do I need?

More control

Less control

Compute Engine

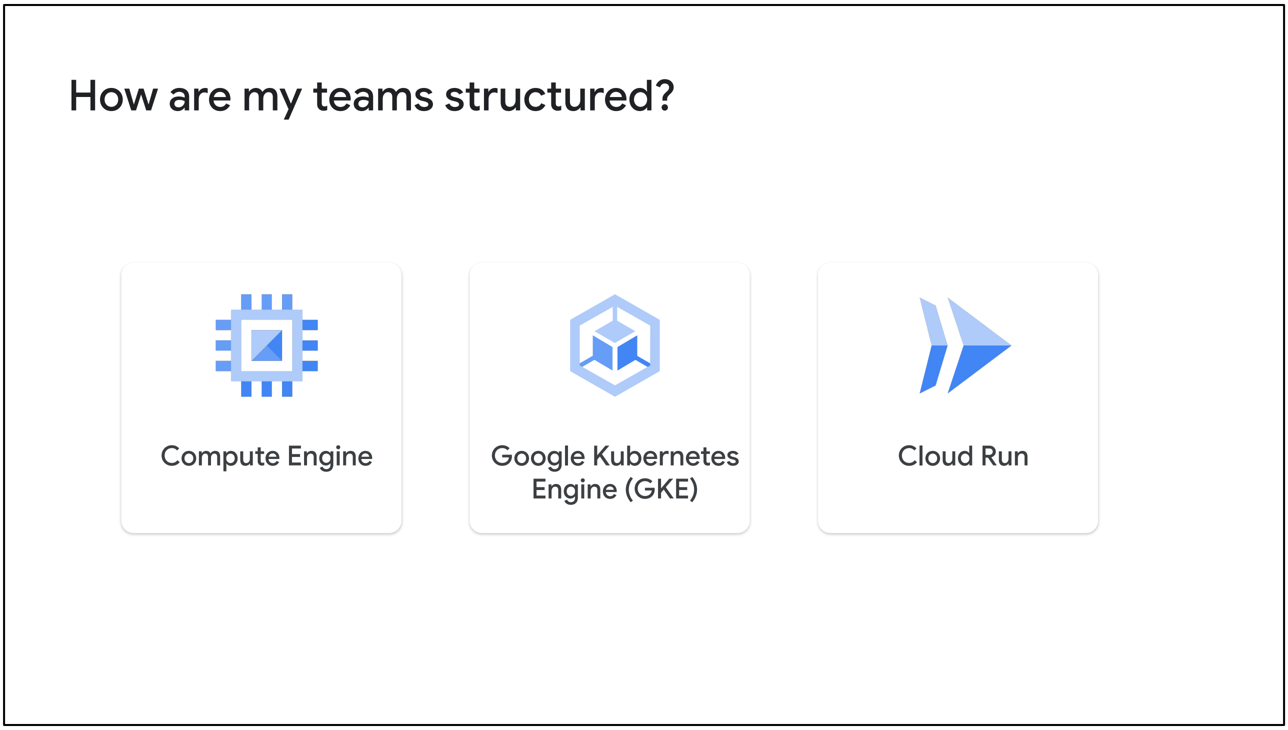
Google Kubernetes Engine (GKE)

Cloud Run

Customer-managed

Serverless

Gaining more control of infrastructure requires more operational effort. When you create a Compute Engine virtual machine, you control the updates for the operating system and software. With GKE, Google manages the virtual machine nodes for your cluster, but you still manage the size of the cluster and decide how to scale each application within the cluster. Cloud Run is serverless. For Cloud Run, you just need to deploy your application, and Google manages the infrastructure and autoscaling.



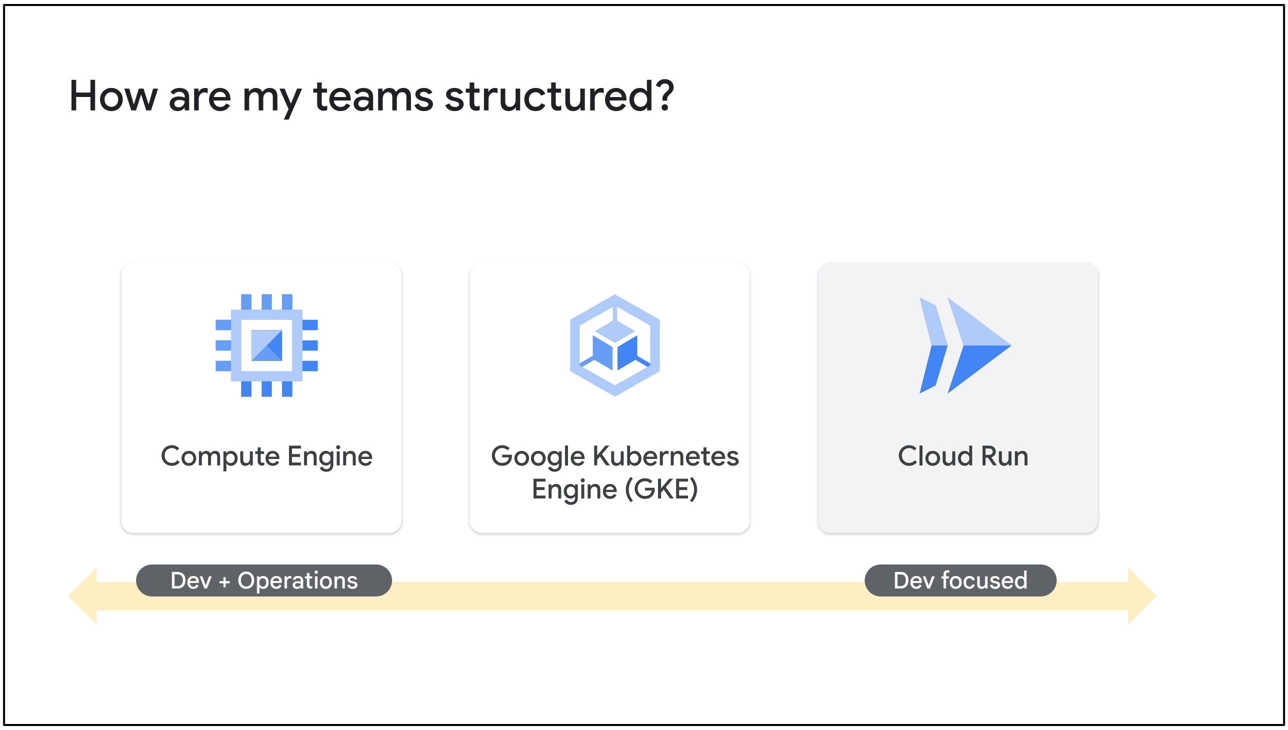
How are my teams structured?

Compute Engine

Google Kubernetes Engine (GKE)

Cloud Run

Another question is, “How are my teams structured?”



How are my teams structured?

Compute Engine

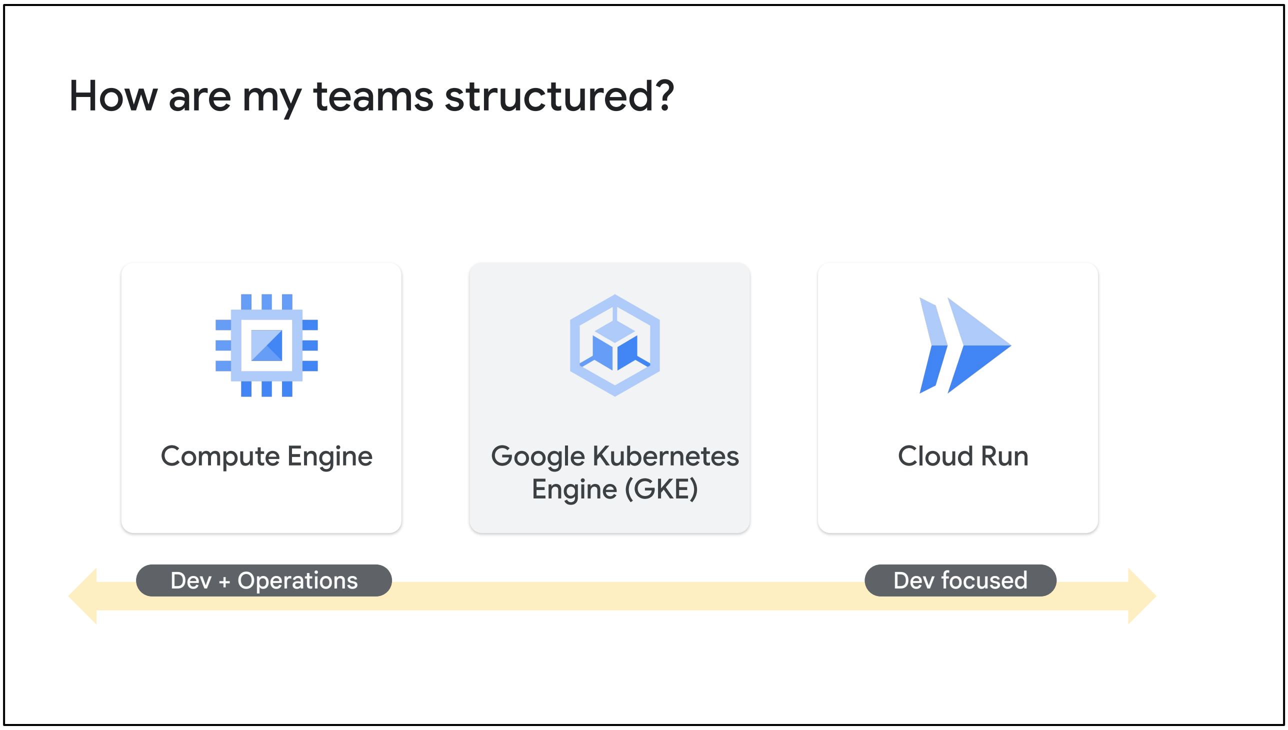
Google Kubernetes Engine (GKE)

Cloud Run

Dev + Operations

Dev focused

If your teams are mostly developer-focused, Cloud Run is probably best for you. If you have both developers and operations teams, you might still use Cloud Run services and functions when appropriate.



How are my teams structured?

Compute Engine

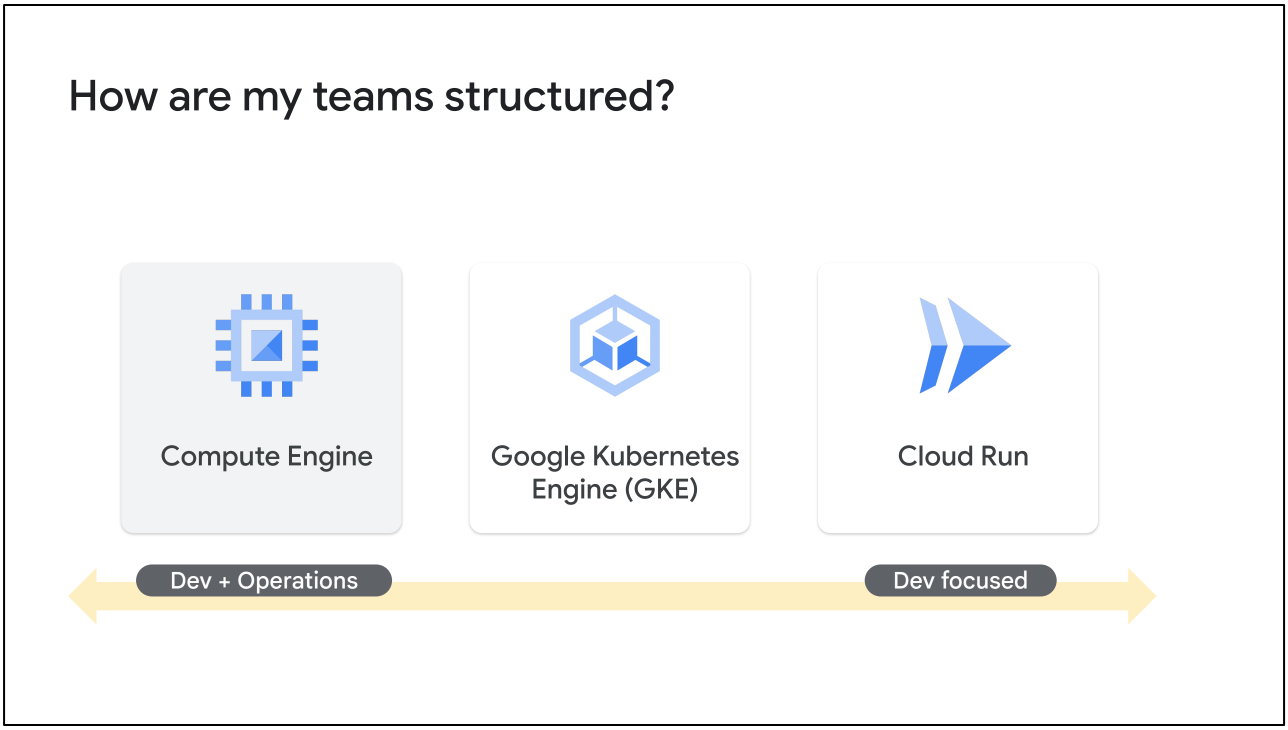
Google Kubernetes Engine (GKE)

Cloud Run

Dev + Operations

Dev focused

You might also decide to use Google Kubernetes Engine to integrate with your hybrid systems and have more control over your workloads. Stateful applications and non-HTTP network protocols can also be used with GKE, but not with Cloud Run.



How are my teams structured?

Compute Engine

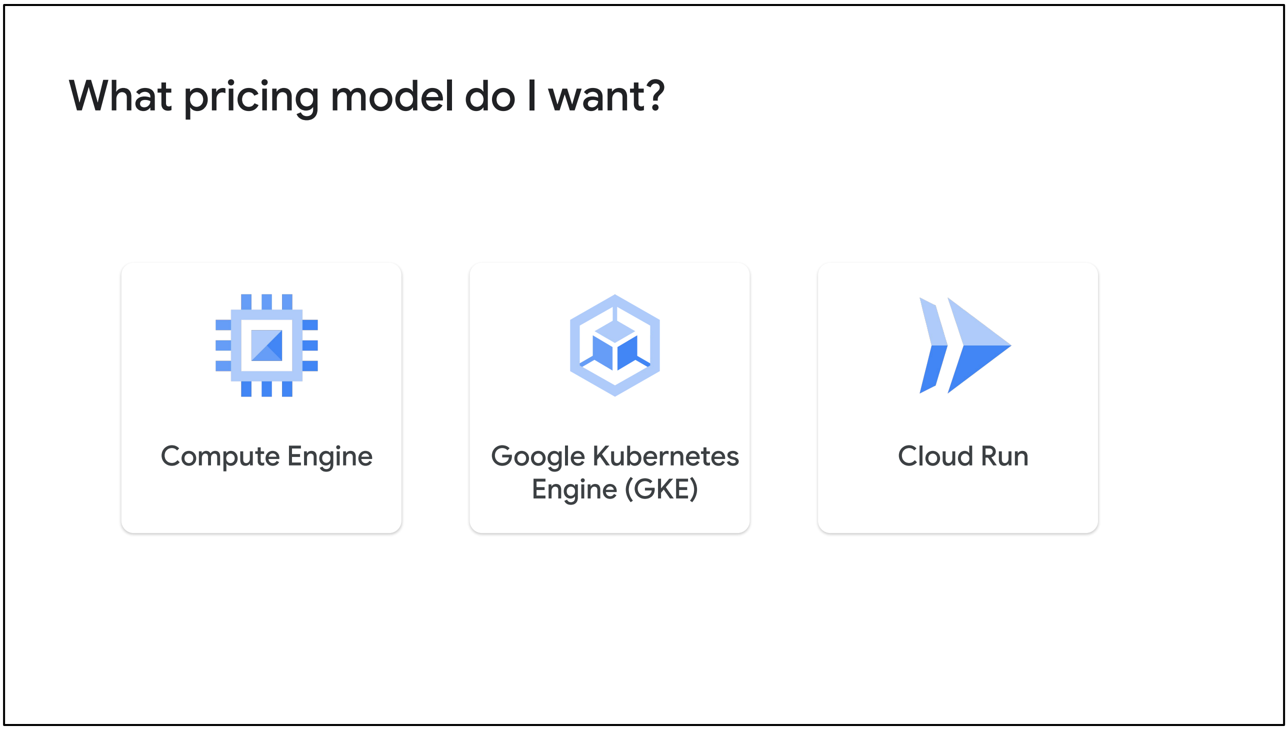
Google Kubernetes Engine (GKE)

Cloud Run

Dev + Operations

Dev focused

If you're modernizing your applications over time, you might need to manage Compute Engine VMs that have been migrated from on-premises data centers. Your operations team must be able to manage the health and security of these virtual machines.



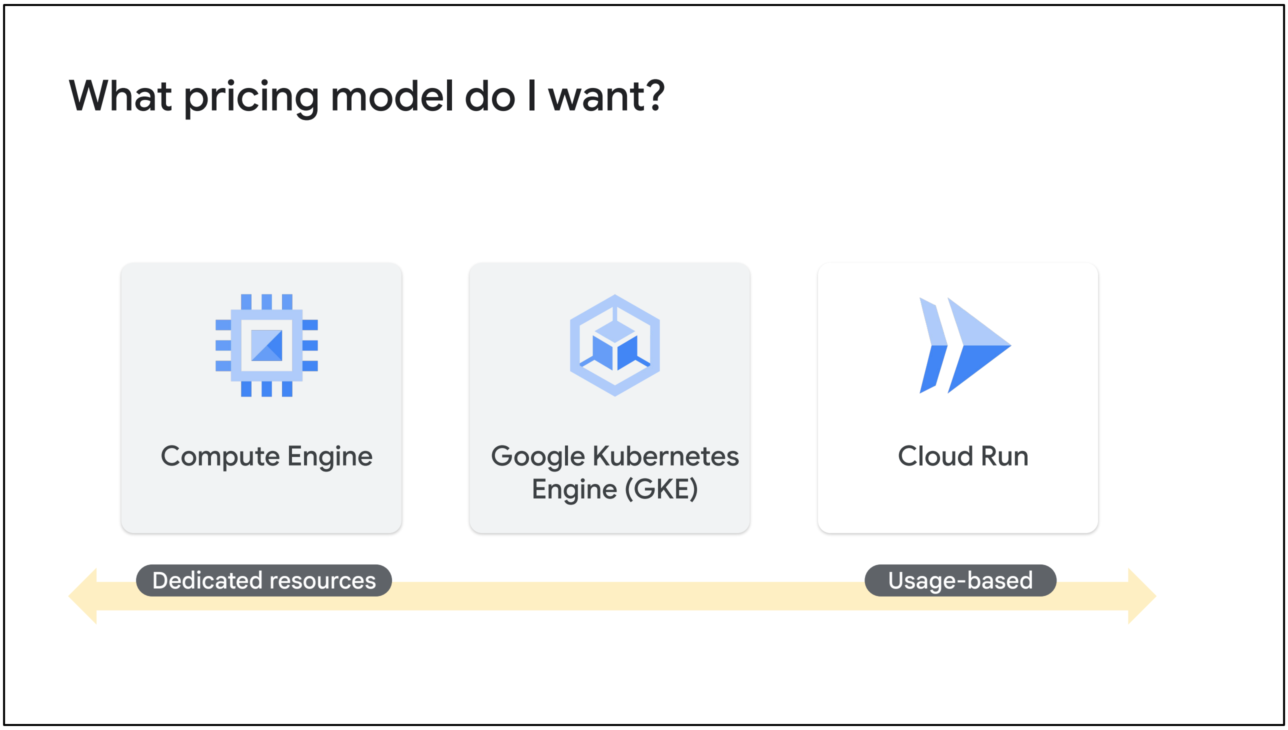
What pricing model do I want?

Compute Engine

Google Kubernetes Engine (GKE)

Cloud Run

Pricing for the platforms is different, which might affect your choice as well.



What pricing model do I want?

Compute Engine

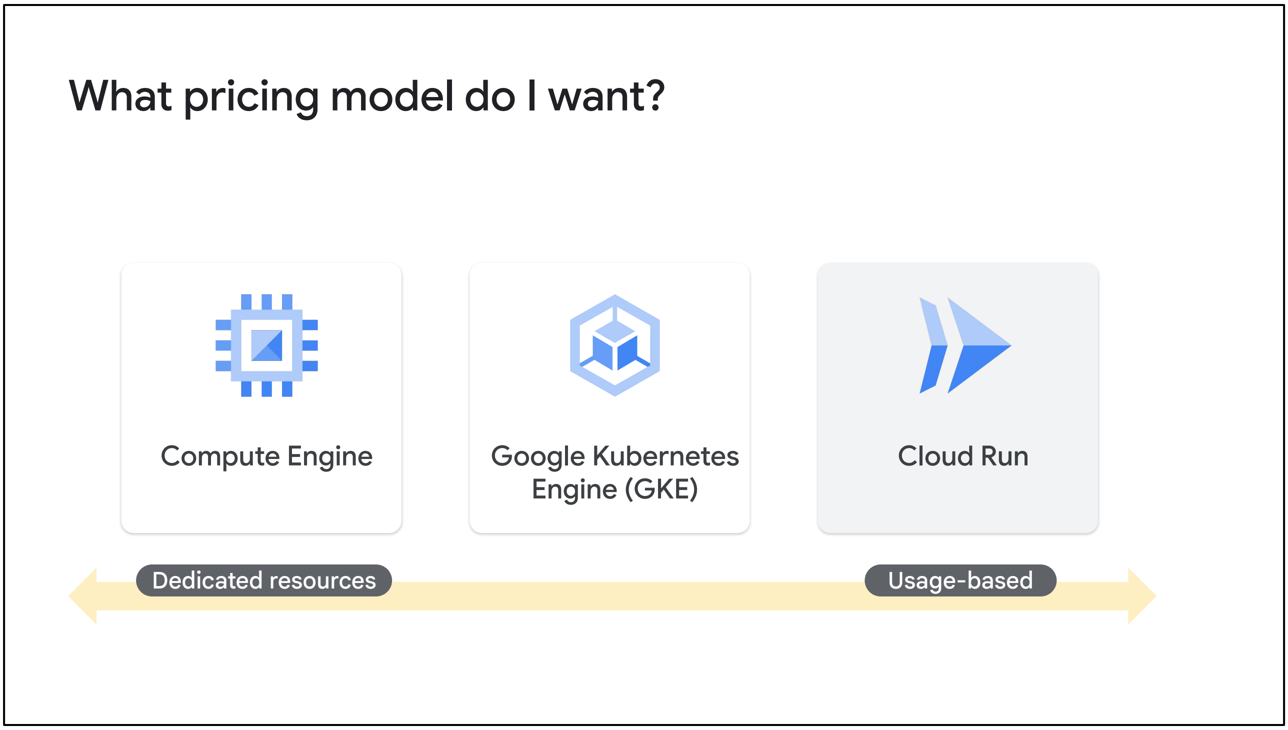
Google Kubernetes Engine (GKE)

Cloud Run

Dedicated resources

Usage-based

Compute Engine and Google Kubernetes Engine charges are based on the usage of dedicated VMs. Charges are predictable, and these platforms can be ideal when you require consistent capacity for your applications.



What pricing model do I want?

Compute Engine

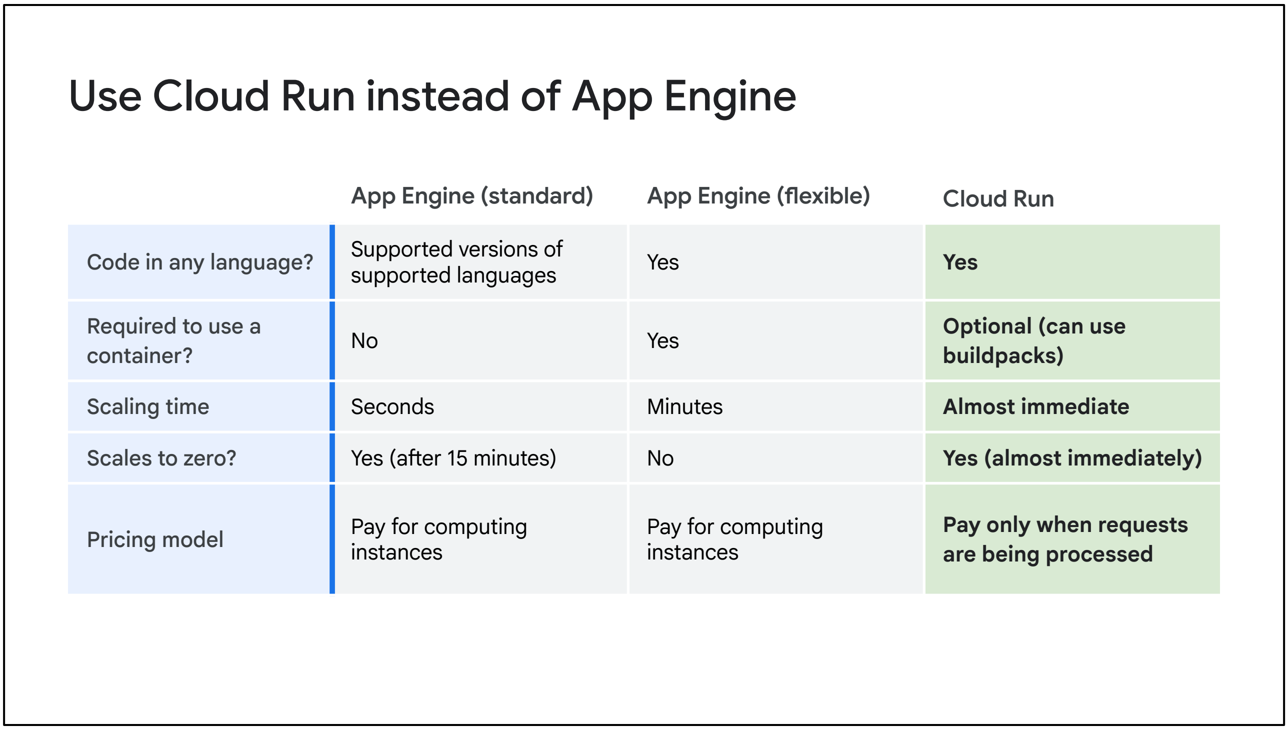
Google Kubernetes Engine (GKE)

Cloud Run

Dedicated resources

Usage-based

Cloud Run services and functions are pay per use, which can result in significant savings, especially when your traffic patterns are inconsistent.



Use Cloud Run instead of App Engine

	App Engine (standard)	App Engine (flexible)	Cloud Run
Code in any language?	Supported versions of supported languages	Yes	**Yes**
Required to use a container?	No	Yes	**Optional (can use buildpacks)**
Scaling time	Seconds	Minutes	**Almost immediate**
Scales to zero?	Yes (after 15 minutes)	No	**Yes (almost immediately)**
Pricing model	Pay for computing instances	Pay for computing instances	**Pay only when requests are being processed**

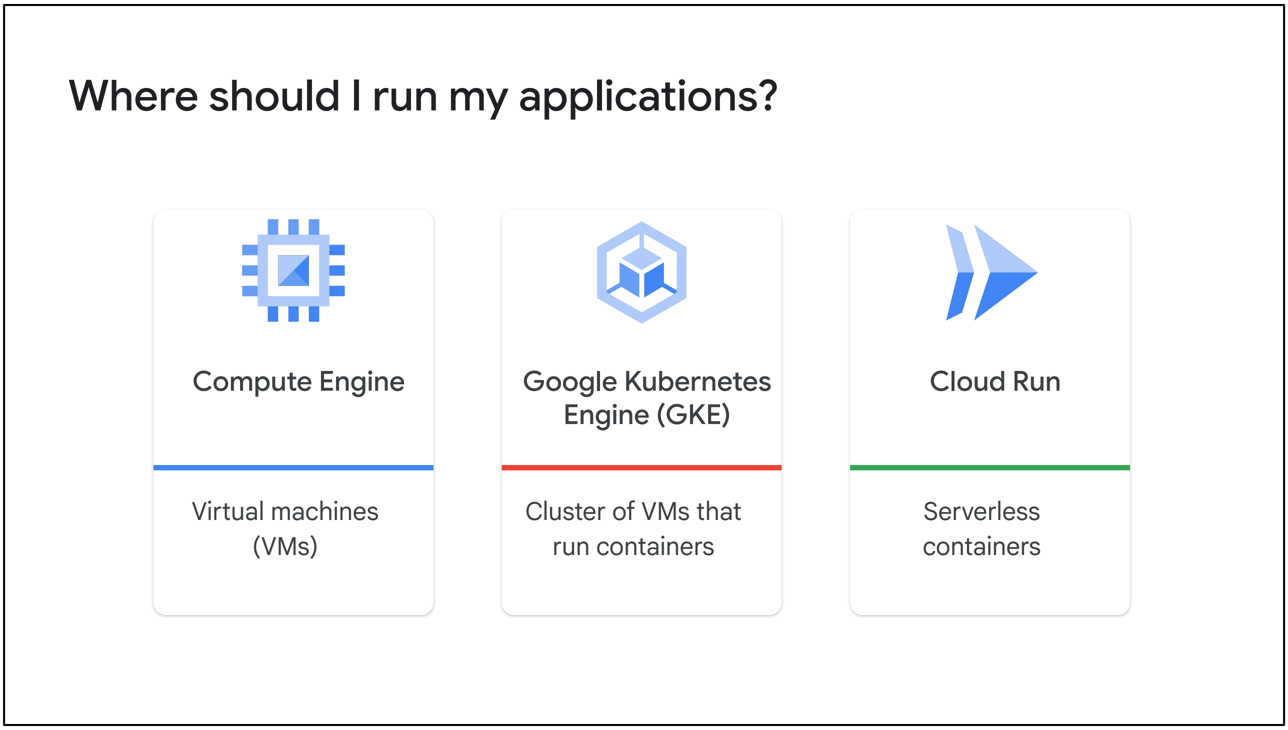
App Engine is a fully managed serverless compute environment. App Engine supports two environments, standard and flexible.

Cloud Run is the latest evolution of Google Cloud serverless, building on the experience of running App Engine for more than a decade. Cloud Run services can handle the same workloads as App Engine services, but Cloud Run offers customers much more flexibility in implementing these services. This flexibility, along with improved integrations with Google Cloud and third-party services, also enables Cloud Run to handle workloads that cannot run on App Engine.

Unlike App Engine, Cloud Run can scale up and down almost immediately in response to traffic spikes. And, by default, you pay for Cloud Run services only when requests are being processed.

If you are creating a new service, you should use Cloud Run instead of App Engine.

[Compare App Engine and Cloud Run:
https://cloud.google.com/appengine/migration-center/run/compare-gae-with-run]



Where should I run my applications?

Compute Engine

Virtual machines (VMs)

Google Kubernetes Engine (GKE)

Cluster of VMs that run containers

Cloud Run

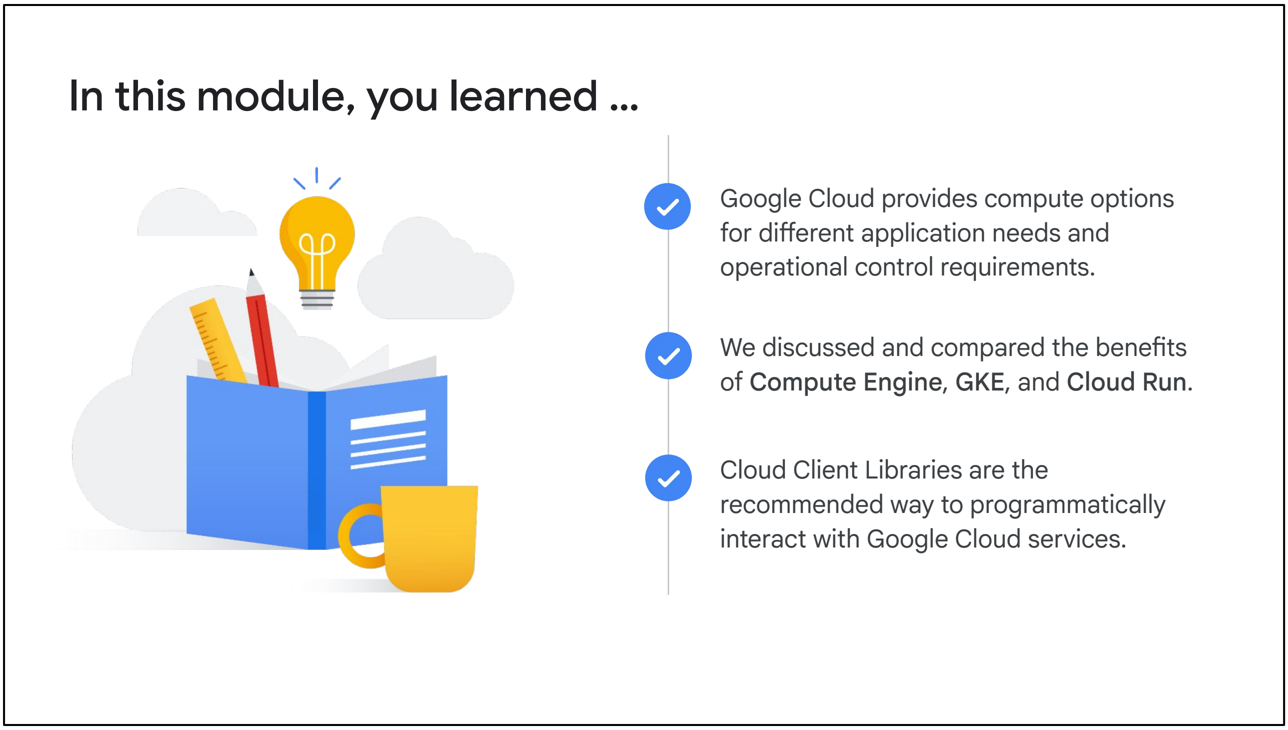
Serverless containers

Returning to the question, "Where should I run my applications?", the best answer is that you should run each workload on the platform that best fits its requirements. You don't need to standardize on a single platform, even within a single application. Larger applications might benefit from using multiple platforms that allow them to solve each problem with the correct tool.

In general, as you move toward the left, you have more control over your application and infrastructure, but managing that infrastructure requires more operational cost and effort.

Most applications written with the Cloud Client Libraries can be easily moved from platform to platform, so you can change your decision later. If you do not have complex infrastructure requirements, start with a serverless platform that lets you focus on the application instead of the infrastructure. If you later want more control over the infrastructure, you can move your application to a platform that requires more operational effort but provides the needed control or flexibility.

[Application Hosting Options: https://cloud.google.com/hosting-options]



In this module, you learned …

- ✓ Google Cloud provides compute options for different application needs and operational control requirements.
- ✓ We discussed and compared the benefits of **Compute Engine**, **GKE**, and **Cloud Run**.
- ✓ Cloud Client Libraries are the recommended way to programmatically interact with Google Cloud services.

Google Cloud offers a range of compute options depending on the needs of your application and the level of operational control that you require.

We compared the benefits of **Compute Engine**, **GKE**, and **Cloud Run**.

Cloud Client Libraries are the recommended way to programmatically interact with Google Cloud services. With this approach, you can move your application to a different platform when your needs change.