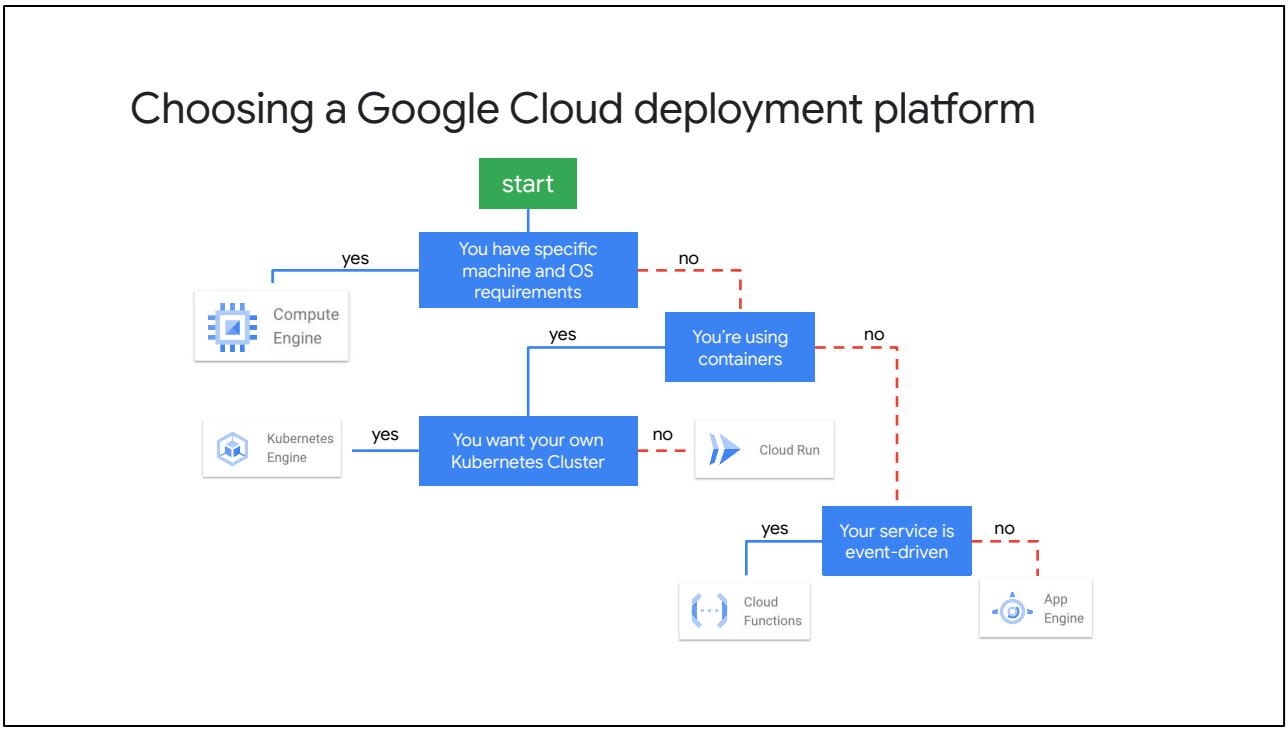
Deploying Applications to Google Cloud

Stephanie Wong
Developer Advocate, Google Cloud

In this module, we discuss the different options of deploying applications to Google Cloud. Google Cloud offers many possible deployment platforms, and the choice is not always immediately obvious.



Let me give you a high-level overview of how you could decide on the most suitable platform for your application.

First, ask yourself whether you have specific machine and OS requirements. If you do, then Compute Engine is the platform of choice.

If you have no specific machine or operating system requirements then the next question to ask is whether you are using containers. If you are, then you should consider Google Kubernetes Engine or Cloud Run, depending on whether you want to configure your own Kubernetes cluster.

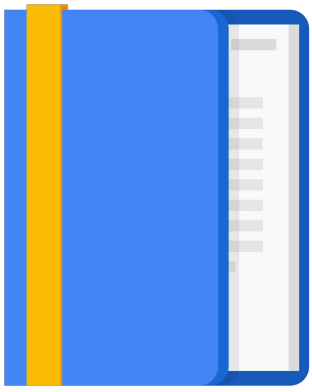
If you are not using containers, then you want to consider Cloud Functions if your service is event-driven, and App Engine if it's not.

We'll talk through each of these services in this module and you will get to explore them in a lab.

Let's get started!

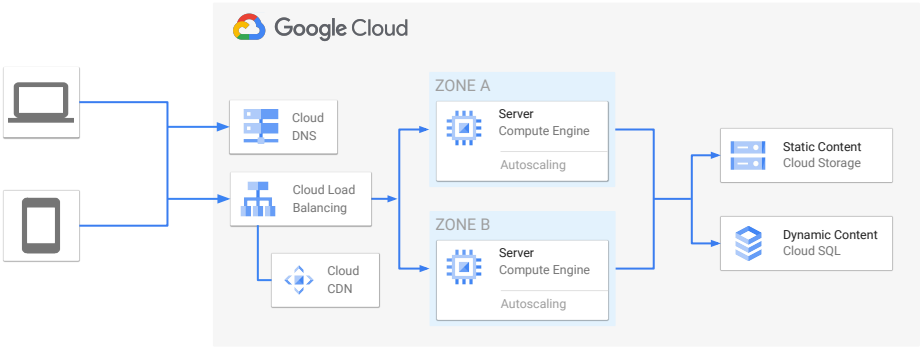
Agenda

- **Google Cloud Infrastructure as a Service**
- Google Cloud Deployment Platforms



Let’s begin by talking about Compute Engine, which is Google Cloud’s infrastructure as a service (or IaaS) offering.

Use Compute Engine when you need complete control over operating systems, for apps that are not containerized or self-hosted databases



Compute Engine is a great solution when you need complete control over your operating systems, or if you have an application that is not containerized, an application built on a microservice architecture, or an application that is a database.

Instance groups and autoscaling as shown on this slide allow you to meet variations in demand on your application. Let's take a closer look at instance groups.

Managed instance groups create VMs based on instance templates

- Instance templates define the VMs: image, machine type, etc.
  - Test to find the smallest machine type that will run your program.
- Use a Startup Script to install your program from a Git repo.

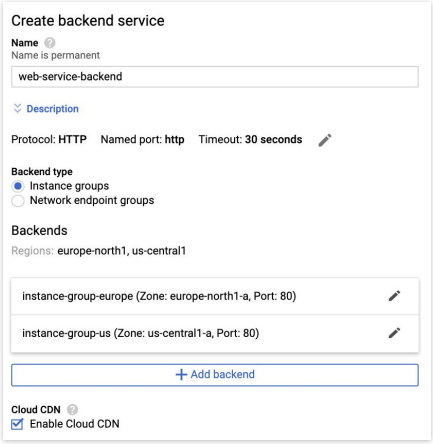
- Instance group manager creates the machines.
- Set up auto scaling to optimize cost and meet varying user workloads.
- Add a health check to enable auto healing.
- Use multiple zones for high availability.

Managed instance groups create VMs based on instance templates. Instance templates are just a resource used to define VMs and managed instance groups. The templates define the boot disk image or container image to be used, the machine type, labels, and other instance properties like a startup script to install software from a Git repository.

The virtual machines in a managed instance group are created by an instance group manager. Using a managed instance group offers many advantages, such as autohealing to re-create instances that don't respond and creating instances in multiple zones for high availability.

Use one or more instance groups as the backend for load balancers

- Use a global load balancer if you have instance groups in multiple regions.
- Enable the CDN to cache static content.
- For external services, set up SSL.
- For internal services, don't provide a public IP address.

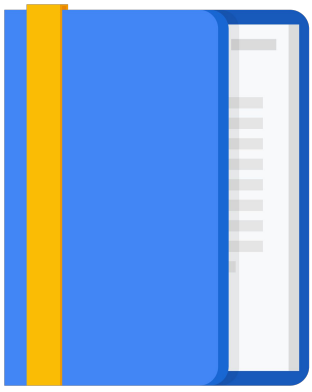


I recommend using one or more instance groups as the backend for load balancers. If you need instance groups in multiple regions, use a global load balancer, and if you have static content, simply enable Cloud CDN as shown on the right.

Agenda

Google Cloud Infrastructure as a Service

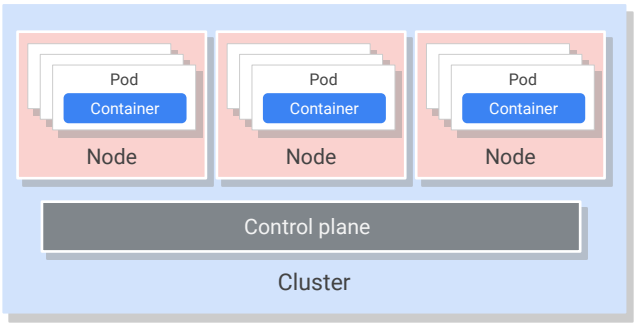
Google Cloud Deployment Platforms



Let’s go through the other deployment platforms: GKE, Cloud Run, App Engine, and Cloud Functions.

Google Kubernetes Engine (GKE) automates the creation and management of compute infrastructure

- Kubernetes clusters have a collection of nodes.
- In GKE, nodes are Compute Engine VMs.
- Services are deployed into pods.
- Optimize resource utilization by deploying multiple services to the same cluster.
- You pay for the VMs.

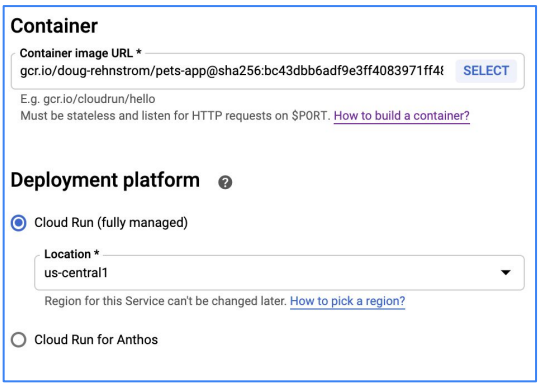


Google Kubernetes Engine, or GKE, provides a managed environment for deploying, managing, and scaling containerized applications using Google infrastructure. The GKE environment consists of multiple Compute Engine virtual machines grouped together to form a cluster. GKE clusters are powered by the Kubernetes open source cluster management system. Kubernetes provides the mechanisms with which to interact with the cluster. Kubernetes commands and resources are used to deploy and manage applications, perform administration tasks and set policies, and monitor the health of deployed workloads.

This diagram on the right shows the layout of a Kubernetes cluster. A cluster consists of at least one cluster control plane and multiple worker machines that are called nodes. These control plane and node machines run the Kubernetes cluster orchestration system. Pods are the smallest, most basic deployable objects in Kubernetes. A pod represents a single instance of a running process in a cluster. Pods contain one or more containers, such as Docker containers, that run the services being deployed. You can optimize resource use by deploying multiple services to the same cluster.

Cloud Run allows you to deploy containers to Google managed Kubernetes clusters

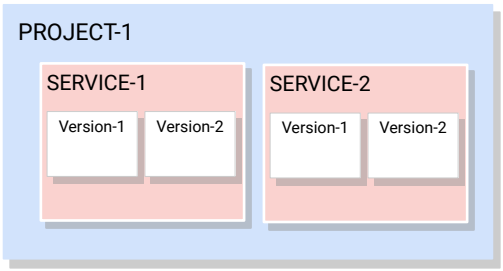
- Cloud Run allows you to use Kubernetes without the cluster management or configuration code.
- Apps must be stateless.
- Need to deploy apps using Docker images in Container Registry.
- Can also use Cloud Run to automate deployment to your own GKE cluster.



Cloud Run, on the other hand, allows you to deploy containers to a Google-managed Kubernetes cluster. A big advantage is that you don't need to manage or configure the cluster. The services that you deploy must be stateless, and the images you use must be in Container Registry. Cloud Run can be used to automate deployment to Anthos GKE clusters. You should do this if you need more control over your services, because it will allow you to access your VPC network, tune the size of compute instances, and run your services in all GKE regions. The screenshot on the right shows a Cloud Run configuration where the container image URL is specified along with the deployment platform, which can be fully managed Cloud Run or Cloud Run for Anthos.

App Engine was designed for microservices

- Each Google Cloud project can contain 1 App Engine application.
- An application has 1 or more services.
- Each service has 1 or more versions.
- Versions have 1 or more instances.
- Automatic traffic splitting for switching versions.

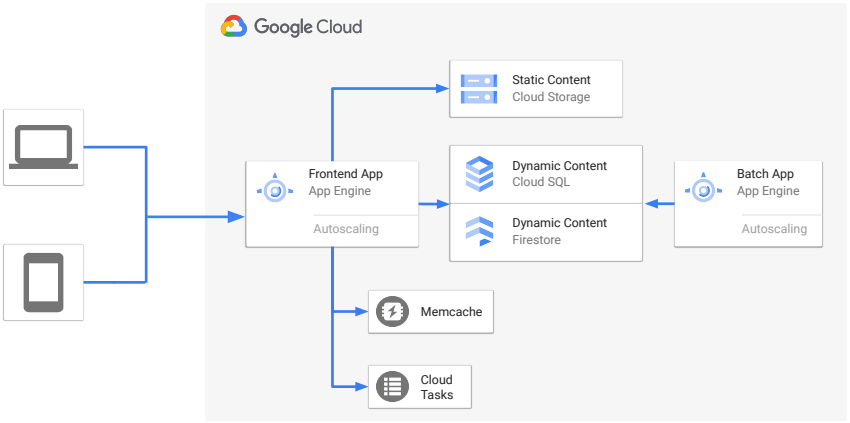


App Engine is a fully managed, serverless application platform supporting the building and deploying of applications. Applications can be scaled seamlessly from zero upward without having to worry about managing the underlying infrastructure. App Engine was designed for microservices. For configuration, each Google Cloud project can contain one App Engine application, and an application has one or more services. Each service can have one or more versions, and each version has one or more instances. App Engine supports traffic splitting so it makes switching between versions and strategies such as canary testing or A/B testing simple.

The diagram on the right shows the high-level organization of a Google Cloud project with two services, and each service has two versions. These services are independently deployable and versioned.

Let me show you a typical App Engine microservice architecture.

Typical App Engine microservice architecture

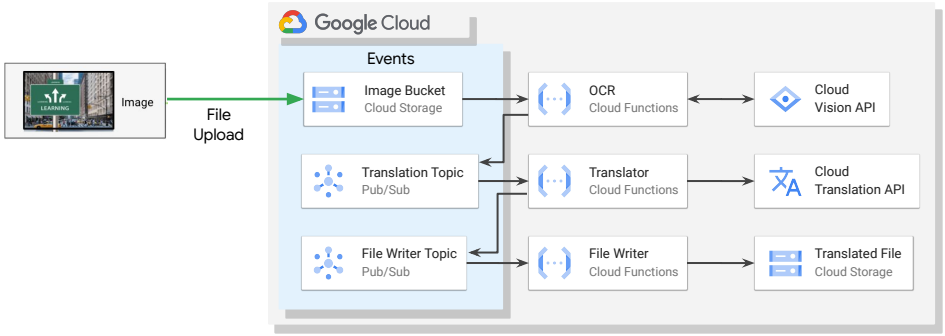


This could be an example of a retailer that sells online. Here App Engine serves as the frontend for both web and mobile clients. The backend of this application is a variety of Google Cloud storage solutions with static content such as images stored in Cloud Storage, Cloud SQL used for structured relational data such as customer data and sales data, and Firestore used for NoSQL storage such as product data. Firestore has the benefit of being able to synchronize with client applications.

Memcache is used to reduce the load on the datastores by caching queries, and Cloud Tasks are used to perform work asynchronously outside a user request (or service-service request). There's also a batch application that generates data reports for management.

Cloud Functions is great way to create loosely coupled, event-driven microservices

- Can be triggered by changes in a storage bucket, Pub/Sub messages, or web requests
- Completely managed, scalable, and inexpensive



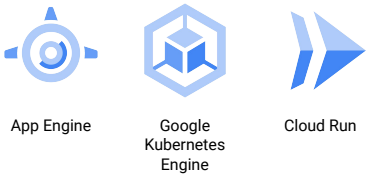
Cloud Functions are a great way to deploy loosely coupled, event-driven microservices. They have been designed for processing events that occur in Google Cloud. The functions can be triggered by changes in a Cloud Storage bucket, a Pub/Sub message, or HTTP requests. The platform is completely managed, scalable, and inexpensive. You do not pay if there are no requests, and processing is paid for by execution time in 100ms increments.

This graphic illustrates an image translation service implemented with Cloud Functions. When an image is uploaded to a Cloud Storage bucket, it triggers an OCR Cloud Function that identifies the text in the image using Google's Cloud Vision API. Once the text has been identified, this service then publishes a message to a Pub/Sub topic for translation, which triggers another Cloud Function that will translate the identified text in the image using the Cloud Translation API. After that, the translator Cloud Function will publish a message to a file write topic in Pub/Sub, which triggers a Cloud Function that will write the translated image to a file.

This sequence illustrates a typical use case of Cloud Functions for event-based processing.

Lab

Deploying Apps to Google Cloud

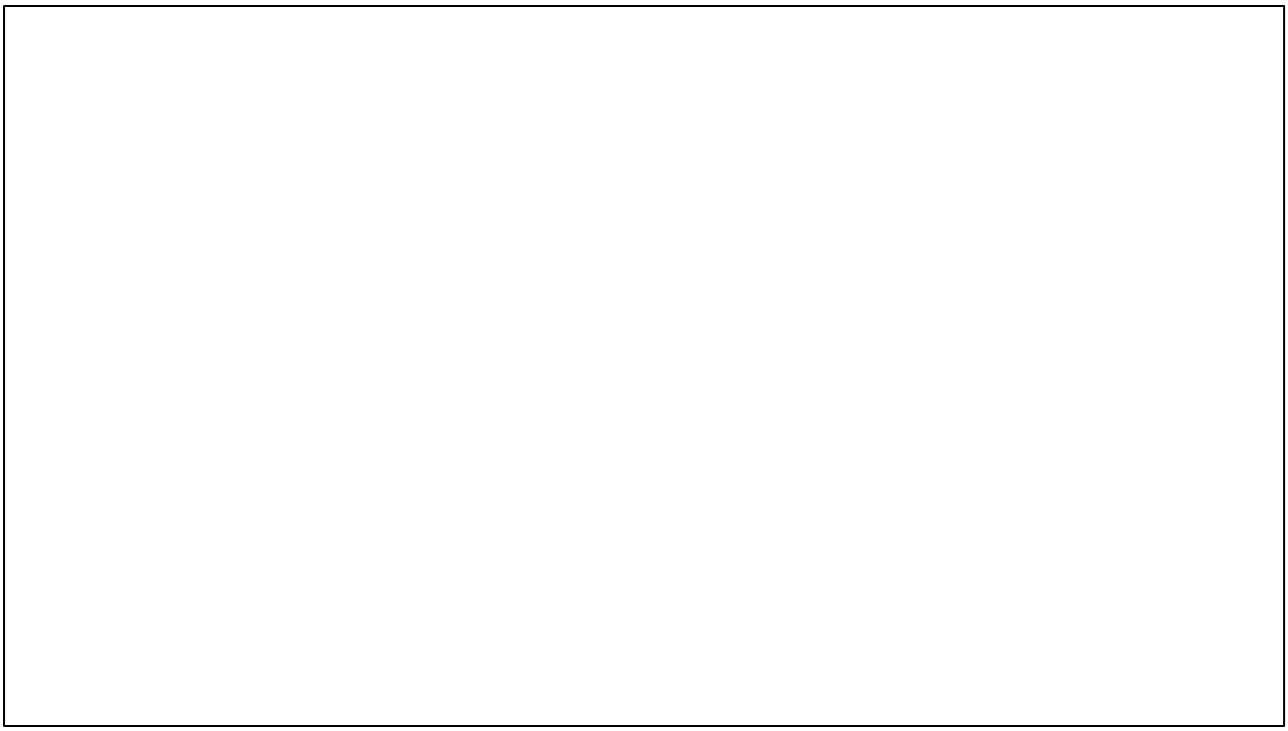


Objectives

- Deploy to App Engine
- Deploy to Google Kubernetes Engine
- Deploy to Cloud Run

In the first lab of this course, you used Cloud Build to create Docker images and store those images in Container Registry. If you follow the 12-factor best practices when writing your applications, you should be able to create applications that are portable across different cloud providers and also portable between different deployment services provided by the cloud. That's why in this lab, you deploy code to App Engine, Google Kubernetes Engine, and Cloud Run. Let me quickly go over why we chose those services for this lab:

Deploying your Dockerized application to a virtual machine using Compute Engine in the first lab was easy but that might not be the most effective option. A more automated way might be using App Engine, which will be the first compute platform of this lab. App Engine is great for those who just want to focus on their code and not worry at all about the underlying infrastructure like networks, load balancers, and autoscalers which are completely managed by App Engine.



Now, sometimes developers want more freedom to customize their environments. Google Kubernetes Engine, or GKE, provides a balance where you have a lot of customization over your environment similar to Compute Engine. However, GKE also helps you optimize your spend by allowing you to deploy multiple services into the same cluster of virtual machines. This provides an excellent balance between flexibility, portability, and cost optimization.

Kubernetes can get pretty complicated though. That's where Cloud Run comes in. Cloud Run allows you to deploy your own stateless, Dockerized services onto Kubernetes clusters that are managed by Google. Google Cloud takes care of the hard parts of managing the cluster and configuring the load balancers, autoscalers, and health checkers, so that you just focus on the code.

Deploying a single application on all of these compute platforms might help you choose the right platform for your own services.

Lab review

Deploying Apps to Google Cloud

In this lab you saw how to deploy an application to App Engine, GKE, and Cloud Run. Hopefully, this lab gave you a better feel for how each of these compute platforms works and will help you choose the right platform for your own services.

I recommend that you follow the 12-factor best practices that we covered earlier in the course, automate as much as possible by using continuous integration and infrastructure as code tools, and containerize your services using Docker. If you do that, your apps will be very portable, and deployment will become easier and more flexible, no matter which compute platform you use.

You can stay for a lab walkthrough, but remember that Google Cloud's user interface can change, so your environment might look slightly different.

Review

Deploying Applications to Google Cloud

In this module we covered the various deployment services provided by Google. These include Compute Engine if you need complete control over your deployment environment; Google Kubernetes Engine if you want the flexibility, portability and automation that is provided by Kubernetes; and App Engine and Cloud Run if you want a completely managed platform as a service.

Again, each of these choices has advantages and disadvantages. Make sure you understand each one so that you can make an informed decision when deploying your services.