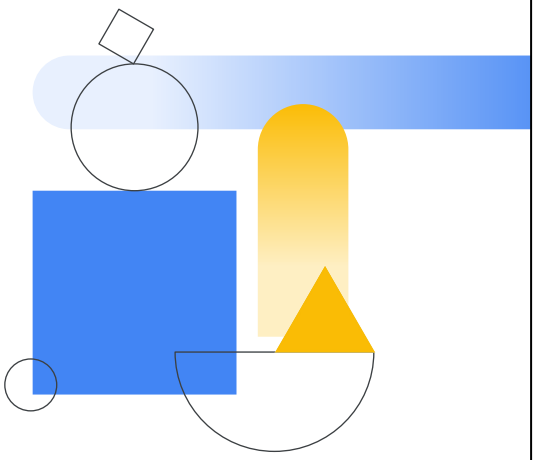
Google Cloud

Deploying Applications to Google Cloud



Learning objectives

01 Choose the appropriate Google Cloud deployment service for your applications.

02 Configure scalable, resilient infrastructure using Instance Groups.

03 Orchestrate microservice deployments using Kubernetes, GKE and Cloud Run.

04 Leverage App Engine for a completely automated platform as a service (PaaS).

05 Create serverless applications using Cloud Functions.

Google Cloud offers several kinds of compute resources for deployment. Each offers different levels of control and features. In this module, we will discuss, compare, and contrast the following five services:

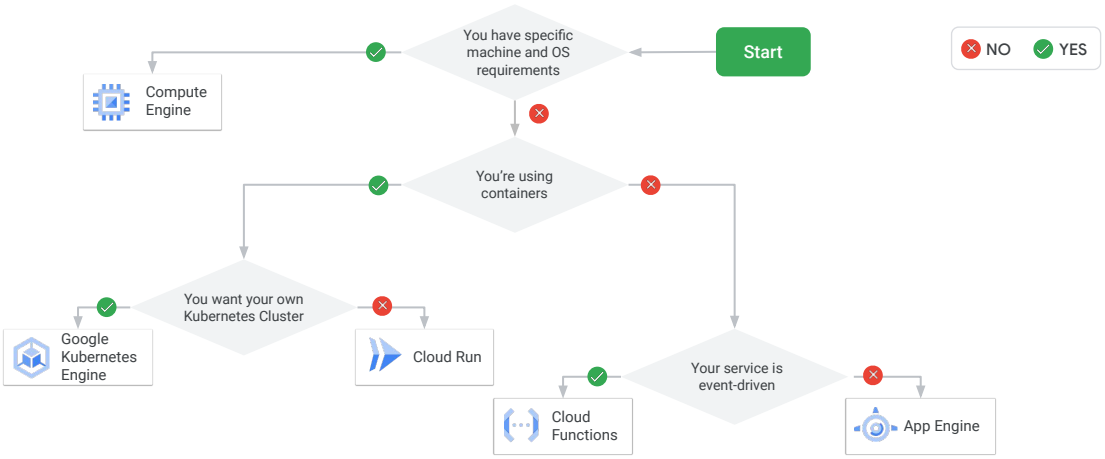
- Compute Engine
- App Engine
- Google Kubernetes Engine
- Cloud Functions
- Cloud Run

Agenda

01	Google Cloud Infrastructure as a Service
02	Google Cloud Deployment Platforms
03	Lab
04	Quiz
05	Review

Google Cloud Infrastructure as a Service

Choosing a Google Cloud deployment platform



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

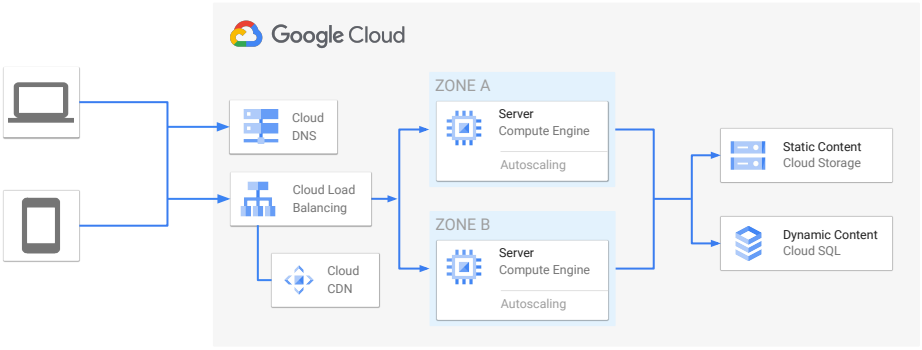
First, ask yourself whether you have specific machine and OS requirements. If you do, then Compute Engine is the platform of choice.

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.

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

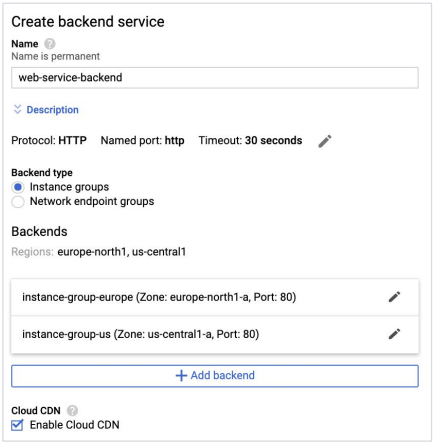
- Instance templates define the VMs: image, machine type, etc.
  - Test to find the smallest machine type that will run your program.
- Use a Startup Script to install your program from a Git repo.
- Instance group manager creates the machines.
- Set up auto scaling to optimize cost and meet varying 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.



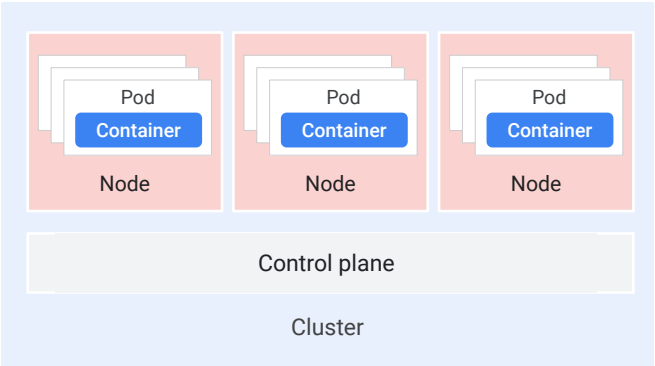
I recommend using one or more instance groups as the backend for load balancers. If you need instance groups in multiple regions, use a global load balancer, and if you have static content, simply enable Cloud CDN as shown on the right.

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.



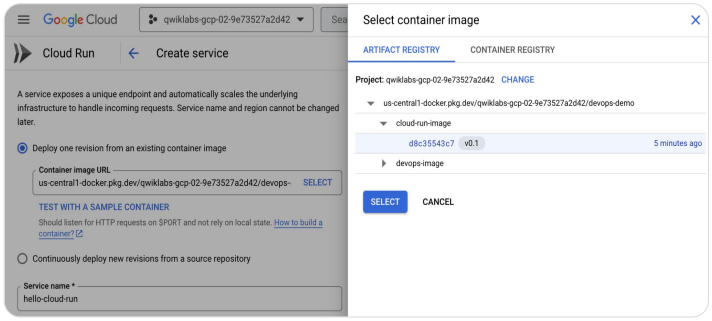
Google Kubernetes Engine, or GKE, provides a managed environment for deploying, managing, and scaling containerized applications using Google infrastructure. The GKE environment consists of multiple Compute Engine virtual machines grouped together to form a cluster. GKE clusters are powered by the Kubernetes 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.

GKE has the Autopilot and Standard modes of operation, which offer you different levels of flexibility, responsibility, and control. Google recommend the fully-managed Autopilot mode, in which Google Cloud manages your nodes for you and provides a workload-focused, cost-optimized, production-ready experience.

You can deploy a container image stored in Artifact Registry to Cloud Run

- Cloud Run can deploy a container image stored in Artifact registry.
- Cloud Run can deploy directly from source.
- A container can be deployed using the Google Cloud console or gcloud command line.



You can deploy a container image stored in Artifact Registry to Cloud Run. You can also deploy directly from source to Cloud Run, which includes automatically creating a container image for your built source and storing the image in Artifact Registry.

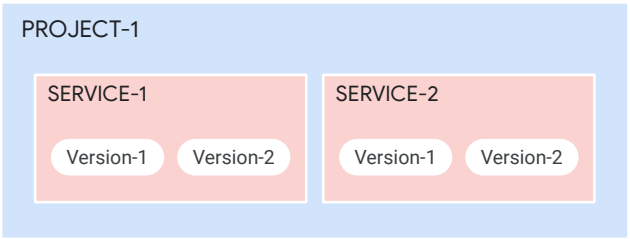
When deploying from source code, Cloud Run can automatically, containerize local source code, push the container image to an Artifact Registry repository, deploy the container image Cloud Run from the repository, and push and pull images using the repository cloud-run-source-deploy, in the region that you specify at deploy time. If the repository does not exist, Cloud Run creates it for you if your account has the required permissions.

You can also deploy a container image by tag or digest that is stored in Artifact Registry. Deploying to a service for the first time creates its first revision. Note that revisions are immutable. If you deploy from a container image tag, it will be resolved to a digest and the revision will always serve this particular digest.

You can deploy a container using the Google Cloud console or the gcloud command line.

App Engine was designed for microservices

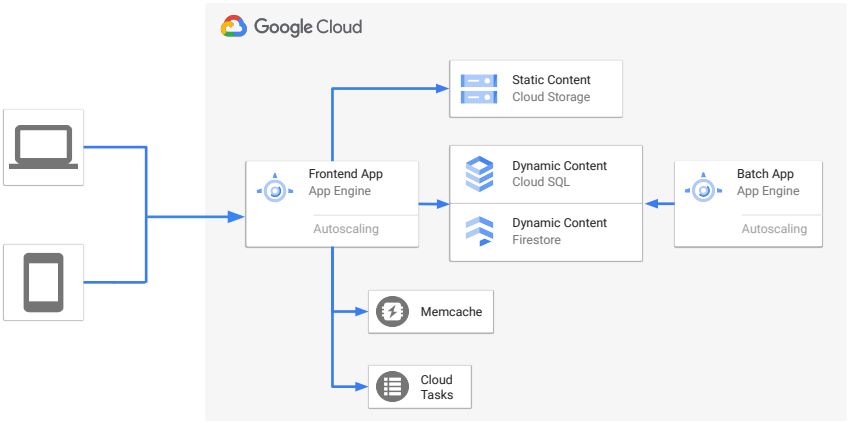
- Each Google Cloud project can contain 1 App Engine application.
- An application has 1 or more services.
- Each service has 1 or more versions.
- Versions have 1 or more instances.
- Automatic traffic splitting for switching versions.



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

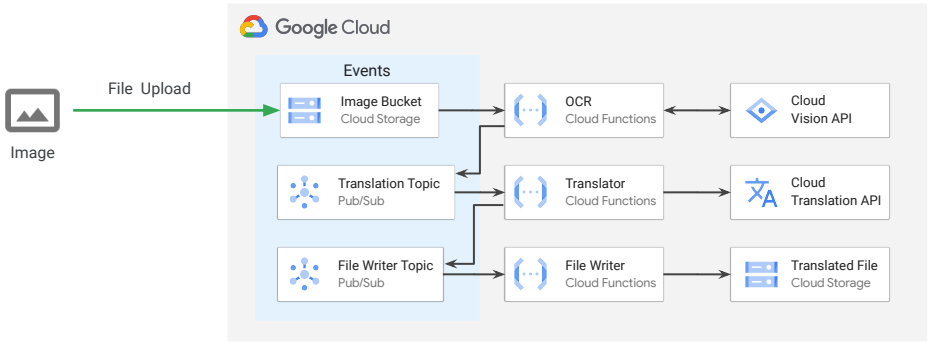


This could be an example of a retailer that sells online. Here App Engine serves as the frontend for both web and mobile clients. The backend of this application is a variety of Google Cloud storage solutions with static content such as images stored in Cloud Storage, Cloud SQL used for structured relational 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



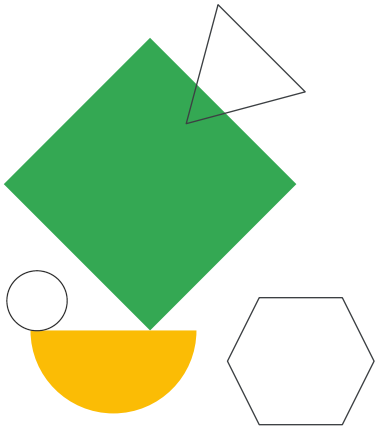
Cloud Functions are a great way to deploy loosely coupled, event-driven microservices. They have been designed for processing events that occur in Google Cloud. The functions can be triggered by changes in a Cloud Storage bucket, a Pub/Sub message, or HTTP requests. The platform is completely managed, 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 Intro

Deploying Apps to Google Cloud

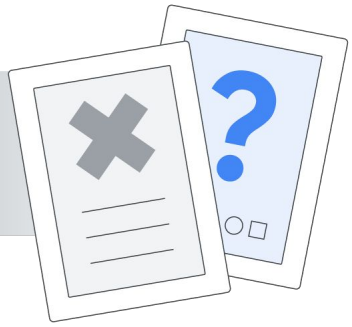


Lab objectives

01 Deploy to App Engine.

02 Deploy to Google Kubernetes Engine.

03 Deploy to Cloud Run.



Quiz

Question #1

Question

You need to deploy an existing application that was written in .NET version 4. The application requires Windows servers, and you don't want to change it. Which should you use?

A. Compute Engine

B. GKE

C. App Engine

D. Cloud Functions

You need to deploy an existing application that was written in .NET version 4. The application requires Windows servers, and you don't want to change it. Which should you use?

A. Compute Engine
B. GKE
C. App Engine
D. Cloud Functions

Question #1

Answer

You need to deploy an existing application that was written in .NET version 4. The application requires Windows servers, and you don't want to change it. Which should you use?

A. Compute Engine 

B. GKE

C. App Engine

D. Cloud Functions

A. This is the correct answer. The approach is a lift and shift which is best supported by Compute Engine, because Compute Engine offers full control over virtual machines including operating system. No repackaging would be required.

B. This answer is not correct. GKE would require repackaging into Docker containers.

C. This answer is not correct. App Engine standard environment does not support .NET, and using App Engine flexible environment would require repackaging the application.

D. This answer is not correct. Cloud Functions is the wrong model; it is for deploying single purpose functions.

Question #2

Question

You have containerized multiple applications using Docker and have deployed them using Compute Engine VMs. You want to save costs and simplify container management. What might you do?

A. Write Terraform scripts for all deployment.

B. Rewrite the applications to run in App Engine standard environment.

C. Rewrite the applications to run in Cloud Functions.

D. Migrate the containers to GKE.

You have containerized multiple applications using Docker and have deployed them using Compute Engine VMs. You want to save costs and simplify container management. What might you do?

A. Write Terraform scripts for all deployment.

B. Rewrite the applications to run in App Engine standard environment.

C. Rewrite the applications to run in Cloud Functions.

D. Migrate the containers to GKE.

Question #2

Answer

You have containerized multiple applications using Docker and have deployed them using Compute Engine VMs. You want to save costs and simplify container management. What might you do?

A. Write Terraform scripts for all deployment.

B. Rewrite the applications to run in App Engine standard environment.

C. Rewrite the applications to run in Cloud Functions.

D. Migrate the containers to GKE.

A. This answer is not correct. While this could be achieved with Terraform, the requirement to save costs and simplify container management would not be met.

B. If the applications are containerized, then rewriting to run in App Engine standard environment is not cost-effective.

C. This answer is not correct. Cloud Functions are for deploying single purpose functions, not applications.

D. This is the correct answer. The applications are containerized, and GKE will help with the resource efficiency and hence costs, automate many aspects of the container management, and provide the best solution for the scenario.

Question #3

Question

You've been asked to write a program that uses Vision API to check for inappropriate content in photos that are uploaded to a Cloud Storage bucket. Any photos that are inappropriate should be deleted. What might be the simplest, cheapest way to deploy this program?

A. Compute Engine

B. GKE

C. Cloud Functions

D. App Engine

You've been asked to write a program that uses Vision API to check for inappropriate content in photos that are uploaded to a Cloud Storage bucket. Any photos that are inappropriate should be deleted. What might be the simplest, cheapest way to deploy this program?

A. Compute Engine
B. GKE
C. Cloud Functions
D. App Engine

Question #3

Answer

You've been asked to write a program that uses Vision API to check for inappropriate content in photos that are uploaded to a Cloud Storage bucket. Any photos that are inappropriate should be deleted. What might be the simplest, cheapest way to deploy this program?

A. Compute Engine

B. GKE

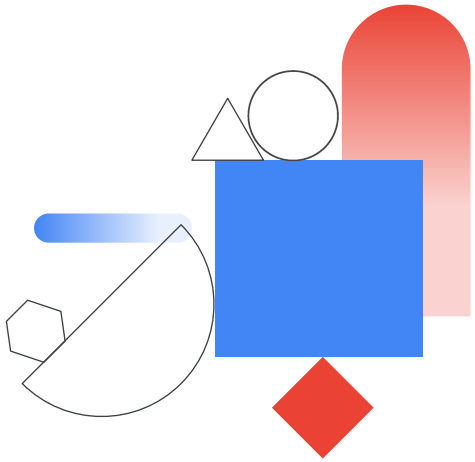
C. **Cloud Functions** 

D. App Engine

C. This is the correct answer. The requirements for simplest and cheapest are met with Cloud Functions. Cloud Functions are for single purpose functions like image analysis. Cloud Functions also can be triggered by Cloud Storage events, so they provide seamless integration. The payment model based on number of requests, processing time of requests (measured in 100ms units), and then other resources consumed is the most suitable of all options offered above. There is a free tier too. Cloud Functions also provides automatic scaling, high availability, and fault tolerance.

Answers A, B, D could all be solutions but would require more development work and more expense for resources used, and as a result, do not meet the requirement of simplest, cheapest way of achieving the required functionality.

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.

More resources

Migration to Google Cloud:
Deploying your workloads

https://cloud.google.com/solutions/migration-to-gcp-deploying-your-workloads

Compute Engine

https://cloud.google.com/compute/

GKE

https://cloud.google.com/kubernetes-engine/

App Engine

https://cloud.google.com/appengine/

The links provide access to some useful resources on Google Cloud deployment platforms.


Google Cloud