

Architecture: Compute








- **Compute Engine**
- **Kubernetes Engine**

Serverless:

- **App Engine**
- **Cloud Functions**
- **Cloud Run**

Architecture: Compute Resources

	 Compute Engine	 Kubernetes Engine	 Cloud Functions	 App Engine	 Cloud Run
Features	<p>Infrastructure as a Service offering for stateful applications</p> <ul style="list-style-type: none">➤ Operating system-level control➤ Supports custom machine types➤ Autoscaling Support	<p>Secured and managed Kubernetes service</p> <ul style="list-style-type: none">➤ Kubernetes applications➤ Pod and cluster autoscaling➤ workload and network security	<p>Serverless execution environment for building and connecting cloud services</p> <ul style="list-style-type: none">➤ Supports Node.js (Node.js 10 or 12), Python 3 (Python 3.7 or 3.8), Go (Go 1.11 or 1.13) or Java (Java 11)➤ Supports Google Cloud Client Libraries	<p>Serverless PaaS offering to run applications without managing infrastructure</p> <ul style="list-style-type: none">➤ Standard Environment supports Python, Java, Node.js, PHP, Ruby and Go➤ Flexible environments supports dockerised containers	<p>Managed compute platform to run stateless containers that are invocable via web requests or Pub/Sub events.</p> <ul style="list-style-type: none">➤ Serverless➤ Built on Knative➤ Fully managed or Cloud Run for Anthos
Use Cases	<ul style="list-style-type: none">➤ On-premises and monolithic workloads➤ Raw compute to meet existing infrastructure requirements <p>Examples:</p> <ul style="list-style-type: none">▪ Relational databases, SAP HANA▪ CRM systems▪ Legacy ERP systems	<ul style="list-style-type: none">➤ Microservices architecture➤ Hybrid and multi-cloud support <p>Examples:</p> <ul style="list-style-type: none">▪ Containerised API deployment▪ cloud native apps	<ul style="list-style-type: none">➤ Event driven and data processing apps➤ Manipulate user generated data and events <p>Examples:</p> <ul style="list-style-type: none">▪ Post a comment on Slack channel following a GitHub commit▪ Statistical analysis▪ Image thumbnail generation	<ul style="list-style-type: none">➤ HTTP services and backend apps➤ Web frameworks, microservices <p>Examples:</p> <ul style="list-style-type: none">▪ Flask▪ Django▪ Express.js▪ Symfony▪ Spring Boot	<ul style="list-style-type: none">➤ Container-based apps and services➤ Industry standard packaging for multi-cloud infrastructure <p>Examples:</p> <ul style="list-style-type: none">▪ Custom runtime environments such as Rust, Kotlin, C++, and Bash▪ Legacy web apps using languages such as Python 2.7, Java 7▪ Containerized apps that need custom hardware and software (OS, GPUs)

Compute: Compute Engine

Pricing

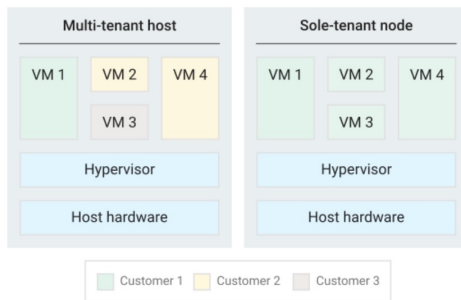
- Per-second billing
- Commitment savings
- Sustained-use savings

Networking

- Regional HTTPS load balancing
- Network Load Balancing
- Global and multi-regional subnetworks
- Inbound/outbound firewall rules

Sole-tenancy

A sole-tenant node is a physical Compute Engine server that is dedicated to hosting only your project's VMs



Compute

Predefined and Custom machine types

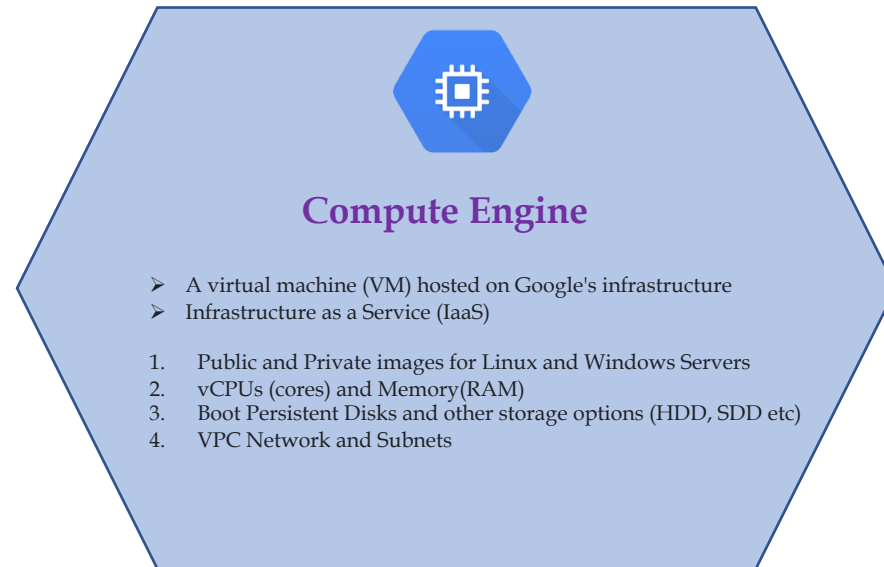
- General purpose (Standard, High-memory, High-CPU)
- Memory-Optimized
- Compute-Optimized

Tau VMs – Optimised for scale-out workloads

- Web Servers, Containerised microservices, media transcoding

Cloud GPUs – For machine learning, scientific computing and 3D visualisation

- Machine Learning, Medical Analysis, Video transcoding, Graphic Visualisation



Live migration for VMs

- live migration to keep VMs running even when a host system event, such as a software or hardware update, occurs.
- Compute Engine live migrates running instances to another host in the same zone instead of requiring VMs to be rebooted.

Storage

- Zonal persistent disk
- Regional persistent disk
- Local SSD
- Cloud Storage buckets
- Filestore

Preemptible VM instances

Much lower price than normal instances but VMs might stop (preempt) based on resource availability.

- ideal for fault-tolerant applications, batch processing tasks etc
- Always stops after they run for 24 hours
- 30 seconds preemption signal

Shielded VM

Shielded VM offers verifiable integrity of VM instances
- Secure Boot, virtual trusted platform module (vTPM) enabled Measured Boot and integrity monitoring

Confidential VMs

- Confidential VMs allows data encryption in use – while it's being processed.
- It doesn't compromise on performance
- simple, easy-to-use deployment

VM access

Linux:

- SSH access – GCP Console, third party client terminal
- CloudShell via Cloud SDK
- Firewall rule tcp:22 allow

Windows:

- RDP access – tcp:3389 allow
- Powershell terminal, third party client

Instance Templates

Instance templates define the machine type, boot disk image or container image, labels, and other instance properties. It's used to create Managed Instance Group or to create individual VMs.





Compute: Compute Engine -> Machine Types

A machine type is a set of virtualized hardware resources available to a virtual machine (VM) instance, including the system memory size, virtual CPU (vCPU) count, and persistent disk limits.

1. General purpose (Standard, High-memory, High-CPU)
2. Memory-Optimized
3. Compute-Optimized
4. Accelerator-optimised

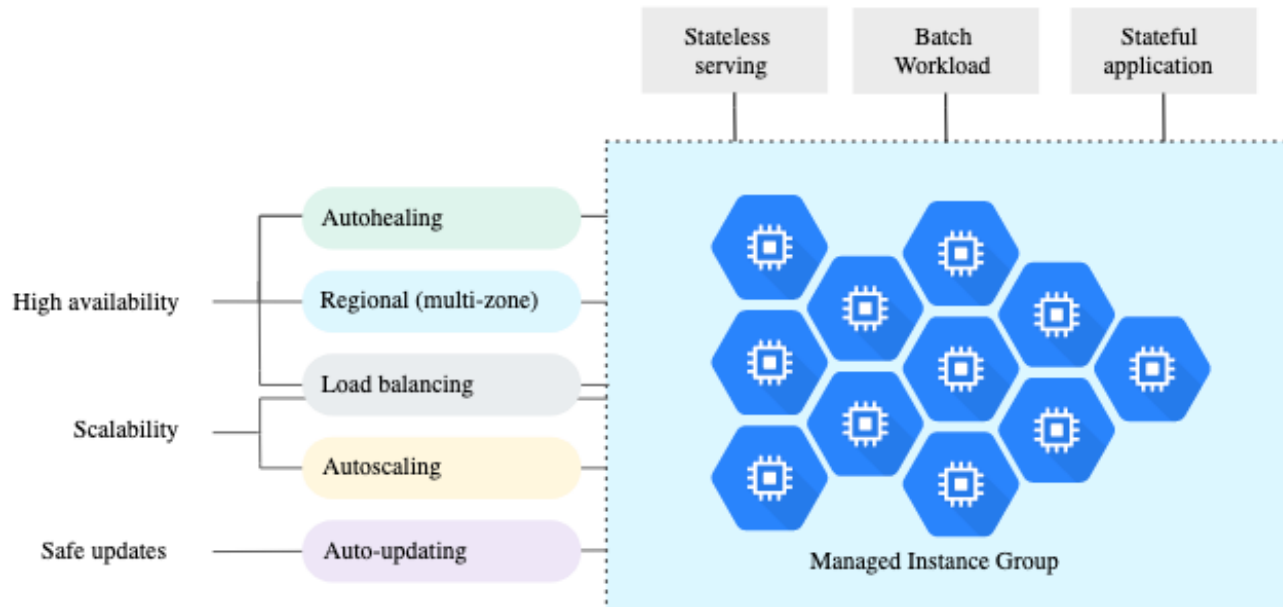
General purpose		Workload optimized		
Cost-optimized	Balanced	Memory-optimized	Compute-optimized	Accelerator-optimized
E2	N2, N2D, N1	M2, M1	C2	A2
Day-to-day computing at a lower cost	Balanced price/performance across a wide range of VM shapes	Ultra high-memory workloads	Ultra high performance for compute-intensive workloads	Optimized for high performance computing workloads
<ul style="list-style-type: none">• Web serving• App serving• Back office apps• Small-medium databases• Microservices• Virtual desktops• Development environments	<ul style="list-style-type: none">• Web serving• App serving• Back office apps• Medium-large databases• Cache• Media/streaming	<ul style="list-style-type: none">• Medium-large in-memory databases such as SAP HANA• In-memory databases and in-memory analytics• Microsoft SQL Server and similar databases	<ul style="list-style-type: none">• Compute-bound workloads• High-performance web serving• Gaming (AAA game servers)• Ad serving• High-performance computing (HPC)• Media transcoding• AI/ML	<ul style="list-style-type: none">• CUDA(Compute Unified Device Architecture)-enabled ML training and inference• HPC (High Performance Computing)• Massive parallelized computation



Compute: Compute Engine -> Instance Groups

An instance group is a collection of virtual machine (VM) instances that can be managed as a single entity. Compute Engine offers two kinds of VM instance groups, managed and unmanaged:

- Managed instance groups (MIGs)
- Unmanaged instance groups

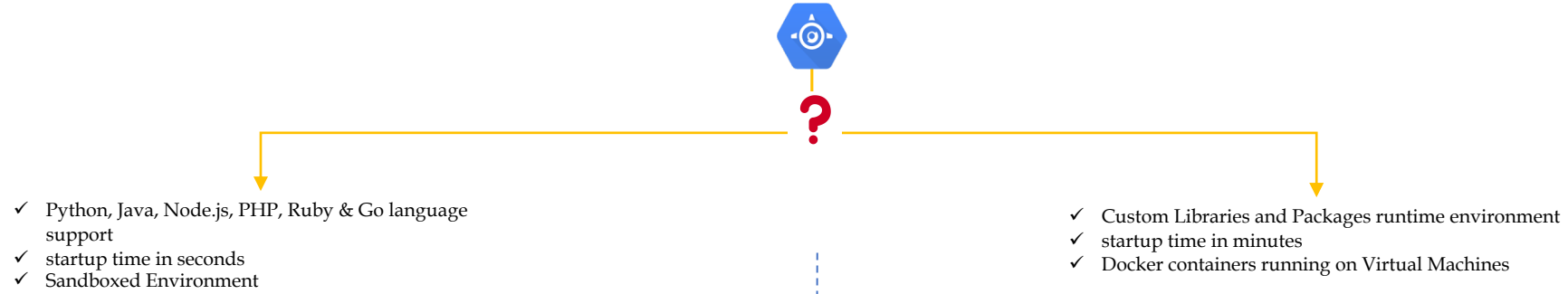


Managed instance groups (MIGs) are suitable for scenarios like these:

- Stateless serving workloads, such as a website frontend
- Stateless batch, high-performance, or high-throughput compute workloads, such as image processing from a queue
- Stateful applications, such as databases, legacy applications, and long-running batch computations with checkpointing



App Engine is a fully managed, serverless platform for developing and hosting web applications at scale. You can choose from several popular languages, libraries, and frameworks to develop your apps, then let App Engine take care of provisioning servers and scaling app instances based on demand.



The App Engine Standard Environment

The App Engine standard environment is based on container instances running on Google's infrastructure. Containers are preconfigured with one of several available runtimes.

- Python
- Java
- Node.js
- PHP
- Ruby
- Go

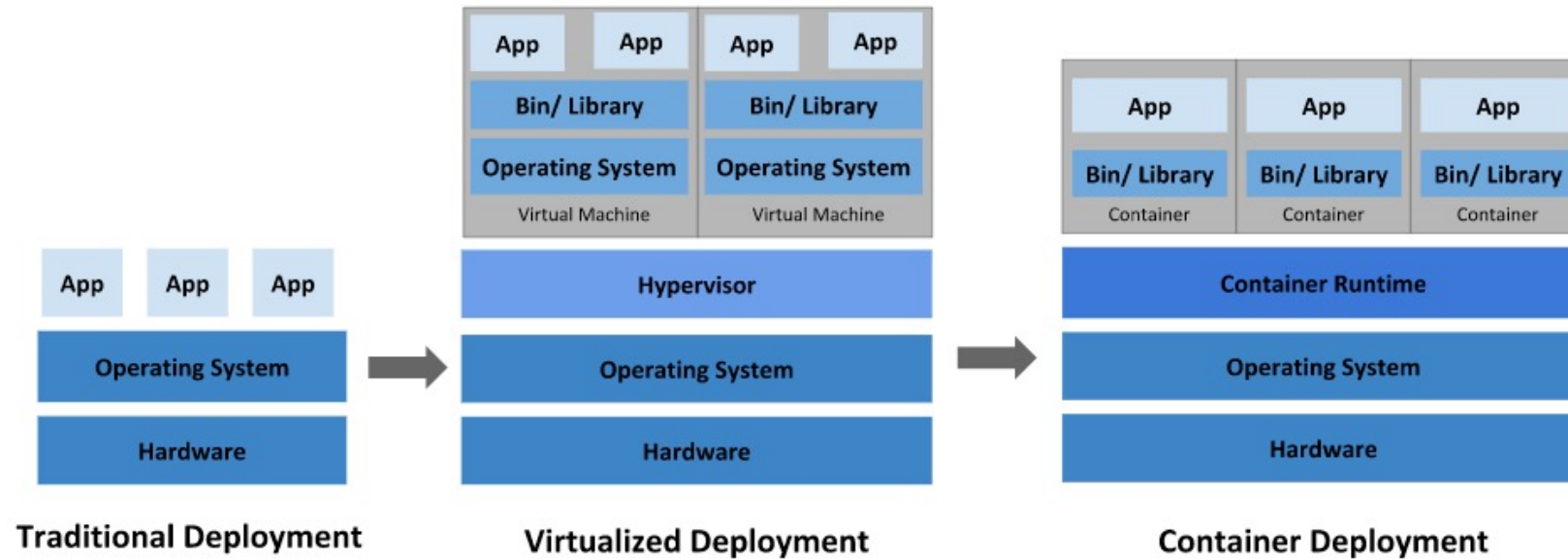
The App Engine standard environment gives 1 GB of data storage and traffic for free, which can be increased by enabling paid applications.

App Engine Flexible Environment

Flexible environment allows developers to customize the supported runtimes or provide own runtime by supplying a custom Docker image or Dockerfile from the open-source community.

Features:

- Customizable infrastructure
- Performance options
- Native feature support
- Managed virtual machines



Why Containers?

- Agile application creation and deployment
- Loosely coupled, distributed, elastic, liberated micro-services
- Continuous development, integration, and deployment
- Environmental consistency across development, testing, and production
- Cloud and OS distribution portability
- Application-centric management
- Resource utilization: high efficiency and density.

Why Kubernetes:

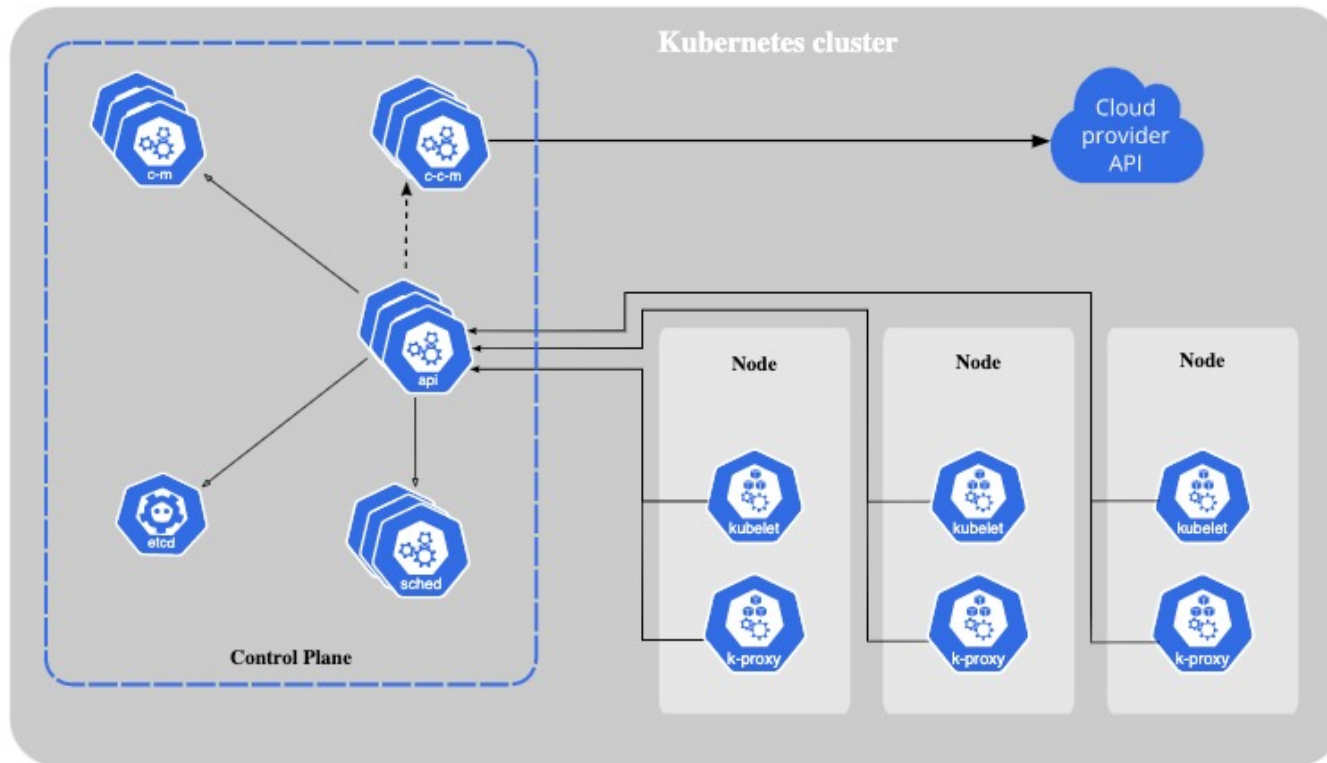
- **Service discovery and load balancing**
- **Storage orchestration**
- **Automated rollouts and rollbacks**
- **Automatic bin packing**
- **Self-healing**
- **Secret and configuration management**



Kubernetes: Architecture & Components

A Kubernetes cluster consists of a set of worker machines, called [nodes](#), that run containerized applications. The worker node(s) host the [Pods](#) that are the components of the application workload. The [control plane](#) manages the worker nodes and the Pods in the cluster.










Open Source Kubernetes architecture



Kubernetes provides several built-in workload resources:

- [Deployment](#) and [ReplicaSet](#).
- [StatefulSet](#).
- [DaemonSet](#)
- [Job](#) and [CronJob](#)

Kubernetes Components:

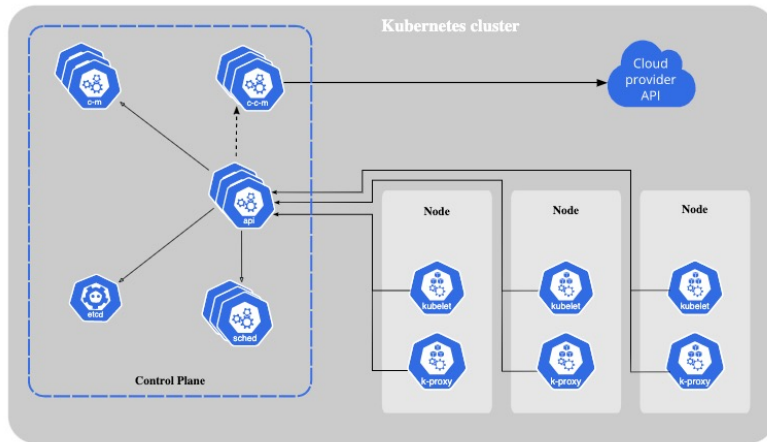
API server		Front end and inter cluster communicator for the Kubernetes Control Plane. Designed to scale horizontally.
Cloud controller manager (optional)		A control plane component that embeds cloud-specific control logic (Node, Route, Service Controllers)
Controller manager		Abstract Components to reduce complexity (Node, Replica, Endpoints, Service Account & Token Controllers)
etcd (persistence store)		Distributed key-value store used to store state information
kubelet		Runs on every node to make sure container is running.
kube-proxy		A Network proxy to maintain network rules on pods.
Scheduler		Low level computer abstractions to run pods.
Control plane		
Node		Node components run on every node, maintaining running pods and providing the Kubernetes runtime environment. <ul style="list-style-type: none">• Kubelet – Runs on every node to make sure container is running.• Kube-proxy – A Network proxy to maintain network rules on pods.



Compute: Google Kubernetes Engine(GKE)

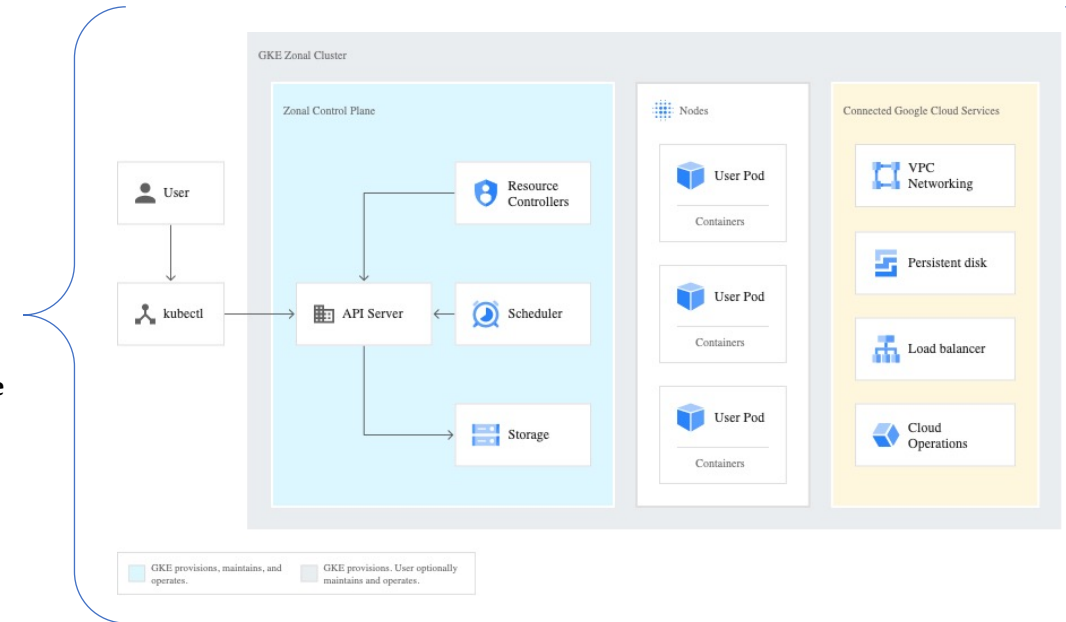
Google Kubernetes Engine (GKE) provides a managed environment for Kubernetes using Google infrastructure.

Open Source Kubernetes architecture



Kubernetes Engine

GKE Cluster architecture



Kubernetes on Google Cloud

- Google Cloud's load-balancing for Compute Engine instances
- Node pools to designate subsets of nodes within a cluster for additional flexibility
- Automatic scaling of cluster's node instance count
- Automatic upgrades for cluster's node software
- Node auto-repair to maintain node health and availability
- Container native networking and security using GKE Sandbox
- Logging and monitoring with Google Cloud's operations suite for visibility into your cluster

Modes of operation

GKE clusters have two modes of operation to choose from:

- Autopilot
- Standard



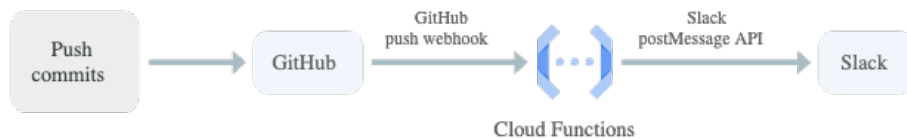
Compute: Cloud Functions

Cloud Functions is a scalable pay-as-you-go functions as a service (FaaS) to run code with zero server management.

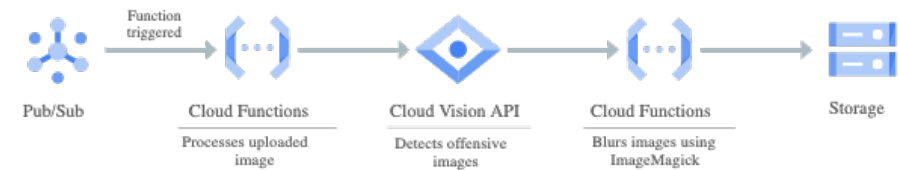
- ✓ With Cloud Functions you can write simple, single-purpose functions that are attached to events emitted from cloud infrastructure and services.
- ✓ Cloud Functions removes the work of managing servers, configuring software, updating frameworks, and patching operating systems.
- ✓ Key networking capabilities for hybrid and multi-cloud scenarios

Use cases

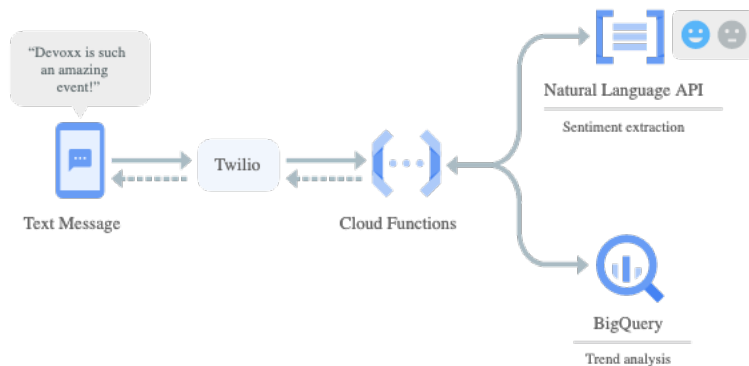
- Integration with third-party services and APIs



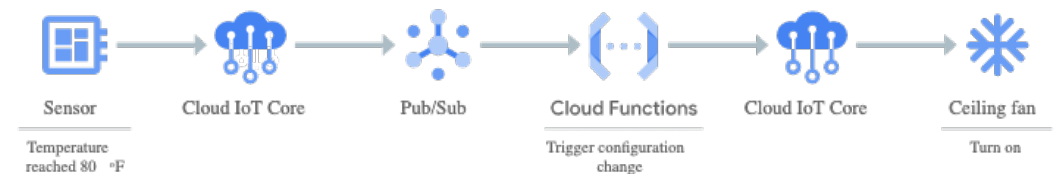
- Real-time stream processing



- Sentiment analysis



- Serverless IoT back ends



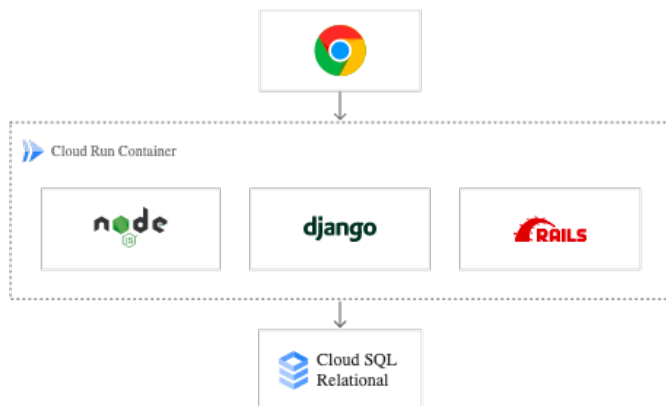


Compute: Cloud Run

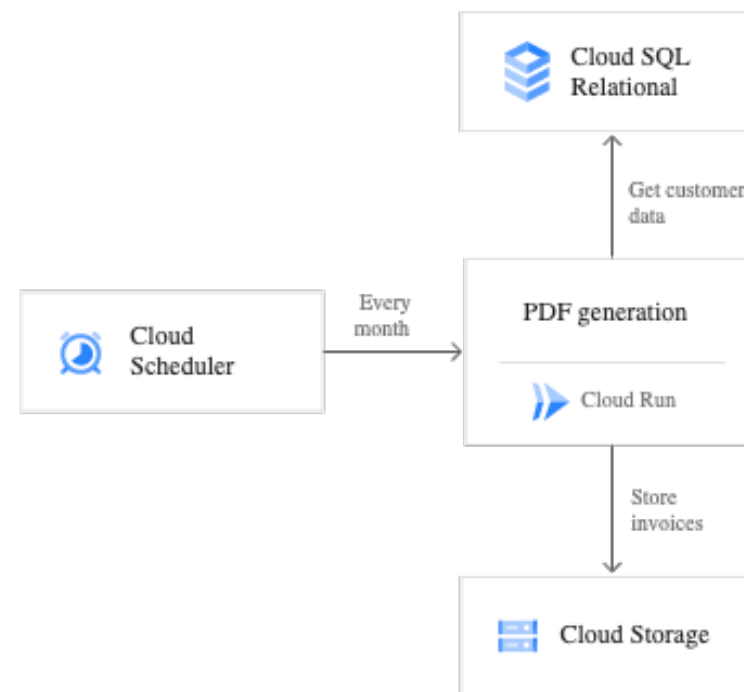
Cloud Run is a managed compute platform that enables you to run stateless containers that are invocable via web requests or Pub/Sub events. Cloud Run is serverless and is built upon the container and Knative open standards, enabling portability of applications. Two Options available – Cloud Run (Fully managed) & Cloud Run for Anthos

Use cases:

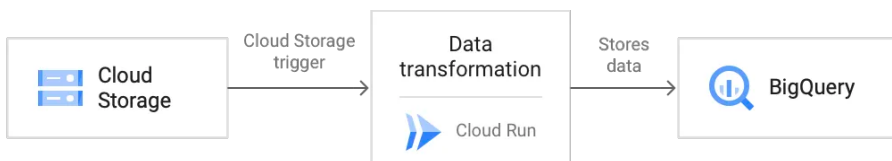
- Web services: Websites



- Automation: Scheduled document generation



- Data processing: Lightweight data transformation



Compute: Compute Engine Hosting Options

Many workloads have specific technical requirements. Platforms are ordered by degree of abstraction.

	Compute Engine	Kubernetes Engine (GKE)	Cloud Run for Anthos on Google Cloud	Cloud Run (fully managed)	App Engine Flexible Environment	App Engine Standard Environment	Cloud Functions
Deployment format	VM image	Cluster	Container	Container	App or Container	App	Function
Custom URLs	✓	✓	✓	✓	✓	✓	✗
Scale-to-zero	✗	✗	~ ¹	✓	✗	✓	✓
Free tier	✓	✗	✗	✓	✗	✓	✓
Persistent disks	✓	✓ ²	✗	✗	✗	✗	✗
Websockets	✓	✓	~ ³	✗	✓ ^β	✗	✗
Run any language	✓	✓	✓	✓	✓	✗	✗
Request timeout	None	None	15 minutes	15 minutes	60 minutes	1 minute	9 minutes
Background processes	✓	✓	✗	✗	✓	~ ⁴	✗
TPU/GPU access	✓	✓	~ ³	✗	✗	✗	✗
VPC connectivity	✓	✓	✓	✓	✓	✓	✓

¹Cloud Run for Anthos on Google Cloud scales pod counts to zero. The node count per cluster cannot scale to zero, and these nodes are billed during no-request periods.

²Container instances on GKE do not persist data upon shutdown. However, Compute Engine persistent disks can be [mounted to container instances on GKE](#).

³While websocket use and TPU/GPU access are technically possible with Cloud Run for Anthos on Google Cloud, they are not officially supported.

⁴App Engine standard environment supports background tasks for basic and manual scaling modes.

^βBeta software [has no SLA](#) and may not be suitable for production workloads.

Architecture: Compute Resources



Compute Engine

Infrastructure as a Service offering for stateful applications

- Operating system-level control
- Supports custom machine types
- Autoscaling Support



Kubernetes Engine

Secured and managed Kubernetes service

- Kubernetes applications
- Pod and cluster autoscaling
- workload and network security



Cloud Functions

Serverless execution environment for building and connecting cloud services

- Supports Node.js (Node.js 10 or 12), Python 3 (Python 3.7 or 3.8), Go (Go 1.11 or 1.13) or Java (Java 11)
- Supports Google Cloud Client Libraries



App Engine

Serverless PaaS offering to run web applications without managing infrastructure

- Standard Environment supports Python, Java, Node.js, PHP, Ruby and Go
- Flexible environments supports dockerised containers



Cloud Run

Managed compute platform to run stateless containers that are invocable via web requests or Pub/Sub events.

- Serverless
- Built on Knative
- Fully managed or Cloud Run for Anthos

Features

Use Cases

- On-premises and monolithic workloads
 - Raw compute to meet existing infrastructure requirements
- Examples:
- Relational databases, [SAP HANA](#)
 - CRM systems
 - Legacy ERP systems

- Microservices architecture
 - Hybrid and multi-cloud support
- Examples:
- Containerised API deployment
 - cloud native apps

- Event driven and data processing apps
 - Manipulate user generated data and events
- Examples:
- Post a comment on Slack channel following a GitHub commit
 - Statistical analysis
 - Image thumbnail generation

- HTTP services and backend apps
 - Web frameworks, microservices
- Examples:
- Flask
 - Django
 - Express.js
 - Symfony
 - Spring Boot

- Container-based apps and services
 - Industry standard packaging for multi-cloud infrastructure
- Examples:
- Custom runtime environments such as Rust, Kotlin, C++, and Bash
 - Legacy web apps using languages such as Python 2.7, Java 7
 - Containerized apps that need custom hardware and software (OS, GPUs)