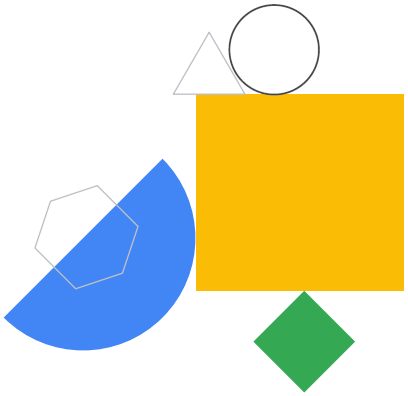
Google Cloud

Fundamentals



Welcome to Fundamentals. In this module we will introduce the digital value chain, which explains how APIs can be used to implement connected digital experiences.

You will learn about Apigee services and the hybrid deployment model that is used to install the Apigee platform.

This module provides a high-level overview of Google Cloud, Kubernetes, and Anthos, which are the building blocks needed to install and manage Apigee hybrid, and also an introduction to REST API concepts.

Agenda

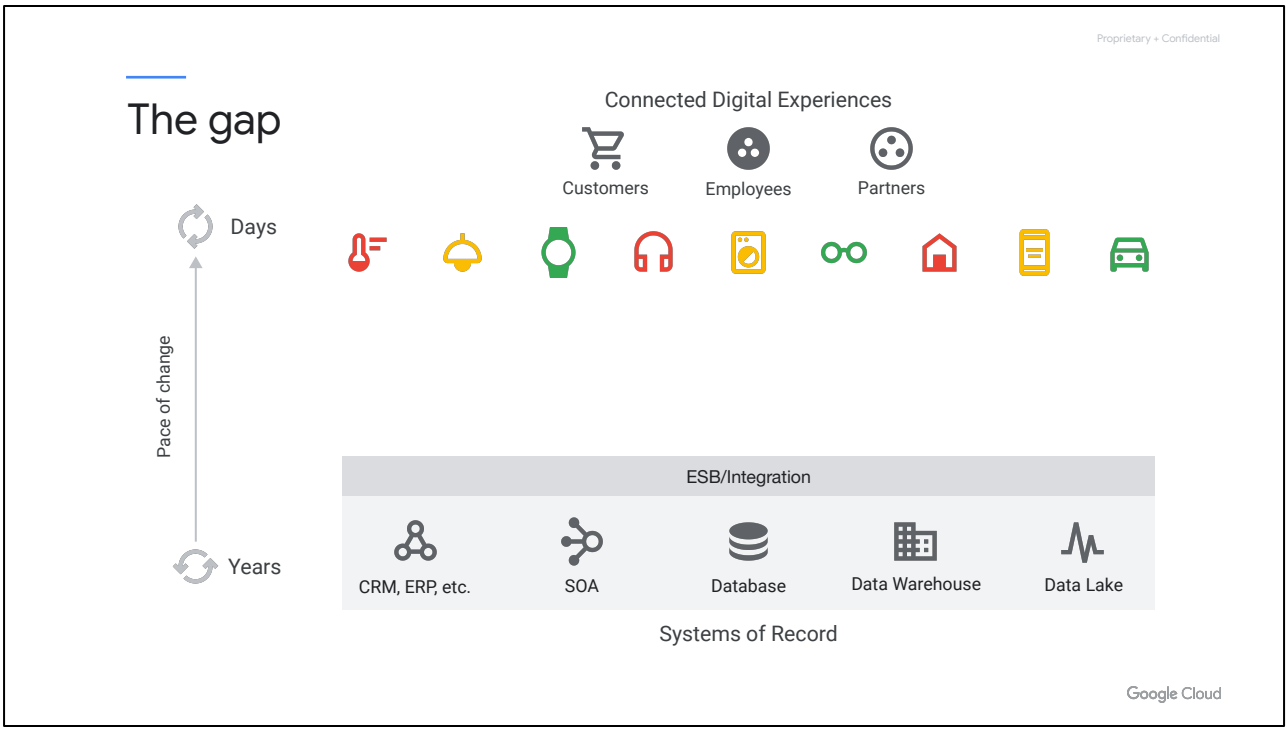
01	Apigee Overview
02	Introduction to Google Cloud
03	Introduction to Kubernetes and Anthos
04	REST Concepts
05	Quiz

Let’s start with an overview of Apigee, Google Cloud's API Management Platform.

We'll discuss the business problems that can be solved using Apigee, see many of the features that Apigee provides, and introduce the components of Apigee and deployment options for the Apigee platform.

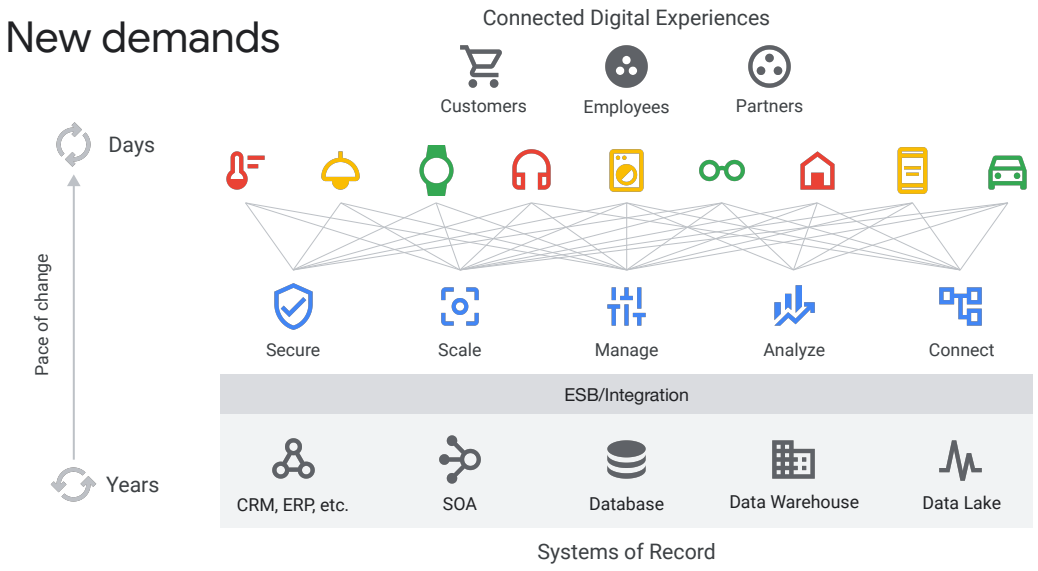
In later lectures, you will learn about the Google Cloud platform and get a high-level overview of Kubernetes and Anthos.

We will also discuss REST concepts in this module.



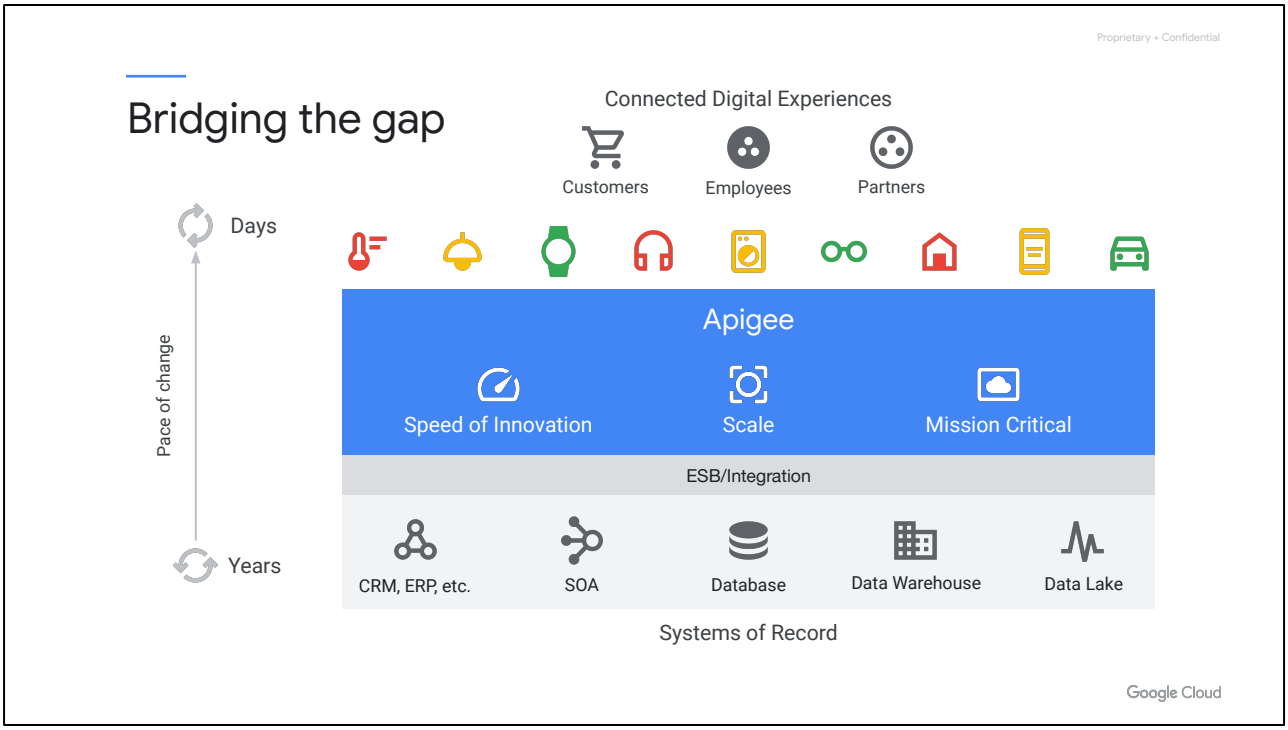
Modern applications provide features and capabilities that legacy systems cannot directly provide.

The pace of change in these digital experiences is much faster than that in traditional systems, and so a gap exists between applications that provide connected experiences and those that provide the business and data systems of record.



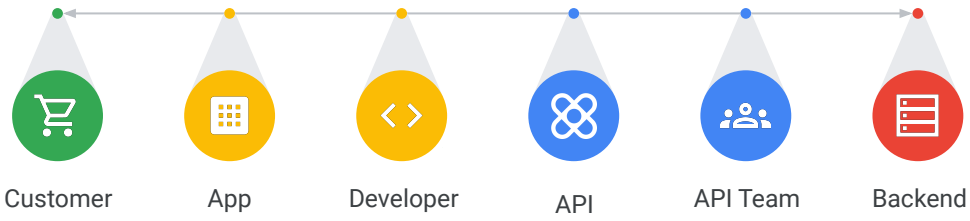
Applications that provide these connected experiences to end users must be able to do so securely and at scale.

They have to manage the entire application lifecycle, connect to different backend systems, and be able to track and analyze the interactions between consumers and producers of data and services.



Apigee is a fully featured API management platform that bridges the gap and enables application developers and API providers to create connected digital experiences for end users.

Digital value chain



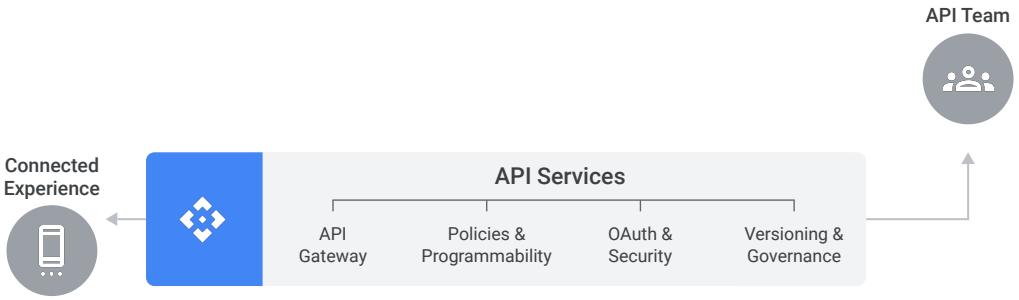
The Digital Value Chain helps enterprises provide customers or end users with connected digital experiences using their backend data and services.

Web or mobile apps are built by internal enterprise developers or by external third-party companies.
App developers leverage the APIs offered by your company.

These **APIs** are built and managed by the **API Team** within the enterprise. APIs integrate with **backend** services and other service endpoints, thus providing a clean and seamless interface to applications.

By focusing on the consumption side of the digital value chain, Apigee improves the app developer experience and makes it easier for your APIs to be successfully consumed.

Apigee services



The Apigee platform includes an API services layer that provides the runtime API gateway functionality.

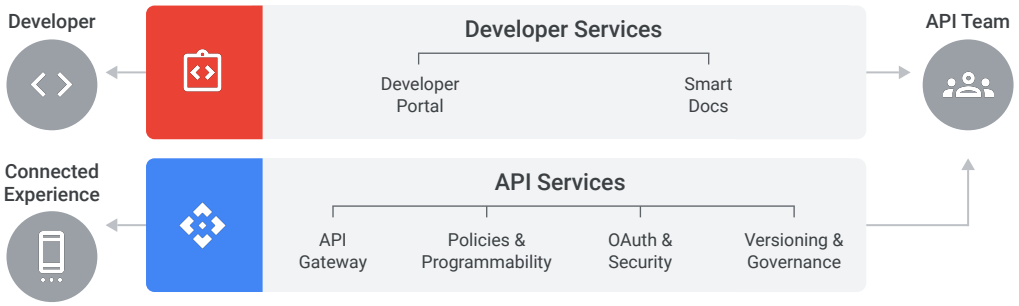
This layer includes the API processing engine, which enables you to add policies and programming to your API proxy.

During proxy execution, these policies are applied to requests from applications and responses from backend systems.

API Services also provide security functionality in the form of OAuth and other security policies for identity verification, authentication, and access control.

It supports features for request and response mediation, revision control and versioning, orchestration, and much more.

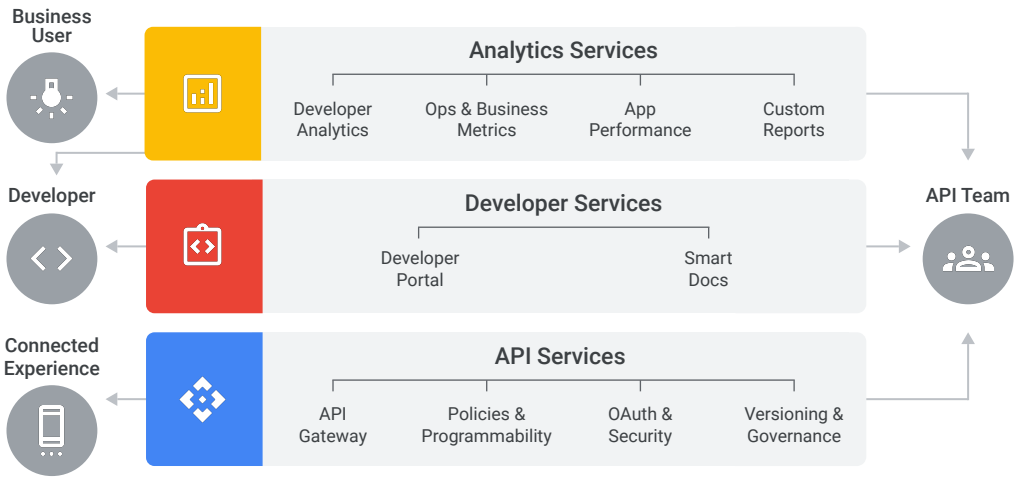
Apigee services



To enable the digital value chain, the Apigee platform includes Developer Services. This includes a developer portal that enables app developers to consume your APIs.

App developers can register their applications and view the API catalog. Developers can browse your API documentation, and even try out your API using Apigee's SmartDocs feature.

Apigee services



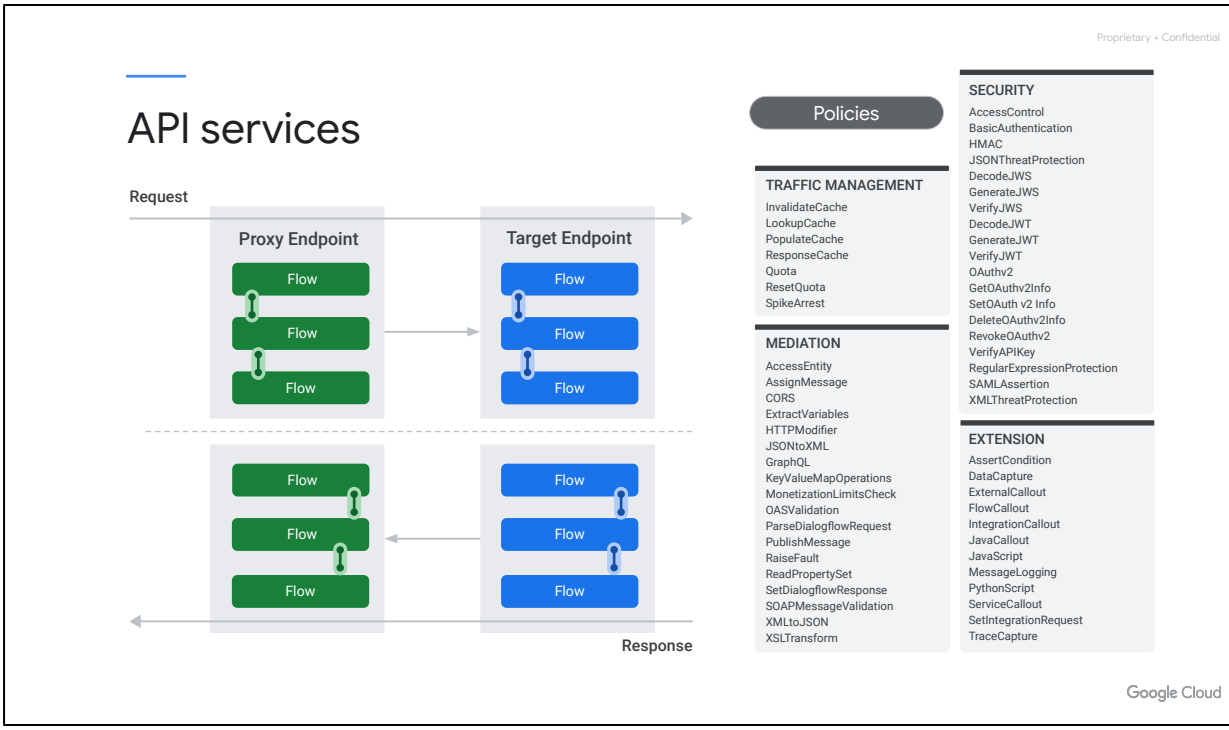
Any API program must be able to measure and track its performance. The Apigee platform includes Analytics services, which enable enterprises to report on various aspects of the API program.

It has many dashboards that chart various dimensions and metrics, including API proxy traffic, response times, target response times, cache performance, developer engagement, and traffic composition.

You can view these dashboards in the Apigee UI or retrieve analytics data using the management API.

You can also create your own custom reports to chart dimensions that are not in the default dashboard reports, or use custom dimensions.

Analytics services is part of the core platform and is a critical component used in API management.
Using this service enables you to gauge the operational health of the platform and determine the overall business value and success of the API program.



API Services provide the runtime execution context for your API proxy.

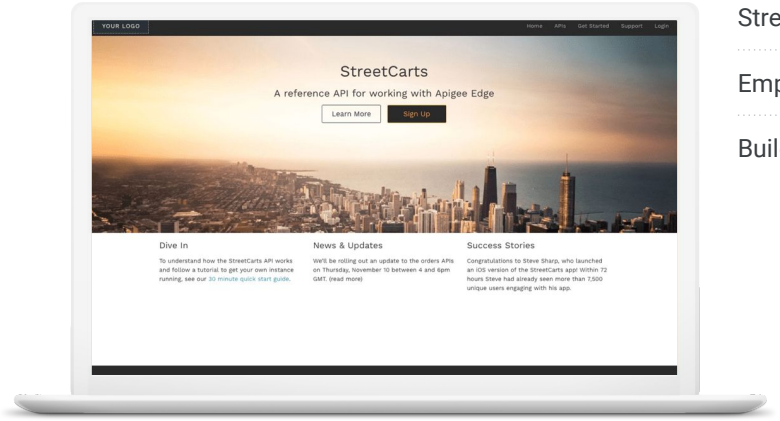
To develop the API proxy logic, you include policies in the proxy or target endpoints of the API proxy within the request or response flows.

A policy is a configuration file written in XML that provides a specific functionality.

Apigee has various pre-built policies that implement traffic management, request and response mediation, and security.

You can also write custom code that can be executed by extension policies.

Developer services: developer portal



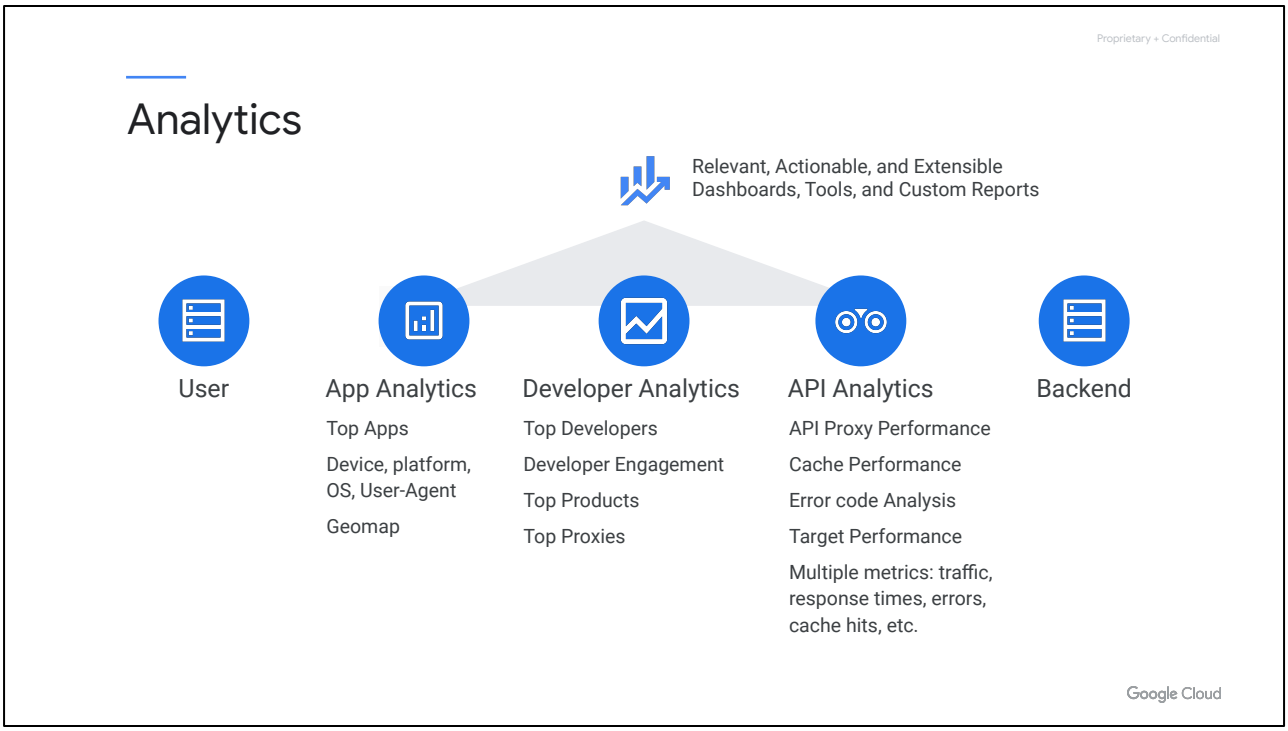
- Streamline API adoption
- Empower developers
- Build an interactive community

Apigee supports three developer portal solutions: an integrated portal, a Drupal-based portal, and a do-it-yourself, or DIY, portal.

The integrated portal is a lightweight portal available in the Apigee hybrid UI. It is the fastest and easiest way to onboard app developers and enable them to register their apps, browse API documentation, and start using your APIs. You can customize the look and feel of the integrated portal using custom themes, menus, and javascript code.

The Drupal portal is based on Drupal 9. It is a fully customizable portal that you host and manage.In addition to the customization options available with the integrated portal, you can implement Apigee monetization, include blogs and forums, and have custom developer and app registration flows.

The DIY portal is a fully customizable portal that you develop and host using the Apigee management APIs. It involves the most effort and highest risk and takes the longest to implement.



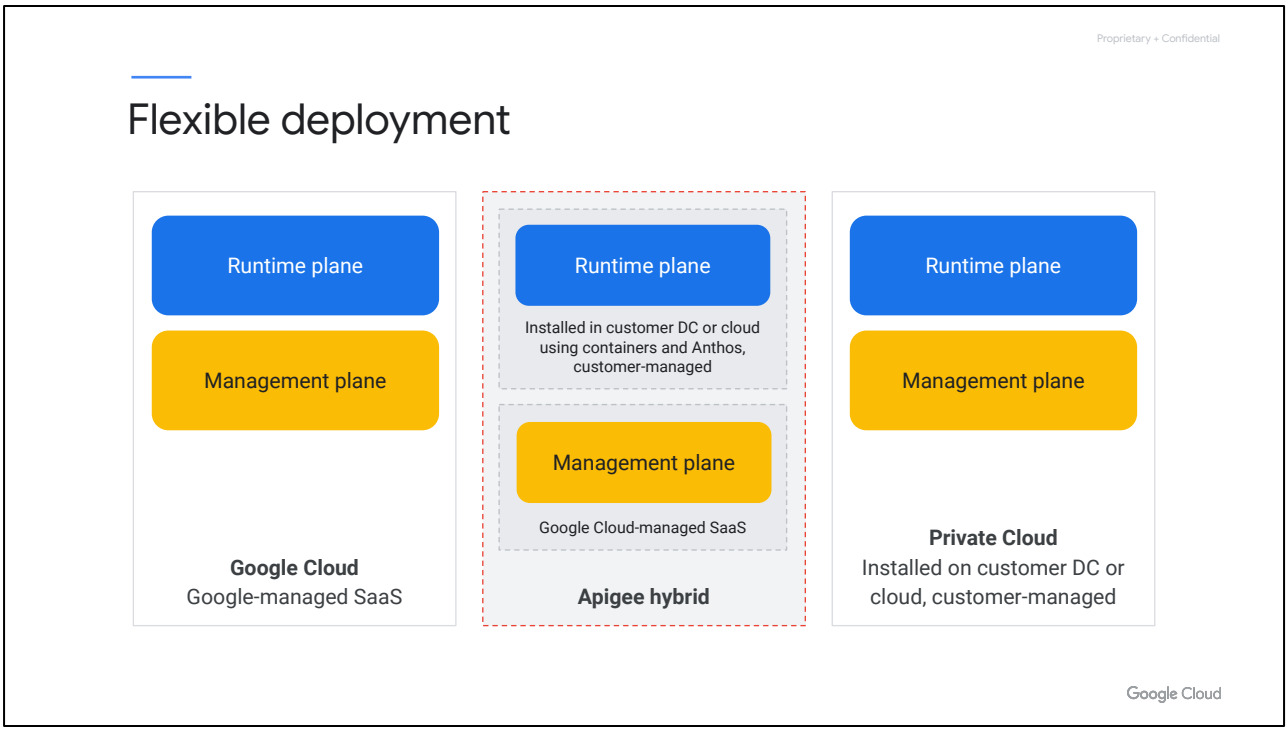
Apigee analytics provides pre-built dashboards for the entire API digital value chain.

These dashboards include charts that report on app analytic dimensions, such as the top apps by API consumption, device type, platform, user-agent, and geographic location.

Developer analytic dashboards include charts on the top developers, API products, and API proxies consumed.

API proxy analytics dashboards have charts on proxy performance, cache performance, error codes, and target performance.

These charts report on different metrics such as total traffic, response times, and cache hits.



[Compare Apigee deployment models]

Apigee offers flexible platform deployment options:

Apigee SaaS *(Public cloud, Apigee X)*
Apigee's Google-managed software-as-a-service deployment simplifies customer adoption and dramatically accelerates time to market for new APIs.
Developers can immediately start building and running APIs at scale. This is the most popular deployment option.
It allows customers to focus on addressing business needs, while letting Google manage the operational overhead of running the software at scale in a secure and reliable way.
Apigee X is the latest implementation of Apigee Edge for public cloud that is hosted on Google Cloud and managed by Apigee.

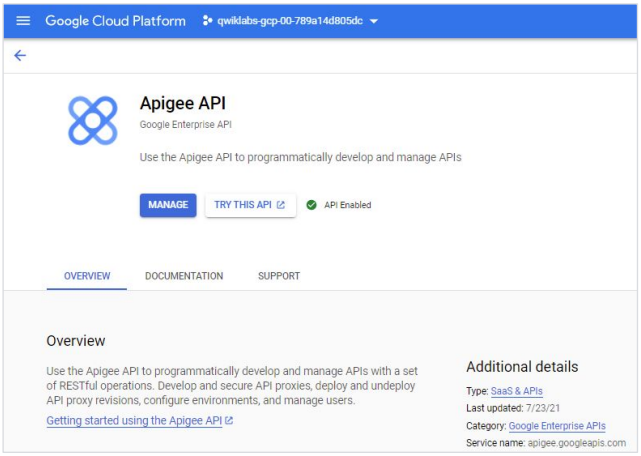
Apigee hybrid
Customers who want to reduce infrastructure management costs but retain the flexibility of running APIs in multiple clouds or on-premises can choose the hybrid deployment model.
This model allows the customer to manage and deploy containerized versions of the API runtime wherever necessary, while delegating the management plane operations to Google.
Apigee's hybrid deployment model takes advantage of Anthos, Google Cloud's hybrid and multi-cloud application platform, which simplifies the management of runtime

components across multiple clouds and data centers.
This course teaches you how to install and manage the Apigee hybrid deployment model.

Apigee Private Cloud
For customers who are willing to manage the Apigee infrastructure, Apigee can be deployed and managed in a private cloud.
Apigee can be installed in an on-premises data center or in an infrastructure-as-a-service cloud.

Management API

- **The Apigee API** is hosted by Apigee in the hybrid management plane on Google Cloud.
- **OAuth 2.0 tokens** are used for authentication when calling the APIs.
- You can programmatically manage API proxies and related configuration using the Apigee APIs.



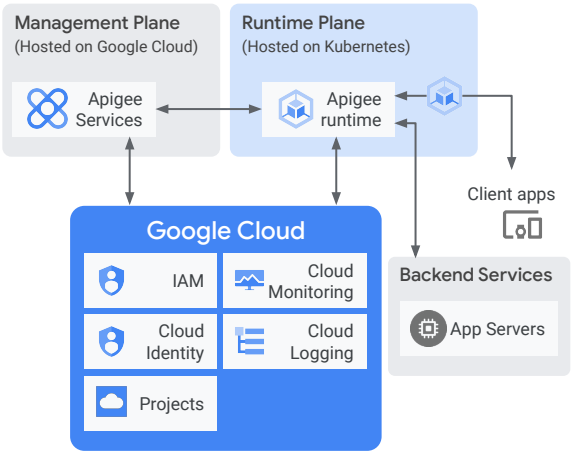
The Apigee management API is a fully featured REST API that is hosted in the Apigee hybrid management plane.

The API must be enabled for your Google Cloud project. The API uses OAuth 2.0 for user authentication. OAuth tokens can be generated using Cloud credentials.

The Apigee API enables you to manage entities such as organizations, environments, developers, and API products.

Apigee hybrid

- Apigee hybrid enables you to deploy APIs in the following platforms: Anthos (GKE / AWS / Azure / on-premises VMware / Bare Metal), EKS, AKS, OpenShift and RKE.
- Apigee hybrid consists of a management plane maintained by Google and a runtime plane that you install on **Kubernetes** using GKE and Anthos clusters.
- Hybrid supports a **multi-cloud strategy** where you manage enterprise API gateways in your data center, on Google Cloud, or on other cloud providers.

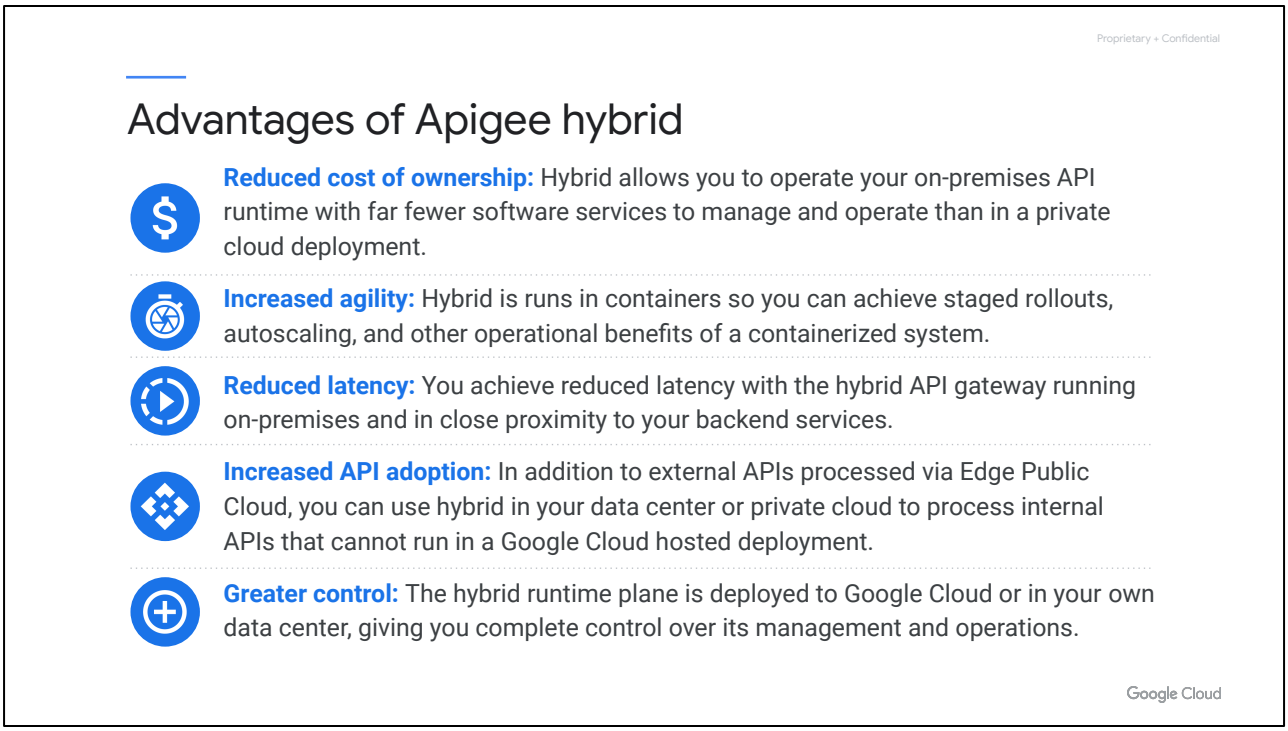


Apigee hybrid enables you to deploy APIs within your data center on-premises, on Google Cloud and on other cloud providers.

Apigee hybrid consists of a management plane hosted and managed by Google and a runtime plane that you install on Kubernetes using GKE, Anthos, EKS, AKS, and other platforms.

Hybrid supports a multi-cloud strategy that enables you to manage enterprise API gateways in your data center, or by using multiple cloud providers.

All API traffic is processed within the boundaries of your network and under your control, but management services such as the UI, API and analytics run in the cloud and are maintained by Google.



There are many advantages of using Apigee hybrid:

- You are required to manage fewer software components with this deployment model than in a fully developed private cloud deployment, thus reducing the cost to install, manage, and operate the runtime plane.
- The software components run in containers on Kubernetes, so you get all the benefits of a container orchestration platform, including rolling updates and autoscaling.
- By installing the hybrid runtime plane in close proximity to backend services, you achieve reduced latency with your APIs.
- Because you can install the hybrid runtime plane in your data center or private cloud, it can be used to process both internal APIs and public externally facing APIs.
- Finally, you have complete control over the management and operations of the hybrid runtime plane.

Agenda

01	Apigee Overview
02	Introduction to Google Cloud
03	Introduction to Kubernetes and Anthos
04	REST Concepts
05	Quiz

In this lecture we will discuss Google Cloud and some of the features that are used by Apigee hybrid.

Google Cloud

- Google Cloud enables you to build, deploy, and scale applications and services on the same infrastructure as Google.
- Google Cloud uses data centers around the globe containing physical infrastructure that is used to provide virtual machines (VMs) and services like compute, storage, networking, big data, and machine learning.
- Each data center location is in a global region.
  - Examples of regions are us-central1 (Iowa, USA), europe-west1 (Belgium), asia-east1 (Taiwan).
- Each region contains one or more zones, which are isolated from each other within the region.
  - Each zone is identified by a name. For example, zone a in the east Asia region is named asia-east1-a.

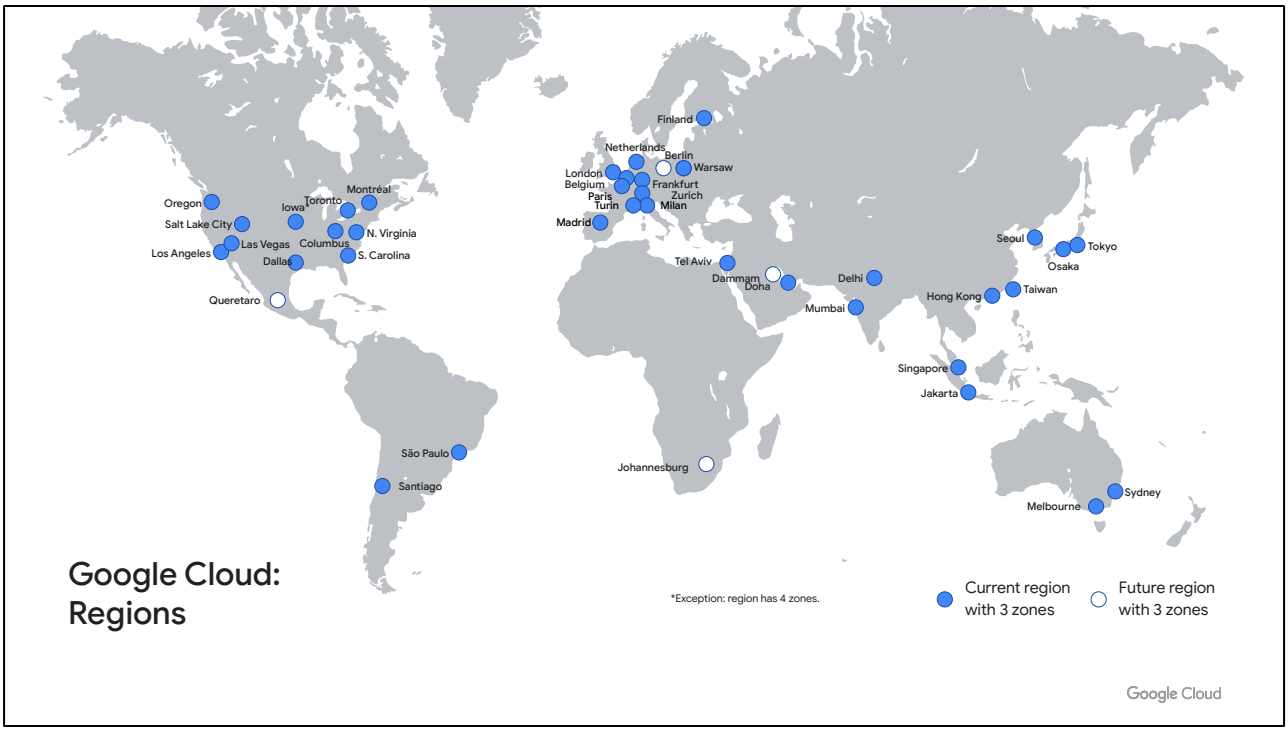
Google Cloud is a suite of compute, network, storage, and other services that run on Google's infrastructure.

With data centers around the globe, you can provision VMs, storage, networking, and other services such as machine learning on Google Cloud.

These data centers are located in regions, and each region is composed of one or more zones.

Cloud resources are deployed in one or more zones within a region. You can think of zones as isolated locations within a region: each with its own power, cooling, etc.

The distribution of resources provides several benefits, including redundancy in case of failure and reduced latency because resources are closer to clients.



[Current Google Cloud regions]

Google Cloud has regions all over the world. This provides a lot of flexibility to locate compute and other resources based on your application requirements.
For instance, you might want to place compute resources for certain applications close to your home office, while choosing other geographic regions for applications that serve customers located globally.

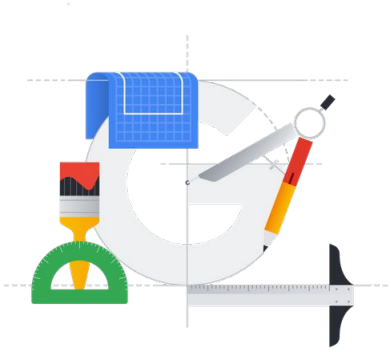
We give you a range of regions around the world—Asia, Australia, North America, South America, and Europe—so you can place resources close to where you need them.

Google Cloud continues to expand into the following regions: Doha (Qatar), Paris (France), Milan (Italy), Madrid (Spain), Turin (Italy), Columbus (US), Berlin (Germany), Dammam (Kingdom of Saudi Arabia), Dallas (Texas), and Tel Aviv (Israel).

Each region also includes at least three availability zones, so you can build highly reliable applications even if you experience large-scale system failures.

Apigee hybrid can be deployed in any available region.

Fundamental principles



1. Infrastructure, security, and networking
2. Intelligence, unlocking data, and accelerating development
3. Openness, enterprise friendliness, and cost efficiency

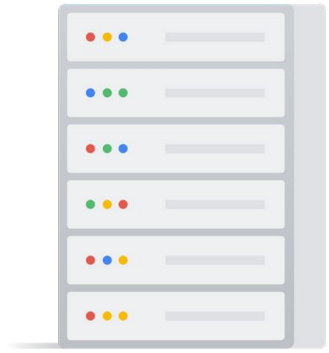
When building our Cloud, we focused on 3 fundamental principles:

- Provide the tools to operate your infrastructure and applications securely.
- Enable you to accelerate development with intelligence and speed.
- Focus on the pieces that affect your business in an open and cost-efficient manner.

A comprehensive platform

Google Cloud is a comprehensive platform that enables you to:

- Securely migrate and modernize application workloads.
- Scalably manage data of all sizes with a suite of fully managed databases.
- Unlock the value of your data with smart analytics.
- Build and operate apps across clouds and on-premises.



Google Cloud is a comprehensive platform that supports ingestion of data, whether in real time through Pub/Sub or with a data transfer service, where we ship a device to your data center to transfer information.

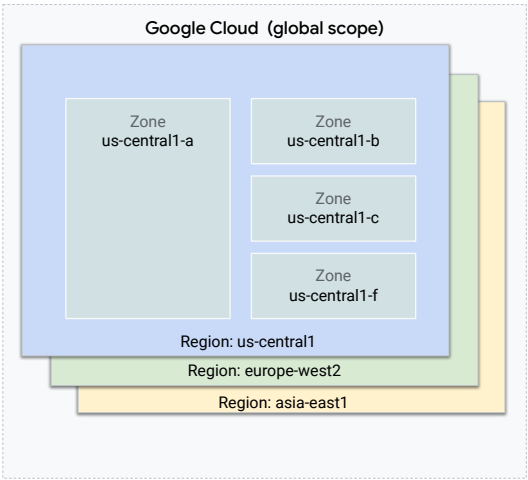
We give you the tools for building data pipelines, for data wrangling, and for cleaning up data.

There are tools for data warehousing, like BigQuery, as well as a variety of tools for doing analytics, using spreadsheets, taking advantage of AI and ML and enabling data scientists.

Using Google Cloud, you can implement your application migration and modernization strategy incrementally and in a cost-efficient manner by running applications and services across multiple clouds and on-premises.

Regions and resources

- Regions are independent geographic areas that consist of *zones*.
- A zone is a deployment area for resources within a region. For high availability, applications are deployed across multiple zones in a region.

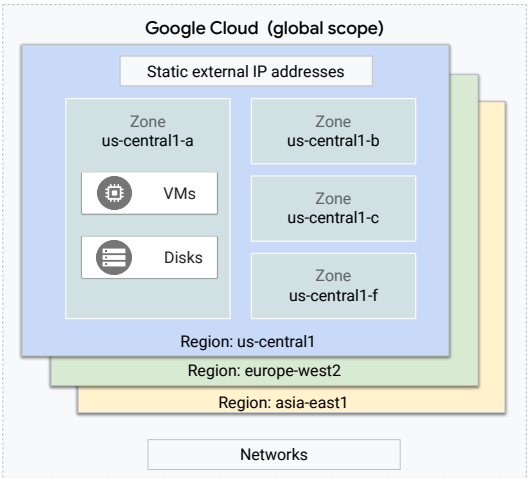


Regions are independent geographic areas that consist of *zones*.

A zone is a deployment area for resources within a region. For high availability, applications are deployed across multiple zones in a region.

Regions and resources

- Regions are independent geographic areas that consist of *zones*.
- A zone is a deployment area for resources within a region. For high availability, applications are deployed across multiple zones in a region.
- Google Cloud resources are fundamental components that make up all Google Cloud services.
- *Resources* are organized hierarchically and help organize your work on Google Cloud.
- Some services and resources can be accessed globally (storage, disk images, networks); some only within a region (static external IP addresses); and others within a zone (VMs, disks).



Google Cloud resources are fundamental components that make up all Google Cloud services.

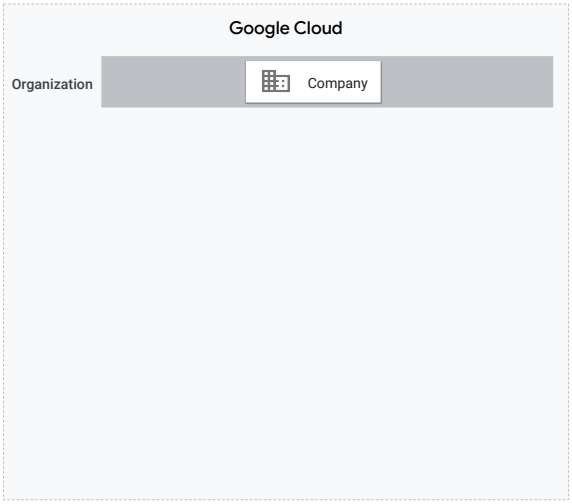
Some services and resources can be accessed globally (cloud storage, disk images, networks); some only within a region (static external IP addresses); and others within a zone (VMs, disks).

The list of available Google Cloud services is long, and it keeps growing.

Locations within regions tend to have round-trip network latencies of under 5 milliseconds on the 95th percentile.

Resource hierarchy

- Companies map their organization structure onto Google Cloud using the resource hierarchy.
- It provides logical attach points for IAM access management policies.
- The Google Cloud organization resource is the root node.



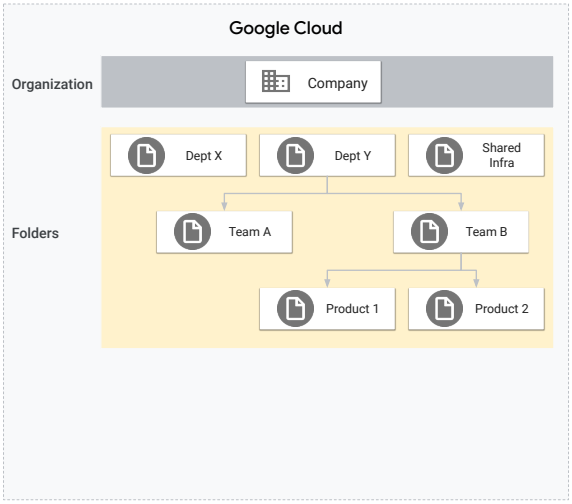
In Google Cloud, resources are organized in a hierarchy.

Companies typically map their organizational structure to the Cloud resource hierarchy, which provides logical attach points for IAM access policies.

The organization resource is the root node of the resource hierarchy. It provides central visibility and control over every other resource that belongs to the organization.

Resource hierarchy

- Companies map their organization structure onto Google Cloud using the resource hierarchy.
- It provides logical attach points for IAM access management policies.
- The Google Cloud organization resource is the root node.
- A folder is an additional grouping mechanism above projects, and is mapped under a Google Cloud organization.

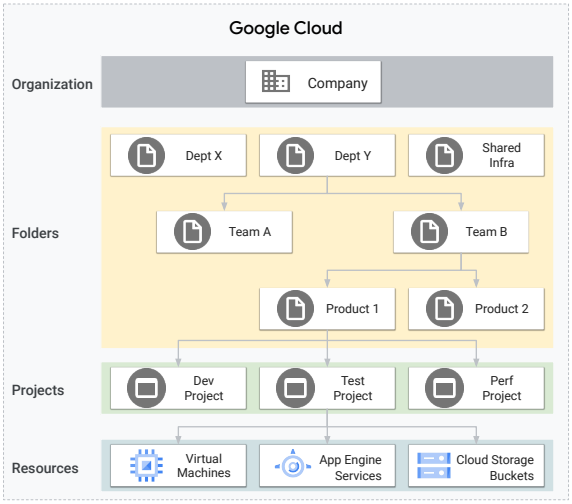


An additional and optional grouping mechanism under the organization node is the folder.

Folders can be mapped to departments, teams or business units within an organization.

Resource hierarchy

- Companies map their organization structure onto Google Cloud using the resource hierarchy.
- It provides logical attach points for IAM access management policies.
- The Google Cloud organization resource is the root node.
- A folder is an additional grouping mechanism above projects, and is mapped under a Google Cloud organization.
- A project resource is the base-level organizing entity for resources like storage buckets and Compute Engine VMs.
- A project is required to create resources, use services and APIs, enable billing, manage permissions, etc.

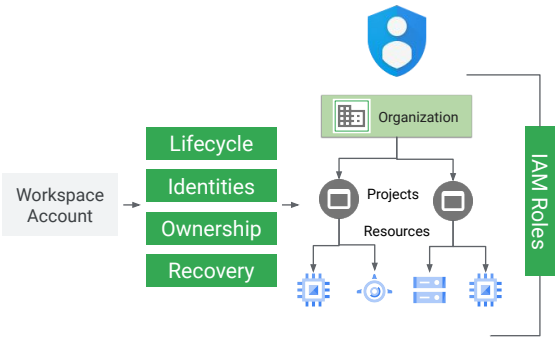


Under the folder or organization node is the project resource.

The Google Cloud project is the base-level entity for organizing resources. It is required to create resources, use Cloud APIs and services, manage permissions, enable billing etc.

Accessing Google Cloud resources

- A Google Workspace or Cloud Identity account is a prerequisite to accessing the Google Cloud organization resource:
  - It provides identity management, recovery, ownership and lifecycle management.
  - It has one Google Cloud organization provisioned with it.
- Google Workspace Super Admins are granted the ability to assign IAM roles by default.
  - They assign the Google Cloud Organization Admin role to appropriate users in the domain.
- IAM policies may be used to control access to all resources in the hierarchy.
- Policies are inherited top-down from the organization level.



To access the organization and other cloud resources, you need a Google Cloud account.

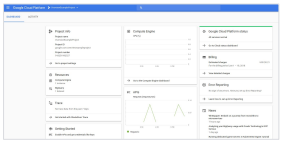
The account must be granted the appropriate IAM roles in order to access organization resources.

For example, to access the organization resource, the user account must be granted the organization admin role.

Access control for resources is managed by IAM policies. An IAM policy contains bindings of user accounts to roles, in addition to other metadata. It is attached to a resource and is inherited by any child resources.

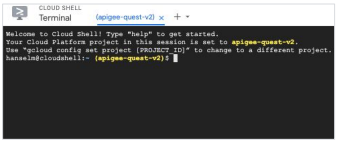
Interacting with services and resources

Google Cloud gives you three basic ways to interact with Google Cloud services and resources.



Google Cloud console

A web-based GUI to manage Google Cloud projects and resources.



Command-line interface

Cloud Shell is a browser-based interactive shell environment and is accessed from the Google Cloud console.

Cloud SDK provides the gcloud command-line tool with commands to manage Google Cloud resources.

Client libraries

Cloud SDK includes client libraries to easily create and manage resources using code in various languages.

Libraries include application APIs for access to Google Cloud services and Admin APIs for managing resources.

Google Cloud

There are 3 basic methods you can use to interact with Google Cloud services and resources. These are the Google Cloud console, the Cloud Shell CLI, and client software libraries.

From the Google Cloud console, you can create a new project or choose an existing project and use the resources that you create in the context of that project.

The cloud SDK provides the gcloud command line tool that you use in cloud shell. It can also be used to programmatically manage cloud resources.

Cloud Shell provides:

- A temporary Compute Engine virtual machine instance
- Command-line access to the instance from a web browser
- A built-in code editor
- Persistent disk storage
- Pre-installed Google Cloud SDK and other tools
- Web preview functionality
- Built-in authorization for access to Google Cloud console projects and resources

Provisioning Apigee hybrid

Hybrid provisioning steps:

1. To install and manage Apigee hybrid, create a Google Cloud account and Google Cloud organization.

To provision and start using Apigee hybrid, follow these steps:

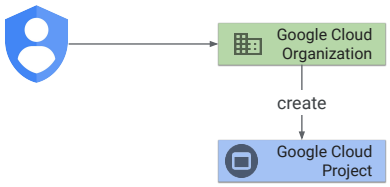
First, you will need a Google Cloud account and organization, which are created when you sign up to use Google Cloud services.

You also use a Google Cloud identity account and organization with Apigee hybrid.

Provisioning Apigee hybrid

Hybrid provisioning steps:

1. To install and manage Apigee hybrid, create a Google Cloud account and Google Cloud organization.
2. Create a Google Cloud project under that organization.

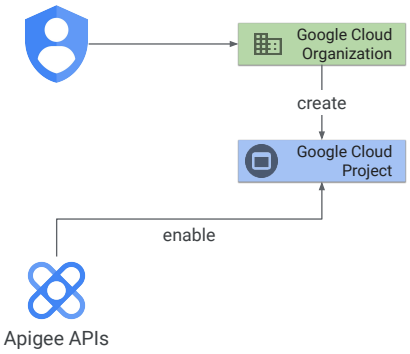


Next, create a project under the cloud organization via the Cloud Console.

Provisioning Apigee hybrid

Hybrid provisioning steps:

1. To install and manage Apigee hybrid, create a Google Cloud account and Google Cloud organization.
2. Create a Google Cloud project under that organization.
3. Enable Apigee APIs for the Google Cloud project. This is needed for Apigee hybrid services to communicate with each other and with other services.



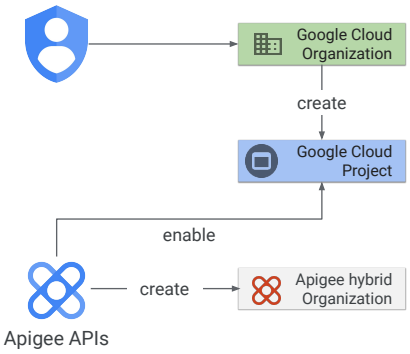
You will need to enable Apigee APIs in your project via the console.

Apigee APIs are used by hybrid services to communicate with other cloud services.

Provisioning Apigee hybrid

Hybrid provisioning steps:

1. To install and manage Apigee hybrid, create a Google Cloud account and Google Cloud organization.
2. Create a Google Cloud project under that organization.
3. Enable Apigee APIs for the Google Cloud project. This is needed for Apigee hybrid services to communicate with each other and with other services.
4. Using the Apigee APIs, create an Apigee hybrid organization.



Use the Apigee API to create an Apigee hybrid organization. This is a separate resource under the cloud organization.

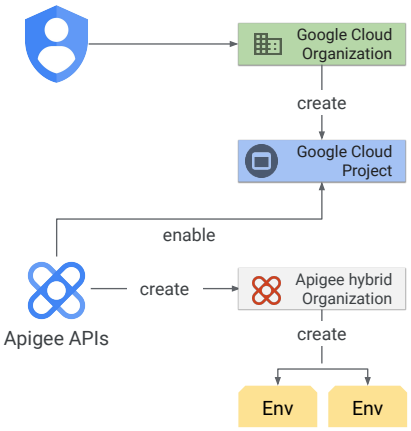
The Apigee hybrid organization ID must match the Google Cloud project ID.

Note that trial hybrid organizations are valid for 60 days.

Provisioning Apigee hybrid

Hybrid provisioning steps:

1. To install and manage Apigee hybrid, create a Google Cloud account and Google Cloud organization.
2. Create a Google Cloud project under that organization.
3. Enable Apigee APIs for the Google Cloud project. This is needed for Apigee hybrid services to communicate with each other and with other services.
4. Using the Apigee APIs, create an Apigee hybrid organization.
5. Create one of more environments under the Apigee hybrid organization to deploy API proxies.

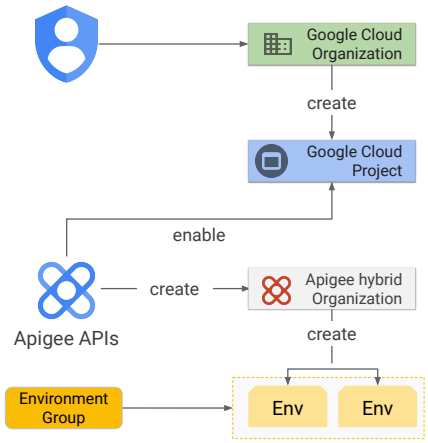


Create one or more environments via the Apigee hybrid UI. An environment is needed to deploy and run your API proxies.

Provisioning Apigee hybrid

Hybrid provisioning steps:

1. To install and manage Apigee hybrid, create a Google Cloud account and Google Cloud organization.
2. Create a Google Cloud project under that organization.
3. Enable Apigee APIs for the Google Cloud project. This is needed for Apigee hybrid services to communicate with each other and with other services.
4. Using the Apigee APIs, create an Apigee hybrid organization.
5. Create one of more environments under the Apigee hybrid organization to deploy API proxies.
6. Create an environment group and assign the environment(s) to the group.



NOTE: For the training labs, the Google Cloud account, organization, project, and hybrid org will be pre-provisioned for you.

Create an environment group via the Apigee hybrid UI and assign one or more environments to the group.

Environment groups allow you to group environments together, and provide the domain names for routing API traffic to the proxies deployed to the environments within the group.

You must have a domain name that you can use for your Apigee hybrid installation and you must create at least one environment group.

Service accounts

- A service account is a special type of account in Google Cloud that enables applications and software components to interact with each other and with other Google Cloud APIs.
- This enhances security by compartmentalizing access and limiting each service account's scope and access privileges.
- Permissions are then applied by assigning one or more roles to the service account.

Apigee hybrid uses service accounts to perform various tasks during normal operation, such as sending logs and metrics data to the management plane, connecting to the management plane, downloading proxy bundles, and executing backups.

As part of the hybrid runtime plane installation process, you create several service accounts to perform these tasks.

These service accounts are assigned roles with the minimum set of permissions that are required to perform the task.

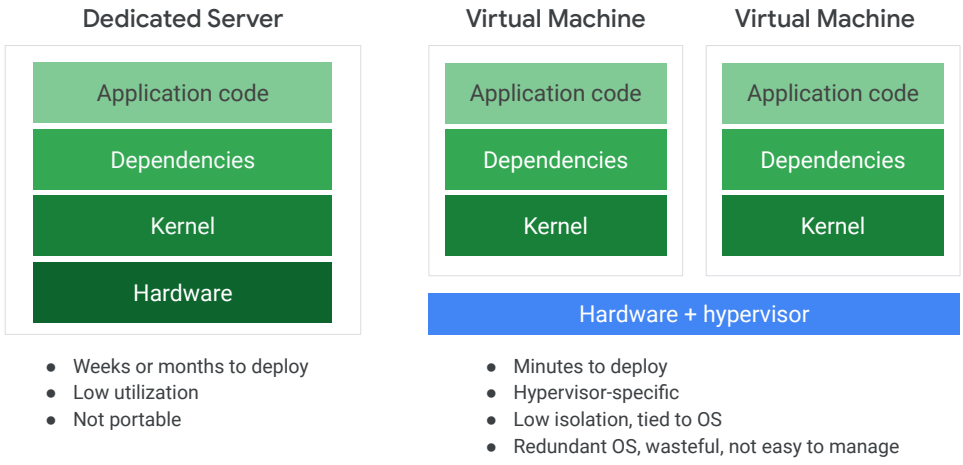
Apigee provides the create-service-account tool to create all the service accounts needed for installation and operation of hybrid.

Agenda

01	Apigee Overview
02	Introduction to Google Cloud
03	Introduction to Kubernetes and Anthos
04	REST Concepts
05	Quiz

In this lecture we will discuss some of the basic concepts of Kubernetes and Anthos.

Virtual evolution



- Weeks or months to deploy
- Low utilization
- Not portable

- Minutes to deploy
- Hypervisor-specific
- Low isolation, tied to OS
- Redundant OS, wasteful, not easy to manage

In the early days of server virtualization, you could run multiple servers and operating systems on the same physical hardware using a hypervisor.
Today you create virtual machine images and deploy them fairly quickly, thus improving hardware utilization and deployment time. But with a VM, you are still bundling the application, dependencies, and operating system together.

This isn't fully portable because you can't move your VM to an environment running a different hypervisor, such as the cloud or your laptop. And every time you start a VM, the OS has to boot, which takes time.

But this was a great step forward from dedicated servers because it raised the abstraction layer, and the VMs allowed you to abstract away the hardware and hypervisor.

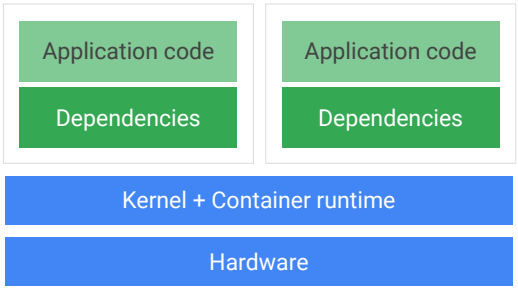
The VM-centric way to solve the app dependency and runtime isolation problem is to run a VM for each application with its own set of dependencies.
The result is that multiple copies of the kernel are running, which makes this deployment model redundant and wasteful.

Containers to the rescue

A container:

- Contains only the application code and dependencies packaged together.
- Is built as a series of layers from an OS image.
- Is ultra portable and runs anywhere on laptops, VMs, and the cloud.
- Is lightweight because it doesn't carry a full OS.
- Provides process isolation, app scalability, and a path to microservices.

To manage and run containers in production environments, you need to scale them, run health checks, and perform container orchestration. This is where Kubernetes comes in.



- Minutes to deploy
- Portable
- Very efficient

When you raise the abstraction level and virtualize the OS, the application and its dependencies can be packaged and run together in a container.

You can run a container anywhere that runs the Linux kernel, which makes it highly portable.

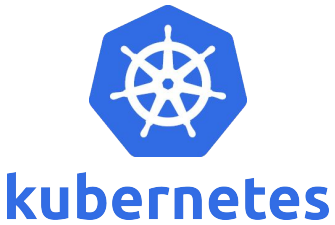
It starts up quickly, so your app scales quickly. It is lightweight because it doesn't carry a full OS and it can use the underlying hardware very efficiently.

This is the right level of abstraction and is the next step in the evolution of managing code.

However, with containers, you still need a system to orchestrate and manage them in production environments.

Kubernetes

- Kubernetes is a container orchestration system with a set of APIs that can be used to deploy and run application containers.
- It comprises a set of control processes that continuously drive the current state toward the provided desired state.
- It can run on laptops, multi-node clusters, in public clouds and on-premises, on VMs, and on bare metal.
- Kubernetes manages containerized applications that run on a set of machines known as a cluster.



Kubernetes is a container orchestration system that is used to deploy and run application containers.

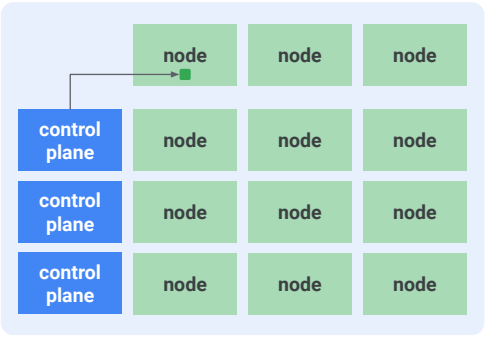
It comprises a set of control processes that continuously drive the current state of application workloads toward the desired state.

It can run on laptops, multi-node clusters, in public clouds, on-premises, on VMs, and on bare metal.

Kubernetes manages containerized applications that run on a set of machines known as a cluster.

Clusters

- A cluster has at least one worker machine called a node, and at least one control plane. The control plane manages the nodes in the cluster.
- Kubernetes manages the pods (jobs) that run containers on those nodes.
- A cluster could have 1000s of nodes with multiple control plans.
- Control plane schedules pods on nodes based on load.
- In GKE, clusters can be single-zone, multi-zone, or regional.
- A regional cluster has control planes and nodes spread across multiple zones in the region.



Kubernetes cluster

In GKE, a cluster consists of at least one control plane and multiple worker machines, called nodes. These control plane and node machines run the Kubernetes cluster orchestration system.

In GKE, clusters can be single-zone, multi-zone, or regional.

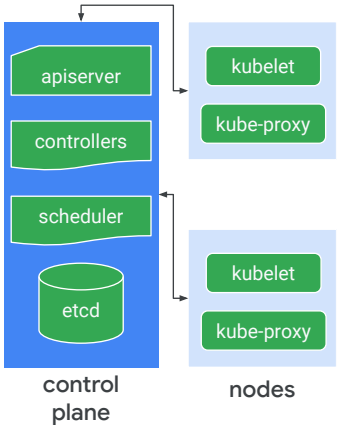
- A single-zone cluster has a single control plane running in one zone. This control plane manages workloads on nodes running in the same zone.
- A multi-zonal cluster has a single control plane running in one zone and has nodes running in multiple zones. Multi-zonal clusters balance availability and cost for consistent workloads.
- A regional cluster has multiple replicas of the control plane running in multiple zones within a given region. Nodes also run in each zone where a replica of the control plane runs.

Because a regional cluster replicates the control plane and nodes, it consumes more Compute Engine resources than a similar single-zone or multi-zonal cluster.

Regional clusters have control planes and nodes spread across zones for high availability and resilience from single zone failure with no downtime during control plane upgrades.

Control plane and node

- The cluster control plane processes include the Kubernetes API server, scheduler, and core resource controllers.
- All interactions with the cluster are done via Kubernetes API calls that are handled by the Kubernetes API Server process running on the control plane.
- The cluster control plane is responsible for scheduling and managing the lifecycle, scaling, and upgrade of containerized application workloads on the nodes.
- A node runs the Docker runtime, Kubernetes node agent (kubelet), and other services.
- Kubelet communicates with the control plane and is responsible for starting and running Docker containers scheduled on that node.



The cluster control plane processes include the Kubernetes API server, scheduler, and core resource controllers.

All interactions with the cluster are done via Kubernetes API calls that are handled by the Kubernetes API Server running on the control plane.

The control plane is responsible for scheduling and managing the lifecycle, scaling, and upgrade of containerized application workloads on the nodes.

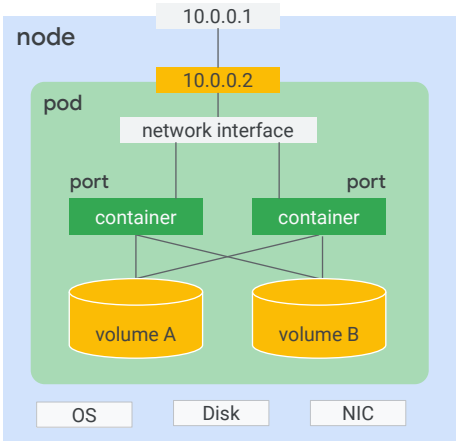
A node runs the container runtime, Kubernetes node agent (kubelet), and other services.

Kubelet communicates with the control plane and is responsible for starting and running containers scheduled on that node.

kube-proxy is a network proxy that maintains the network rules that allow communication with the pods running on the nodes.

Pods

- Kubernetes manages pods that run containers on nodes. In Google Cloud, nodes are VMs in Compute Engine.
- A pod is the basic unit of deployment for containers:
  - It can run one or more containers (sidecar).
  - It shares networking and storage among the containers.
  - It has a unique IP address and set of ports within the cluster.
  - It has its own storage volumes in memory or persistent disk.
- Pods are ephemeral. Deployments manage the lifecycle of their constituent pods and perform horizontal scaling.



Kubernetes Pod

A pod is the smallest deployable unit of compute that you can create and manage in Kubernetes.

A pod is a group of one or more containers with shared storage and network resources and a specification for how to run the containers.

Pods are not usually created directly, but instead are created using Kubernetes workload resources like Deployments or Jobs.

Pods in a Kubernetes cluster are used in two main ways: Pods that run a single container, and Pods that run multiple containers that work together in a sidecar pattern.
These co-located containers form a single cohesive unit of service.

In Apigee hybrid, some of the runtime components use sidecar containers to generate and authenticate OAuth tokens while communicating with the management plane.

Pods are ephemeral, disposable entities. When a Pod is created, it is scheduled to run on a node in the cluster.

The Pod remains on that node until it finishes execution, the Pod object is deleted, the Pod is evicted for lack of resources, or the node fails.

Deployments and ReplicaSets

- A deployment is a declarative way to create and manage pods and ReplicaSets in Kubernetes.
- A desired state is defined in a deployment, and the Deployment Controller changes the actual state to the desired state at a controlled rate.
- A ReplicaSet's purpose is to maintain a stable set of replica pods running at any given time.
- A deployment is defined using a YAML file that specifies:
  - A ReplicaSet with the number of pods.
  - A selector label for choosing which pods to include in the deployment.
  - Pods with their labels and containers in a template.

A deployment is a declarative way to create and manage pods in Kubernetes. It defines a ReplicaSet that specifies the desired number of pod replicas needed.

The Deployment Controller in Kubernetes changes the actual state of the deployment to the desired state at a controlled rate.

The purpose of a ReplicaSet is to maintain a stable set of replica pods running at any given time.

A deployment is defined using a YAML file that specifies the desired number of pods and a selector label that identifies the pods to be included in the deployment.

It also includes a specification that identifies the containers that will run in the pod.

The example Deployment in the slide manages 4 pod replicas with the role label set to “web.”

StatefulSets

- StatefulSets represent a set of pods with unique, persistent identities and stable hostnames that GKE maintains regardless of where they are scheduled.
- The state information and other resilient data for any given StatefulSet pod are maintained in persistent disk storage associated with the StatefulSet.
- StatefulSets are designed to deploy stateful applications and clustered applications that save data to persistent storage.
- StatefulSets are suitable for deploying Kafka, MySQL, Redis, ZooKeeper, Cassandra, etc.

A StatefulSet is the workload object that is used to manage stateful applications. It manages the deployment and scaling of a set of pods, *and provides guarantees regarding the ordering and uniqueness* of these Pods.

Like a Deployment, a StatefulSet manages pods that are based on an identical container spec.

Unlike a Deployment, a StatefulSet maintains a persistent identity for each of the pods, which is maintained across any rescheduling. These identifiers make it easier to match existing volumes to new pods that replace any that have failed.

Apigee hybrid uses a StatefulSet to deploy Cassandra, the datastore that stores objects used by the runtime plane.

DaemonSets

- DaemonSets manage groups of replicated pods.
- DaemonSets attempt to adhere to a one-pod-per-node model, either across the entire cluster or across a subset of nodes.
- As nodes are added to a node pool, DaemonSets automatically add pods to the new nodes as needed.
- DaemonSets are useful for deploying ongoing background tasks that do not require user intervention.
  - Log collection daemon like fluentd is an example of a DaemonSet.

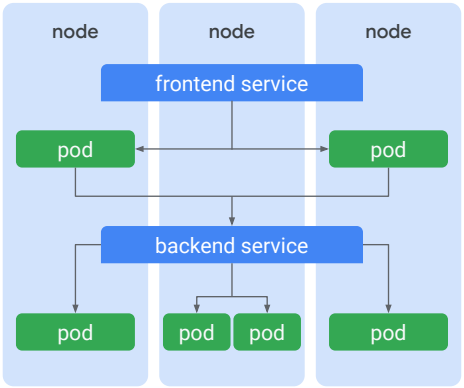
A *DaemonSet* ensures that all (or some) nodes in the cluster run a copy of a pod.

As nodes are added to the cluster, the DaemonSet pods are added to them. As nodes are removed from the cluster, those pods are garbage collected.

Typical uses of a DaemonSet include running a logs collection daemon on every node or running a node-monitoring daemon on every node.

Services

- A service is a network abstraction in Kubernetes that provides a stable endpoint for a group of pods.
- It allows other pods or deployments to reference them as a unit and communicate with them.
- Services use selectors to determine what pods they operate on. If pods have the correct labels, they are automatically picked up and exposed by the service.
- With a service, you get a stable (fixed) IP address that lasts for the life of the service, even as the IP addresses of the member pods change.
- A service also provides load balancing. Clients call the service IP address, and their requests are balanced across the pods that are members of the service.



In Kubernetes, a service is an abstraction that defines a logical set of pods and a policy by which to access them.

The set of pods targeted by a service is usually determined by a selector.

A service has a fixed IP address that lasts for the life of the service, even as the IP addresses of the member pods change.

Because a pod is ephemeral, its IP address changes as it is deleted and re-created. Therefore it doesn't make sense to use Pod IP addresses directly.

Clients call the service IP address instead, and their requests are load-balanced across the pods that are members of the service.

Service definitions

- A service is defined using a YAML file. The service definition file contains:
  - kind of resource - Service

```
kind: Service
apiVersion: v1.1
metadata:
  name: frontend-svc
spec:
  selector:
    role: web
  type: LoadBalancer
  ports:
  - name: http
  protocol: TCP
  port: 80
  targetPort: 80
```

service

Like other Kubernetes resources, a service is defined in a manifest file in yaml format.

In the example shown, the kind of resource is specified as a Service.

Service definitions

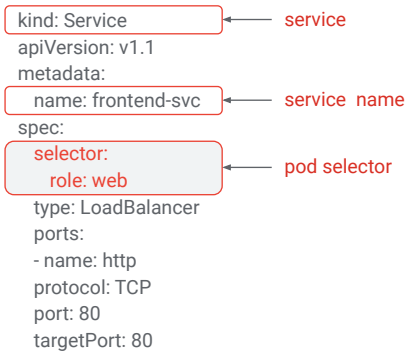
- A service is defined using a YAML file. The service definition file contains:
  - kind of resource - Service
  - service name

```
kind: Service          <-- service
apiVersion: v1.1
metadata:
  name: frontend-svc   <-- service name
spec:
  selector:
    role: web
  type: LoadBalancer
  ports:
  - name: http
  protocol: TCP
  port: 80
  targetPort: 80
```

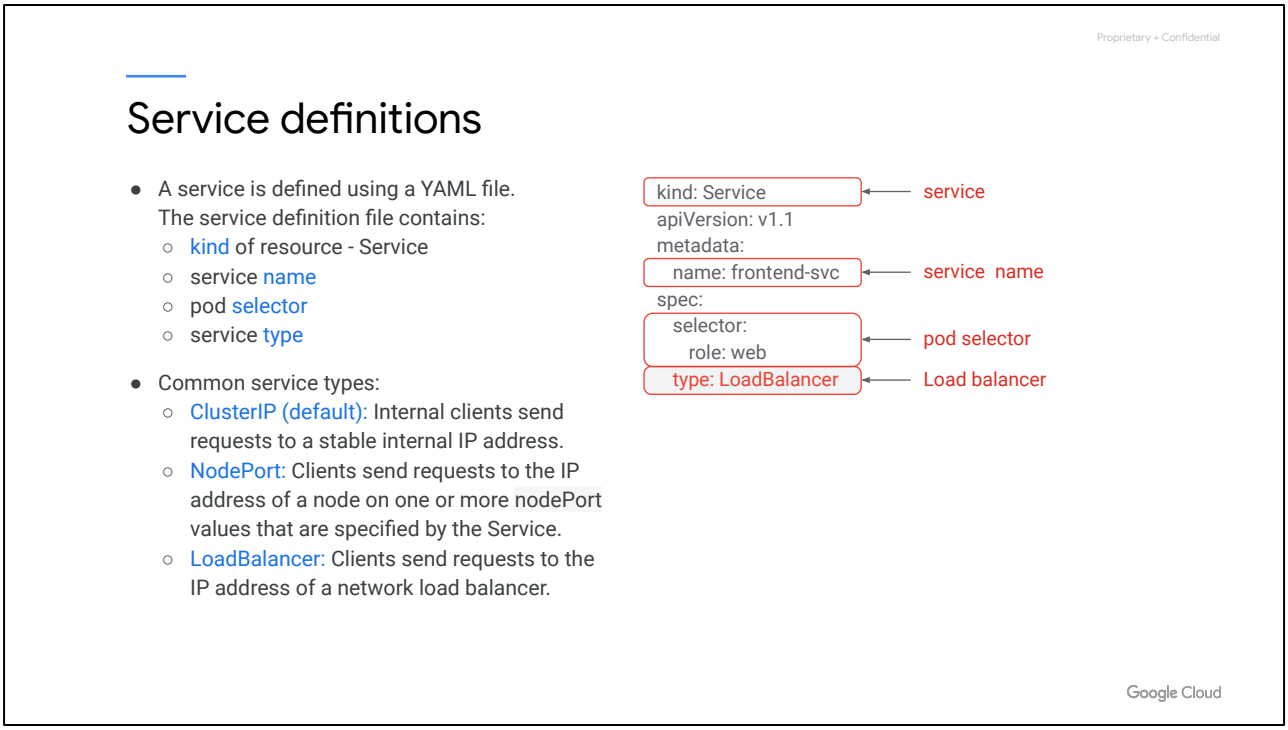

The service is given a name; in this example it is named frontend-svc.

Service definitions

- A service is defined using a YAML file.
  The service definition file contains:
  - kind of resource - Service
  - service name
  - pod selector



The service also has a selector set to the role: web. This selects all pods that have a role label set to web and are part of the frontend service.



The service type defined is LoadBalancer.

A Load balancer service is externally accessible and can be reached by clients that know the DNS name or IP address of the service.

In addition to LoadBalancer, two other types of services that are commonly used in Kubernetes are ClusterIP and NodePort.

See a full list of service types here:
https://cloud.google.com/kubernetes-engine/docs/concepts/service

There are five types of services:

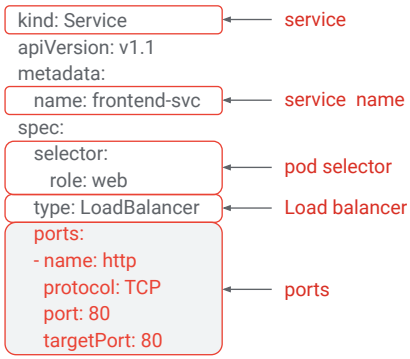
- **ClusterIP (default):** Internal clients send requests to a stable internal IP address.
- **NodePort:** Clients send requests to the IP address of a node on one or more nodePort values that are specified by the service.
- **LoadBalancer:** Clients send requests to the IP address of a network load balancer.
- **ExternalName:** Internal clients use the DNS name of a service as an alias for an external DNS name.
- **Headless:** You can use a headless service in situations where you want a pod grouping but don't need a stable IP address.

The NodePort type is an extension of the ClusterIP type. So a service of type NodePort has a cluster IP address.

The LoadBalancer type is an extension of the NodePort type. So a service of type LoadBalancer has a cluster IP address and one or more nodePort values.

Service definitions

- A service is defined using a YAML file.
  The service definition file contains:
  - kind of resource - Service
  - service name
  - pod selector
  - service type
  - source and target ports



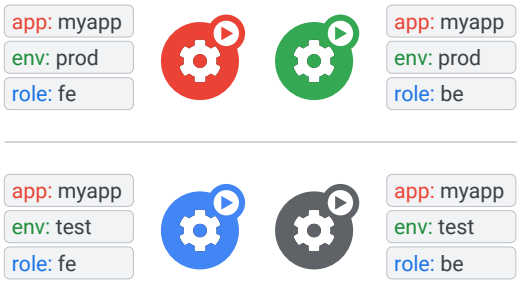
External clients call the service by using the load balancer's IP address and the TCP port specified by port value in the yaml.

The request is forwarded to one of the member pods on the TCP port specified by targetPort.

Labels

- Labels are arbitrary metadata that can be attached to Kubernetes objects in order to group them for selection.
- A label is a key-value pair that can be assigned to an object at creation time and subsequently added and modified at any time.
- An object can have more than 1 label defined. Each key must be unique for that object.

For example, here are 4 pods running, each with 3 labels attached to them: app, env, and role.



Labels are key/value pairs that are attached to Kubernetes objects, such as pods.

Labels are used to specify identifying attributes of objects that are meaningful and relevant to users.

They can be used to organize and to select subsets of objects.

In the example shown, there are 4 pods with 3 labels attached to each of them: app, env, role.

Labels: Pods selection

To list all the pods running the myapp application in production, you could filter requests with the app = myapp, env=prod label selectors.

To list all the pods running the myapp application in production, you could filter requests with the app = myapp, env=prod label selectors.

Labels: Pods selection

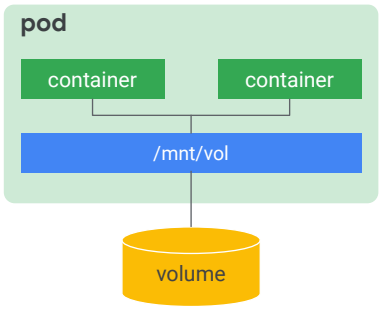
To list all the pods running the myapp application in production, you could filter requests with the app = myapp, env=prod label selectors.

To list all the pods running the myapp application in the test environment, you could filter requests with the app = myapp, env=test label selectors.

To list all the pods running the myapp application in the test environment, you could filter requests with the app = myapp, env=test label selectors.

Volumes

- A volume in Kubernetes is a directory that is accessible to all of the containers in a pod.
- A **pod** specifies what volumes it contains, the path where containers mount the volume, the storage medium, etc.
- Ephemeral volume types have the same lifetimes as their enclosing pods, are created when the pod is created, and are removed when a pod is terminated.
- Volume types backed by durable storage exist independent of the pod. Their data is preserved when the pod is terminated.
- Volume types:
  - Ephemeral: emptyDir, configMap, secret
  - Persistent: persistentVolumeClaim



A Kubernetes volume is a directory that is accessible to all of the containers in a pod.

To use a volume, a pod specifies what volumes to provide for the pod and where to mount those into containers.

After the volume is mounted, the containers in the pod are brought online and the rest of pod initialization is completed.

There are many different types of volumes in Kubernetes.

Volumes, such as those backed by block stores, may outlive the pod. They will be unmounted when a pod goes away and are re-mounted on new pods if needed.

Some volumes, like ConfigMaps and Secrets, are coupled to the life of the pod and cease to exist when the pod ceases to exist.

One class of volume is a persistent volume which has a life cycle of its own, independent of the pod.

Secrets are a way to store sensitive, encrypted data, such as passwords or keys. These are used to keep developers from having to bake sensitive information into their pods and containers. Secrets are stored in a temporary file system so that they're never written into non-volatile storage.

ConfigMaps are used for non-sensitive string data. Storing configuration, setting command line variables, and storing environment variables are natural use cases for ConfigMaps.

A full list of volume types is here:
https://cloud.google.com/kubernetes-engine/docs/concepts/volumes

Custom resource definition

- A custom resource in Kubernetes is an extension of the Kubernetes API that represents a customization of a particular Kubernetes installation.
- After a custom resource is installed, users can create and access its objects using *kubectl*, just as they do for built-in resources like *pods*.
- A custom resource is combined with a *custom controller* to provide a *declarative API* to manage the Kubernetes objects and keep them in sync with the desired state.
- The CustomResourceDefinition (CRD) resource allows you to define custom resources. Defining a CRD object creates a new custom resource with a name and schema that you specify.
- Custom resources can be created and managed using *kubectl*, Kubernetes client libraries, or the Kubernetes REST API.

A Custom resource is an extension of the Kubernetes API that represents a customization of a particular Kubernetes installation.

After a custom resource is installed, users can create and access its objects using kubectl, just as they do for built-in resources like *pods*.

When combined with a custom controller, a custom resource provides a declarative API that allows you to specify the desired state of the resource, which is then kept in sync with the current state of the resource in the cluster.

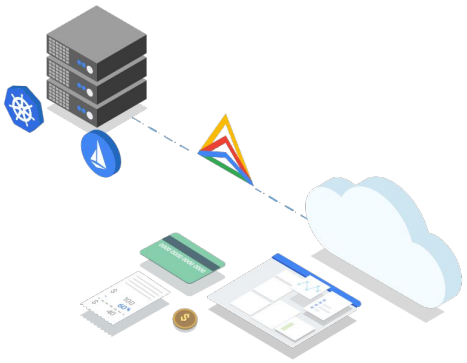
A custom resource can be created by defining a CustomResourceDefintion or CRD object in Kubernetes using a specified name and schema.

Apigee hybrid defines a custom resource definition called ApigeeDeployment.

The ApigeeDeployment CRD is used to create, update, and release stateless hybrid components such as message processors in a Kubernetes cluster.

Anthos

- Anthos is a modern application management platform that provides a consistent development and operations experience for cloud and on-premises environments.
- It is used to manage Kubernetes installations for deploying and managing workloads across the enterprise.
- It uses Anthos clusters as the primary computing environment.



We discussed some of the Kubernetes resources that are used by Apigee hybrid. Lets now talk about Anthos, the hybrid and multi-cloud platform also used by Apigee hybrid.

Anthos is a modern application management platform that provides a consistent development and operations experience for cloud and on-premises environments.

Anthos is used to manage Kubernetes installations for deploying and managing workloads across the enterprise.

The primary computing environment for Anthos uses Anthos clusters, which extend GKE for use on Google Cloud, on-premises, or other cloud providers.

Fun fact: Anthos is “flower” in Greek. Flowers grow on-premises, but they need rain from the clouds (Google Cloud) to flourish.

Anthos components

- Anthos clusters: GKE enriched for hybrid/multi-cloud.
- Anthos clusters on VMware: runs on VMware vSphere.
- Anthos Service Mesh: an enterprise-grade service mesh based on Istio.
- Cloud Run for Anthos: a managed serverless execution environment based on Knative.
- Anthos Config Management: used to automate policy and security at scale for hybrid and multi-cloud.
- Google Cloud Marketplace for Anthos: provides third-party applications.

Anthos comprises a set of core components that implement various features in your hybrid cloud installation.

All of these components run on Google Cloud and on-premises, while some of them also run on other cloud providers currently supported by Anthos.

These components implement infrastructure and cluster management, configuration management, workload migration, service management, serverless workloads, logging and monitoring, and the application marketplace.

Anthos component terminology

Core Component	Cloud	On-prem
Kubernetes Engine	GKE	Anthos clusters on VMware
Configuration Management	Anthos Config Management	Anthos Config Management
Service Mesh	Anthos Service Mesh	Anthos Service Mesh
Serverless Workloads	Cloud Run for Anthos	Cloud Run for Anthos
Logging and Monitoring	Cloud Logging and Cloud Monitoring	Cloud Logging and Cloud Monitoring
Migration	Migrate for Anthos	
Marketplace	Kubernetes Applications	Kubernetes Applications

For infrastructure and cluster management, Anthos uses GKE on Google Cloud, Anthos clusters on other supported cloud providers, and Anthos clusters on VMware for clusters running on-premises.

For configuration management, Anthos Config Management is used on Google Cloud, other supported cloud providers, and on-premises.

To migrate workloads running on-premises to Google Cloud, there is Migrate for Anthos.

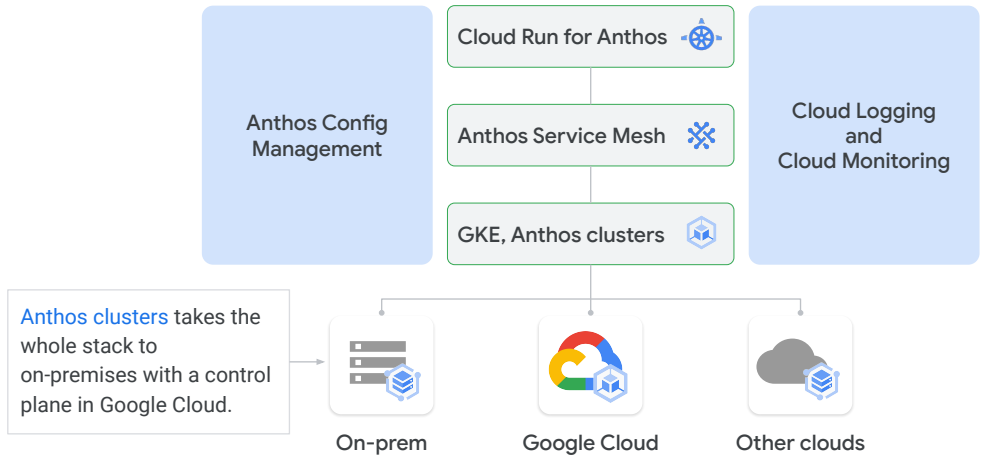
Services on Google Cloud, other supported cloud providers, and on-premises can be managed using Anthos Service Mesh.

Cloud Run for Anthos enables you to run serverless workloads on-premises or on Google Cloud.

For logging and monitoring, Cloud Logging and Cloud Monitoring is available on Google Cloud and on-premises.

The Kubernetes applications in Cloud Marketplace are available for Google Cloud and on-premises.

Anthos stack



Google Cloud

The primary computing environment for Anthos relies on GKE on Google Cloud, on-premises, or other supported clouds to manage Kubernetes installations in the environments where you deploy your applications.

With Anthos on Google Cloud, GKE manages the node components in the customer's project.

With Anthos clusters on VMware, all components are hosted in the customer's on-premises environment, and with Anthos clusters on other supported clouds, all components are hosted in the customer's cloud environment.

To manage service communication, networking, authentication, and other features within your clusters, install Anthos Service Mesh. Anthos Service Mesh is based on Istio, which is an open-source implementation of the service mesh infrastructure layer.

To declare and manage cluster resources consistently across multiple environments, use Anthos Config Management.

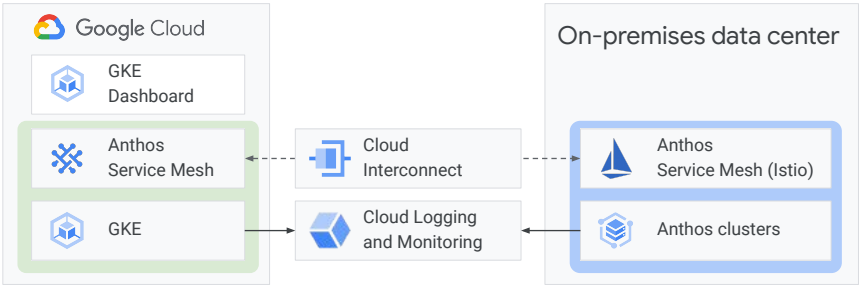
Cloud Run for Anthos is powered by Knative which is an open source project that supports serverless workloads on Kubernetes. It provides a developer-focused experience for creating and running modern applications on-premises or on Google Cloud.

Cloud Logging and Cloud Monitoring provide a unified place to view logs and monitor

the health of your cluster resources.

Note: Anthos GKE on-prem is now called Anthos clusters on VMware.

Anthos networking



On-prem and cloud environments can be connected using:

- A site-to-site VPN using Cloud VPN.
- Dedicated or Partner Cloud Interconnect.

Anthos clusters must also be able to reach Google's API endpoints to access:

- Cloud Logging and Cloud Monitoring for logging and monitoring.
- GKE Dashboard to register clusters with Google Cloud Console.

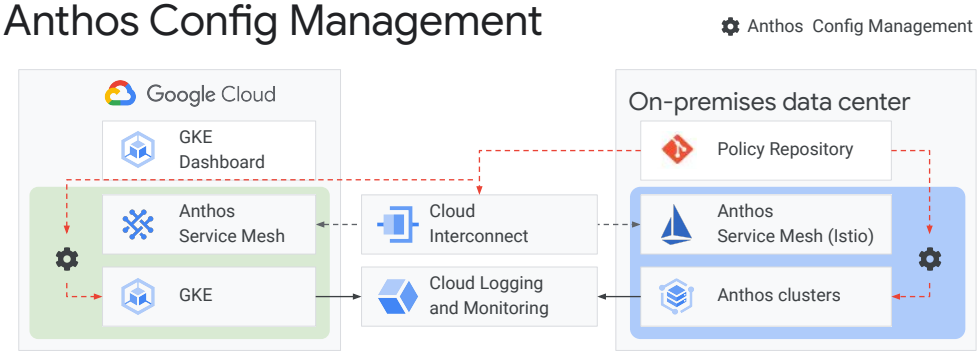
Anthos environments outside of Google Cloud must be able to reach Google's API endpoints for Cloud Logging and Cloud Monitoring services and to register clusters with Google Cloud Console.

You can connect your on-premises, other cloud, and Google Cloud environments in various ways.

The easiest way is by implementing a site-to-site VPN between the environments using Cloud VPN.

For more stringent latency and throughput requirements, you can choose between Dedicated Interconnect and Partner Interconnect.

Anthos Config Management



Anthos Configuration Management:

- A single source of truth for your cluster configuration.
- Configuration is kept in the form of a git repository located either on-premises or in the cloud.
- Agents enforce configurations locally in each environment, managing the complexity of running clusters across environments.

As the number of clusters increases and starts spanning multiple environments, the cluster administrator has additional complexity to manage in terms of **resources and consistency**.

With Anthos Config Management, you create a common configuration for all administrative policies that apply to your Kubernetes clusters both on-premises and in the cloud.

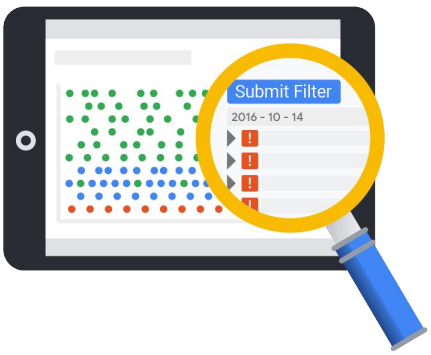
Configuration is treated as data that is stored in a git repository and serves as a single source of truth for your cluster configurations.

Anthos Config Management evaluates changes and rolls them out to all clusters so that your desired state is always reflected in a consistent manner.

Imagine that you had a single cluster: you configure a desired state (creating an object / modify a service / roll out a new version) and the API server manifests that desired state. Now that you have more than one cluster across different environments, you can configure a desired state and save it in a central location that makes sure your desired state is reflected in all your environments.

Anthos logging and monitoring

- Anthos uses Cloud Logging to provide a unified place to store and analyze logs generated by your cluster's internal components and running workloads.
  - Workload logs are automatically enriched with labels like the pod name and cluster that generated them.
- Anthos uses Cloud Monitoring to automatically store your application's critical metrics for use in debugging, alerting, and post-incident analysis.



Cloud Logging and Cloud Monitoring are the built-in observability solutions for Google Cloud.

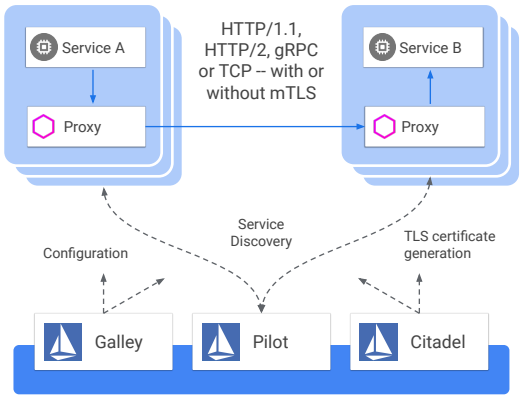
They offer a fully managed logging solution, metrics collection, monitoring, dashboards, and alerting.

Cloud Monitoring monitors GKE on-prem clusters in a way similar to GKE clusters on Google Cloud.

It is a powerful cloud-based observability solution that is easy to configure and provides visibility to your Anthos environments running on-premises or on Google Cloud.

Istio

- Istio is an OSS implementation of the service mesh model.
- Istio uses sidecar proxies to enhance network security, reliability, and visibility.
- Using Istio, these functions are abstracted away from the application's primary container and implemented in a common sidecar proxy that is delivered as a separate container in the same pod.



Istio is an open source implementation of the service mesh model that lets you connect, secure, control, and observe services running in your cluster.

Istio makes it easy to create a network of deployed services with load balancing, service-to-service authentication, monitoring, and other functionality with few or no code changes in service code.

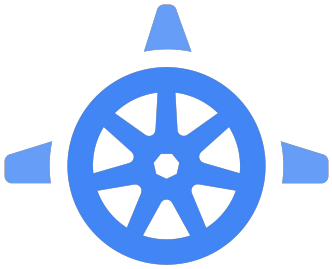
You can add Istio support to services by deploying a special sidecar proxy throughout your environment that intercepts all network communication between services, and then configure and manage Istio using its control plane.

The Istio control plane contains these components:

- Pilot provides service discovery and propagates configuration to the different proxy services.
- Galley is used for configuration management in Istio on Kubernetes and other platforms.
- Citadel is used for TLS certificate management within the service mesh.

Cloud Run for Anthos

- Based on Knative, the Kubernetes-based open API and runtime platform for deploying and managing serverless workloads
- Works with Anthos and on GKE without an Anthos subscription
- Automates workload deployment to your own GKE cluster
- Run your workloads on-premises or on Google Cloud
- Workloads are fully portable. Run them on:
  - GKE on Google Cloud
  - Anthos running on-premises
  - A third-party cloud platform that supports Knative



Cloud Run for Anthos

Cloud Run for Anthos is a standalone product that works with Anthos as well as on GKE without an Anthos subscription.

Like Cloud Run, it is powered by Knative, an open source project that supports serverless workloads on Kubernetes.

Cloud Run for Anthos abstracts away complex Kubernetes concepts, allowing you to easily leverage the benefits of Kubernetes and serverless together. It provides access to custom machine types, additional networking support, and Google Compute Engine hardware accelerators. It allows you to run your workloads on-premises or on Google Cloud.

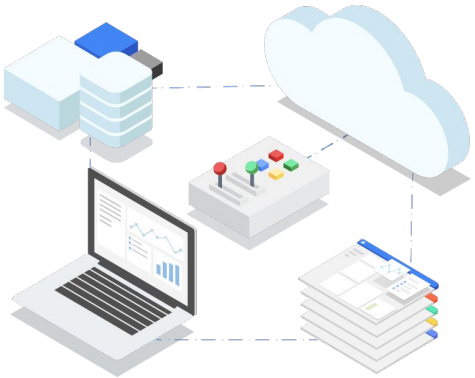
Cloud Run for Anthos provides you with more control over your services because it allows you to access your VPC network and run your services in all GKE regions.

Anthos Service Mesh

Anthos Service Mesh provides:

- A fully managed service mesh that manages your Istio environment in GKE.
- Service metrics and logs for all traffic within the mesh's GKE cluster.
- In-depth telemetry in the Anthos Service Mesh dashboard that enables you to filter and slice your metrics and logs data on a wide variety of attributes.
- Clear and simple insight into the health of your service with service level objectives (SLOs).

https://cloud.google.com/service-mesh/docs



Anthos Service Mesh is a suite of tools that helps you monitor and manage a reliable service mesh.

A service mesh is an infrastructure layer that enables managed, observable, and secure communication across services running in your cluster.

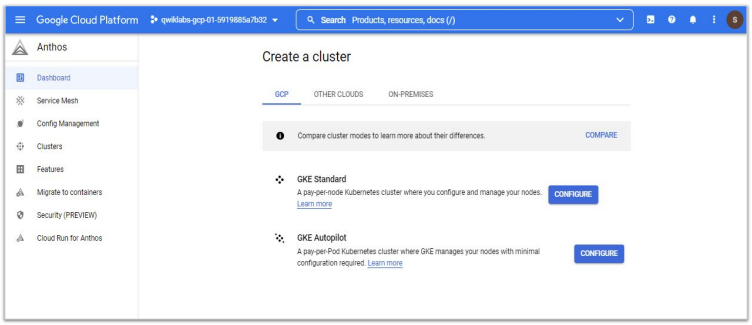
Anthos Service Mesh is based on Istio, the open source service mesh platform, and is deployed as a uniform layer across your entire infrastructure.

It enables you to create, deploy, and manage services on Google Cloud, on-prem, or other supported clouds.

Unified user interface

The Google Cloud Console provides a secure, unified user interface comprising:

- The Anthos Dashboard that lets you view the state of your Anthos managed clusters and workloads.
- Anthos Service Mesh Dashboard that provides service-level metrics on all services running in the mesh.



The Anthos dashboard in the Google Cloud Console provides you with a secure, unified user interface to view and manage your Kubernetes clusters running on Google Cloud, on-premises, or on other supported clouds.

It includes an out-of-the-box structured view of all your Anthos resources.

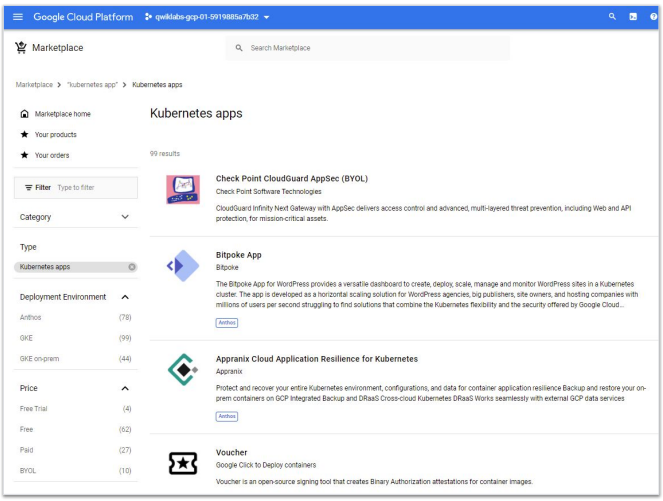
The dashboard landing page gives you a runtime status summary for all your services and clusters.

It also includes the Anthos Service Mesh Dashboard, which provides service-level metrics for observability into the health and performance of your cluster services.

Application Marketplace

Google Cloud Marketplace:

- A solution catalog containing third-party applications
- Using these apps, you can provide additional functionality to your existing clusters wherever they are running.
- Apps are directly supported by the software vendor
- Solutions with direct integration to existing Google Cloud billing



For easy installation and management of third-party applications, you can use Google Cloud Marketplace.

The marketplace contains apps that deploy to your Anthos clusters regardless of where they are running.

You can also search and filter for Anthos apps and easily find Anthos-compatible solutions.

Cloud Marketplace solutions have direct integration with your existing Google Cloud billing and are supported directly by the software vendor.

Benefits of Anthos

Development

- Quickly and easily build and deploy container-based applications and microservices.
- Build CI/CD workflows for configuration as well as code with Anthos Config Management.
- Write code-free instrumentation using Istio and Cloud Operations suite for uniform observability.
- Using Google Cloud Marketplace apps, quickly and easily drop off-the-shelf products into clusters.
- Using Migrate for Anthos and GKE, containerize existing applications to run on Google Kubernetes Engine.

Operational

- Quickly provide and manage infrastructure that is compliant with corporate standards.
- Deploy new clusters with a single command on GKE and GKE On-Prem (gkectl).
- Centralize config management, deployment, and rollback using Anthos Config Management.
- Use single-pane-of-glass visibility across all clusters to monitor infrastructure and application performance.
- Secure microservices using Istio, which provides in-cluster mTLS and certificate management.

Anthos provides a wide range of benefits for both development and operations.

For development, Anthos provides a container management platform based on Kubernetes.

Developers can use this platform to quickly and easily build and deploy existing container-based applications and microservices-based architectures.

They can build CI/CD workflows for configuration and code and have code-free instrumentation with logging and monitoring.

Operational benefits include centralized configuration management, centralized deployment and management of clusters, and single-pane-of-glass visibility across all infrastructure.

With Anthos, you can enforce security standards on clusters using auditable configuration and secure microservices using mutual TLS.

Agenda

01	Apigee Overview
02	Introduction to Google Cloud
03	Introduction to Kubernetes and Anthos
04	REST Concepts
05	Quiz

Let’s discuss some high level concepts of REST and the HTTP protocol that is used by REST APIs.

HTTP overview

Hypertext Transfer Protocol (HTTP) is an application protocol for distributed hypermedia information systems.

- Hypertext is structured text that uses logical links (hyperlinks) between nodes containing text. HTTP is the protocol to exchange or transfer hypertext.
- Communication between a host and a client occurs via a request/response pair.

Uniform Resource Locator (URL):

http(s)://<host>:<port>/<path to resource>?<params-name>=<param-value>

HTTP Methods:

- GET: Retrieve an existing resource.
- POST: Create a new resource.
- PUT/PATCH: Update an existing resource.
- DELETE: Delete an existing resource.
- HEAD: Retrieve message headers only, for a given resource.
- TRACE: Retrieve the hops that a request takes to round trip from the server.
- OPTIONS: Retrieve the server capabilities.

Status Codes: HTTP(S) Status Code Definitions

HTTP is an application-layer protocol that allows a client to fetch and update resources from a server.

It can be used to fetch hypertext documents, text, and binary data. It can also be used to create, update, and delete data on a server.

Clients and servers use a request-response message pair to facilitate this data exchange over HTTP.

A request consists of an HTTP method or verb that defines the client operation, the path of the resource in the form of a URL, the HTTP version, and any optional headers.
Depending on the method used, a request body is also included in the request.

A server response includes the HTTP version, status code of the operation, a status message, HTTP headers, and optionally a response.

HTTP: Request and response message formats

Message format:

- A start line
- Zero or more header fields followed by CRLF
- An empty line indicating the end of the message-header
- Optionally a message-body

Mssage Header

message-header = field-name ":" [field-value]

Message Body

If present, then usually Content-Type and Content-Length headers specify the type and size of the body content.

The request and response message formats contain optional headers and an optional message body, depending on the HTTP method used.

For methods POST, PUT, or PATCH that create or update resources on the server, a body is typically included in the request and response messages.

In addition, the request message usually contains standard headers to indicate the format and length of the body data in the request and response.

REST overview

REST = "Representational State Transfer" ; defined by Roy Fielding in his PhD dissertation.

- REST is an architectural style and is a common, easy-to-understand design pattern for APIs.
- RESTful APIs:
  - Use HTTP for communication and adhere to common web HTTP concepts.
  - Generally use JSON (Javascript Object Notation) for message payloads.
  - Are self-documenting and easy for app developers to consume.

Resource	POST	GET	PUT/PATCH	DELETE
/dogs	Create a new dog	List dogs	Bulk update dogs	Delete all dogs
/dogs/bo	Error	Show Bo	If exists, update Bo. If not, error	Delete Bo

REST was defined in a 2000 PhD doctoral dissertation at the University of California, Irvine. Roy Fielding and his colleagues designed the REST architectural style while version 1.1 of HTTP was being designed.

Roy Fielding and his team used HTTP concepts to create a simple pattern for APIs.

REST APIs leverage common web HTTP concepts like URLs, verbs, and status codes.

Request and response message payloads are generally specified using JavaScript Object Notation or JSON.

The use of these common web patterns in REST APIs tends to reduce the app developer learning curve and helps make REST APIs self-documenting and easy to adopt.

REST APIs

Resource structure:

```
GET /owners/5678/dogs
```

Queries:

```
GET /dogs?color=red&state=running&location=park
```

Partial response:

```
GET /owners/5678/dogs?fields=id,name,picture
```

Pagination:

```
GET /dogs?limit=25&offset=50
```

API Versioning:

```
/dogs/v1
```


REST APIs use well-defined URLs that contain the resource type being operated on.

URLs contain the plural noun form of the resource type, for example, *owners* or *dogs*.

If the API operates on a specific resource object, the ID of the resource is included in the URL path following the resource noun.

For APIs that operate on a subset of resources, query parameters are used in the URL.

There are techniques to request partial responses and pagination using URL query parameters.

When you need to version, a best practice is to version your APIs using a version string as a path fragment in the URL.

JSON overview

- JSON (JavaScript Object Notation) is a lightweight data-interchange format based on the JavaScript Programming Language.
- JSON is a language -independent data format.

JSON home - http://www.json.org

Syntax specification

```
object
    {}
    { members }
members
    pair
    pair , members
pair
    string : value
array
    []
    [ elements ]
elements
    value
    value , elements
value
    string
    number
    object
    array
    true
    false
    null
```

Example

```
{
  "firstName": "John",
  "lastName": "Smith",
  "isAlive": true,
  "age": 25,
  "address": {
    "streetAddress": "21 2nd Street",
    "city": "New York",
    "state": "NY",
    "postalCode": "10021-3100"
  },
  "phoneNumbers": [
    {
      "type": "home",
      "number": "212 555-1234"
    },
    {
      "type": "mobile",
      "number": "646 555-4567"
    }
  ],
  "children": [],
  "spouse": null
}
```


JSON is a lightweight data interchange text format that is easy for humans to read and write.

It is also easy for machines to parse and generate.

It is based on a subset of the JavaScript programming language standard.

JSON is built on two structures: a collection of name/value pairs and an ordered list of values.

Curl overview

- Curl is an open source command line tool and library for transferring data with URL syntax.
- It supports multiple protocols like HTTP, HTTPS, IMAP, IMAPS, LDAP, LDAPS, POP3, and POP3S.
- It supports SSL certificates, HTTP form-based upload, proxies, user+password authentication, and more.
- Distribution: https://github.com/curl/curl

```
curl https://retailstore.com/v1/categories

[{"color":"#506d7d","id":0,"name":"Appliances"},
{"color":"#94cbb9","id":1,"name":"Automotive"},{
"color":"#ffc20e","id":2,"name":"Baby"},{"color"
:"#f7a969","id":3,"name":"Books"},{"color":"#6c5
07d","id":4,"name":"Cameras"},{"color":"#f07d65"
,"id":5,"name":"Clothing"},{"color":"#e36b22","i
d":6,"name":"Electronics"},{"color":"#b34b37","i
d":7,"name":"Fitness"},{"color":"#97779e","id":8
,"name":"Gifts"},{"color":"#71a973","id":9,"name
":"Health"},{"color":"#b75f5f","id":10,"name":"H
ome"}]
```

```
curl -v http://www.apigee.com
* Rebuilt URL to: http://www.apigee.com/
*   Trying 184.72.221.111...
* Connected to www.apigee.com (184.72.221.111) port 80 (#0)
> GET / HTTP/1.1
> Host: www.apigee.com
> User-Agent: curl/7.43.0
> Accept: */*
>
< HTTP/1.1 301 Moved Permanently
< Server: nginx
< Date: Tue, 19 Apr 2016 20:23:12 GMT
< Content-Type: text/html
< Content-Length: 178
< Connection: keep-alive
< Location: http://apigee.com/
<
<html>
<head><title>301 Moved Permanently</title></head>
<body bgcolor="white">
<center><h1>301 Moved Permanently</h1></center>
<hr><center>nginx</center>
</body>
</html>
* Connection #0 to host www.apigee.com left intact
```

Google Cloud

CURL is a command-line tool and library to get and send data using URL syntax.

It supports multiple protocols and performs SSL certificate validation when used with a secure protocol like HTTPS.

Through the libcurl library, curl supports standard HTTP methods such as GET, POST, and PUT.

It is a popular tool used to test REST APIs from the command line.

Curl options

A few examples that illustrate some of the common options used:

Perform a GET on specified URL:

```
curl http://www.apigee.com
```

Perform a GET on specified URL and set output as verbose:

```
curl -v http://www.apigee.com
```

Specify the username and password to use for server authentication:

```
curl -u <user>:<password> http://<host>:<port>/<resource>
```

Perform a POST on the specified resource, passing <data>:

```
curl -u <user>:<password> -X POST http://<host>:<port>/<resource> -d '<data>'
```

Perform a POST on the specified resource, passing <header> and <data>:

```
curl -u <user>:<password> -H "<header>" -X POST http://<host>:<port>/<resource> -d '<data>'
```

Perform a DELETE on the specified resource <id>, passing header <header>:

```
curl -u <user>:<password> -H "<header>" -X DELETE http://<host>:<port>/<resource>/<id>
```


Here are some examples of using curl.

You can easily make a GET call to a URL by specifying it as an argument to the curl command. The output of the command contains the response from the URL.

When you specify the -v option, curl outputs verbose information as it submits the request and receives the response from the requested URL.

You can also supply authentication options like a username and password on the curl command line.

And you can specify which HTTP methods to use when making API calls with curl by using the -X option.

The -H option is used to supply optional headers in the curl request.

Curl supports a wide variety of command line options that are documented on the curl man page.

Agenda

01	Apigee Overview
02	Introduction to Google Cloud
03	Introduction to Kubernetes and Anthos
04	REST Concepts
05	Quiz

Question #1

Question

How does Apigee help in the digital transformation of an enterprise?

A. Using Apigee, you can easily and securely integrate with backend services.

B. Apigee enables the entire digital value chain, from exposure to consumption of services and data, using APIs.

C. Apigee can be used to break up a monolithic application into microservices.

D. Apigee provides capabilities of a service mesh that you can use to run microservices.

Question #1

Answer

How does Apigee help in the digital transformation of an enterprise?

A. Using Apigee, you can easily and securely integrate with backend services.

B. Apigee enables the entire digital value chain, from exposure to consumption of services and data, using APIs.

C. Apigee can be used to break up a monolithic application into microservices.

D. Apigee provides capabilities of a service mesh that you can use to run microservices.

Explanation:

A. Incorrect. Integration of backend services is not sufficient to digitally transform an enterprise, and integration can be achieved with products other than Apigee.

B. Correct! Apigee provides tools to implement all of the entities in the digital value chain to achieve API development, integration, and consumption.

C. Incorrect. Apigee provides API management capabilities that allow applications to utilize APIs that consume backend services.

D. Incorrect. Apigee is used to implement APIs for services that can run in a mesh.

Question #2

Question

Which deployment models are supported by Apigee? Select three.

A. Customer-managed management and runtime planes

B. Apigee-managed management and runtime planes

C. Customer-managed management plane and Apigee-managed runtime plane

D. Apigee-managed management plane and customer-managed runtime plane

Question #2

Answer

Which deployment models are supported by Apigee? Select three.

A. Customer-managed management and runtime planes 

B. Apigee-managed management and runtime planes 

C. Customer-managed management plane and Apigee-managed runtime plane

D. Apigee-managed management plane and customer-managed runtime plane 

Explanation:

A. Correct! The Apigee private cloud deployment model hosts the management and runtime planes in the customer datacenter or other cloud.

B. Correct! The Apigee SaaS deployment model hosts the management and runtime planes in Google Cloud.

C. Incorrect. This deployment model is not supported.

D. Correct! The Apigee hybrid deployment model hosts the management plane in Google Cloud, with the runtime plane hosted and managed by the customer in their datacenter or other cloud.

Question #3

Question

Which of the following are benefits of using Apigee hybrid? Select three.

A. The hybrid runtime plane is deployed to Google Cloud, to other clouds, or in your own data center, giving you complete control over its management and operations.

B. Apigee hybrid runs on Kubernetes, so you can achieve autoscaling and other operational benefits of a containerized system.

C. With Apigee hybrid, you operate your API runtime plane with fewer software services than in a private cloud deployment, resulting in reduced cost of ownership.

D. Installing the Apigee hybrid runtime plane on GKE provides automatic access to all other Google Cloud APIs and services.

Question #3

Answer

Which of the following are benefits of using Apigee hybrid? Select three.

A. The hybrid runtime plane is deployed to Google Cloud, to other clouds, or in your own data center, giving you complete control over its management and operations.

B. Apigee hybrid runs on Kubernetes, so you can achieve autoscaling and other operational benefits of a containerized system.

C. With Apigee hybrid, you operate your API runtime plane with fewer software services than in a private cloud deployment, resulting in reduced cost of ownership.

D. Installing the Apigee hybrid runtime plane on GKE provides automatic access to all other Google Cloud APIs and services.

Explanation:

A. Correct! The customer has full control over the operation and management of the hybrid runtime plane.

B. Correct! Apigee hybrid components are containerized, giving you all the benefits of running on Kubernetes.

C. Correct! The Apigee hybrid runtime plane contains a subset of the components that are installed in the Apigee Edge for Private Cloud product, as the other components are hosted by Apigee in Google Cloud.

D. Incorrect. Hybrid runtime services will still need access to Google Cloud services through purchase entitlements, API enablement, IAM, etc.

Question #4

Question

Which set of HTTP methods contains one that is invalid?

A. GET, POST, PUT

B. PATCH, HEAD, DELETE

C. PUT, TAIL, GET

D. OPTIONS, TRACE, PATCH

Question #4

Answer

Which set of HTTP methods contains one that is invalid?

A. GET, POST, PUT

B. PATCH, HEAD, DELETE

C. PUT, TAIL, GET 

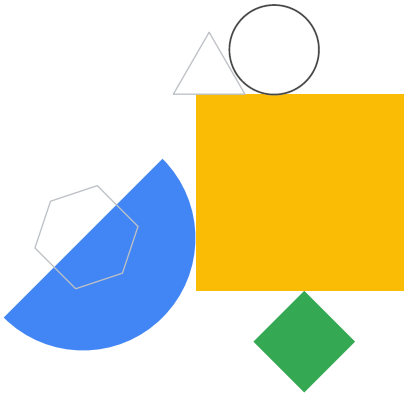
D. OPTIONS, TRACE, PATCH

Explanation:

A. Incorrect. GET, POST, PUT are valid HTTP methods.
B. Incorrect. PATCH, HEAD, DELETE are valid HTTP methods.
C. Correct! TAIL is not a valid HTTP method.
D. Incorrect. OPTIONS, TRACE, PATCH are valid HTTP methods.

Google Cloud

Review: Fundamentals



In this module you learned about Apigee and the services it provides to help customers implement their digital value chain.

You also learned about Apigee hybrid as one of the deployment models for installing Apigee.

The topics included a high-level overview of Google Cloud, Kubernetes, and Anthos, which are the building blocks needed to install and manage Apigee hybrid. We also covered a brief introduction to REST concepts.

In the next module, you will learn about the Apigee hybrid architecture, Apigee terminology, and networking concepts that are needed to install and manage the hybrid platform.