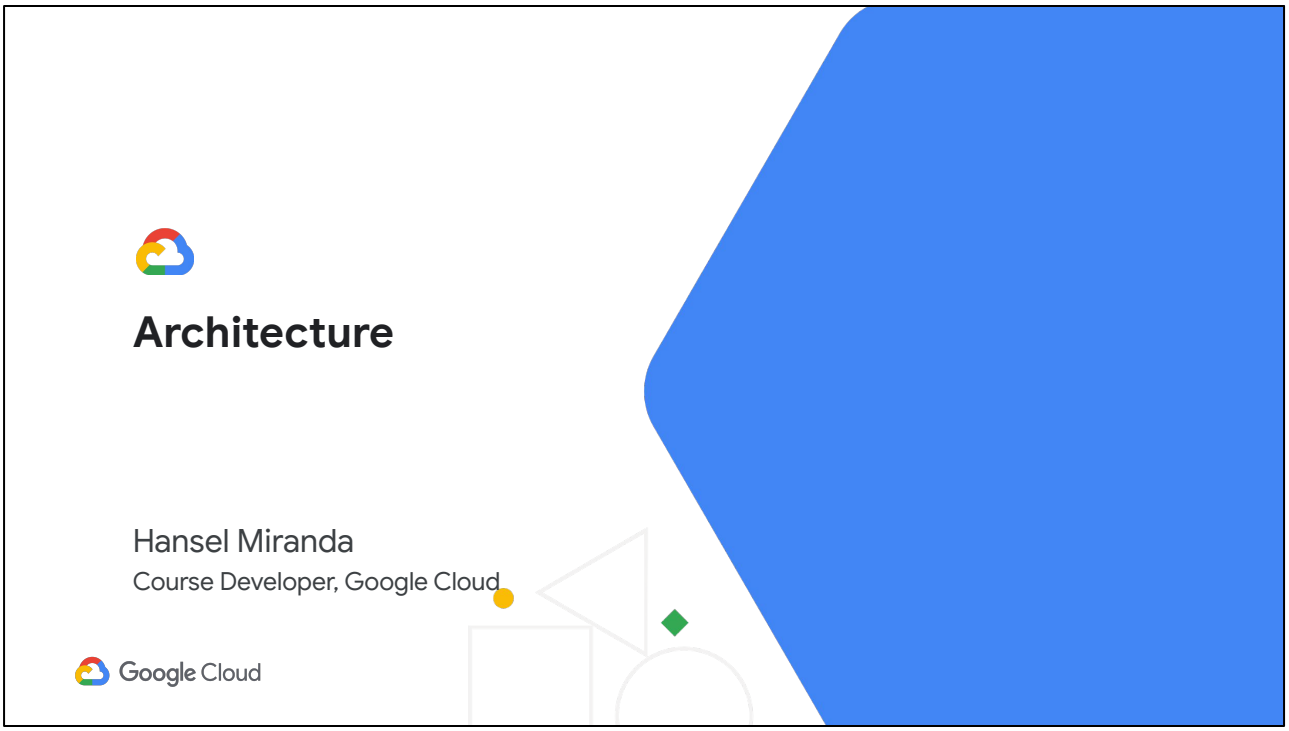


In this module, you will learn about hybrid terminology and core concepts like the Apigee Organization and Environment.

The Apigee hybrid architecture, its components, and their operation are discussed.

You will also learn about the hybrid management plane, the hybrid UI, and the management API, and how they are used to manage your hybrid platform.

You will learn about networking in Apigee hybrid and how virtual hosts are configured for environments.

Agenda

Terminology and Organizational Structure

Architecture

Networking

Apigee API

Google Cloud

In this lecture we will discuss the terminology and some of the core concepts used in Apigee.

We will also discuss organizational structure and the entities that are required in Apigee hybrid.

In later lectures, you will learn about the Apigee hybrid architecture, networking, and the management API.

Terms

- API: An interface that enables an application to consume capabilities or data from another application.

Google Cloud

An API is an interface that enables an application to consume capabilities or data from another application.

By defining stable entry points to application logic and data, APIs enable developers to easily access and reuse application logic built by other developers over the network.

An API also implies a contract. The contract provides some level of assurance that, over time, the API will change in a predictable manner.

Terms

- API: An interface that enables an application to consume capabilities or data from another application.
- API proxy: A bundle of XML configuration files and optional resources such as Javascript files. The bundle describes the request and response cycle for a given API and contains policies that implement security, mediation, transformation, enrichment, etc.

Google Cloud

In Apigee, an API proxy is a bundle of XML configuration files and optional resources that decouples the application-facing API from your backend service.

It enables the backend to undergo changes to the service, while allowing apps to continue to call the same API without any interruption.

Terms

- API: An interface that enables an application to consume capabilities or data from another application.
- API proxy: A bundle of XML configuration files and optional resources such as Javascript files. The bundle describes the request and response cycle for a given API and contains policies that implement security, mediation, transformation, enrichment, etc.
- Policy: A module that implements a specific management function as part of the proxy request/response flow.

Google Cloud

An Apigee policy is a module that implements a specific management function as part of the proxy in its request or response flow.

Policies enable you to add common types of capabilities to an API easily and reliably without the need to write code.

They provide features such as security, rate-limiting, transformation, and mediation.

You can control the behavior of your API by implementing policies in your API proxy.

Terms

- API: An interface that enables an application to consume capabilities or data from another application.
- API proxy: A bundle of XML configuration files and optional resources such as Javascript files. The bundle describes the request and response cycle for a given API and contains policies that implement security, mediation, transformation, enrichment, etc.
- Policy: A module that implements a specific management function as part of the proxy request/response flow.
- Virtual host: Configuration to route incoming API requests using one or more host aliases or domain names.

Google Cloud

In order for an API proxy to receive requests in Apigee hybrid, it must be deployed to an environment that is mapped to a virtual host.

A virtual host is a piece of configuration that allows hybrid to route API requests using multiple host aliases.

The host alias is typically the DNS domain name that maps to an IP address, and it is used as the endpoint by client applications that send requests to the API.

Terms

- API: An interface that enables an application to consume capabilities or data from another application.
- API proxy: A bundle of XML configuration files and optional resources such as Javascript files. The bundle describes the request and response cycle for a given API and contains policies that implement security, mediation, transformation, enrichment, etc.
- Policy: A module that implements a specific management function as part of the proxy request/response flow.
- Virtual host: Configuration to route incoming API requests using one or more host aliases or domain names.
- Target server: Configuration that decouples concrete endpoint URLs from TargetEndpoints. A TargetServer is referenced by name in a TargetEndpoint HTTPConnection.

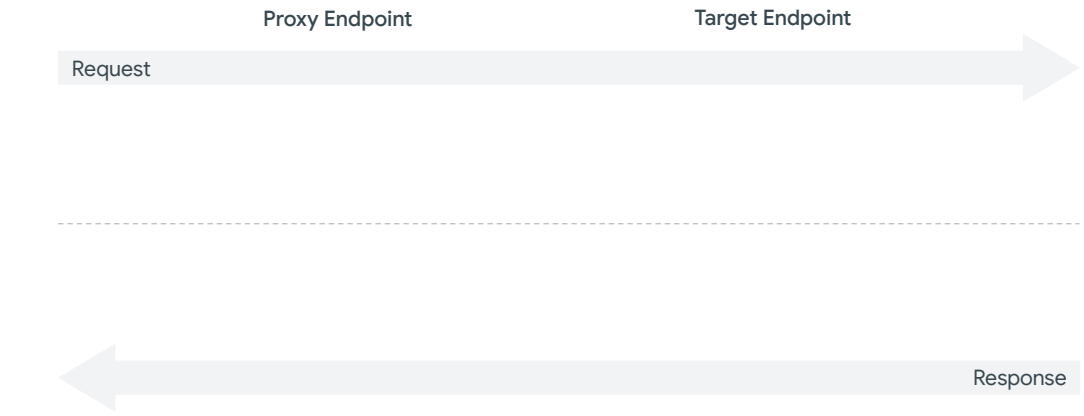
Google Cloud

A target server is a piece of configuration that abstracts the location of a backend resource out of an API proxy.

You create a target server configuration by providing its logical name, the hostname and port of the backend server, and optional TLS configuration.

The API proxy forwards requests from client applications to the target backend by referencing the name of the target server in the target endpoint configuration.

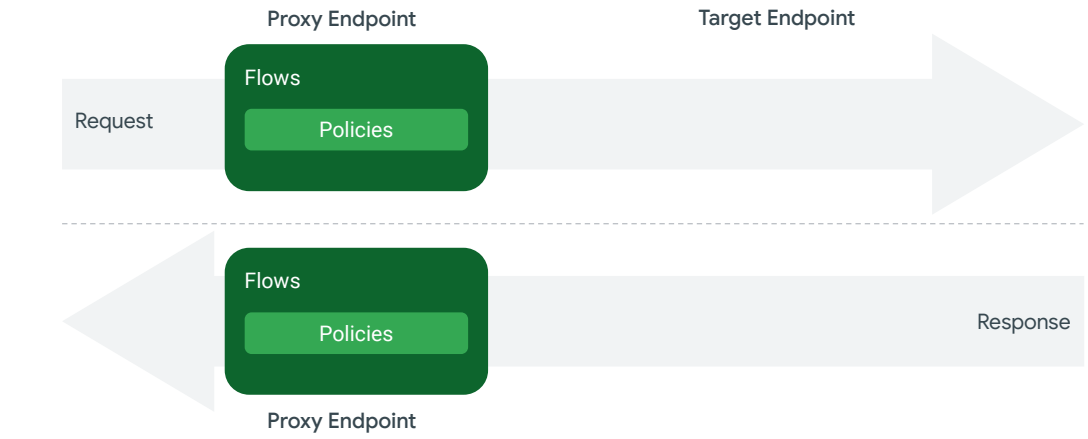
API Proxy: Request/Response cycle



Google Cloud

An API proxy configuration consists of 2 types of endpoints: ProxyEndpoint and TargetEndpoint.

API Proxy: Request/Response cycle



Google Cloud

The proxy endpoint defines the way client apps consume your APIs. You configure the ProxyEndpoint to define the URL of your API proxy and whether apps access the API over HTTP or HTTPS.

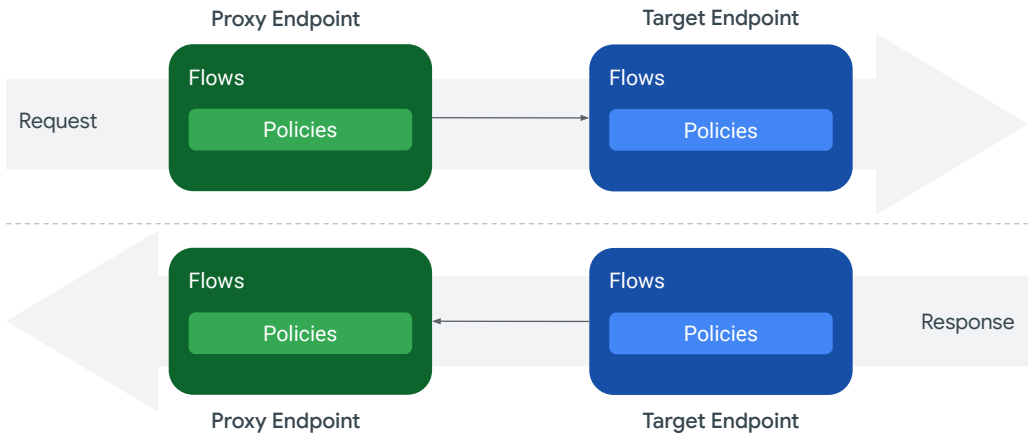
You usually attach policies to the ProxyEndpoint to enforce security, quota checks, rate-limiting, and access control.

Policies are placed in flows in the proxy endpoint. Flows are sequential stages along the API request and response processing path.

You build the logic of the API proxy by placing policies in one or more flows.

Some flows are conditional and execute when their condition evaluates to true at runtime.

API Proxy: Request/Response cycle



Google Cloud

The target endpoint defines the way the API proxy interacts with your backend services. You configure the TargetEndpoint to forward requests to the proper backend service, including any security settings, protocol, and other connection information.

You can attach policies to the TargetEndpoint to augment the request for the backend service, add authentication data, or rewrite the target URL if needed.

You can also reformat the response from the backend using policies in the target endpoint.

The same types of flows can be used in the proxy and target endpoints.

Policies

Traffic management			
Configure cache, control traffic quotas and spikes.			

Google Cloud

Apigee hybrid comes with a set of policies available for use in your API proxies. They are grouped into 4 categories.

The traffic management policies enable you to configure caching, control quotas, and mitigate the effects of traffic spikes to your API proxy.

Policies

Traffic management	Mediation		
			
Configure cache, control traffic quotas and spikes.	Transform request and response messages, parse and validate input payloads, and raise faults.		

Google Cloud

Using mediation policies, you perform message transformation, parsing, validation of the API request and response, and raise faults and alerts.

You can also access the metadata of various entities used by your API proxy, such as API products, developers, and apps.

Policies

Traffic management	Mediation	Security	
			
Configure cache, control traffic quotas and spikes.	Transform request and response messages, parse and validate input payloads, and raise faults.	Control access to your APIs with OAuth, validate API keys, and detect threats in request content.	

Google Cloud

With security policies, you can control access to your APIs with OAuth and API key validation, and implement threat protection features to detect malicious data in the request sent to the API.

If handling XML payloads, you can generate SAML tokens and verify SAML assertions in your API proxy.

There are policies to generate, decode, and verify signed JSON web signatures, and JSON web tokens, which can be used to secure access to your API proxy.

Policies

Traffic management	Mediation	Security	Extension
			
Configure cache, control traffic quotas and spikes.	Transform request and response messages, parse and validate input payloads, and raise faults.	Control access to your APIs with OAuth, validate API keys, and detect threats in request content.	Implement orchestration using service callouts, message logging, and custom functionality using Java, JavaScript, and Python code.

Google Cloud

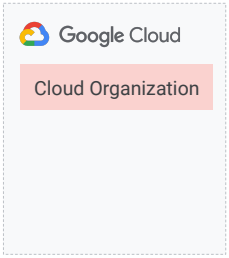
The extension group of policies lets you define custom functionality by allowing you to plug in code written in Java, JavaScript, and Python to your API proxy.

You can also log messages to a syslog server and orchestrate calls to other third-party endpoints from within your proxy.

Shared flows are common reusable sets of policies that can be used by multiple API proxies.

Using the FlowCallout policy in this group, you can invoke a shared flow from within your API proxy.

Google Cloud and Apigee hybrid entities



Organization: Root node in the Google Cloud resource hierarchy that provides central visibility and control over organization resources.

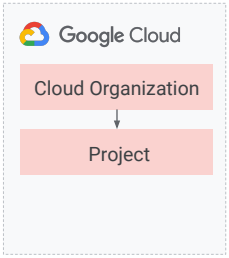
Google Cloud

You should be familiar with a few top-level entities when using Apigee hybrid.

At the top is the Google Cloud organization. This is the root node in the resource hierarchy that is used in Apigee hybrid.

It provides central visibility and control over all its child resources.

Google Cloud and Apigee hybrid entities



Organization: Root node in the Google Cloud resource hierarchy that provides central visibility and control over organization resources.

Project: Base-level organizing entity for resources like Cloud Storage buckets, Compute Engine instances, hybrid organization, and environments.

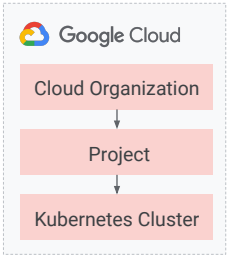
Google Cloud

Under the organization, a Google Cloud project is used.

The project is the base entity under which all cloud resources are organized.

You can create a project using the Google Cloud Console.

Google Cloud and Apigee hybrid entities



Organization: Root node in the Google Cloud resource hierarchy that provides central visibility and control over organization resources.

Project: Base-level organizing entity for resources like Cloud Storage buckets, Compute Engine instances, hybrid organization, and environments.

Cluster: Set of machines that run containerized applications and are managed by Kubernetes.

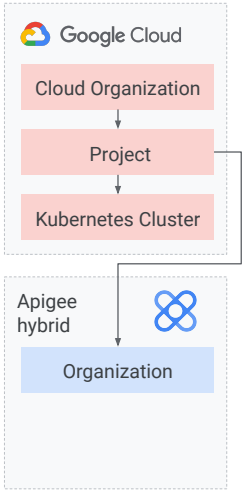
Google Cloud

The Apigee hybrid runtime plane resources are deployed in a cluster that runs on Kubernetes.

The cluster comprises a set of virtual machines that run the Apigee hybrid workloads and other system components.

It can be created using the Cloud Console or the gcloud command line interface and is associated to the Google Cloud project.

Google Cloud and Apigee hybrid entities



Organization: Root node in the Google Cloud resource hierarchy that provides central visibility and control over organization resources.

Project: Base-level organizing entity for resources like Cloud Storage buckets, Compute Engine instances, hybrid organization, and environments.

Cluster: Set of machines that run containerized applications and are managed by Kubernetes.

Apigee Organization: Boundary or scope of data that provides multi-tenancy and logical grouping of objects in Apigee. There is a 1:1 relationship with the Google Cloud project.

Google Cloud

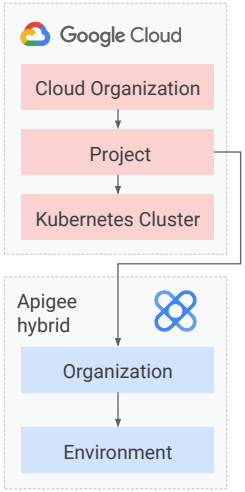
The hybrid organization is a resource under the Google Cloud project and is a prerequisite for installing and running Apigee hybrid.

It serves as a logical boundary of data and facilitates multi-tenancy within the Apigee system.

There is a 1-to-1 relationship between the hybrid organization and the cloud project.

You can create a hybrid organization using the Apigee API.

Google Cloud and Apigee hybrid entities



Organization: Root node in the Google Cloud resource hierarchy that provides central visibility and control over organization resources.

Project: Base-level organizing entity for resources like Cloud Storage buckets, Compute Engine instances, hybrid organization, and environments.

Cluster: Set of machines that run containerized applications and are managed by Kubernetes.

Apigee Organization: Boundary or scope of data that provides multi-tenancy and logical grouping of objects in Apigee. There is a 1:1 relationship with the Google Cloud project.

Environment: A runtime execution context for API proxies that provides development lifecycle and testing within the hybrid organization.

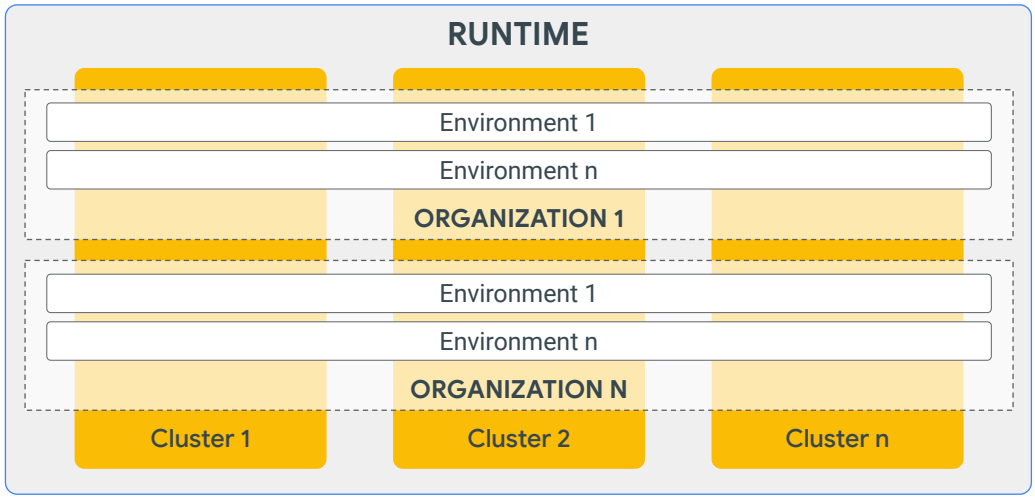
Google Cloud

An environment provides a runtime execution context for API proxies.

There can be more than one environment under an Apigee hybrid organization.

You can create an environment using the Apigee hybrid UI or Apigee API.

Multitenancy



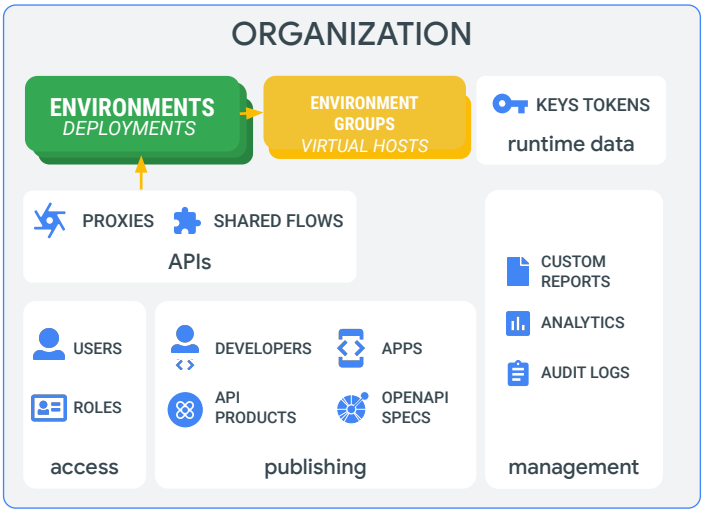
Google Cloud

More than one organization can be created in Apigee hybrid to support multi-tenancy. A hybrid runtime can be installed in Kubernetes clusters in different regions, with the organizations and environments spanning the installation.

You must consider the cost and availability of cluster resources when implementing multi-tenancy in your hybrid installation.

Organization

- An organization is the top-level entity in the Apigee object hierarchy.
- Customers can have multiple organizations.
- Assets can only be copied between organizations via management APIs, typically using a CI/CD process.
- Documentation and access to APIs are provided by developer portals. A developer portal is tied to exactly one organization.



Google Cloud

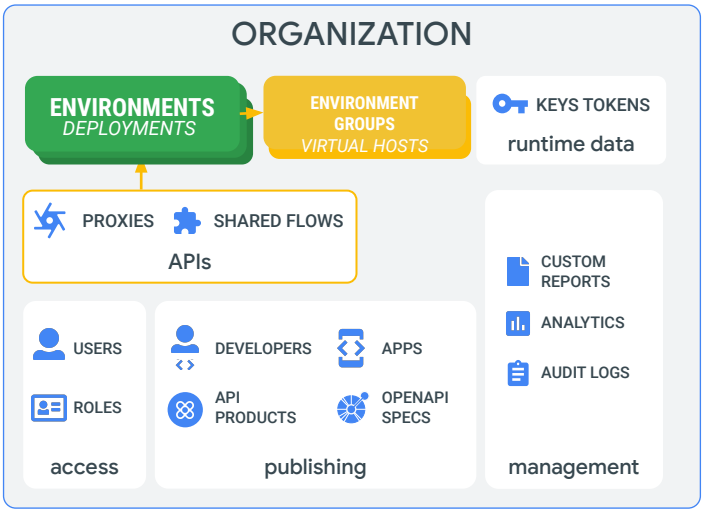
A hybrid organization is the top-level entity in Apigee hybrid. The hybrid organization resource is used to scope API entities that are used during the execution of the API proxy.

You can have more than one Apigee hybrid organization. A separate organization is usually used to manage API entities and proxies in production installations.

Data is not directly shared between organizations. You can, however, export and import data between organizations using the Apigee APIs.

APIs

- API proxies and shared flows are scoped at the organization level



Google Cloud

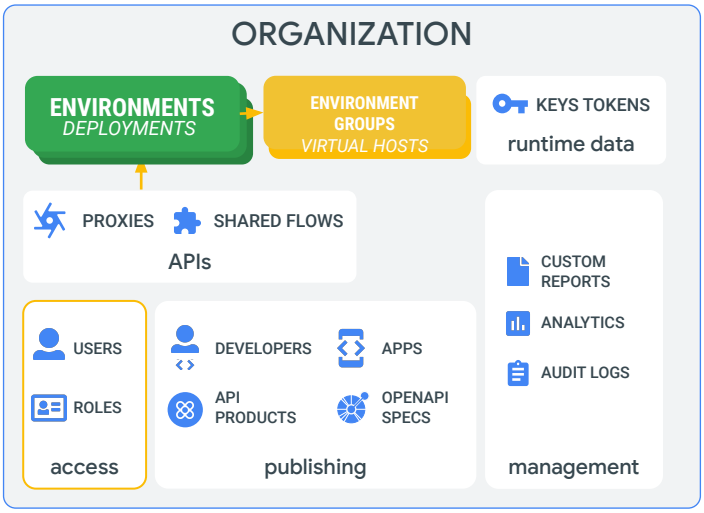
API proxies are defined at the organization level and are deployed to environments.

Shared flows are reusable sets of commonly used policies that can be used by more than one API proxy, and like API proxies, they are defined at the organization level.

A shared flow must be deployed to an environment before it can be used.

Access

- A user is granted access to the hybrid organization and its entities by assigning roles to the account in the Google Cloud project.
- User roles can be further refined at the hybrid environment level.



Google Cloud

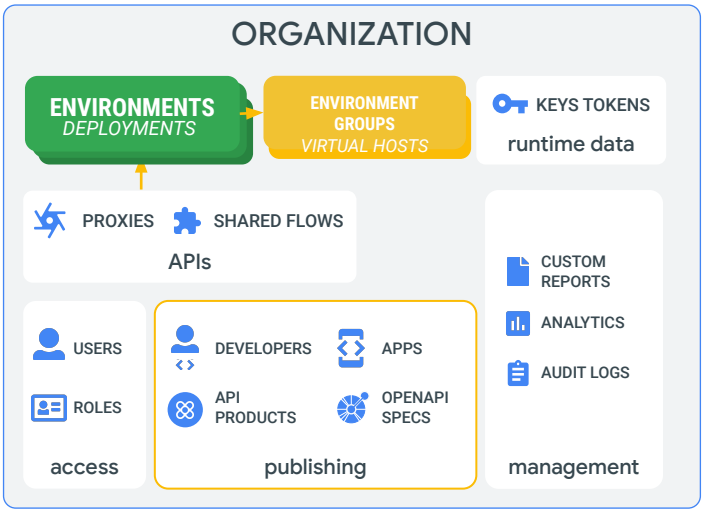
A user represents an authenticated account that can access a hybrid organization and the entities within that organization, such as the environments, API proxies, and keystores.

To add a new user to your Apigee organization, you grant access to the user's account first in the Google Cloud project by assigning desired roles to the account.

You can refine access for the user in individual hybrid environments by using the Apigee hybrid UI to assign roles.

Publishing

- Write OpenAPI specifications to document APIs.
- Create API products containing API proxies.
- Publish documentation and API products to the developer portal.
- Developers register apps to use the API products on the developer portal.



Google Cloud

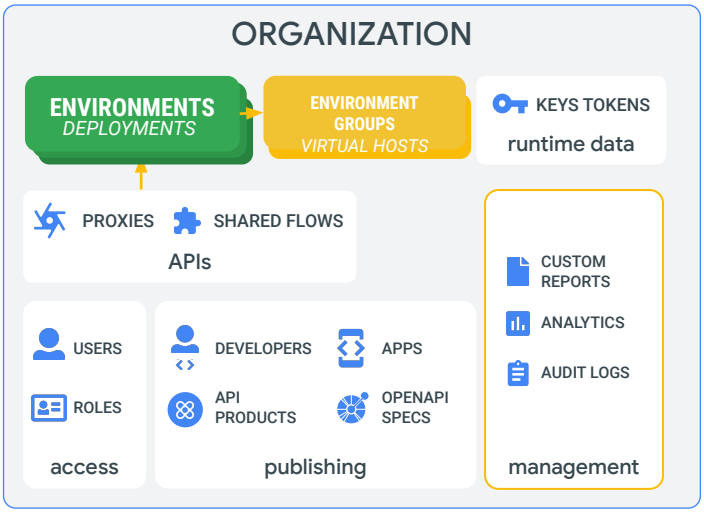
You package your API proxies into API products for consumption by app developers on the Apigee developer portal.

Using openAPI specifications, you create documentation for your APIs and publish it on the developer portal.

Developers register their apps by subscribing to one or more API products on the portal.

Management

- Analytics data is collected for your APIs and API products.
- View analytics dashboards in the hybrid UI or access the data via the Apigee API.
- Admin activity audit logs are generated for the project resources.



Google Cloud

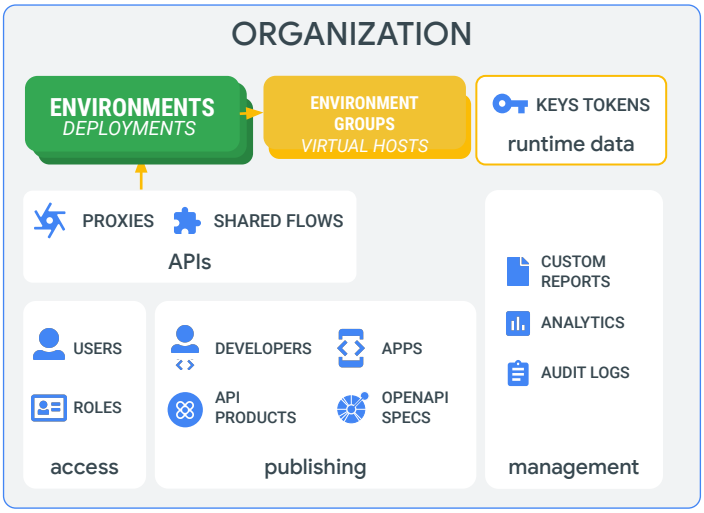
Analytics data is collected for all your APIs and API products and viewable in the Apigee hybrid UI.

Apigee hybrid writes **Admin Activity** audit logs, which record operations that modify the configuration or metadata of hybrid resources in the Google Cloud project.

The log entries are viewable in the Google Cloud Console.

Runtime data

- Apikeys and OAuth tokens are used to identify and authenticate apps invoking APIs.



Google Cloud

API keys and OAuth tokens are generated and linked to apps when developers register them to use an API product.

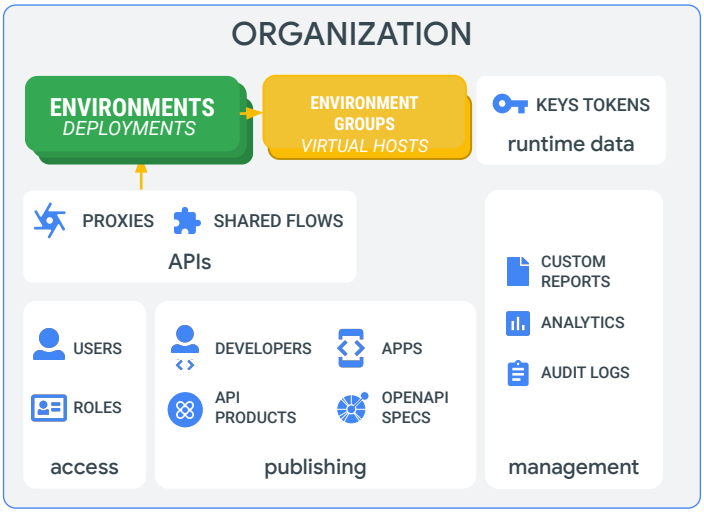
When an app presents a key or token in the API request, it can be identified and authenticated, which allows access to the API product and its proxies.

The API product specifies the environments in which it is available and can be used.

You can configure an API product with OAuth scopes when using OAuth, and use quota settings to rate-limit the number of requests made to the API proxies that make up the product.

Environment groups

- Environment groups allow you to group environments together.
- They are associated to virtual hosts in the hybrid runtime plane configuration.
- Environments within a group share the same hostaliases or domain names.



Google Cloud

In Apigee hybrid, you assign one or more environments to an environment group.

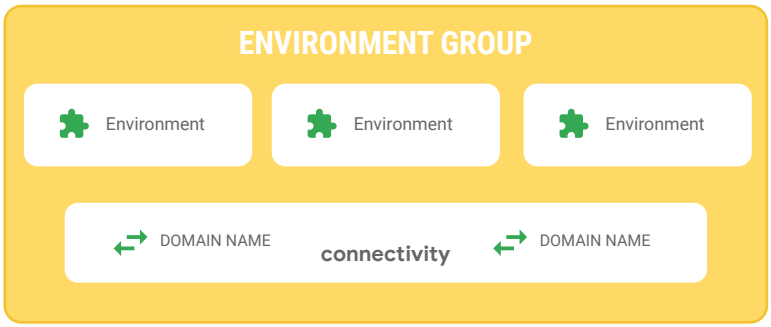
Environment groups allow you to group environments together, and provide the hostaliases or domain names for routing API traffic to the proxies deployed to the environments within the group.

You must create at least one environment group, and you must assign at least one hostalias or domain name to the group.

In the hybrid runtime plane configuration, a virtual host is associated to the name of an environment group.

Environment group

- Environments within an environment group share the same hostnames (domain names).
- You can group environments by function, by hostname address, by region, or by any other dimension.
- An environment group defines hostnames for routing requests to environments.



Google Cloud

All environments added to an environment group share the same hostaliases or domain names that are configured for that group.

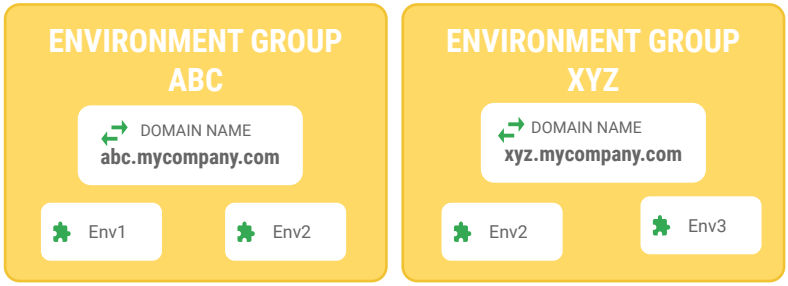
You can group environments by business function, by domain name, by region, or by any other dimension.

An environment group defines one or more host aliases or domain names that map to the environments in the group.

This allows Apigee hybrid to route incoming API requests to the correct runtime environment and API proxy that is deployed to that environment.

Environments and groups

- An environment can belong to multiple environment groups.
- Domain names must be unique to only one environment group.
- An API proxy that is deployed to an environment that belongs in multiple groups can be invoked using different domain names.



A proxy with basepath /foo that is deployed to environment Env2 can be invoked using basepaths:

abc.mycompany.com/foo
xyz.mycompany.com/foo

Google Cloud

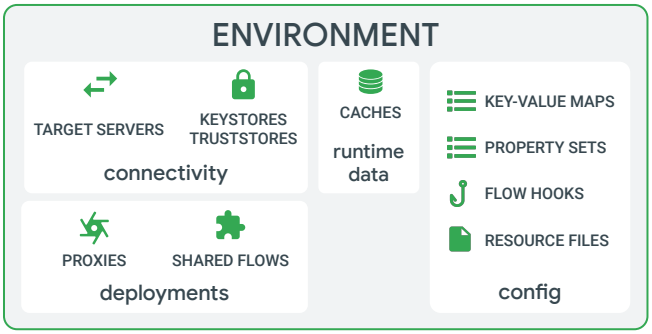
An Apigee hybrid environment can belong to multiple environment groups.

Domain names that are configured must be unique to only one environment group.

An API proxy that is deployed to an environment that belongs in multiple groups, can be invoked using different domain names.

Environment

- An environment is a runtime execution context for API proxies.
- A proxy revision must be deployed to an environment before the proxy can receive runtime traffic.
- Deploying different revisions of a proxy to different environments provides the ability to support the API lifecycle, with proxies moving from development through testing and into production.



Google Cloud

An environment provides a runtime execution context for API proxies.

A proxy must be deployed to an environment in order to process API requests.

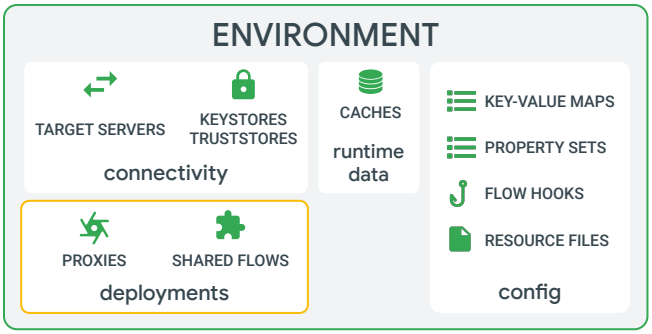
You can have multiple environments in an organization in Apigee hybrid. API proxies with related functionality can be deployed to the same environment.

Environments are also used to support the API development lifecycle.

A proxy revision can be promoted from a development environment through the testing environment and eventually into production using a CI/CD process.

Deployments

- Proxy and shared flow revisions are deployed to an environment.



Google Cloud

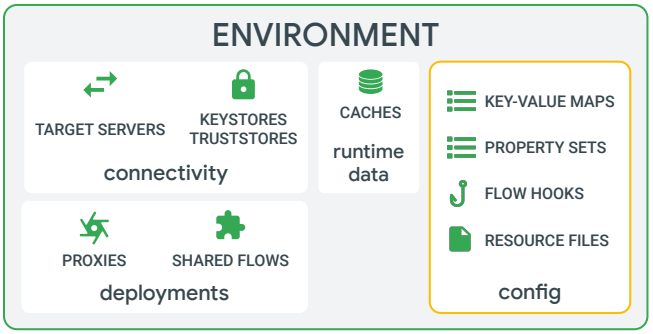
A proxy revision is deployed to an environment.

The proxy can then receive and process API requests.

A shared flow revision must also be deployed to an environment. It is then available for use by any proxy that is deployed to that same environment.

Config

- Environment scoped configuration objects include KVMs, flow hooks, resources.
- Property sets are environment scoped collections of name-value pairs.



Google Cloud

Each environment in Apigee hybrid can have its own set of key-value maps, which are collections of name/value string pairs. This data can be used by API proxies in that environment at runtime.

The KVM can be created by using the hybrid UI or API, and its data can be populated using the KeyValueMapOperations policy.

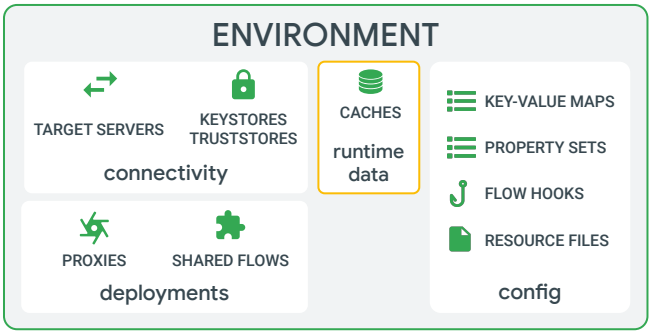
KVMs can be encrypted to store passwords or other sensitive information.

Property sets are another way to store name/value string pairs. They can be created and populated via the Apigee hybrid UI or API and stored in the management plane. They are downloaded to the runtime plane for use by the API proxies.

Other environment-scoped configurations include flow hooks that execute shared flows and resources like Javascript files.

Runtime data

- Caches can be used to cache API response and other data.
- Improve your API response times and throughput by using caches.



Google Cloud

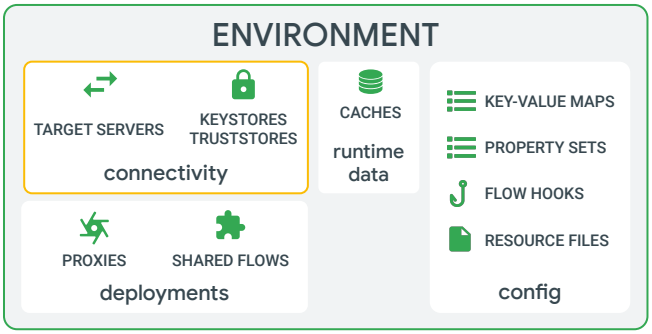
Caches are used to store API responses or other data for use by the API proxy at runtime.

They can eliminate unnecessary requests to backends or third-party services by the API proxy, thus resulting in improved latency and throughput and better scalability.

Cached data should not be shared between proxies in different environments, so they are only scoped at the environment level.

Connectivity

- Target servers configure backend server hostnames and ports.
- Keystores and truststores store private keys and certificate chains for TLS.

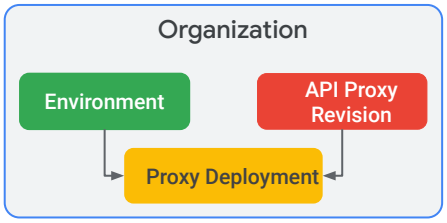


Google Cloud

Target servers are used to decouple backend URL configuration from the API proxy code. This allows the proxy to connect to environment-specific backends without modifying the proxy.

Keystores and truststores store private keys and certificates to support communication over TLS between client apps and Apigee and between Apigee and backend servers.

Environments, API proxies, and deployments



- API proxies process API traffic and are created or imported into an organization.
- API proxy revisions must be deployed into environments in order to process traffic.
- A proxy deployment is an association between an API proxy revision and an environment.
- An API proxy can be deployed in one or more environments.

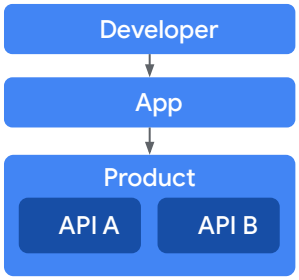
Google Cloud

When you make a change to an API proxy, you can deploy it as a new revision to a specific environment in your cluster.

A proxy deployment is an association between an API proxy revision and an environment.

Products, apps, and developers

- Products bundle together API proxies to offer them to application developers.
- Developers are external API users who interact with your APIs by developing apps that use them.
- Apps are associations between developers and products.



Google Cloud

API products bundle a set of API proxies with optional quotas and other metadata and offer them to app developers.

App developers register their apps to use one or more API products on the Apigee developer portal.

Each registered app receives an apikey that associates the app to the set of API products.

Apigee hybrid enforces access control to the API proxies in the product when the app provides the apikey in the request to the API.

Agenda

Terminology and Organizational Structure

Architecture

Networking

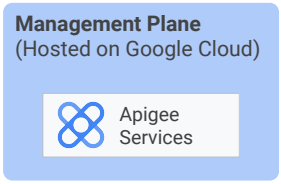
Apigee API

Google Cloud

In this lecture, we will discuss the architecture of Apigee hybrid and the various components that are used to operate the API management platform.

Apigee hybrid

Apigee-run management plane:
A set of services hosted on Google Cloud that includes the UI, Apigee API, and analytics.



Google Cloud

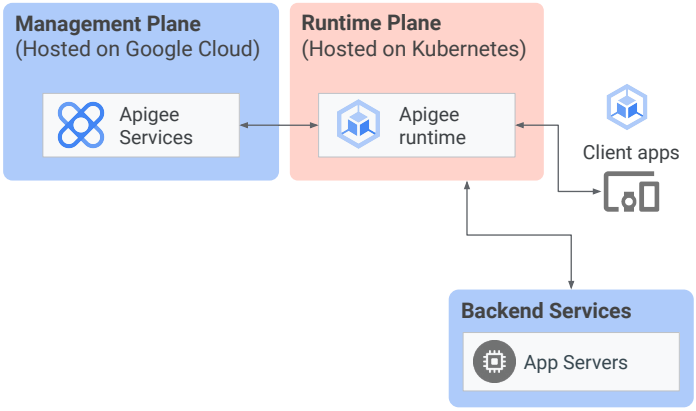
Apigee hybrid consists of a control or management plane that is hosted on Google Cloud and maintained by Google.

The management plane consists of the Apigee hybrid UI, the Apigee management API, and analytics services.

Apigee hybrid

Apigee-run management plane: A set of services hosted on Google Cloud that includes the UI, Apigee API, and analytics.

Customer-managed runtime plane: A set of containerized services that is set up and maintained in your own Kubernetes cluster. All API traffic is processed within the runtime plane.



Google Cloud

Apigee hybrid also consists of a runtime plane that is installed and maintained by the customer.

The runtime plane is installed in a cluster running on a Kubernetes platform.

All API traffic passes through the runtime plane and is processed within this plane.

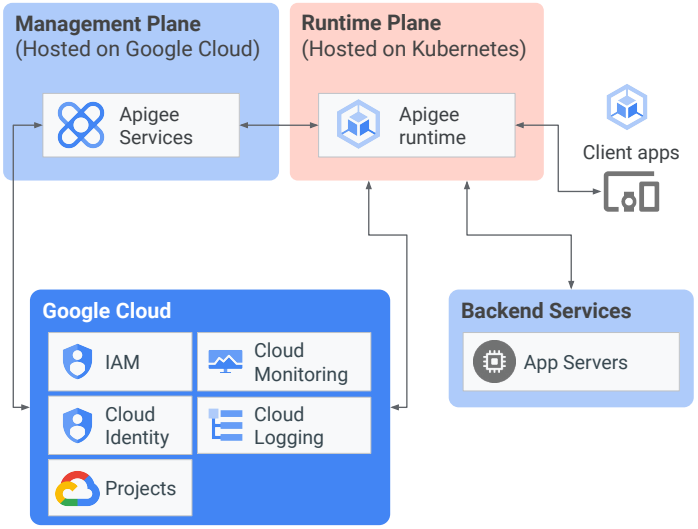
The runtime plane components are containerized so you can achieve auto-scaling and other operational benefits of containers.

Apigee hybrid

Apigee-run management plane: A set of services hosted on Google Cloud that includes the UI, Apigee API, and analytics.

Customer-managed runtime plane: A set of containerized services that is set up and maintained in your own Kubernetes cluster. All API traffic is processed within the runtime plane.

Google Cloud: A suite of cloud services hosted by Google.



Google Cloud

The management plane interacts with other Google APIs and services running in Google Cloud, such as Identity and Access Management.

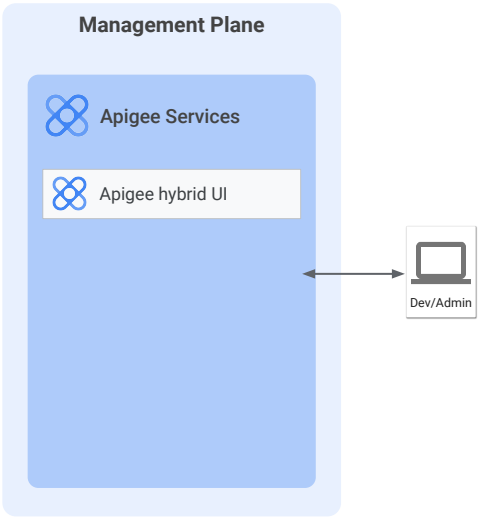
The runtime plane also interacts with Cloud Logging and Cloud Monitoring services.

You can view runtime component logs and monitor the health of your runtime cluster using these Cloud services.

Management plane

The management plane runs on Google Cloud and includes these primary services:

- Apigee hybrid UI: A UI to create and deploy API proxies, configure policies, and create API products and applications.



Google Cloud

The Apigee hybrid user interface is a service that runs in the management plane.

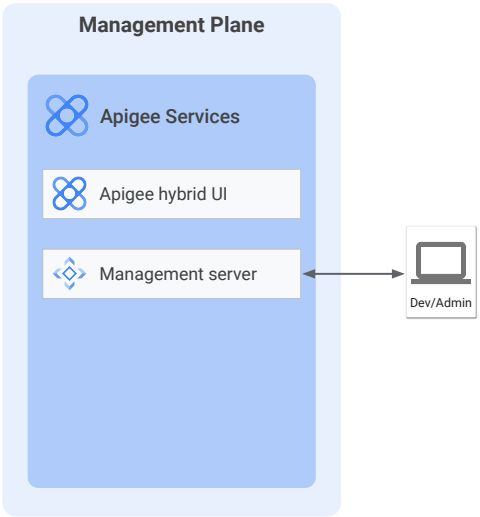
It enables API developers to create and deploy API proxies, configure policies, create API products, and create developer apps.

Administrators use the Apigee hybrid UI to monitor proxy deployment status and create and manage environments and other configuration objects.

Management plane

The management plane runs on Google Cloud and includes these primary services:

- Apigee hybrid UI: A UI to create and deploy API proxies, configure policies, and create API products and applications.
- Management Server: A programmatic interface to Apigee APIs to manage your hybrid organization and environments, deploy API proxies, etc.



Google Cloud

The management server provides a set of Apigee APIs.

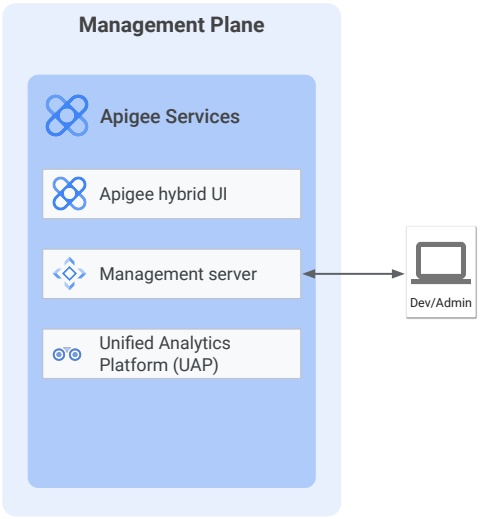
These APIs provide a programmatic interface to manage your hybrid organization, environments, and other resources, such as proxy deployments, apps, and API products.

You can use these APIs in CI/CD workflows to automate hybrid management tasks.

Management plane

The management plane runs on Google Cloud and includes these primary services:

- Apigee hybrid UI: A UI to create and deploy API proxies, configure policies, and create API products and applications.
- Management Server: A programmatic interface to Apigee APIs to manage your hybrid organization and environments, deploy API proxies, etc.
- Unified Analytics Platform (UAP): A platform that receives and processes analytics and deployment status data from the runtime plane.



Google Cloud

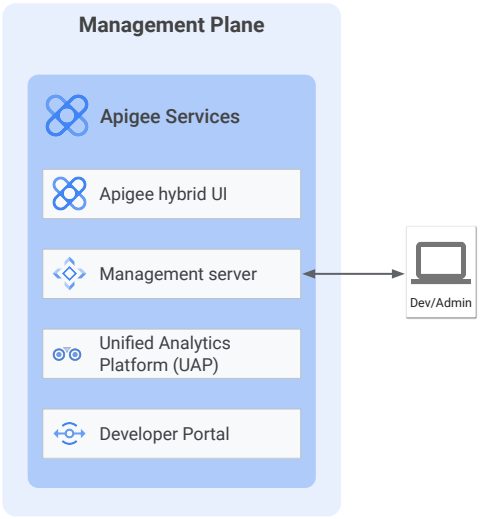
The unified analytics platform receives and processes analytics and deployment status data from the runtime plane.

It makes this data available to the Apigee hybrid UI and API for developers and administrators to consume.

Management plane

The management plane runs on Google Cloud and includes these primary services:

- Apigee hybrid UI: A UI to create and deploy API proxies, configure policies, and create API products and applications.
- Management Server: A programmatic interface to Apigee APIs to manage your hybrid organization and environments, deploy API proxies, etc.
- Unified Analytics Platform (UAP): A platform that receives and processes analytics and deployment status data from the runtime plane.
- Integrated Developer Portal: A lightweight application developer portal that provides standard registration and app creation flows.



Google Cloud

The integrated developer portal is a simple self-service portal development tool available in the Apigee hybrid UI.

It enables API developers and administrators to quickly create a developer portal and publish API products and their API documentation.

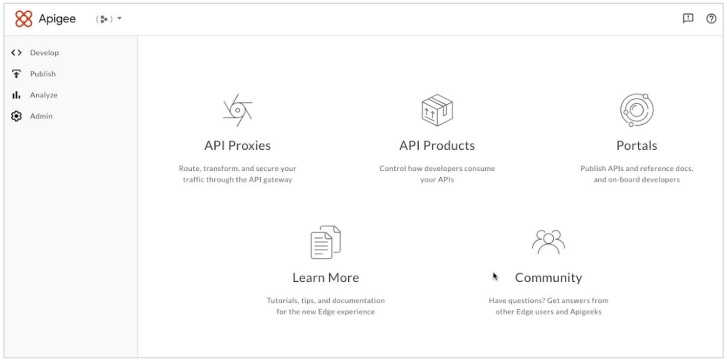
Application developers can browse the API catalog, read the documentation, and self-register their apps to consume the API products on the developer portal.

Hybrid UI

A UI interface hosted by Apigee in the management plane.

Used for:

- API development tasks, such as creating and editing API proxies and shared flows.
- Environment, environment group and configuration data management.



apigee.google.com

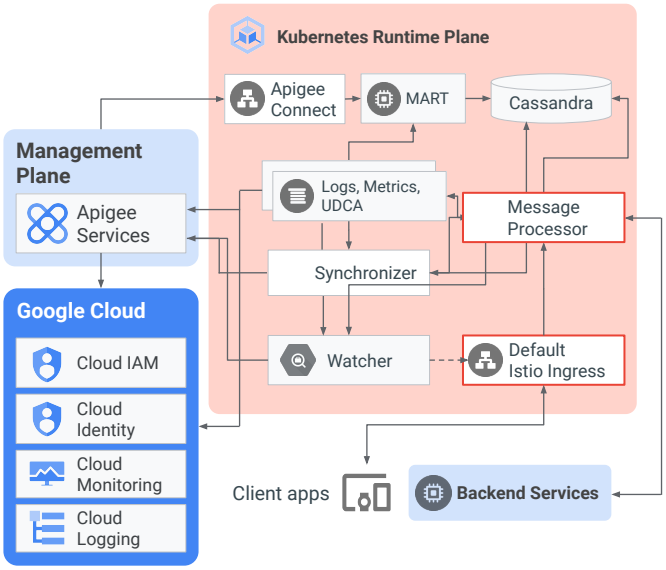
Google Cloud

API developers and administrators can perform various tasks using the Apigee hybrid UI, based on their role and access permissions:

- You can manage environments and environment groups using the hybrid UI.
- To create, update, and delete environment resources, you must also update the hybrid runtime plane configuration separately.
- You can refine user access to each environment by assigning user roles that are different from what is granted at the project level.
- You can deploy or undeploy proxy revisions to and from the hybrid environment and view deployment status of the proxy.
- You can create trace sessions to debug your API proxy and download trace session data for viewing offline.
- You can create and manage other objects, such as target servers, key-value maps, and shared flows, for use by your API proxies.

Runtime plane

- The runtime plane is a set of containerized runtime services that you set up and maintain in your own Kubernetes cluster.
- All API traffic from client apps passes through and is processed within the runtime plane.
- The runtime plane includes the following major components:
  - Message Processors (MP)
  - Default Istio Ingress (runtime traffic)



Google Cloud

The runtime plane runs in a Kubernetes cluster that you maintain and deploy on a supported Kubernetes platform.

The primary services that execute on the runtime plane are:

Message Processors:

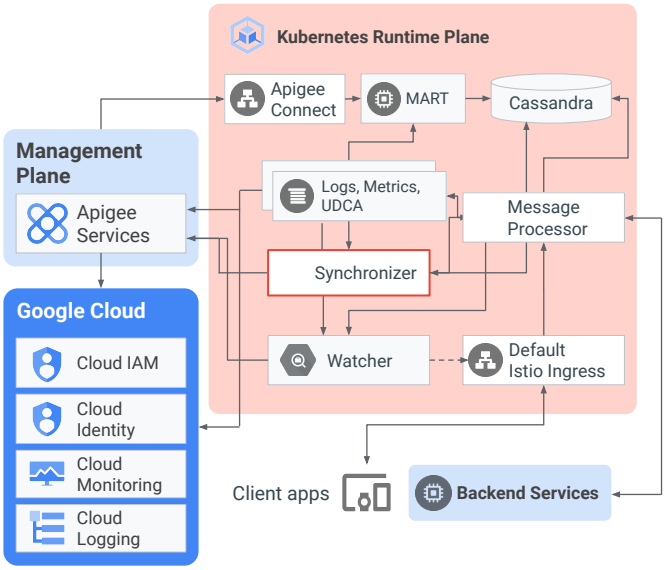
- Message Processors process API traffic from client applications and integrate with backend services.

An Istio Ingress gateway is used to route API requests that come from outside the cluster to the runtime message processor pods.

It uses a combination of the domain name and base path to map the request to the correct runtime pod.

Runtime plane

- The runtime plane is a set of containerized runtime services that you set up and maintain in your own Kubernetes cluster.
- All API traffic from client apps passes through and is processed within the runtime plane.
- The runtime plane includes the following major components:
  - Message Processors (MP)
  - Default Istio Ingress (runtime traffic)
  - Synchronizer



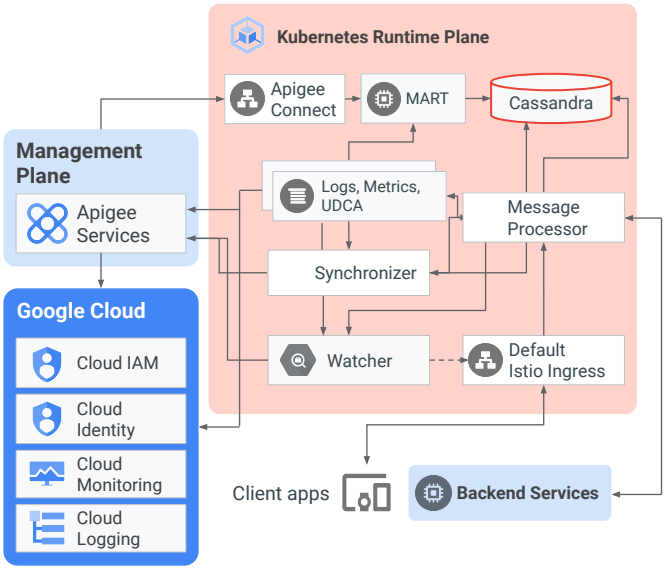
Google Cloud

Synchronizer:

- The synchronizer fetches configuration data about an API environment from the management plane and propagates it across the runtime plane.

Runtime plane

- The runtime plane is a set of containerized runtime services that you set up and maintain in your own Kubernetes cluster.
- All API traffic from client apps passes through and is processed within the runtime plane.
- The runtime plane includes the following major components:
  - Message Processors (MP)
  - Default Istio Ingress (runtime traffic)
  - Synchronizer
  - Cassandra



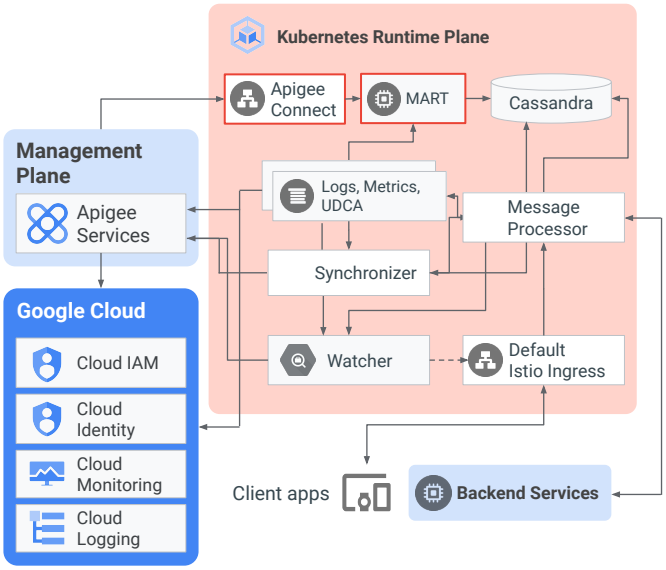
Google Cloud

Cassandra:

- Cassandra is the hybrid runtime plane datastore that provides data persistence in the runtime plane.

Runtime plane

- The runtime plane is a set of containerized runtime services that you set up and maintain in your own Kubernetes cluster.
- All API traffic from client apps passes through and is processed within the runtime plane.
- The runtime plane includes the following major components:
  - Message Processors (MP)
  - Default Istio Ingress (runtime traffic)
  - Synchronizer
  - Cassandra
  - MART
  - Apigee Connect (management traffic)



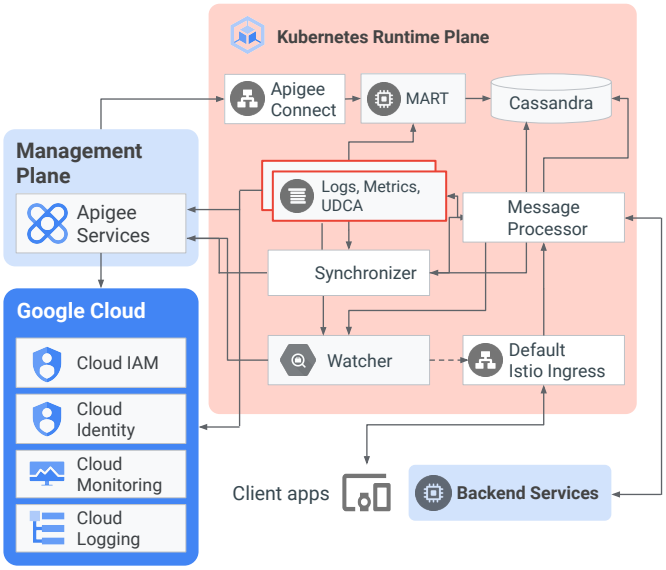
Google Cloud

The management API for runtime data, or MART, is a service used by the management plane to access and update entities in the runtime plane datastore.

The Apigee Connect component allows the hybrid management plane to connect securely to the MART service in the runtime plane, without requiring you to expose the MART endpoint outside the cluster.

Runtime plane

- The runtime plane is a set of containerized runtime services that you set up and maintain in your own Kubernetes cluster.
- All API traffic from client apps passes through and is processed within the runtime plane.
- The runtime plane includes the following major components:
  - Message Processors (MP)
  - Default Istio Ingress (runtime traffic)
  - Synchronizer
  - Cassandra
  - MART
  - Apigee Connect (management traffic)
  - UDCA



Google Cloud

The Universal Data Collection Agent is a service running within the data collection pod in the runtime plane.

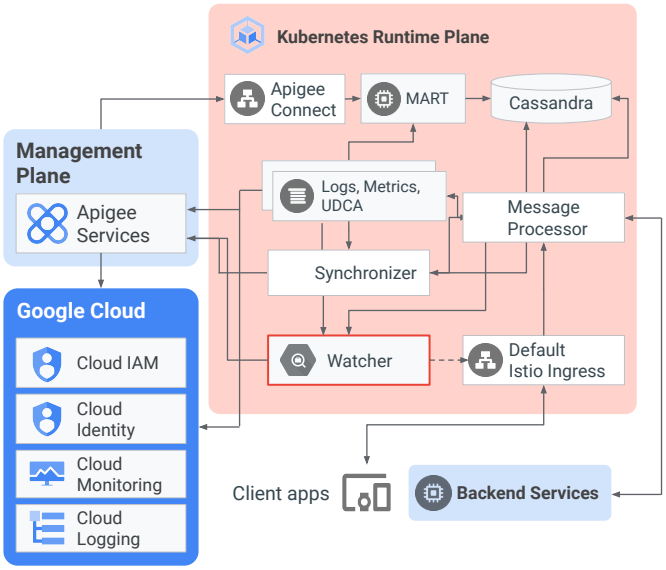
It receives analytics and trace data from the Message Processors and sends it to the Unified Analytics Platform in the management plane.

Log information is collected from the runtime components and sent to Cloud Logging.

All runtime components generate metrics that are collected and sent to Cloud Monitoring.

Runtime plane

- The runtime plane is a set of containerized runtime services that you set up and maintain in your own Kubernetes cluster.
- All API traffic from client apps passes through and is processed within the runtime plane.
- The runtime plane includes the following major components:
  - Message Processors (MP)
  - Default Istio Ingress (runtime traffic)
  - Synchronizer
  - Cassandra
  - MART
  - Apigee Connect (management traffic)
  - UDCA
  - Watcher



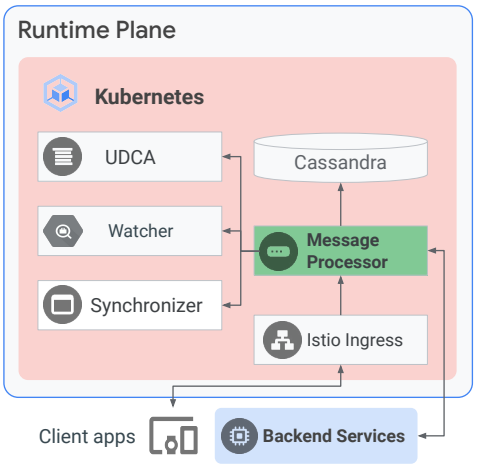
Google Cloud

The Apigee Watcher component retrieves virtual host changes for an organization and propagates these changes to the Istio ingress gateway running in the cluster.

Message Processor

The Message Processor runtime component:

- Provides API request processing and policy execution in the runtime plane.
- Loads proxies, resources, target servers, and other configuration from Synchronizer and runtime data from Cassandra.
- Is exposed to requests that come from outside the cluster by an Istio Ingress controller.
- Is configured as the runtime resource and implemented as an ApigeeEnvironment (CRD) in the Kubernetes cluster.
- Is scoped to a single Apigee environment.



Google Cloud

Hybrid message processors provide API request processing and policy execution in the runtime plane.

Message processors load proxies, shared flows and other API resources from the synchronizer component.

Other entities, such as API keys and tokens, are accessed from the runtime data store.

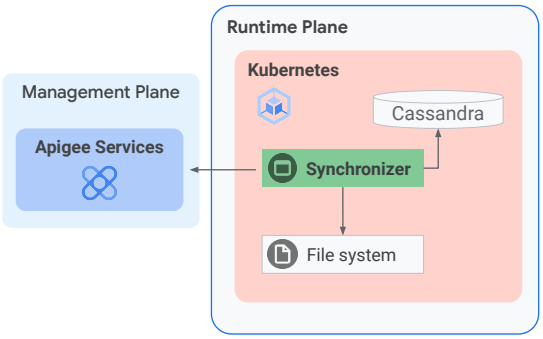
A message processor deployment is scoped to a single Apigee environment.

An Istio ingress controller routes API requests from outside the cluster to the message processors.

Synchronizer

The Synchronizer runtime component:

- Fetches configuration data (contract) from the management plane and propagates it across the runtime plane.
- Periodically polls the Management Server for configuration changes and downloads those changes to the local file system and to Cassandra.
- Stores the downloaded contract as a JSON file in a location where the runtime MPs can access it.
- Is implemented as an ApigeeEnvironment (CRD) in the Kubernetes runtime cluster and scoped to the environment.



Google Cloud

The Synchronizer runtime component fetches configuration data about an API environment from the management plane and propagates it across the runtime plane.

It periodically polls the management server for changes and downloads a new configuration whenever changes are detected.

The configuration data is retrieved and stored locally on the file system and in Cassandra, where the Message Processors accesses it.

The downloaded configuration data allows the runtime plane to function independently from the management plane.

If the connection between the management and runtime plane fails, services on the runtime plane continue to function.

The configuration data downloaded by the Synchronizer includes Proxy bundles, shared flows, API resources, target server, and other configuration needed by the API proxies in the runtime plane.

The configuration data downloaded by the Synchronizer includes:

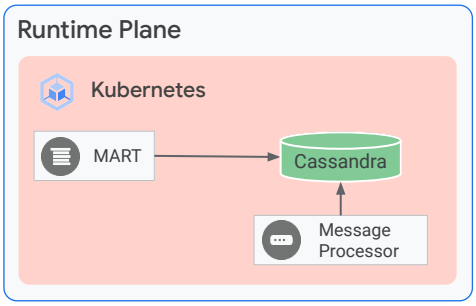
- Proxy bundles and shared flows
- Flow hooks
- Environment information
- Shared API resources
- Target server definitions

- Key Value Map (KVM) names
- Data masks

Cassandra

The Cassandra runtime component:

- Is the datastore that provides Core Persistence Services (CPS) for the hybrid runtime plane.
- Is deployed as a StatefulSet on a ring of nodes (nodepool) in the cluster and scoped to an organization (one or more).
- Uses persistent volumes to store data, with the size of the volume defined during installation.
- Stores the following entities in the datastore:
  - Key management system (KMS)
  - Key Value Map (KVM)
  - Response cache
  - OAuth data
  - Quotas



Google Cloud

Apache Cassandra is the runtime datastore that provides data persistence for the runtime plane.

It is deployed as a StatefulSet on a node pool in your Kubernetes cluster.

The Cassandra database stores information about the following entities:

- Key management system (KMS) data, such as developers, apps, API keys, and API products
- Key Value Map (KVM) data
- Response cache
- OAuth tokens
- Quotas

Locating these entities close to the runtime processing services helps support security and scalability requirements.

Apigee hybrid uses dynamically created persistent volumes to store data in the Cassandra database.

Cassandra pods bind to the existing persistent volumes on restart.

MART (Management API for RunTime)

The MART runtime component:

- Is an API server that provides access to configuration, KMS, caches, and KVMs that are stored in the runtime plane datastore.
- Authenticates requests from the hybrid management plane using OAuth tokens.
- Cannot be invoked directly by end users.
- Is implemented as an ApigeeOrganization (CRD) in Kubernetes, and scoped to a single organization.

Google Cloud

Data that is used by your API proxies at runtime is stored in the Cassandra data store.

This includes data on apps, app developers, API products, and API keys and data stored in key-value maps, property sets, etc.

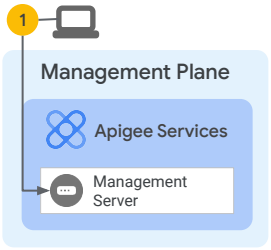
You use the Apigee hybrid UI or API to create and update this data.

These management operations use API calls to the MART service that process this data in the runtime datastore.

Let’s review the sequence of a MART operation.

MART operation

1. A developer or admin calls an Apigee API on the management server (MS).

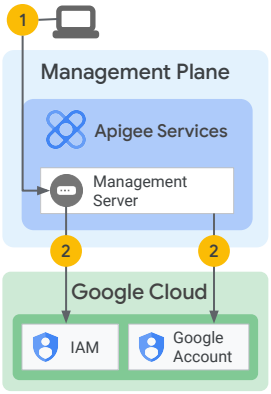


Google Cloud

An API developer or administrator calls an Apigee management API or uses the hybrid UI to process runtime data.

MART operation

1. A developer or admin calls an Apigee API on the management server (MS).
2. The MS authenticates and authorizes the caller using Cloud IAM. It then generates an OAuth token using a trusted service account.

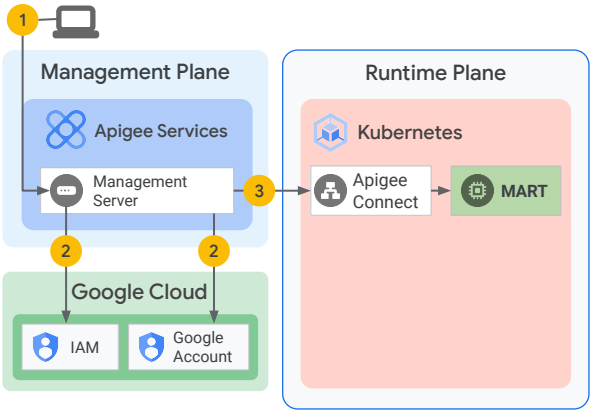


Google Cloud

Using Google's IAM services, the management server authenticates and authorizes the caller. If the caller is authenticated, the management server uses a pre-configured trusted service account to generate an OAuth token.

MART operation

1. A developer or admin calls an Apigee API on the management server (MS).
2. The MS authenticates and authorizes the caller using Cloud IAM. It then generates an OAuth token using a trusted service account.
3. The MS calls the MART server via Apigee Connect with this access token.

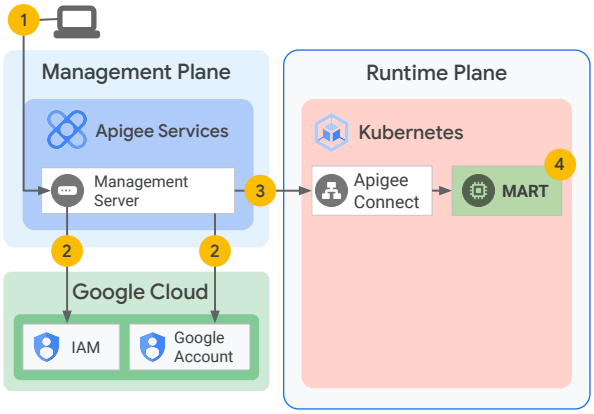


Google Cloud

It then makes a request to the MART server via Apigee Connect and includes the OAuth token in the request.

MART operation

1. A developer or admin calls an Apigee API on the management server (MS).
2. The MS authenticates and authorizes the caller using Cloud IAM. It then generates an OAuth token using a trusted service account.
3. The MS calls the MART server via Apigee Connect with this access token.
4. MART validates the token and processes the request.

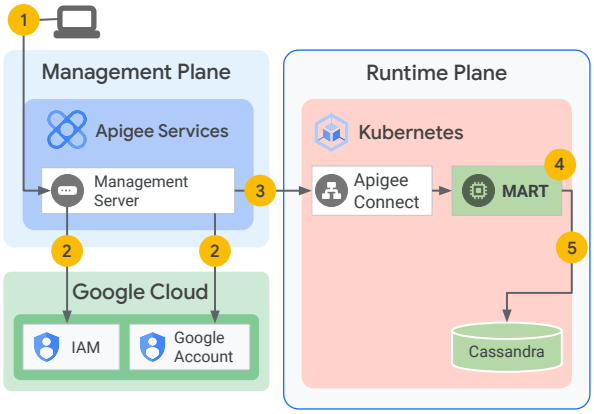


Google Cloud

MART receives the request, authenticates and authorizes it, and then validates the request.

MART operation

1. A developer or admin calls an Apigee API on the management server (MS).
2. The MS authenticates and authorizes the caller using Cloud IAM. It then generates an OAuth token using a trusted service account.
3. The MS calls the MART server via Apigee Connect with this access token.
4. MART validates the token and processes the request.
5. To retrieve or update runtime data, MART connects to the Cassandra datastore.

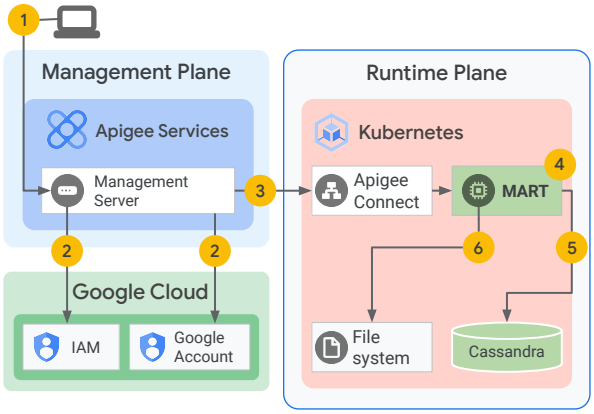


Google Cloud

If the request is valid, MART processes the request by reading or updating the Cassandra runtime datastore.

MART operation

1. A developer or admin calls an Apigee API on the management server (MS).
2. The MS authenticates and authorizes the caller using Cloud IAM. It then generates an OAuth token using a trusted service account.
3. The MS calls the MART server via Apigee Connect with this access token.
4. MART validates the token and processes the request.
5. To retrieve or update runtime data, MART connects to the Cassandra datastore.
6. MART logs its activity to the host file system.



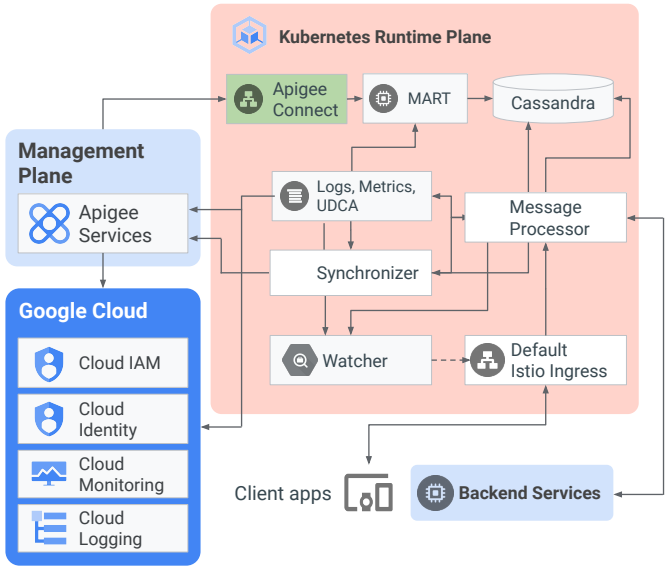
Google Cloud

MART then logs its activity on the host file system.

Apigee Connect

The Apigee Connect runtime component:

- Is an agent installed in the runtime plane.
- Enables the hybrid management plane to securely connect to the MART service in the runtime plane.
- Prevents the MART endpoint from being exposed on the internet.
- Is enabled by default in the runtime plane.



Google Cloud

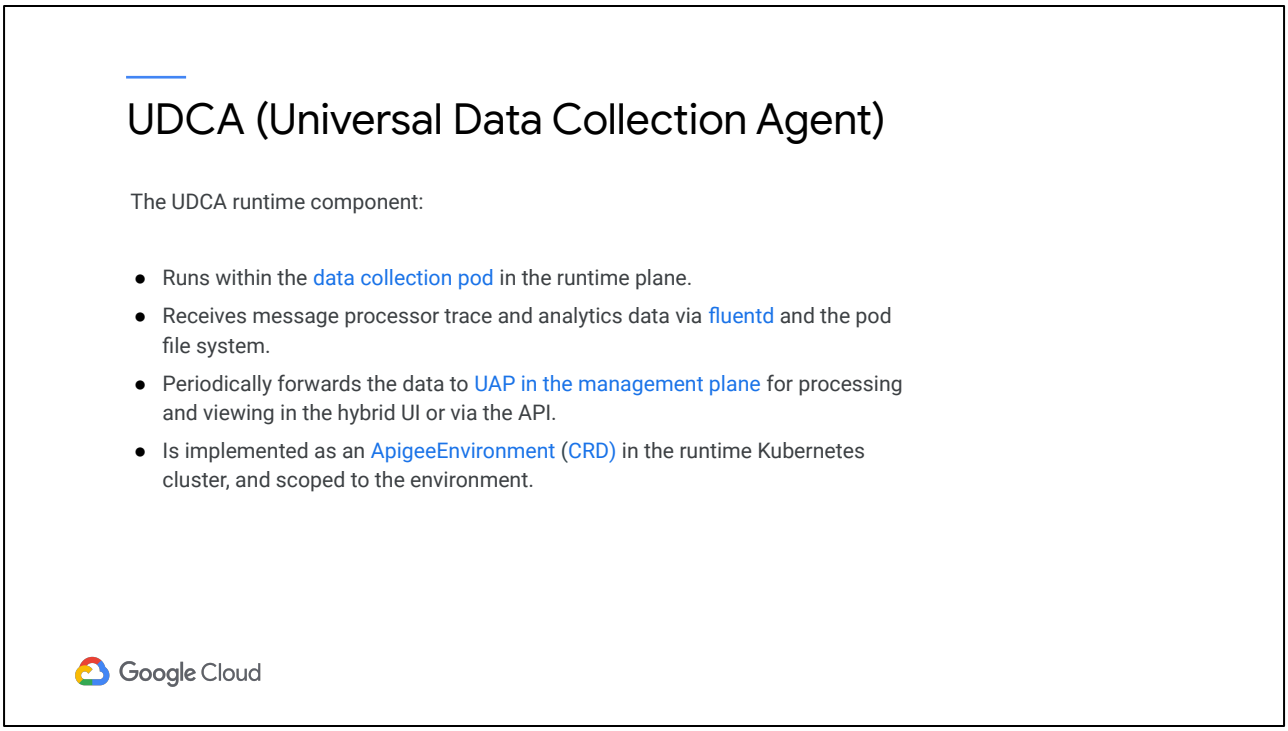
Apigee Connect allows the hybrid management plane to securely connect to the MART service in the runtime plane without requiring you to expose the MART endpoint outside the cluster.

Apigee Connect is a gRPC service that registers with the management plane on startup.

When a hybrid UI or API call that processes runtime API data is made, a management plane service selects an active connection and makes a call to Apigee Connect.

Apigee Connect uses persistent connections. The connection remains open, and the client—in this case Apigee Connect—is responsible for re-establishing it if it fails.

To use Apigee Connect, you must enable the Apigee Connect API in the Google Cloud API library and configure a service account with the Apigee Connect Agent role as part of your cluster setup.



The Universal Data Collection Agent is a service that runs within the data collection pod in the runtime plane.

All message processor services in Apigee hybrid stream trace and analytics data to a data collection pod in the cluster.

The data collection pod stores the streamed data on the pod's file system via a fluentd service.

The Universal Data Collection Agent periodically extracts the stored data and sends it to the Unified Analytics Platform service in the management plane.

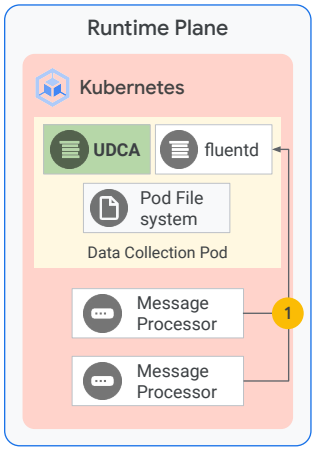
The UAP processes the incoming analytics and deployment status data and makes it available to you via the hybrid UI or the Apigee API.

Apigee hybrid implements the universal data collection agent as an ApigeeEnvironment custom resource definition in the cluster.

Let’s review the sequence of a UDCA operation.

UDCA operation

1. The Message Processors stream trace and analytics data to the data collection pod. This pod contains the fluentd and UDCA services.

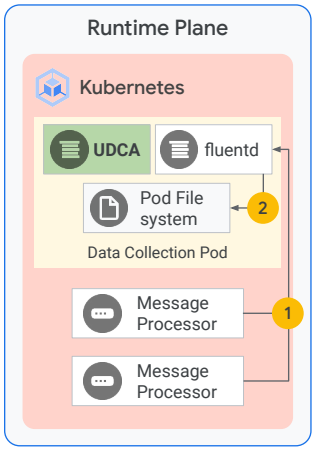


Google Cloud

The Message Processors stream trace and analytics data to the data collection pod in the hybrid runtime plane. This pod contains the fluentd and UDCA services.

UDCA operation

1. The Message Processors stream trace and analytics data to the data collection pod. This pod contains the fluentd and UDCA services.
2. The fluentd service buffers the data and writes it to the pod file system.

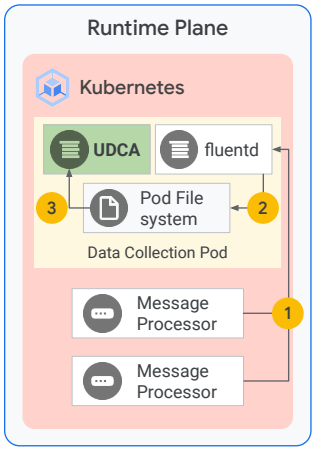


Google Cloud

The fluentd service buffers the data and writes it to the pod file system.

UDCA operation

1. The Message Processors stream trace and analytics data to the data collection pod. This pod contains the fluentd and UDCA services.
2. The fluentd service buffers the data and writes it to the pod file system.
3. UDCA continuously polls the pod file system for new data files.

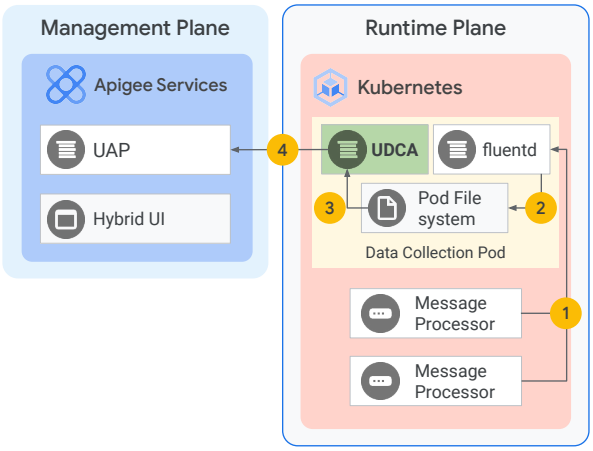


Google Cloud

UDCA continuously polls the pod file system for new data files. The polling interval can be configured for your hybrid installation.

UDCA operation

1. The Message Processors stream trace and analytics data to the data collection pod. This pod contains the fluentd and UDCA services.
2. The fluentd service buffers the data and writes it to the pod file system.
3. UDCA continuously polls the pod file system for new data files.
4. As soon as a new data file is available, UDCA uploads it to the UAP service in the management plane.

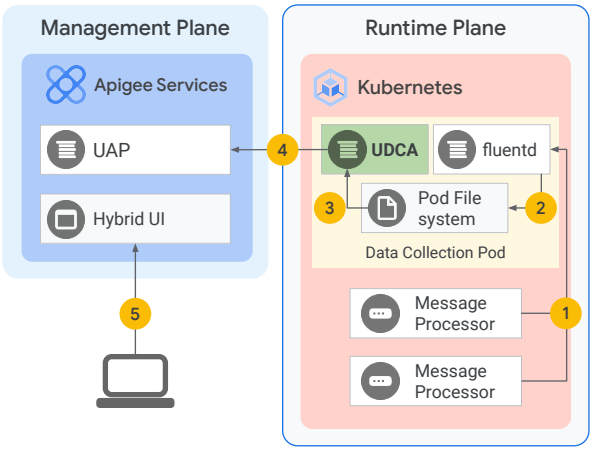


Google Cloud

When a new data file is available, UDCA quickly uploads it to the UAP service in the management plane.

UDCA operation

1. The Message Processors stream trace and analytics data to the data collection pod. This pod contains the fluentd and UDCA services.
2. The fluentd service buffers the data and writes it to the pod file system.
3. UDCA continuously polls the pod file system for new data files.
4. As soon as a new data file is available, UDCA uploads it to the UAP service in the management plane.
5. The API developer or administrator accesses the data via the hybrid UI or Apigee API.



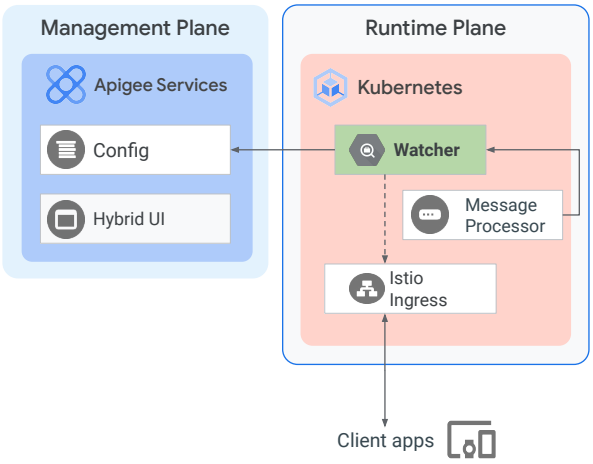
Google Cloud

You then access the data via the Apigee hybrid UI or API.

Watcher

The Apigee Watcher runtime component:

- Fetches virtual host and base path configuration from the management plane.
- Creates or updates ApigeeRoutes based on new or updated configuration.
- Uses the apigee-watcher service account with the Apigee Runtime Agent role to authenticate with the management plane.
- Generates logs that are collected and sent to Cloud Logging.
- Exposes metrics that are scraped by Prometheus and sent to Cloud Monitoring.
- Is implemented as an ApigeeOrganization (CRD) and scoped to the hybrid organization.



Google Cloud

The Watcher component is a hybrid organization-level object that is implemented as an ApigeeOrganization (CustomResourceDefinition) in the Kubernetes cluster.

It fetches virtualhost and basepath configuration from the management plane and propagates that configuration to the ingress gateway using ApigeeRoute resource objects.

Watcher also sends API proxy deployment status data from the runtime message processors to the management plane.

Watcher authenticates to the management plane using a separate service account that has the Apigee Runtime Agent role.

Like the other hybrid runtime components, it generates logs and metrics that can be used for debugging and monitoring purposes using Cloud Logging and Cloud Monitoring.

ApigeeRoute (AR)

ApigeeRoute (AR):

- Is a Kubernetes custom resource definition (CRD) object.
- Contains hostnames and Istio ingress-related configuration such as associated base paths and ports.
- Represents an environment group definition in the Kubernetes cluster.

ApigeeRoute Controller:

- Is a Kubernetes custom controller.
- Updates the Istio Virtual Service and Gateway resources in the cluster when an ApigeeRoute configuration is modified.

Apigee hybrid leverages Anthos Service Mesh to configure an Istio ingress (envoy) proxy for traffic into the cluster.

Google Cloud

Environment groups that are defined in the hybrid management plane are represented as ApigeeRoute custom resource definition objects in the Kubernetes cluster.

ApigeeRoute objects are created or updated in the cluster by the runtime plane Watcher component.

Apigee hybrid leverages Anthos Service Mesh to configure an Istio ingress (envoy) proxy for traffic into the cluster.

The ApigeeRoute controller updates the Istio virtualservice and gateway resources in the cluster when an ApigeeRoute object is modified.

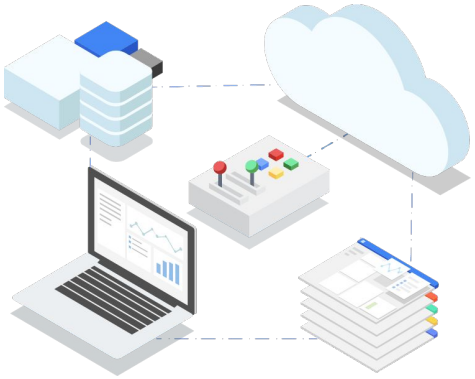
Anthos Service Mesh (ASM)

Anthos Service Mesh provides:

- A fully managed service mesh that manages your Istio environment in GKE.
- In-depth telemetry in the Anthos Service Mesh dashboard.
- Clear and simple insights into the health of your services with service level objectives (SLOs).

Apigee hybrid:

- Uses the Istio ingress gateway component of Anthos Service Mesh.
- Does not use Istio sidecar containers with runtime components.



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

Google Cloud

Anthos Service Mesh is a fully managed service mesh that manages your Istio environment in Google Kubernetes Engine.

It provides in-depth telemetry in the Anthos Service Mesh dashboard on Google Cloud Console.

Using ASM, you get clear and simple insights into the health of your services with service level objectives (SLOs).

Apigee hybrid uses the Istio ingress gateway component of Anthos Service Mesh.

Google Cloud services

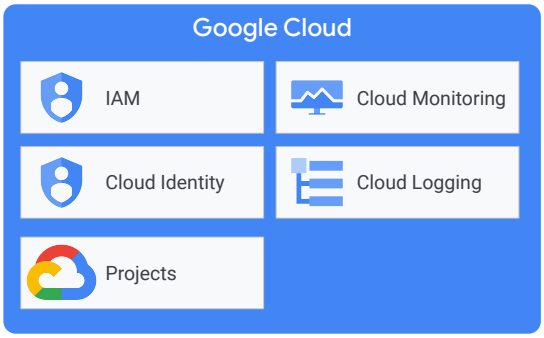
Apigee hybrid leverages a set of key Google Cloud services:

Cloud Identity: For user authentication using Google Cloud accounts and authorization using Google Cloud service accounts.

IAM: For role access management for hybrid services.

Google Cloud Projects: To contain resources used in Apigee hybrid, including organizations and environments.

Cloud Logging and Monitoring: For logging, monitoring, and metrics data analysis.



Google Cloud

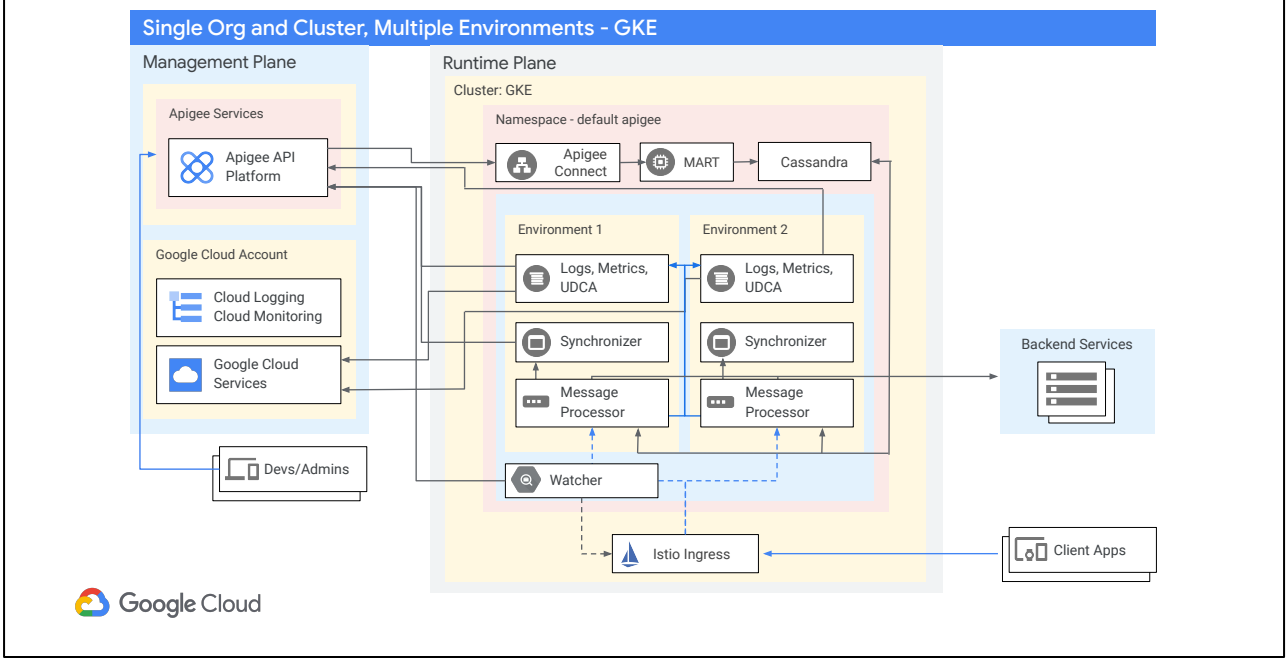
Apigee hybrid uses various Google Cloud services for its operation.

It uses Cloud Identity for user authentication and service account authorization and uses IAM roles and permissions for access management.

Apigee hybrid uses default Apigee roles created in IAM.

All hybrid resources are organized under the cloud project for management and billing.

Cloud Logging and Cloud Monitoring are used by hybrid components for logging, metrics collection, monitoring, and troubleshooting.



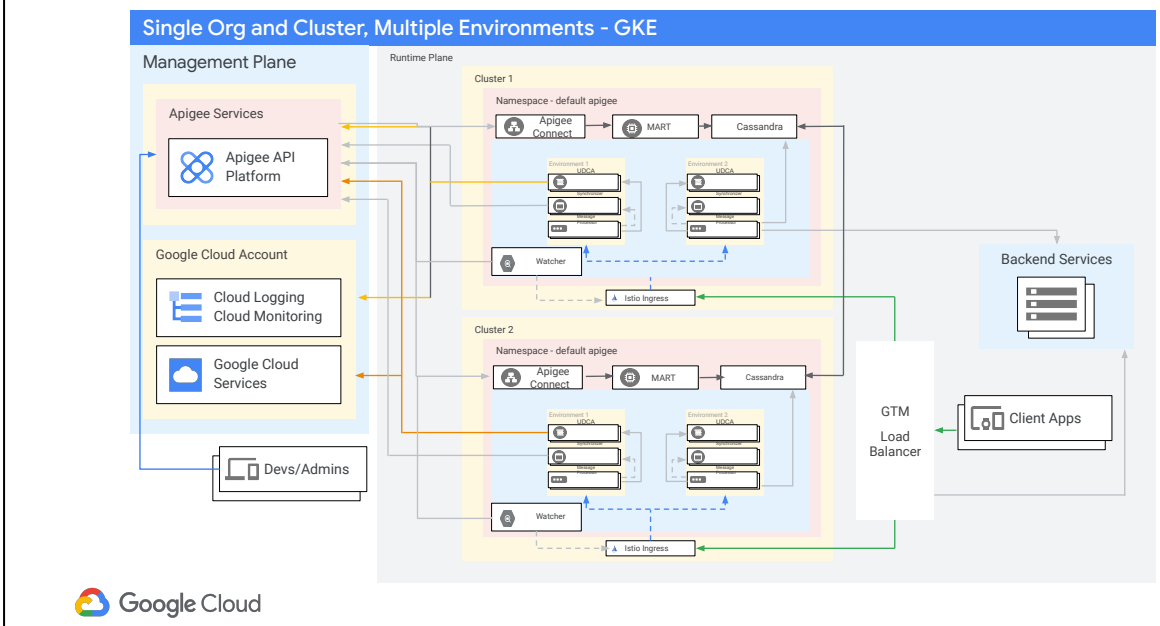
This diagram shows a typical hybrid runtime installation. In production systems there are 2 nodepools: one for data that is used by cassandra, and the second one for the other runtime components.

In this deployment pattern there is one Kubernetes cluster.

All components are hosted within the cluster, including the runtime message processor components that service 2 hybrid environments.

Ingress is via the Istio Ingress gateway, which exposes the runtime service outside the cluster and routes traffic to the environments.

Developers and Admins access management functions through Apigee Services (UI and Apigee API) and via the Google Cloud Console.



This hybrid installation shows a more complex deployment pattern.

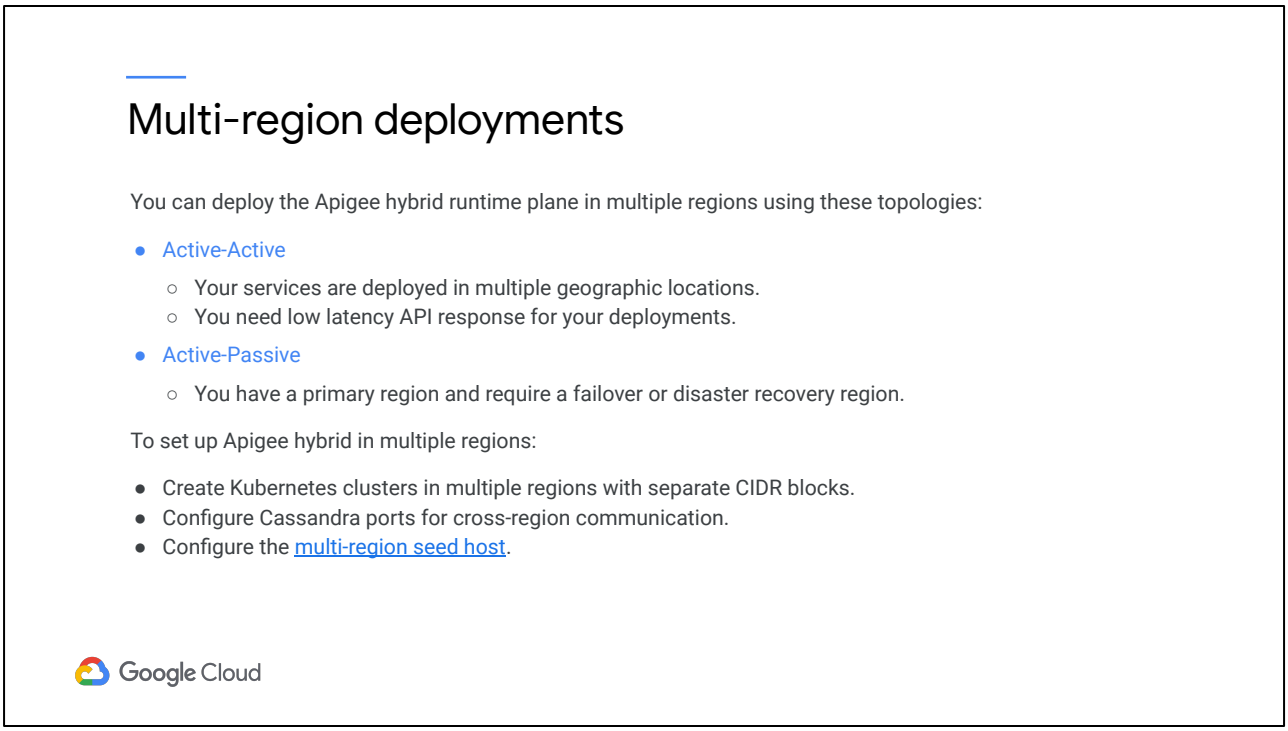
Connections to the management plane are only shown from one environment for clarity, but each environment would have the same kind of outbound and inbound connections.

The setup is identical in terms of traffic that flows from the runtime plane to the management plane, and vice versa.

With this pattern, we introduce a global traffic manager and load balancer to distribute traffic between the two runtime plane clusters.

Additionally, Cassandra replication enables all runtime plane data to be synced across each cluster.

With this setup we can now have different numbers of cassandra nodes in each cluster (although this should still align with the recommendation that the number of nodes is a multiple of the replication factor).



The Apigee hybrid runtime plane can be deployed in multiple regions.

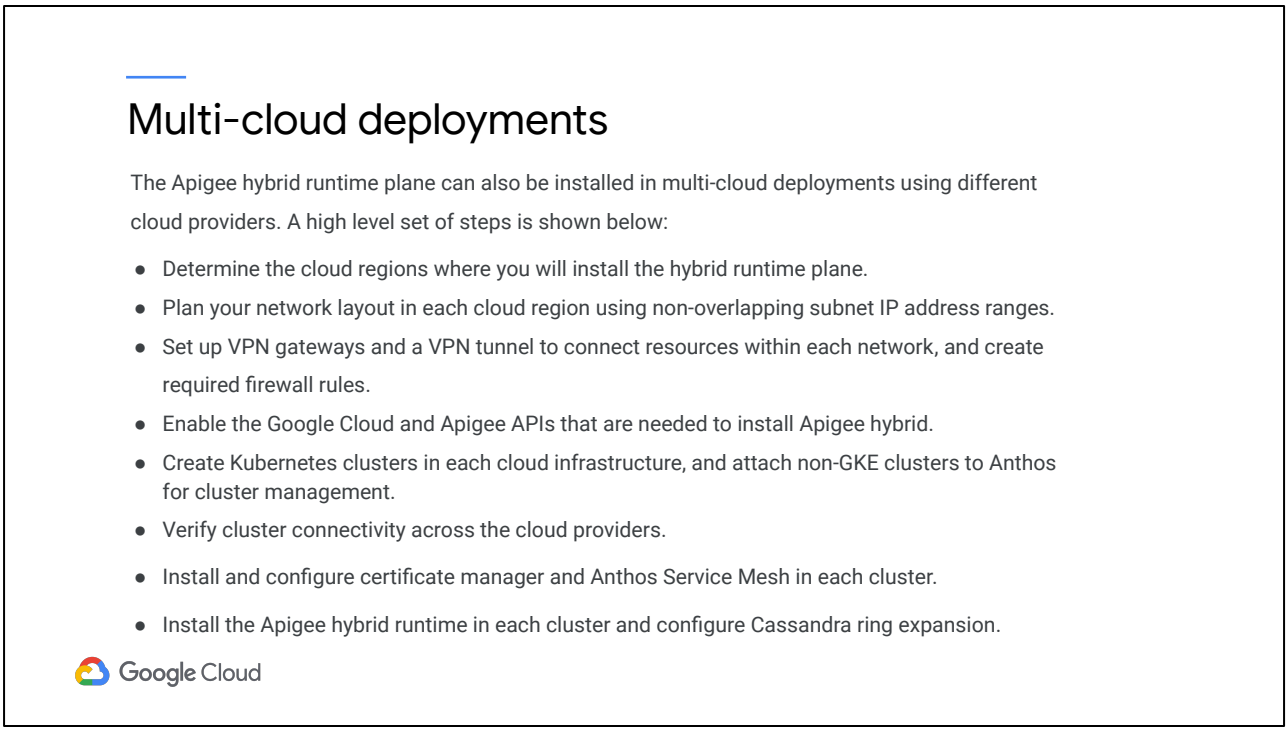
It can be deployed in an active-active topology for low latency API responses from your multi-regional services.

It can also be deployed in an active-passive topology for failover or disaster recovery scenarios.

To set up Apigee hybrid in multiple regions, you set up Kubernetes clusters in each region using separate CIDR blocks.

You must also open Cassandra ports for cross-region communication because Cassandra propagates hybrid data to all clusters.

Details on how to expand an existing Cassandra cluster to a new region and configure a multi-region seed host are provided at the link included on the slide.



You can install the Apigee hybrid runtime plane using a multi-cloud deployment model.

Here is a high-level list of tasks needed to accomplish this:

Plan your network layout in each cloud region and ensure that your subnets do not use overlapping IP address ranges.

Set up VPN gateways and a VPN tunnel to connect resources within each network and create required firewall rules.

Enable the Google Cloud and Apigee APIs that are needed to install Apigee hybrid.

Create Kubernetes clusters in each cloud infrastructure, and attach non-GKE clusters to Anthos for cluster management.

Verify cluster connectivity across the cloud providers.

Install and configure certificate manager and Anthos Service Mesh in each cluster. These are prerequisites for running Apigee hybrid.

Install the Apigee hybrid runtime distribution in each cluster, and configure Cassandra ring expansion.

Agenda

Terminology and Organizational Structure

Architecture

Networking

Apigee API

Google Cloud

In this lecture, we will discuss Apigee hybrid networking within the runtime plane and between the runtime and management planes.

Runtime internal connections

The runtime components connect to each other to perform various functions:

- MART and MPs connect to Cassandra for storing and retrieving runtime data.
- Cassandra nodes communicate on port 7001.
- The default Istio ingress gateway connects to runtime message processors on port 8443.
- MPs send logs and analytics data to fluentd on port 20001.
- Prometheus collects metrics data from various components over TLS.

Google Cloud

It is important to understand the networking and port requirements needed to operate Apigee hybrid.

Individual runtime plane services use slightly different security protocols and different ports to communicate with each other.

Port numbers used by the runtime services are documented on the Apigee hybrid website at docs.apigee.com/hybrid/ports.

The MART and message processor runtime services connect to Cassandra using mutual TLS.

The default Isitio ingress gateway connects to the message processors on port 8443 using 1-way TLS.

Runtime component ports

Runtime external connections

Inbound Connections

- API calls from the management plane to MART via Apigee Connect use TLS..
- Clients apps send runtime traffic to the default ingress gateway on configured TCP ports.

Outbound Connections

- All runtime components use port 443 to send traffic to the management plane and to Google Cloud services.
- MPs send runtime egress traffic to backend services on configured TCP ports.

Google Cloud

External connections to the hybrid runtime plane can be grouped into inbound or outbound connections.

Inbound connections include API calls from the management plane to MART via Apigee Connect, and client app requests to the default Istio ingress gateway.

Management API calls use OAuth over TLS. You can configure the ingress gateway to use TLS based on your API requirements.

Outbound connections include egress traffic to external backend services and connections to the management plane and to Google Cloud services.

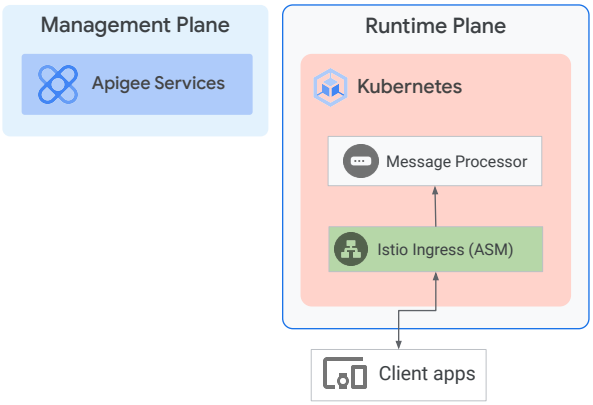
You can configure egress traffic from the message processors to your backend services to use 1-way or 2-way TLS.

Runtime components use port 443 to communicate with the management plane and services in Google Cloud over TLS.

Ingress gateway

The hybrid runtime plane uses an Istio ingress gateway for API runtime traffic:

- istio-ingressgateway
  - Used for API proxy traffic between client apps and the runtime.
  - Implemented as a Deployment in the Kubernetes runtime cluster.



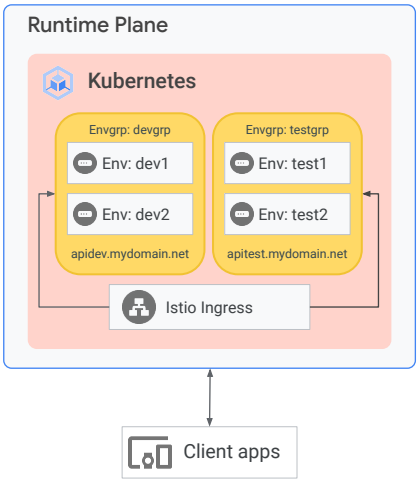
Google Cloud

The Istio ingress gateway manages inbound requests from client apps. The Apigee hybrid installation uses an Anthos Service Mesh Istio Ingress controller that is scoped to the cluster.

It is used to route traffic to the hybrid environments that are configured in the cluster.

Virtual hosts and environments

- The Ingress gateway is used to manage connections that are exposed outside the cluster. All HTTPS requests to an API proxy are first handled by the ingress gateway.
- In hybrid, you configure an environment group with one or more hostnames (domain names).
- Virtual hosts allow Apigee hybrid to handle API requests to hostnames associated with an environment group.
- Using the virtualhosts[].name property, you map the virtual host to an environment group in the runtime plane configuration file.
- The ingress gateway routes the incoming request based on its DNS name to the correct environment group and runtime service.



Google Cloud

Virtual hosts allow Apigee hybrid to handle API requests to multiple domain names and route API proxy requests to specific hybrid environments.

The virtualhosts runtime plane configuration enables you to map an environment group using the virtualhosts.name property.

This configuration enables the ingress gateway to route an API proxy request to the environment group and runtime service that handles the request.

Configuring virtualhosts

This sample overrides.yaml file has 2 virtualhosts defined: devgrp and testgrp.

- The name property of the virtualhost must match the name of an environment group.
- Host aliases are configured using the Apigee UI or API when the environment group is created.
- The virtualhost property requires a TLS key and certificate.
  - The key and certificate are used to provide secure communication with the ingress gateway.
  - The certificate must be compatible with the host aliases used in the specified environment group.

overrides.yaml

```
...
virtualhosts:
  - name: devgrp
    sslCertPath: ./certs/fullchain.pem
    sslKeyPath: ./certs/privkey.pem

  - name: testgrp
    sslCertPath: ./certs/fullchain.pem
    sslKeyPath: ./certs/privkey.pem
...
```

Google Cloud

Here is a sample runtime plane configuration file that defines two virtual hosts named **devgrp** and **testgrp**.

Each virtualhost maps to an environment group with the same name.

One or more hostAliases (domain names) must be configured when creating the environment group. The hostAliases must be unique across multiple environment groups.

The virtualhost property requires a TLS key and certificate. The key and certificate are used to provide secure communication with the ingress gateway and must be compatible with the host aliases defined in the specified environment group.

Using VPC Service Controls

- VPC Service Controls helps mitigate the risk of data exfiltration from Google Cloud services.
- VPC Service Controls defines a service perimeter that acts as a boundary between a project and other services.
- Apigee hybrid integrates with VPC Service Controls to isolate resources in your Google Cloud projects.
  - Both the Google Cloud project and its associated runtime are included within that project's service perimeter.
- Resources within a perimeter are accessed only from clients within authorized VPC networks using Private Google Access.
- Clients that have private access to resources within a perimeter do not have access to unauthorized resources outside the perimeter.

Google Cloud

VPC Service Controls helps mitigate the risk of data exfiltration from Google Cloud services.

With VPC Service Controls, you create perimeters that protect the resources and data of services that you explicitly specify.

Apigee hybrid integrates with VPC Service Controls to isolate resources in your Google Cloud projects.

Resources within a perimeter are accessed only from clients within authorized VPC networks using Private Google Access from either Google Cloud or on-premises.

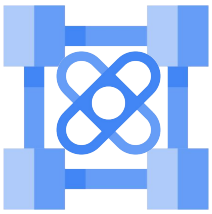
Clients that have private access to resources within a perimeter do not have access to unauthorized (potentially public) resources outside the perimeter.

VPC Service Controls provides an additional layer of security defense for Google Cloud services that is independent of Identity and Access Management (IAM).

Setting up VPC Service Controls with Apigee hybrid

To set up VPC Service Controls, follow these steps:

- Set up private connectivity for your hybrid hosts.
- Configure the service perimeter to secure Apigee services and additional services within the perimeter. Additional services include:
  - Anthos Service Mesh (ASM)
  - Cloud Monitoring
  - Google Kubernetes Engine (if the runtime plane is on GKE)
  - Container Registry (if used as a local repository)
- Copy Apigee hybrid images to the private repository.
- Update your hybrid configuration overrides file to use the image URLs for the private repository for the hybrid runtime components.
- Apply your override file changes to the hybrid runtime cluster.



More details are provided here.

Google Cloud

To set up VPC Service Controls with Apigee hybrid, you must first set up private connectivity for your hybrid hosts. This involves configuring routes, firewall rules, and DNS entries to let the Google APIs access those private IPs.

Next, you must secure Apigee services and additional services within the service perimeter. The additional services include Anthos Service Mesh, Cloud Monitoring, Google Kubernetes Engine (if you are running on GKE), and Container Registry (if you are using this as your local repository).

You then copy the Apigee hybrid images to the private repository by downloading a specific version of the signed Apigee images, tagging the images, and then uploading them to your image repository. A tool is available in the Apigee hybrid distribution that helps you implement this task.

Next, you update your overrides configuration file to refer to the image URLs to the private repository for the hybrid runtime components.

Finally, you apply your overrides configuration file changes to the hybrid runtime cluster.

Kubernetes network policies

- The Apigee hybrid runtime plane uses multiple Kubernetes pods.
- Pods communicate with each other on specific IP addresses and ports within the cluster.
- Using Kubernetes network policies, you can control traffic flow between pods at the IP address or port level.
- You use a selector to specify what traffic is allowed to and from the pods that match the selector.
- IP-based network policies are defined using IP blocks (CIDR ranges).
- To use network policies in Kubernetes, you must use a network plugin that supports the NetworkPolicy resource.

Google Cloud

Apigee hybrid uses multiple Kubernetes pods in its runtime plane that communicate with each other on specific IP addresses and ports within the cluster.

However, not all pods need to communicate with every other pod.

You can use Kubernetes network policies to control the traffic between pods at the IP address or port level.

The entities that a Pod can communicate with are identified through a combination of other pods that are allowed, namespaces that are allowed, and IP blocks.

When defining a pod or namespace based NetworkPolicy, you use a selector to specify what traffic is allowed to and from the pods that match the selector.

With IP-based network policies, you define policies based on IP blocks (CIDR ranges).

To use network policies, you must use a network plugin or controller that supports the NetworkPolicy resource.

Using network policies with Apigee hybrid

- In Apigee hybrid, not every pod needs to communicate with every other pod.
- It is a good practice to create network policies for your hybrid component pods, based on their scope of operation, to control pod traffic flow,.
- Components that are installed by Apigee hybrid are cluster, organization, or environment scoped. The components:
  - apigee-metrics and apigee-logger are cluster scoped.
  - apigee-connect and apigee-mart are organization scoped.
  - apigee-cassandra is organization scoped but can host more than one org.
  - apigee-runtime, apigee-udca, and apigee-synchronizer are environment scoped.
- For example, to secure the apigee-cassandra pod for ingress traffic:
  - Create network policies to allow access only from apigee-mart, apigee-runtime, other apigee-cassandra pods, and apigee-metrics components.

Google Cloud

In the Apigee hybrid runtime plane, not every pod communicates with every other pod in the cluster.

It is a good practice to create network policies for your hybrid component pods to control pod traffic flow, based on their scope of operation.

You create network policies to restrict the component pods to only communicate with the other hybrid runtime pods based on their function.

For example, to secure access to the apigee-cassandra pod, create network policies to allow only ingress traffic from specific hybrid components.

These components include apigee-mart, apigee-runtime, the other apigee-cassandra pods, and the apigee-metrics components.

Similarly, you create additional network policies to secure access to the other hybrid runtime components.

Secure access with network policies

Sample NetworkPolicy configuration to secure access to apigee-cassandra from apigee-mart:

- The podSelector property identifies the apigee-cassandra component to apply the policy.
- The direction of traffic (policyType) is configured as Ingress.
- The source pod from which traffic is allowed uses the podSelector label of apigee-mart.

To apply the network policy to your cluster:

- Install a CNI network plugin like Calico.
- Apply the network policy configuration to the cluster using kubectl.

Google Cloud

```
apiVersion: networking.k8s.io/v1
kind: NetworkPolicy
metadata:
 name: cassandra-mart
spec:
 podSelector:
   matchLabels:
     app: apigee-cassandra
 policyTypes:
   - Ingress
 ingress:
 - from:
   - namespaceSelector:
       matchLabels:
         app: apigee
   - podSelector:
       matchLabels:
         app: apigee-mart
   ports:
   - protocol: TCP
     port: 9042
   - protocol: TCP
     port: 9142
```

Here is a sample NetworkPolicy configuration to secure access to apigee-cassandra from apigee-mart:

Note that the podSelector property identifies the apigee-cassandra component to apply the policy to.

The direction of traffic (policyType) is configured as Ingress.

The source pod from which traffic is allowed uses the podSelector label of apigee-mart.

The protocol and ports are as specified for the apigee-cassandra component.

To apply the configuration to your cluster, install a CNI plugin like Calico and apply the network policy using the kubectl command.

Agenda

Terminology and Organizational Structure

Architecture

Networking

Apigee API

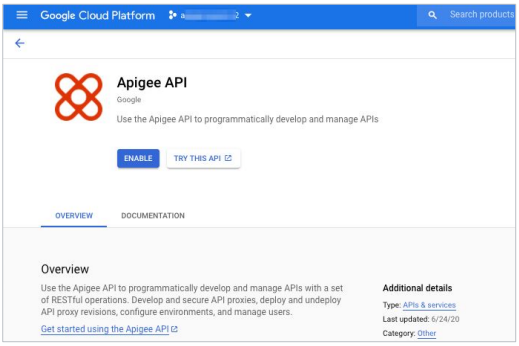
Google Cloud

In this lecture, we briefly discuss the Apigee API.

The Apigee API is a management API that you use to programmatically manage Apigee hybrid entities.

Apigee API

- The Apigee API is hosted by Apigee in the hybrid management plane on Google Cloud.
- The API must be enabled for the project via the Google Cloud Console.
- The Apigee API provides communications between your project and other hybrid services and Cloud APIs.
- The Apigee APIs support OAuth 2.0 for user authentication.
- Service account credentials are required to generate the access token, which can then be used to make secure API calls.



Google Cloud

The Apigee API is hosted by Apigee in the management plane on Google Cloud. It must be enabled for your project.

To use the Apigee API, you must have a valid OAuth 2.0 access token.

The token can be generated by using your Google Cloud account or service account credentials.

A service account with the appropriate role can be created using IAM in the Cloud Console and can be used to generate the token.

The token expires in 60 minutes, after which a new token must be generated.

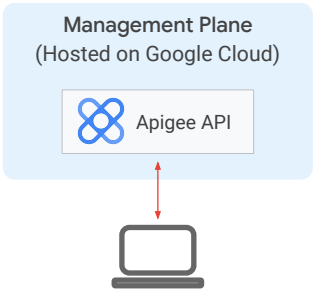
Additional Google Cloud APIs that must be enabled to use hybrid are:

- **Apigee Connect API:** Provides communication between the Apigee management plane and the MART service in the runtime plane.
- **Cloud Pub/Sub API:** Required for quota to operate.
- **Cloud Resource Manager API:** Used by hybrid to validate service accounts.
- **Compute Engine API:** Used for cluster management (GKE-based clusters only).
- **Kubernetes Engine API:** Enable if you plan to use Google Kubernetes Engine (GKE), or GKE on-prem (Anthos) on-prem for your hybrid runtime installation.

Programmatic access

Using the Apigee API, you can programmatically:

- Create, edit, and delete API proxies.
- Deploy and undeploy proxy revisions and shared flows.
- Configure environments.
- Manage apps, developers, keys, and other KMS data.
- ...and much more.



Google Cloud

The Apigee API can be used to programmatically manage various Apigee entities, such as API proxies, API products, developers, and apps.

Using the Apigee API for CI/CD workflows is a typical use case.

As part of these workflows, you can perform CRUD operations—create, read, update, delete—on API proxies and other configuration objects by using scripts that invoke the Apigee APIs.

Apigee API resources

- The Apigee hybrid service has the following service endpoint (base URL), and all URIs below are relative to this service endpoint:

  https://apigee.googleapis.com
- Relative to the service endpoint, the API exposes various REST operations on a variety of resources.
- Some of these resources include:
  - v1.organizations
  - v1.organizations.environments
  - v1.organizations.apis
  - v1.organizations.apiproducts

The complete list of resources and operations is documented here:
https://cloud.google.com/apigee/docs/reference/apis/apigee/rest

Google Cloud

The Apigee APIs are hosted in Google Cloud at the apigee.googleapis.com endpoint.

The APIs expose operations on various resources as REST endpoints.

Organizations, environments, APIs, and API products are some of the resources that can be managed using these APIs.

The complete list of Apigee APIs is available on the Google Cloud Apigee API documentation site.

Example: Calling the /v1/organizations API

- To enable the Apigee API for your Google Cloud project, follow the instructions here.
- Generate an OAuth 2.0 access token using your service account credentials using these steps:
  - Download the service account key .json file.
  - Set the environment variable for your service account key JSON file:

```
$ export GOOGLE_APPLICATION_CREDENTIALS=~/sa-credentials-file.json
```

  - Generate and set the environment variable that contains the OAuth token:

```
$ export TOKEN=`gcloud auth application-default print-access-token`
```

- Call an Apigee API passing in the access token:

```
$ curl -X GET https://apigee.googleapis.com/v1/organizations/{org} -H
"Authorization: Bearer $TOKEN"
```

Google Cloud

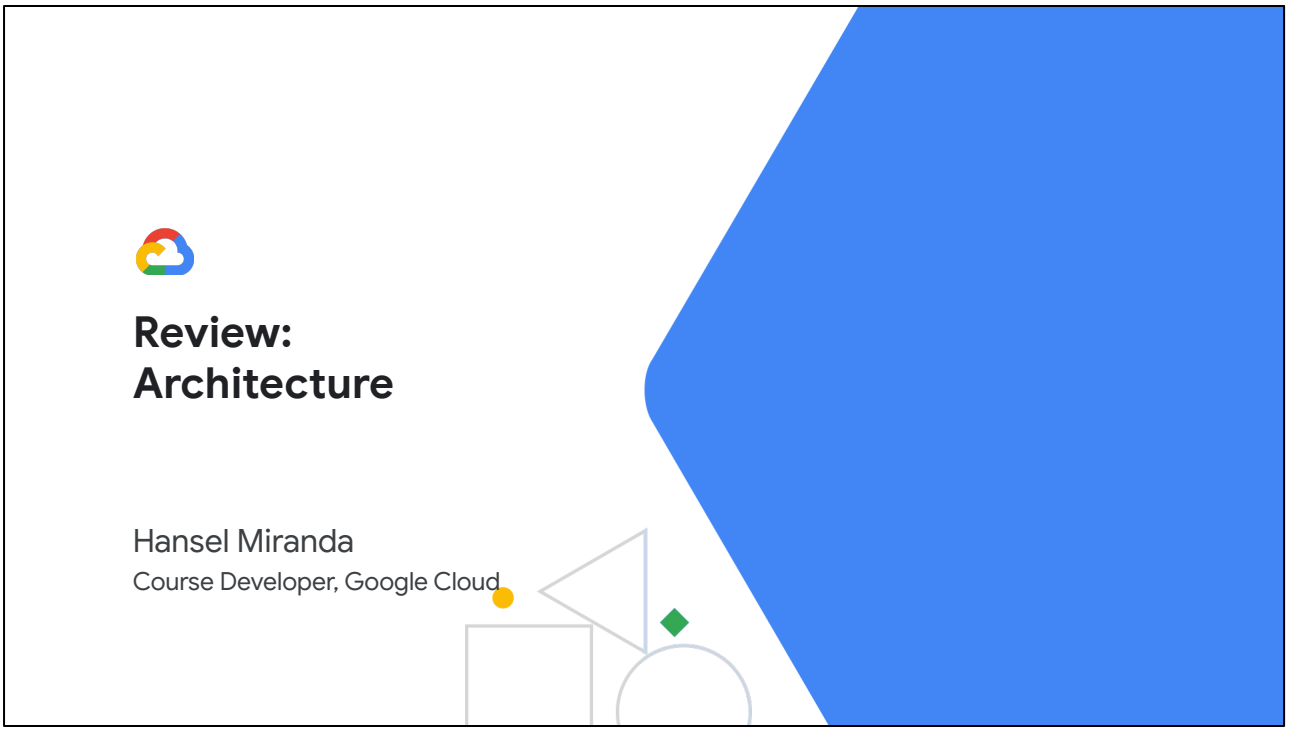
Here is an example of invoking the /v1/organizations Apigee API.

The Apigee API is enabled using the Google Cloud Console.

An OAuth 2.0 access token is generated using service account credentials. The service account in your Google Cloud project with the Apigee Organization Admin role is used.

Using curl, the /v1/organizations API is invoked by passing in the token value as a bearer token in the request header and passing in the name of the hybrid organization in the URL path.

This API returns information about the organization in the response.



In this module, you learned about the terminology and core concepts, such as the Apigee Organization and environment, used in Apigee hybrid.

We discussed the Apigee hybrid architecture, its components, and their operation.

You learned about the hybrid management plane, the hybrid UI, and the Apigee API, and how they are used to manage your hybrid platform.

You also learned about networking in Apigee hybrid and how virtual hosts are configured for environments.

Next, you will learn about the installation process and complete a lab to install Apigee hybrid on Google Kubernetes Engine.