



Security

David Mehi

Customer Success Engineer, Google Cloud



Google Cloud

This is the security module in the course on managing and securing the Apigee hybrid API platform.

In this module, you will learn about the security features and role-based access control used in Apigee hybrid.

You will also learn how traffic is secured within the hybrid runtime plane and with the management plane in Google Cloud.

Agenda

Infrastructure Security

Data Storage and Encryption

RBAC



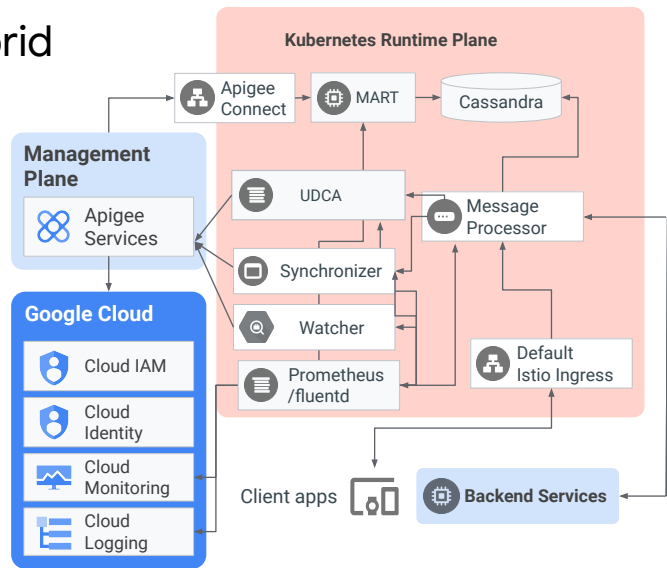
In this lecture, we will discuss the security features used in Apigee hybrid and how traffic within the runtime plane and with the management plane is secured.

In later lectures, you will learn how data is collected and stored and how role-based access control is implemented in Apigee hybrid.

Securing traffic in hybrid

Traffic between various hybrid components is secured as described below:

- **Management plane** ⇄ **Apigee Connect** uses OAuth over TLS.
- **Client Apps** ⇄ **Default Istio Ingress** can be configured to use TLS 1.2/OAuth.
- **MP** ⇄ **Backend services** can be configured to use TLS 1.2/OAuth.
- All **runtime components to the Apigee management plane** use OAuth over TLS 1.2.
- **Prometheus/fluentsd** ⇄ **Google Cloud services** use TLS.



Traffic between the Apigee hybrid runtime components and the management plane is over TLS and uses OAuth access tokens.

Ingress traffic from client applications into the cluster and egress traffic from the runtime message processors to backend target servers can be secured using TLS and OAuth.

All logging and metrics data sent from the hybrid runtime plane to Google Cloud services goes over TLS.

Securing traffic between runtime components

Traffic between various hybrid components is secured as described below:

- MART and MPs connect to Cassandra using mTLS.
- Intra-node communication between Cassandra nodes uses mTLS.
- The default Istio Ingress gateway connects to MPs over TLS.
- MPs send Analytics data to fluentd using mTLS.
- Prometheus scrapes metrics data from various components over TLS.



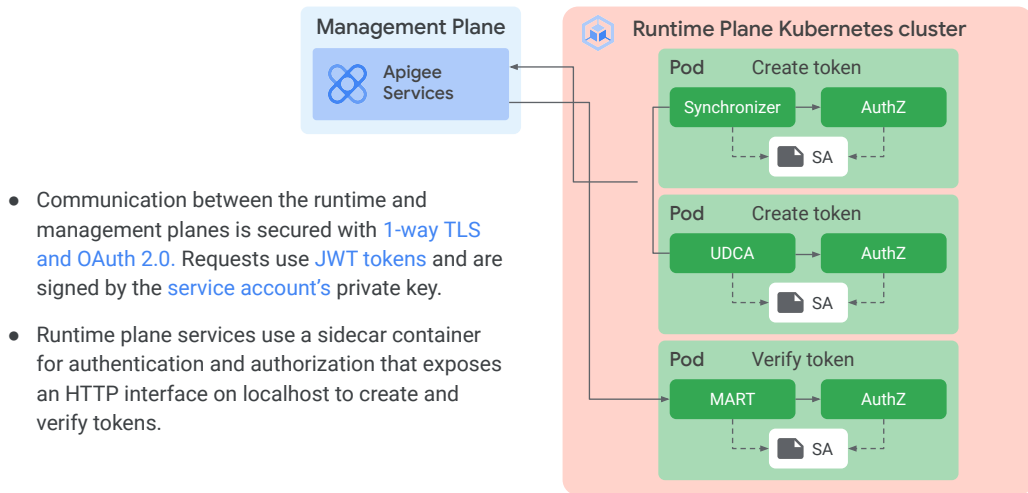
All internal communication between the hybrid runtime plane components uses TLS.

In addition, components that connect to the runtime Cassandra data store use client authentication or mutual TLS.

Intra-node Cassandra communication also uses mutual TLS.

Apigee hybrid stores keys and certificates used by the runtime components in Kubernetes secrets.

Generation and verification of runtime plane tokens



When communicating with management plane services, the synchronizer and UDCA components use OAuth access tokens.

Each component employs a sidecar authorization container that generates an OAuth 2.0 JWT token, which is signed using the component's service account private key.

The service accounts are stored as Kubernetes secrets and made available to the runtime pods.

The MART runtime component uses the authorization container to verify OAuth 2.0 tokens sent with API requests from the hybrid management plane to the runtime plane.

To verify the token, the container uses the public key of a configured service account available from a Google Cloud endpoint.

Service accounts

- A [service account](#) is a special type of account that enables components and applications of a system to interact with each other and with other APIs.
- In hybrid, they act as [trusted issuers](#) for the runtime and management planes and are used by hybrid components to perform various operational tasks.
- During the installation of Apigee hybrid, you create [multiple service accounts](#). Each account is assigned to a specific hybrid component with its own set of roles and permissions.
- Service accounts have to be registered with the Apigee hybrid management plane before they can be used.
- Apigee publishes service accounts that it uses to connect to runtime planes, which can be accessed via the [hybrid.issuers.list](#) management API.



Apigee hybrid uses service accounts to perform various tasks that include sending logs and metrics data to Google Cloud and downloading proxy bundles from the management plane.

A service account is a type of Google account that represents a system or application user that needs to be authenticated and be authorized to access Google APIs and services.

Apigee hybrid requires you to create several service accounts during installation. Each service account requires a specific role or set of roles that enable it to perform certain functions.

Service accounts in hybrid

Service Account	Role	Description
apigee-cassandra	Storage Object Admin	Allows Cassandra backups to Google Cloud Storage (GCS)
apigee-logger	Logs Writer	Allows logging data collection
apigee-mart	Apigee Connect Agent	Allows MART service authentication
apigee-metrics	Monitoring Metric Writer	Allows metrics data collection
apigee-synchronizer	Apigee Synchronizer Manager	Allows the synchronizer to download proxy bundles and configuration data and enables operation of trace
apigee-udca	Apigee Analytics Agent	Allows the transfer of trace and analytics data to management plane
apigee-watcher	Apigee Runtime Agent	Allows retrieval of virtual host changes for an org, and configures Isito ingress.



This is a list of service accounts used in Apigee hybrid. You create these service accounts when you install Apigee hybrid.

The service accounts have a specific assigned role and are used by the runtime plane components to perform the function described.

For example, the apigee-synchronizer service account is used by the synchronizer component to download API proxy bundles and configuration data from the management plane.

The Apigee hybrid distribution includes the create-service-account tool that you can use to create service accounts. You can also use the Google Cloud Console or the gcloud SDK.

It is a good practice to use only one role for each service account in order to limit access for each runtime component.

When installing the hybrid runtime plane, you configure the paths to the service account key files on your machine in the overrides.yaml configuration file.

[About service accounts | Apigee](#)

Securing environments

- When you create an environment, you must assign it to an environment group.
- The environment group is mapped to a virtual host configuration in the overrides.yaml file.
- As part of the virtual host configuration, you provide a TLS key and certificate to secure traffic between client apps and the ingress gateway.
- Self-signed certificates are supported for test environments but not recommended for production clusters.



overrides.yaml

```
...
namespace: my-namespace
org: my-organization
...
envs:
  - name: test
...

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

A hybrid environment must be assigned to an environment group.

The environment group is configured with one or more host aliases or DNS names, and is mapped to a virtual host configuration in the overrides.yaml file.

You configure the virtual host with the path to a TLS certificate and key that is used to secure that host.

Client apps that send API requests to the host domain must be able to trust the certificate in order to establish the TLS connection.

You can use a self-signed certificate for a development or test installation of Apigee hybrid.

Certificates and DNS entries can be generated using the [Let's Encrypt](#) CA, the [Certbot client](#), and Google Cloud Platform [Cloud DNS](#).

Securing Synchronizer

- [Synchronizer's](#) primary job is to periodically poll the management plane, download config data about proxies, shared flows, target servers, etc., and make those available to the local runtime.
- In order to download Apigee configuration data, Synchronizer must first be authorized via the [setSyncAuthorization](#) Apigee API. This is accomplished by:
 1. Generating an [OAuth token](#) using a gcloud authenticated user with the [Apigee Org Admin](#) role.
 2. Calling the API passing in the bearer token and the synchronizer service account that has the [Apigee Synchronizer Manager](#) role.

```
1.
$export TOKEN=$(gcloud auth
print-access-token)

2.
curl -X POST -H "Authorization: Bearer
$TOKEN" -H
"Content-Type:application/json" \
https://apigee.googleapis.com/v1/organiz
ations/{org}:setSyncAuthorization" -d \
'{"identities":["serviceAccount:synchron
izer-manager-service-account-name"]}'
```



The synchronizer runtime component securely accesses the management plane using OAuth over TLS to download API proxy and configuration data to the runtime plane.

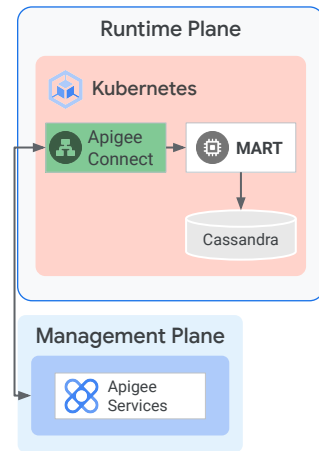
It uses a service account whose key file path is configured in the runtime plane configuration file.

In order to authorize the synchronizer account, you must call the `setSyncAuthorization` Apigee API as part of the hybrid installation process.

To invoke the API, you must first generate an OAuth access token using your Apigee organization admin or gcloud authenticated user credentials.

Securing MART with Apigee Connect

- Apigee Connect is a runtime component that allows the management plane to securely connect to MART.
- With Apigee Connect, you do not expose the MART endpoint, and you do not need to configure the MART ingress gateway and certificate.
- All MART traffic between the management and runtime plane passes through the secure Apigee Connect connection.
- The management plane uses OAuth tokens on all API requests to the MART service.



Apigee Connect 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.

To use Apigee Connect, you must enable the Apigee Connect API in the Google Cloud API library for your project.

A service account with the Apigee Connect Agent role must be created and used to configure the Apigee Connect runtime component in the `overrides.yaml` configuration file.

On startup, Apigee Connect uses the service account credentials to establish a secure bi-directional gRPC stream with the management plane.

All communication between the management plane and Apigee Connect is encrypted using TLS. In addition, all MART API requests from the management plane use OAuth JWT tokens for authentication.

Using Apigee Connect

To use Apigee Connect, follow these steps:



To configure Apigee Connect for your hybrid installation, follow these steps:

Using Apigee Connect

To use Apigee Connect, follow these steps:

1. Enable the Apigee Connect API in the Google Cloud API library.



Enable the Apigee Connect API in the Google Cloud API library.

Using Apigee Connect

To use Apigee Connect, follow these steps:

1. Enable the Apigee Connect API in the Google Cloud API library.
2. Add the [Apigee Connect Agent role](#) to the MART service account created earlier:

```
$ gcloud projects add-iam-policy-binding {project_id}
--member serviceAccount:{mart_svc_acct_email} --role
roles/apigeeconnect.Agent
```



Add the Apigee Connect Agent role to the MART service account.

This is a pre-defined role with the `apigeeconnect.endpoints.connect` permission that is needed to set up the Apigee Connect agent.

Using Apigee Connect

To use Apigee Connect, follow these steps:

1. Enable the Apigee Connect API in the Google Cloud API library.
2. Add the [Apigee Connect Agent role](#) to the MART service account created earlier:

```
$ gcloud projects add-iam-policy-binding {project_id}  
--member serviceAccount:{mart_svc_acct_email} --role  
roles/apigeeconnect.Agent
```
3. Update the overrides.yaml file to enable Apigee Connect.

overrides.yaml

```
...  
# Apigee Connect Agent  
connectAgent:  
  enabled: true  
  serviceAccountPath:  
    ./svc-accts/mart_sa_key.json  
...
```



To enable Apigee Connect, update your overrides.yaml configuration file, specifying the path to the MART service account key file.

Using Apigee Connect

To use Apigee Connect, follow these steps:

1. Enable the Apigee Connect API in the Google Cloud API library.
2. Add the [Apigee Connect Agent role](#) to the MART service account created earlier:

```
$ gcloud projects add-iam-policy-binding {project_id}
--member serviceAccount:{mart_svc_acct_email} --role
roles/apigeeconnect.Agent
```
3. Update the overrides.yaml file to enable Apigee Connect.
4. Update the Apigee hybrid org to enable the Apigee Connect feature using an [organizations.update](#) API call.

overrides.yaml

```
...
# Apigee Connect Agent
connectAgent:
  enabled: true
  serviceAccountPath:
    ./svc-accts/mart_sa_key.json
...
```



To enable the Apigee Connect feature, update the Apigee hybrid organization by using an `organizations.update` Apigee API call.

Using Apigee Connect

To use Apigee Connect, follow these steps:

1. Enable the Apigee Connect API in the Google Cloud API library.
2. Add the [Apigee Connect Agent role](#) to the MART service account created earlier:

```
$ gcloud projects add-iam-policy-binding {project_id}  
--member serviceAccount:{mart_svc_acct_email} --role  
roles/apigeeconnect.Agent
```
3. Update the overrides.yaml file to enable Apigee Connect.
4. Update the Apigee hybrid org to enable the Apigee Connect feature using an [organizations.update](#) API call.
5. Start the agent by applying the config changes:

```
$ apigeectl apply -f your_overrides_file.yaml --org
```

overrides.yaml

```
...  
# Apigee Connect Agent  
connectAgent:  
  enabled: true  
  serviceAccountPath:  
    ./svc-accts/mart_sa_key.json  
...
```



To apply the configuration changes to the cluster, use the `apigeectl` command, passing in the path to the overrides config file and supplying the organization flag.

Apigee Connect is enabled by default for new installations of Apigee hybrid version 1.3.0 and newer. You are most likely to need these steps if you are upgrading from an older version of Apigee hybrid.

Securing Cassandra

- In hybrid, [TLS is enabled by default](#) for any communication between Cassandra nodes and between clients and Cassandra nodes.
 - All client communication to Cassandra requires authentication over mTLS.
 - The username and passwords can be stored in a Kubernetes secret which is configured in the overrides file.
- These user accounts can be used by clients that communicate with Cassandra:
 - [DML User](#): Used by the client to read and write data to Cassandra (KMS, KVM, Cache and Quota)
 - [DDL User](#): Used by MART for any of the data definition tasks
 - [Admin User](#): Used for any admin activities on Cassandra
 - [Default User](#): Default user when authentication is enabled
- Passwords are stored as [Kubernetes secrets](#).

overrides.yaml

```
...
cassandra:
  auth:
    secret: secret_filename
    default: ## un:cassandra
      password: "iloveapis123"
    admin: ## un:admin_user
      password: "iloveapis123"
    ddl: ## username: ddl_user
      password: "iloveapis123"
    dml: ## username: dml_user
      password: "iloveapis123"
...
```



In Apigee hybrid, TLS is enabled by default for any communication between Cassandra nodes and between clients and Cassandra nodes in the cluster.

Any hybrid runtime component that communicates with Cassandra requires authentication.

Apigee hybrid provides user accounts that are used by clients to communicate with Cassandra.

The DML user account is used by the runtime components to read and write API runtime data to the Cassandra data store.

The DDL user account is used by the MART runtime component for data definition tasks like keyspace creation, update, and deletion.

The admin user account is used for any administrative activities performed on the cassandra cluster.

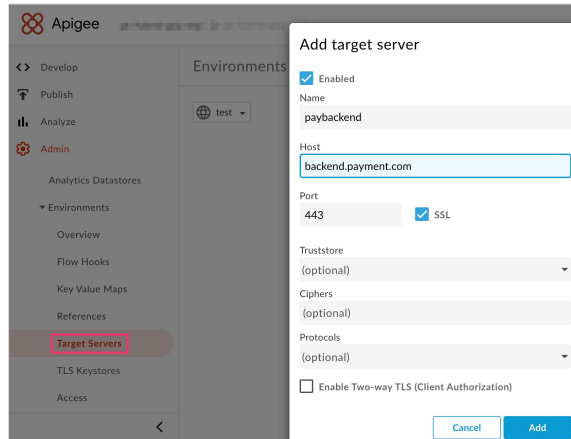
Apigee hybrid provides default passwords for the Cassandra user accounts. You can change the default passwords in the overrides.yaml configuration file, and apply the change to your cluster using the apigeectl command.

Apigee hybrid also supports JMX and Jolokia users to authenticate and communicate with the Cassandra JMX interface and JMX API.

[Using Kubernetes secrets to configure Cassandra authentication](#)

Securing traffic to the backend

- [Message Processor to backend services](#) can be configured to use TLS/mTLS based on requirements.
- Create a [TargetServer](#) configuration using the hybrid UI or Apigee API.
- You can set up [1-way or 2-way TLS](#) between the MP and backend service in the TargetServer configuration.
- Keys and Certificates are managed by creating [Keystores & Truststores](#) in the hybrid UI or via the Apigee API.



You can use TLS to secure traffic to backend servers. You can also use OAuth access tokens in your API proxies to implement authentication and authorization.

Use the Apigee hybrid UI or API to create a TargetServer definition. Configure it to use 1-way or 2-way TLS, depending on your requirements.

To store TLS keys and certificates, create keystores and truststores in the hybrid UI or API.

Securing the runtime plane

- A typical Apigee hybrid installation is made of multiple pods. Each of these pods requires specific access to ports, but not every pod needs to communicate with every other pod.
 - In Kubernetes, you can lock down the ingress and egress connections created to and from the pods by using [network policies](#).
- Follow these [guidelines](#) to harden the security for your GKE cluster running Apigee hybrid.
- Review the [Security overview](#) document that provides an overview of each layer of your Kubernetes infrastructure and explains how you can configure its security features to best meet your needs.



You can secure the Apigee hybrid runtime plane by also implementing Kubernetes network policies in your cluster.

A network policy is a specification of how groups of pods are allowed to communicate with each other and other network endpoints.

A typical Apigee hybrid installation is made of multiple pods that require access to specific ports, and not every pod needs to communicate with every other pod.

Use network policies to restrict communication between pods and to pods that have access outside the Kubernetes network.

You can further harden the security of your runtime cluster by following the recommended guidelines of your Kubernetes infrastructure provider.

Agenda

Infrastructure Security

Data Storage and Encryption

RBAC



In this lecture, we will discuss the types of data entities used in Apigee hybrid and the mechanisms used to collect, store, and make that data available for use.

Apigee hybrid data

Apigee hybrid uses various types of data:

- Data managed by you and stored in Cassandra in the [runtime plane](#):
 - Key management system (KMS) data, including developers, apps, API products, OAuth tokens, authorization codes, and API keys
 - Key value map (KVM) data
 - Environment-scoped cache data
 - Quota data, including buckets and counters
- Data managed by Apigee and stored in the [management plane](#):
 - API proxies
 - Target servers
 - Truststores and keystores
 - Audit logs



Apigee hybrid stores API runtime data in the Cassandra data store in the Kubernetes cluster that you manage.

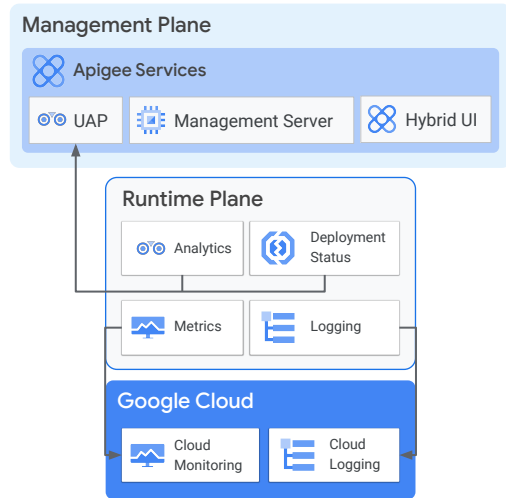
This data includes KMS entities like developers, apps, API products, and API keys. It also includes key value map entries, cache objects, and quota counters.

Apigee manages and stores API proxies, Target server configuration, TLS keys, and certificates in truststores and keystores in the management plane on Google Cloud. This data is downloaded by the synchronizer and made available in the runtime plane.

Apigee hybrid also stores audit logs in the management plane.

Metrics and analytics data

- Apigee hybrid collects three categories of informational data:
 - [Logging data](#)
 - [Metrics data](#)
 - [Analytics and Deployment Status data](#)
- Logging and Metrics data is used for debugging and troubleshooting.
 - Data is streamed and accessed via Cloud Logging, Cloud Monitoring, or other tools.
- Analytics and deployment status data is batched and accessed by the hybrid UI and Apigee APIs.



Apigee hybrid collects logging, metrics, analytics, and proxy deployment status data during its normal operation.

Logging and metrics data is stored on Google Cloud under your project and accessed via the Cloud Logging and Cloud Monitoring user interfaces in Google Cloud Console.

Analytics, proxy deployment status, and trace data are stored in the Apigee management plane on Google Cloud.

Data encryption

- By default, the following data is stored **encrypted** in the hybrid runtime plane:
 - Key Management Service (KMS) data
 - Key-value map (KVM) data
 - Cache data
- Encryption keys for KMS, KVM, and cache have **scope** that determines the set of resources that can be encrypted.
- Hybrid uses default **data encryption keys** (DEKs) and does not require any additional configuration.

Encryption Key	Scope
KMS data	Organization only
KVM data	Organization or Environment
Cache	Environment only



KVM, KMS, and cache data in the runtime plane are stored encrypted by default.

Apigee hybrid provides a set of default Base64-encoded keys that are used to encrypt KVM, KMS, and cache data.

The Apigee hybrid installer stores the keys in the runtime plane as Kubernetes Secrets and uses them to encrypt your data with AES-128 standard encryption.

The keys are under your control and never used by the hybrid management plane at any time.

Data encryption keys (DEKs)

- The default DEKs can be replaced with your own keys by specifying them as base64-encoded strings in the overrides.yaml file and applying them to your cluster.
- The hybrid installer stores the DEKs in the runtime plane as [Kubernetes Secrets](#) and uses them to encrypt data with AES-128 encryption.
- **DEKs cannot be changed or rotated** once applied to the cluster.
- It is recommended to use your own DEKs for production clusters.

overrides.yaml

```
defaults:
  org:
    ...
    kmsEncryptionKey: base64-encoded-key
    kvmEncryptionKey: base64-encoded-key
    ...
  Env:
    ...
    kmsEncryptionKey: base64-encoded-key
    cacheEncryptionKey: base64-encoded-key
    ...
```



The default data encryption keys can be replaced with your own keys by configuring them in your overrides.yaml configuration file.

You cannot change the encryption keys after the runtime data is encrypted.

We recommend that you use your own data encryption keys for production installations.

[How to create an encoded key](#)

Kubernetes secrets

- A [Secret](#) is an object that contains a small amount of sensitive data, such as a password, a token, or a key.
- It is used with a *pod* as files in a volume mounted on one or more of its containers using the *secret* volume type.
- It can also be exposed as environment variables to the container.
- The *secret* volume type is backed by a RAM-backed filesystem, so they are never written to non-volatile storage.
- To list all the secrets in the hybrid *apigee* namespace, run:

```
$kubectl get secrets -n apigee
```

- To view data for a specific secret run:

```
$kubectl get secret {secret-name} -o yaml
```

- You can also create secrets and access them in an API proxy as flow variables.



Kubernetes Secrets let you store and manage sensitive information, such as passwords, OAuth tokens, and keys.

A Secret is used with a pod as files in a volume mounted on its containers, or it can be exposed as an environment variable to the container.

You can create Kubernetes secrets and access them in an API proxy as flow variables in Apigee hybrid.

[Storing data in a Kubernetes secret](#)

Demo

Implement API key verification
in Apigee hybrid



Lets view a short demo on using the trace feature in Apigee hybrid.

Agenda

Infrastructure Security

Data Storage and Encryption

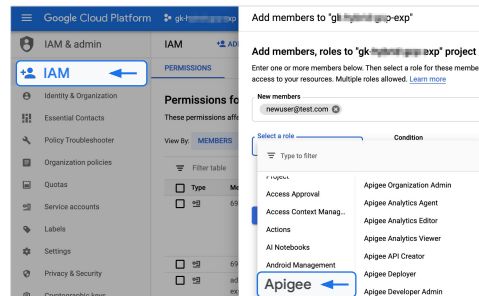
RBAC



In this lecture, we will discuss how role-based access control, or RBAC, is implemented in Apigee hybrid.

Users and roles

- In Apigee hybrid, a [user](#) represents an authenticated Google Cloud account.
- The account must have access to a [Google Cloud project](#) and [Apigee hybrid organization](#) and the resources within them.
- [Access control](#) for the user account is implemented by granting roles at the project level and at the environment level within the hybrid organization.
- The resources that a user can access depend on roles assigned in the project and hybrid environment.
- User accounts can be [federated](#) between Google Cloud and Active Directory.



An Apigee hybrid user is represented by an authenticated Google Cloud account.

The user is granted access by using roles at the project level, which are inherited in all hybrid environments by default.

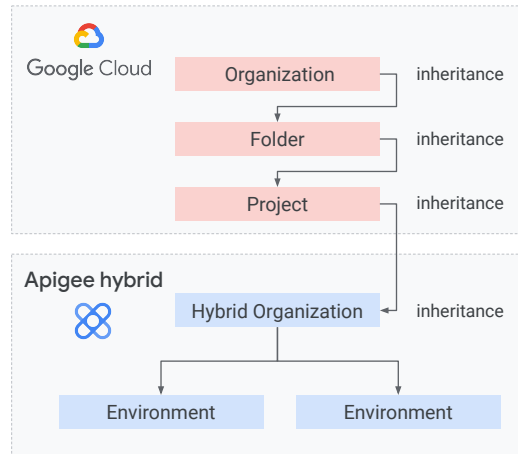
You can refine a user's access by using the hybrid UI to specify one or more roles per environment in your hybrid organization.

The user acts in that role only for the selected environment.

You can federate user accounts between Active Directory and Google Cloud Identity or Google Workspace.

Role inheritance

- Roles assigned to users at the project level are **inherited** by the hybrid organization resource.
- Permissions set on the project apply to all environments, unless you specify more fine-grained permissions for each environment.
- If you assign a role to a user for a given environment, the user has that role only for the selected environment.
- It is recommended to set a **minimum level of permissions** for a user at the project level; then refine those roles for each environment in the hybrid organization as needed.

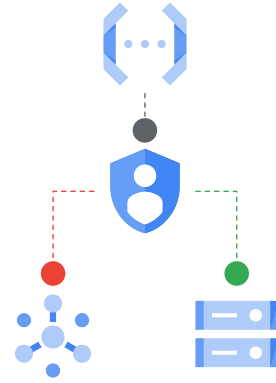


A user inherits roles from the Google Cloud project in the hybrid organization and environments.

You typically assign roles with the least permissions to the user in the project, and then refine those roles in the hybrid organization and individual environments based on the functionality that the user can perform.

Principle of least privilege

- The [principle of least privilege](#) states that a resource should only have access to the exact set of resources it needs in order to function.
 - For example: A database backup service should be restricted to read-only permissions on the one database being backed up.
- To implement this principle, consider the following best practices:
 - Avoid excessive use of broad primitive roles like *Owner*, *Editor*.
 - Use predefined or custom roles for granular access to resources.
 - Assign roles to groups, instead of to individuals, to easily manage user attrition.
 - Reduce default service account permissions and instead use custom dedicated service accounts.
 - Use networking features (firewalls, VPCs) to control access to project resources.



It is a best practice to follow the principle of least privilege when assigning roles that allow users to access resources.

The principle states that a user or resource should only have access to the exact set of resources needed in order to perform a given function.

Avoid granting users broad primitive roles like Owner or Editor.

To manage user attrition, assign roles to groups instead of to individual users. This makes it efficient to change user access by simply managing the user's group membership.

Consider using predefined or custom roles to provide more granular access to resources.

[Practicing the principle of least privilege](#)

Apigee-specific hybrid roles

Role	Description
Analytics Editor	Creates and analyzes reports on API proxy traffic for an Apigee organization.
Analytics Viewer	Views analytics data for an organization.
API Creator	A developer who creates and tests API proxies. Deprecated. Use API Admin role.
Deployer	Deploys and undeploys API proxies to the runtime. Deprecated. Use Environment Admin role.
Developer Admin	Manages application developer access to apps.
Org Admin	A super user who has full access to all Apigee resources in the Apigee organization.
Read Only Admin	An administrator who can run reports and view everything in the Apigee organization without the ability to create or change anything.
API Admin	A developer that creates and tests API proxies, read API products and apps, and edit shared flows, and KVMs. This role replaces the API Creator and Deployer roles.
Environment Admin	Deploys API proxy revisions and edit environment configuration.

[Apigee pre-defined roles and permissions](#)



Apigee hybrid provides a set of predefined roles that you can assign to users based on their job function.

The org admin role has full access privileges to all resources in the Apigee hybrid organization.

API developers can be assigned the API Admin and Environment Admin roles, and a business user can be assigned the Analytics Viewer or Analytics Editor role.

A user with the Developer Admin role has edit access to Apps, API keys, and App developers in hybrid.

Note: The API Creator and Deployer roles are deprecated. Use the API Admin and Environment Admin roles instead.

Assigning roles

- A new user is first added to the project and granted one or more roles. This gives the user the **same role in all environments** in the hybrid organization.
- In the hybrid UI, you can **refine** a user's access by specifying one or more **roles per environment** in your hybrid organization.
 - Select **Admin > Environments > Access** in the left navigation menu, and click **+Grant Access** to add user roles for the environment.

TYPE	EMAIL	ROLES
Individual	ringo@starr.com	Administrator
Individual	john@lennon.org	Administrator, DevOps, Developer
Service Account	service-42@compute-gcp...	Developer
Team	wings-team@paulmccartney.info	DevOps
Team	george@wilburys.biz	Developer



A new user who is added to your project is granted one or more roles based on job function.

By default, the new user inherits the same role for all the project resources, including the hybrid organization and environments.

You can use the Apigee hybrid UI to refine the user roles for each hybrid environment.

When defined for a hybrid environment, the user only functions in the specified role in that environment.

Managing roles using the API

- You can use Apigee APIs, Google Cloud APIs, Google Cloud Console and the gcloud tool to manage user roles in hybrid.
- Use the [/organizations](#) API to list the Apigee organizations and associated projects that you have permission to access.
- Use the [/organizations/ */environments/ *:getIamPolicy](#) and [/organizations/ */environments/ *:setIamPolicy](#) operations to get and update the IAM policy bindings for user roles and resources.
- The above operations use a [Policy](#) object that is a collection of bindings. A binding binds one or more members (user account) to a single role.

```
{
  "bindings": [
    {
      "role":
"roles/resourcemanager.organization
Admin",
      "members": [
        "user:mike@example.com",
        "group:admins@example.com",
        "domain:google.com",
        "serviceAccount:my-project-id@appsp
ot.gserviceaccount.com"
      ]
    },
    ...
  ]
}
```



You can use the Google Cloud Console, the Apigee API, or the gcloud tool to set a user's roles in Apigee hybrid.

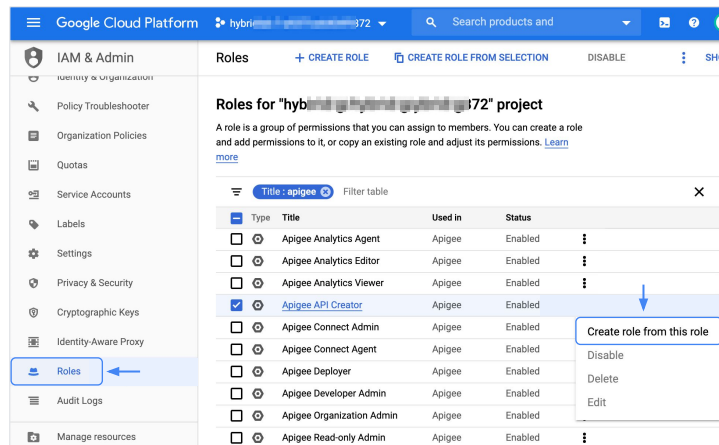
Use the `getIamPolicy` and `setIamPolicy` operations of the `/organizations/environments` Apigee API to get and set a user's role in a hybrid environment.

The API uses a Policy object that contains a collection of role bindings assigned to a set of user accounts.

[Manage users, roles, and permissions using the API | Apigee](#)

Creating custom roles

- Using Google's IAM, you can also create custom roles for use in Apigee hybrid.
- From the Roles page in the Cloud Console, you can create a new custom role from an existing role, or create one by setting permissions individually.



Apigee hybrid supports the creation of custom roles using Google's Identity and Access Management.

You can use the Cloud Console to create a custom role from an existing Apigee hybrid role.

Assigning custom role permissions

- Add or remove permissions for the custom role when creating it.
- After the custom role is created, you can assign users in your hybrid project to that role using the UI or API.

Create Role

ID *
CustomApigeeAPICreator

Role launch stage
General Availability

[+ ADD PERMISSIONS](#)

35 assigned permissions

Filter table

Permission	Status
<input checked="" type="checkbox"/> apigee.apiproductattributes.get	Supported
<input checked="" type="checkbox"/> apigee.apiproductattributes.list	Supported
<input checked="" type="checkbox"/> apigee.apiproducts.get	Supported
<input checked="" type="checkbox"/> apigee.apiproducts.list	Supported
<input type="checkbox"/> apigee.apps.get	Supported
<input type="checkbox"/> apigee.apps.list	Supported
<input checked="" type="checkbox"/> apigee.environments.get	Supported



In the IAM user interface of the Cloud Console, add or remove permissions for the custom role.

After that, use the Cloud Console UI or gcloud API to assign the custom role to users in your project.



Review: Security

David Mehi

Customer Success Engineer, Google Cloud

In this module, you learned how communications between the Apigee hybrid runtime and management plane components are secured. You also learned how components within the runtime plane are secured.

You learned about the different types of data used in hybrid and their secure storage mechanisms. You also learned about role-based access control used for users of Apigee hybrid.

In the next module, you will learn about capacity planning for Apigee hybrid and how the platform can be scaled to handle your API traffic.