



# Deployment and Environment Management

Hansel Miranda  
Course Developer, Google Cloud



In this module you will learn about API proxy and environment management on the hybrid platform.

You will complete a lab to add a new environment to your hybrid runtime plane in Google Kubernetes Engine.

# Agenda

## Build and Deploy API Proxies

Tracing and Debugging

Developer Portal Solutions

Managing Environments

Lab



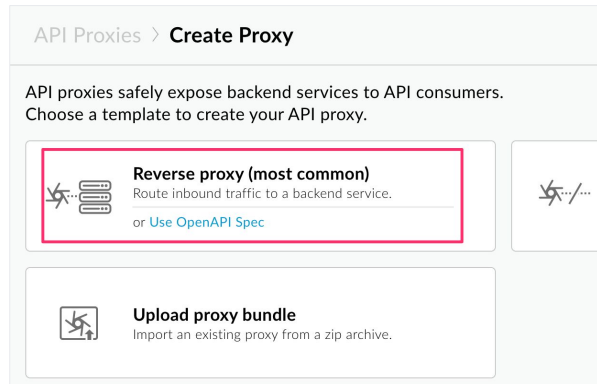
In this lecture we will discuss how to build and deploy API proxies to your hybrid runtime environment.

In later lectures, you will learn how to trace and debug your API proxies and learn about the various developer portal solutions that are available for use.

We will discuss how to manage environments and complete a lab to add an environment in Apigee hybrid.

## Managing API proxies

- Use the [proxy creation wizard](#) in the hybrid UI to create API proxies.
- Any changes to the proxy can be saved incrementally before deployment.
- Use the [hybrid UI](#) or [Apigee API](#) to deploy the proxy revision to your runtime cluster.
- Deploying proxy changes always generates a new revision.
- A deployed proxy revision is [immutable](#).



You use the Apigee hybrid user interface or Apigee API to create an API proxy.

The hybrid UI has a proxy creation wizard that guides you through the steps to create the API proxy.

You can use the wizard to set optional security policies and the target API endpoint, and then deploy the proxy to one or more existing hybrid environments. The UI also allows you to upload an API proxy bundle that you have already created.

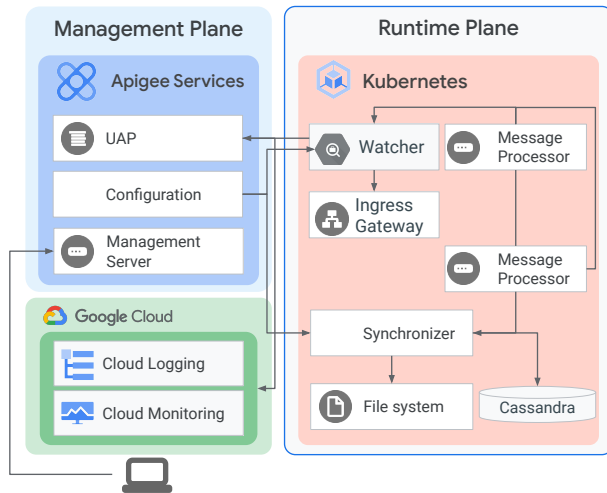
During API development, you iteratively make and save changes to your proxy. When development is complete, you use the hybrid UI or Apigee API to deploy the proxy.

When you deploy a proxy to an environment, a proxy revision is generated. A proxy revision is immutable.

Any future deployment of API proxy changes to an environment always generates a new proxy revision.

## Deployment process

- Hybrid propagates new revisions of your API proxies via the [Synchronizer and Message Processors \(MPs\)](#).
- Synchronizer polls the management plane at regular intervals. This is configurable, with a minimum interval of 60 seconds.
- MPs [act independently](#) of each other. Any MP can be in a different state than the other MPs.
- The proxy deployment process uses an [eventually consistent](#) deployment model.
- MPs send [deployment status](#) to Watcher, which then updates the status to UAP, Cloud Logging, and Cloud Monitoring.



The API proxy deployment process involves the management plane and some hybrid runtime plane components.

The synchronizer component in the runtime plane polls the management plane at regular intervals and downloads any new or modified API proxy and other configuration data from the management plane.

This downloaded data is called a contract and is made available to the runtime plane.

The runtime message processor component retrieves the contract from the synchronizer component and deploys the proxy. It then sends deployment status data to the Watcher component in the runtime plane.

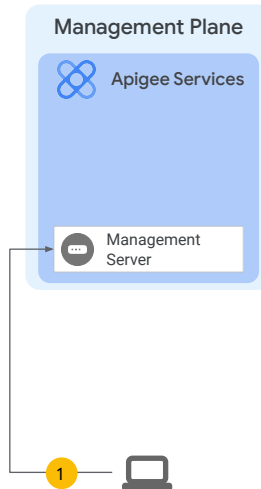
Watcher forwards the deployment status to Google Cloud and to the management plane; from there you can use the hybrid UI to view proxy deployment status or use the Apigee API to retrieve the status information.

The synchronizer polling frequency is configurable with a minimum frequency of 60 seconds.

The Message Processor polling frequency is not configurable and is less than 60 seconds.

## Deployment operation

1. An API developer uses the hybrid UI or API to initiate the deployment process via the management server.

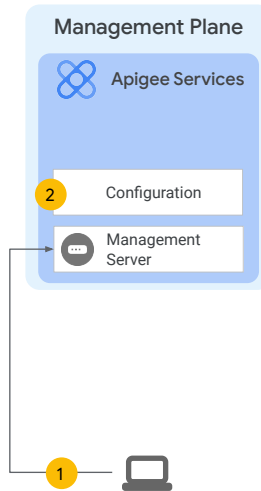


Lets review the sequence of steps in the API proxy deployment process.

An API developer deploys a proxy using the hybrid UI or API. The request is sent to the management server running in the Apigee hybrid management plane on Google Cloud.

## Deployment operation

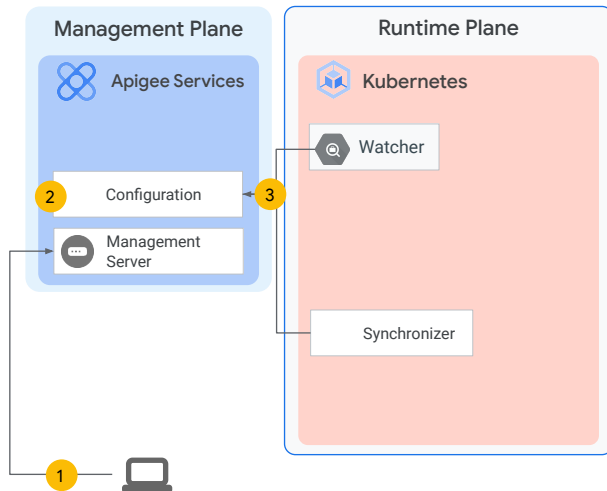
1. An API developer uses the hybrid UI or API to initiate the deployment process via the management server.
2. The management plane generates a new configuration for the MPs and Ingress gateway.



The management plane generates a new configuration for the message processors and Ingress gateway.

## Deployment operation

1. An API developer uses the hybrid UI or API to initiate the deployment process via the management server.
2. The management plane generates a new configuration for the MPs and Ingress gateway.
3. The synchronizer and watcher runtime components check the management plane for updated configuration.

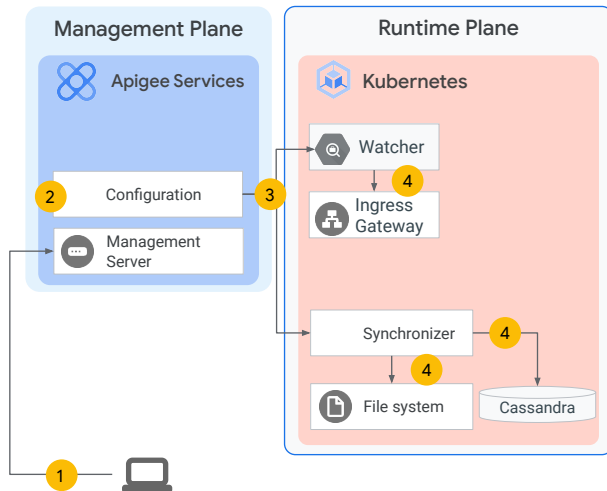


The synchronizer component in the runtime plane polls the management plane at regular intervals for changes to the API proxies and related data.

The watcher component in the runtime plane also checks the management plane for configuration changes to basepath and virtual host information.

## Deployment operation

1. An API developer uses the hybrid UI or API to initiate the deployment process via the management server.
2. The management plane generates a new configuration for the MPs and Ingress gateway.
3. The synchronizer and watcher runtime components check the management plane for updated configuration.
4. Synchronizer downloads the bundle to the file system and Cassandra, and Watcher updates the ingress gateway.



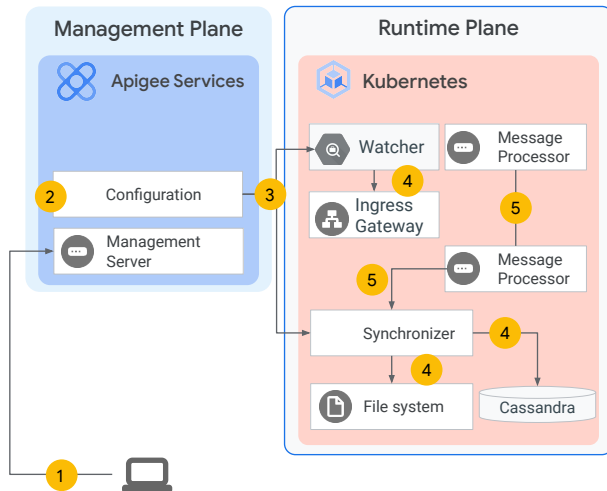
The configuration data, including API proxy bundles, is retrieved by the synchronizer and stored locally on the file system. It is also saved to the Cassandra datastore.

In addition to the proxy bundles, the configuration data downloaded by the Synchronizer includes shared flows, flow hooks, shared resources, target server definitions, TLS settings, key-value map names, and data masks.

The watcher component updates the ingress gateway with the modified routing configuration.

## Deployment operation

1. An API developer uses the hybrid UI or API to initiate the deployment process via the management server.
2. The management plane generates a new configuration for the MPs and Ingress gateway.
3. The synchronizer and watcher runtime components check the management plane for updated configuration.
4. Synchronizer downloads the bundle to the file system and Cassandra, and Watcher updates the ingress gateway.
5. Message processors deploy any changed bundle configurations.

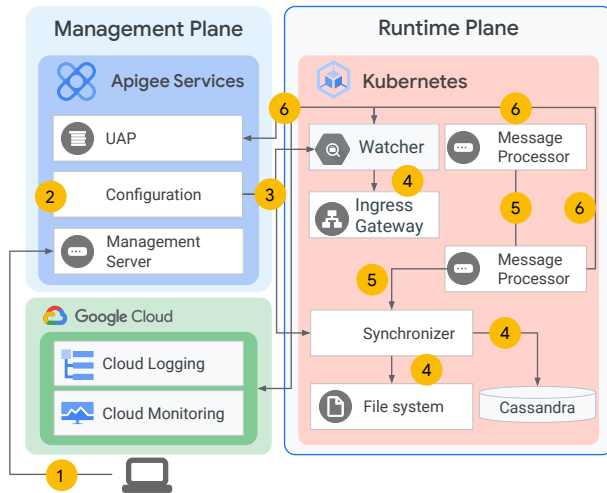


The Message Processor runtime components poll the synchronizer at regular intervals to detect any changed bundles or configuration.

The Message Processors then load the changed proxy bundles and configuration from the synchronizer and deploy them.

## Deployment operation

1. An API developer uses the hybrid UI or API to initiate the deployment process via the management server.
2. The management plane generates a new configuration for the MPs and Ingress gateway.
3. The synchronizer and watcher runtime components check the management plane for updated configuration.
4. Synchronizer downloads the bundle to the file system and Cassandra, and Watcher updates the ingress gateway.
5. Message processors deploy any changed bundle configurations.
6. Deployment status is reported to Watcher, the management plane, and Google Cloud.



The message processor runtime components then send the status of the proxy deployment to the Watcher component in the runtime plane.

Watcher reports the status of the message processors and ingress gateway to the management plane and to Cloud Logging and Cloud Monitoring in Google Cloud.

## Deploying API proxies via the UI

- Deploy an API proxy using the hybrid UI in the management plane.
- Deploy a specific proxy revision to a given [environment](#).

Deployments	
Environment	Revision
prod	3
Test	5
Proxy Endpoints	4
	3
	2
	1
Name	Undepoly
default	







You use the hybrid user interface to deploy API proxies. The hybrid UI runs in the management plane.

You can deploy a specific proxy revision to an environment by selecting it from the revision drop-down list in the proxy overview page.


You can also deploy the proxy revision from the develop tab.

## Proxy deployment status

The hybrid UI indicates the proxy revision deployment status in terms of the number of updated/synced MP pods and displays a status icon:

-  The proxy has not been deployed to the selected environment.
-  There are no errors or warnings for that proxy in the selected environment.
-  Indicates possible proxy configuration errors.
-  An error with one or more pods in the selected environment requires your attention.

Deployments

Environment	Revision	Status
prod	1	<a href="#">Details</a>
Test	<div><b>Environment: test</b>  Revision 1 fully deployed</div>	

Proxy Endpoints

Name
default



The status column in the deployments section of the overview page provides details on the status of the proxy deployment.

The status indicator provides information on the success or failure of the proxy deployment.

If issues are encountered during the deployment, information about the cause of the issues is included in the status details.

A warning or error indication may require you to further troubleshoot the issue with the proxy configuration or runtime plane components.

In Apigee hybrid, proxy deployment is asynchronous, and the status may take some time to update in the UI and API.

## Deploying proxies via the Apigee API

- Deploy an API proxy to a specific environment in your Apigee organization by issuing a POST request to the [/organizations/\\* /environments/\\* /apis/\\* /revisions/\\* /deployments](#) Apigee API.
- \$TOKEN must be set to your OAuth 2.0 access token.
- To [undeploy](#) the proxy revision, issue a DELETE request to the same API.
- View the full set of operations exposed for this API [here: organizations.environments.apis.revisions.deployments](#)

### Sample Request:

```
curl
"https://apigee.googleapis.com/v1/organizations/{org}/environments/{env}/apis/{apiproxy}/revisions/{revision}/deployments" -X POST -H
"Content-type:application/x-www-form-urlencoded" -H "Authorization: Bearer $TOKEN"
```

### Sample Response:

```
{
  "environment": "test",
  "apiProxy": "helloworld",
  "revision": "1",
  "deployStartTime": "1559149080457"
}
```



You can also use the Apigee API to deploy an API proxy.

You deploy an API proxy revision to a specific environment in your Apigee organization by issuing a POST request to the deployments resource.

The organization name, environment name, api proxy name, and proxy revision number must be passed in the request as path parameters.

An OAuth token generated using your Apigee org admin credentials must be passed in as a bearer token header in the request.

## Proxy deployment status via the API

- Retrieve deployment status of an API proxy by issuing a GET request to the [/organizations/\\* /environments/\\* /apis/\\* /revisions/\\* /deployments](#) Apigee API.
- The API returns the deployment details of an API proxy revision and actual state as reported by the runtime pods.
- deploymentStatus can be [deployed](#), [pending](#), or [error](#).
- View the full details for this API here: [organizations.environments.apis.revisions.deployments.get](#)

### Sample Request:

```
curl -H "Authorization: Bearer $TOKEN"  
https://apigee.googleapis.com/v1/organizations/{org}/environments/{env}/apis/{api}/revisions/{rev}/deployments
```

### Sample Response:

```
{  
  "environment": "test",  
  "apiProxy": "helloworld",  
  "revision": "1",  
  "deployStartTime": "1559149080457",  
  "pods": [  
    {  
      "podName": "apigee-runtime-myorg-test...",  
      "deploymentStatus": "deployed",  
      ...  
    }  
  ],  
  ...  
}
```



You can retrieve the deployment status of an API proxy in an environment by issuing a GET request to the deployments resource.

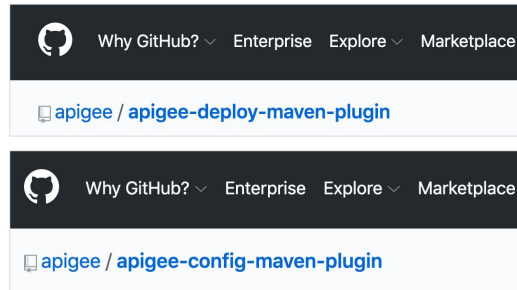
As in the previous API call, the same path parameters and OAuth token must be provided in the request.

An OAuth token generated using your Apigee org admin credentials must be passed in as a bearer token header in the request.

The API response includes information about the API proxy revision and the deployment status as reported by the runtime pods.

## Deploying proxies via command-line tools

- The [Apigee Maven deploy plugin v2.0.0](#) supports command-line deployment of API proxies to Apigee hybrid.
- The [Apigee Maven config plugin v2.0.0](#) supports command-line deployment of configuration objects needed by API proxies to Apigee hybrid.
- The [apigeecli](#) tool allows you to manage API proxies and other entities on Apigee hybrid.
- A sample [CI/CD use case](#) uses the Apigee maven deploy and config plugins mentioned above.



A few command-line tools are available that support API proxy deployment to Apigee hybrid.

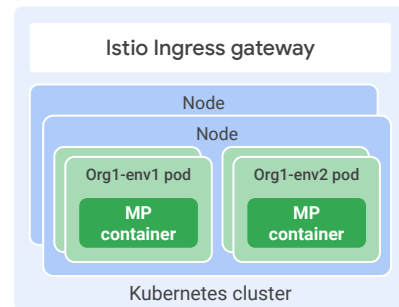
The Apigee maven deploy and config plugins enable you to deploy API proxies and configuration data.

These tools can be used as part of a continuous integration and continuous delivery process in your API development project.

The apigeecli tool is another command-line tool that can be used to create and manage API proxies, products, and other entities used in Apigee hybrid.

## Deployment considerations

- An API proxy revision can be deployed to one or more environments in the organization.
- A best practice is to limit the number of proxies and shared flows (max 50) that are deployed to a given environment.
- To choose the number of proxies to deploy per environment, consider the following:
  - **MP boot-up time**: increases based on the number of proxies deployed, so the time during autoscaling increases.
  - **Noisy neighbor problem**: when one proxy crashes, all the proxies deployed to the same MP become unavailable while the MP restarts.



When deploying API proxies in Apigee hybrid, try to limit the number of proxies deployed to the same runtime environment.

Consider the message processor boot time that increases with the number of proxies deployed, thus affecting the time taken for scaling the runtime pods in your cluster.

Another consideration is the noisy neighbor problem: when one proxy crashes a message processor, all of the proxies deployed to the same message processor become unavailable while it restarts. By limiting the number of proxies deployed to an environment, you minimize the impact of a single proxy crashing.

Having multiple environments that share the same host alias allows you to divide up proxies by environment without causing additional friction for your API consumers.

There is a limit on the number of API proxy revisions that are retained in history. Review the Apigee hybrid product limits below:

[Hybrid product limits](#)

---

# Agenda

Build and Deploy API Proxies

Tracing and Debugging

Developer Portal Solutions

Managing Environments

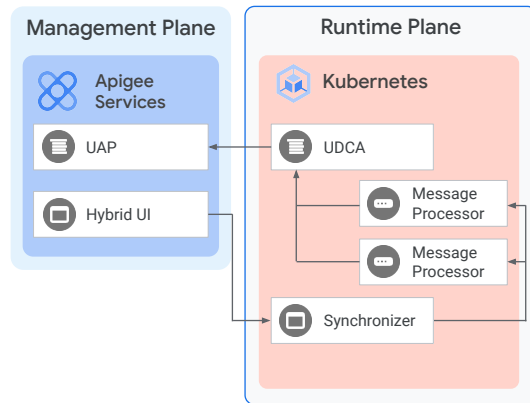
Lab



In this lecture we will discuss how to debug your API proxy using the trace tool in Apigee hybrid.

## Tracing API proxies

- Apigee hybrid lets you trace and debug the entire request/response flow of your API proxies.
- Trace in hybrid is **asynchronous**. As a result, hybrid trace data is not available in real time.
- There is a slight delay in the availability of trace data as compared to using trace with Apigee Edge for the Cloud.
- Trace data is persisted in the management plane for up to 24 hours. This duration cannot be changed.
- A trace or debug session on a given MP ends when its **timeout** is reached or the **maximum number of requests** has been processed in that session.



Trace is a tool for troubleshooting API proxies running on Apigee. Trace lets you probe the details of each step through an API proxy flow.

Trace collects data from the request and response and proxy flow variables, allowing you to inspect the state of the API proxy as it executes policies at runtime.

Trace is asynchronous in Apigee hybrid, so there is a slight delay in the availability of trace data in the hybrid UI or Apigee API.

Lets review the steps involved when using trace in Apigee hybrid.

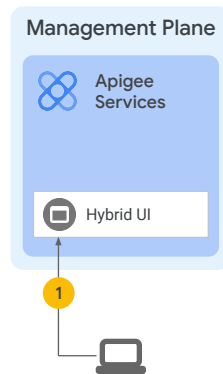
When you start a trace session, two properties determine when it ends on a given Message Processor:

- **timeout:** The length of time that a session collects data for. The default is 300 seconds (or 5 minutes). The maximum value is 600 seconds (or 10 minutes).
- **count:** The maximum number of requests that are recorded in a single session *per Message Processor*. Because the number of Message Processors in most clusters is variable, the effects of the count can be unpredictable. Apigee does not recommend customizing this setting.

You cannot configure a trace session's timeout in the hybrid UI. You *can* configure this timeout with the [trace APIs](#) when you create a new session.

## Trace operation

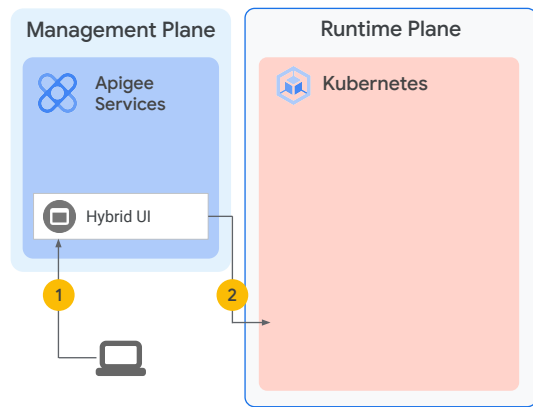
1. You create a [trace session](#) request via the hybrid UI or Apigee API.



You initiate a trace session via the Apigee hybrid UI or Apigee API.

## Trace operation

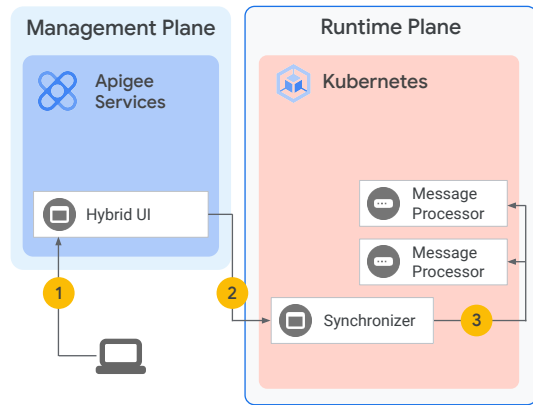
1. You create a [trace session](#) request via the hybrid UI or Apigee API.
2. The management plane issues the trace request to the runtime plane using the pub/sub model.



Using a pub/sub model, the management plane sends the trace request to the runtime plane.

## Trace operation

1. You create a [trace session](#) request via the hybrid UI or Apigee API.
2. The management plane issues the trace request to the runtime plane using the pub/sub model.
3. Synchronizer subscribes to trace session request notifications. It receives the request and propagates it to the Message Processors (MPs).

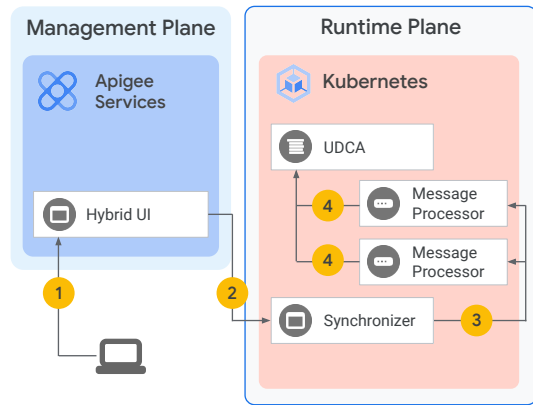


The synchronize runtime component subscribes to notifications about trace session requests.

It receives a request and propagates it to the runtime message processor components.

## Trace operation

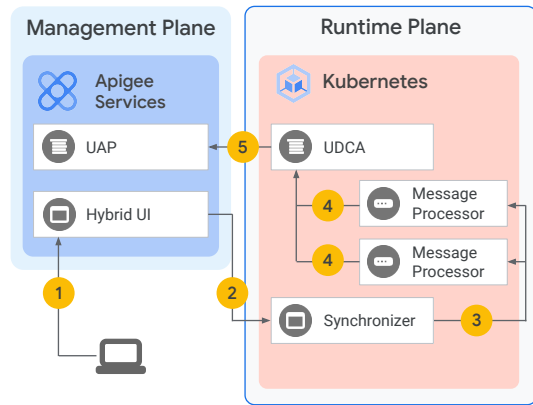
1. You create a [trace session](#) request via the hybrid UI or Apigee API.
2. The management plane issues the trace request to the runtime plane using the pub/sub model.
3. Synchronizer subscribes to trace session request notifications. It receives the request and propagates it to the Message Processors (MPs).
4. The MPs collect trace data for the API requests/responses and stream it to the UDCA data collection pod.



The message processor traces the requests as they are received and collect and stream the trace data to the UDCA data collection pod in the cluster.

## Trace operation

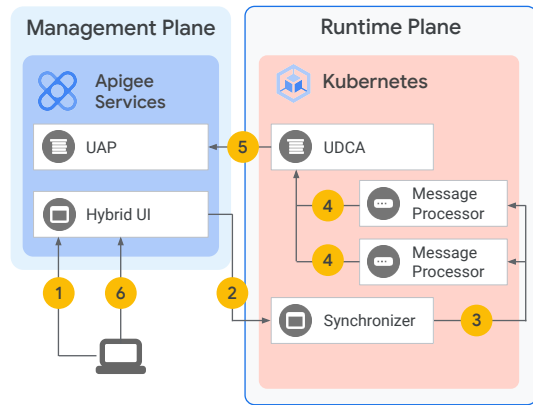
1. You create a [trace session](#) request via the hybrid UI or Apigee API.
2. The management plane issues the trace request to the runtime plane using the pub/sub model.
3. Synchronizer subscribes to trace session request notifications. It receives the request and propagates it to the Message Processors (MPs).
4. The MPs collect trace data for the API requests/responses and stream it to the UDCA data collection pod.
5. The UDCA collects this data and sends it to the UAP service running in the management plane.



The UDCA runtime component collects the trace data and sends it to the UAP service in the management plane, which makes it available to the hybrid UI and API.

## Trace operation

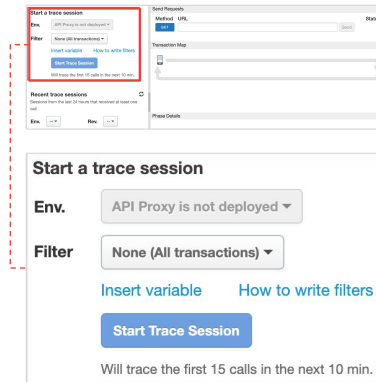
1. You create a [trace session](#) request via the hybrid UI or Apigee API.
2. The management plane issues the trace request to the runtime plane using the pub/sub model.
3. Synchronizer subscribes to trace session request notifications. It receives the request and propagates it to the Message Processors (MPs).
4. The MPs collect trace data for the API requests/responses and stream it to the UDCA data collection pod.
5. The UDCA collects this data and sends it to the UAP service running in the management plane.
6. You then access the trace session data via the hybrid UI or Apigee API.



You can then view the trace session in the hybrid UI or use the Apigee API to retrieve trace session data.

## Using Trace in the hybrid UI

- Trace sessions can be initiated from the [Trace](#) tab in the proxy deployment view.
- Select the [environment](#) and [proxy revision](#) to be traced.
- Optionally, add a [filter](#) to trace only those requests that match the filter condition.
- You can download active trace session data from the UI and delete the data if it is no longer needed.
- You can start additional trace sessions in parallel.



You can initiate trace sessions from the trace tab in the Apigee hybrid UI by selecting the environment and proxy revision.

You can selectively trace API requests by applying an optional filter that uses conditions on request metadata and flow variables.

For further troubleshooting and analysis, you can also download trace session data for viewing in the hybrid UI.

## Trace session data

- Trace session data is available to download or view in the UI for 24 hours and is then deleted.
- You can download a file containing the trace data for offline viewing.
- Trace data downloaded via the hybrid UI includes all transactions in the session in a [Messages](#) array.
- Trace data downloaded via the API contains a [single, specified transaction](#) per API request.
- You can upload the previously saved trace session data into the [Offline Trace](#) view in the hybrid UI to view trace results.

```
{
  "DebugSession": {
    "Retrieved":
      "2019-06-08T13:08:13.395Z",
    "Organization": "myorg",
    "Environment": "prod",
    "API": "myproxy",
    "Revision": "1",
    "SessionId":
      "a2a271aa-4242-4ac6-97cb-aec8dcb115a9"
  },
  "Messages": [
    {...}
  ]
}
```



Active trace session data can be viewed in the hybrid UI for a period of 24 hours.

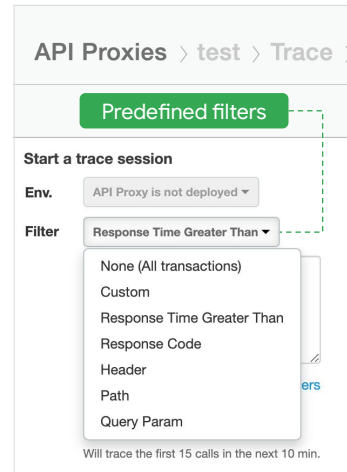
If further analysis is required, a trace session can be downloaded using the hybrid UI or Apigee API.

Use the offline trace view to upload previously downloaded trace sessions for viewing in the hybrid UI.

For troubleshooting purposes, Apigee support personnel must be added to the hybrid organization to view offline trace sessions.

## Trace filters

- When you create a new trace session, you can use the hybrid UI or Apigee API to add a filter for that session.
- A filter is a [conditional statement](#) that the hybrid evaluates against request and response messages to determine whether trace data should be captured in the debug session.
- Filters support the same syntax that is used in Apigee Edge conditions, such as grouping, logical AND and OR, path expressions, operators, and literals.
- Filters can access all flow variables, including custom variables defined in your API proxy.
- A set of [predefined filters](#) is available to use in the hybrid UI.



A filter is a conditional statement that is applied to API transactions to determine whether trace data should be captured in that session.

You create a filter by using Apigee system or custom flow variables and combining them into logical expressions with boolean operators.

Apigee hybrid provides a set of predefined filters that can be used in trace sessions.

These filters use common data elements, such as response time and response code values, request header, and query and path parameter values, which can be applied to your trace sessions.

## Using Trace via the API

To initiate a debug session via the Apigee API, follow these steps:

1. Generate an OAuth token using your Google account credentials.
2. Send a POST request to the [Debug Sessions API](#).
  - a. Optionally set a session's timeout in seconds in the request (but not after a session is started). Default session timeout is 5 mins (max is 10 mins).
  - b. Optionally set filter conditions.

If the request is successful, the API responds with a JSON object that describes the newly created debug session.

[Using Trace via the API](#)



### Sample request:

```
curl -H "Authorization: Bearer $TOKEN" -X "POST"

https://apigee.googleapis.com/v1/organizations/{org}/environments/{env}/apis/{api_proxy}/revisions/{rev}/debugsessions -d '{
  "timeout": "42",
  "filter": "filter_condition"
}'
```

### Sample response:

```
{
  "name": "56382416-c4ed-4242-6381-591bbf2788cf",
  "validity": 300,
  "count": 10,
  "tracesize": 5120,
  "timeout": "42"
}
```

You can initiate a trace session by sending a POST request to the debug sessions API.

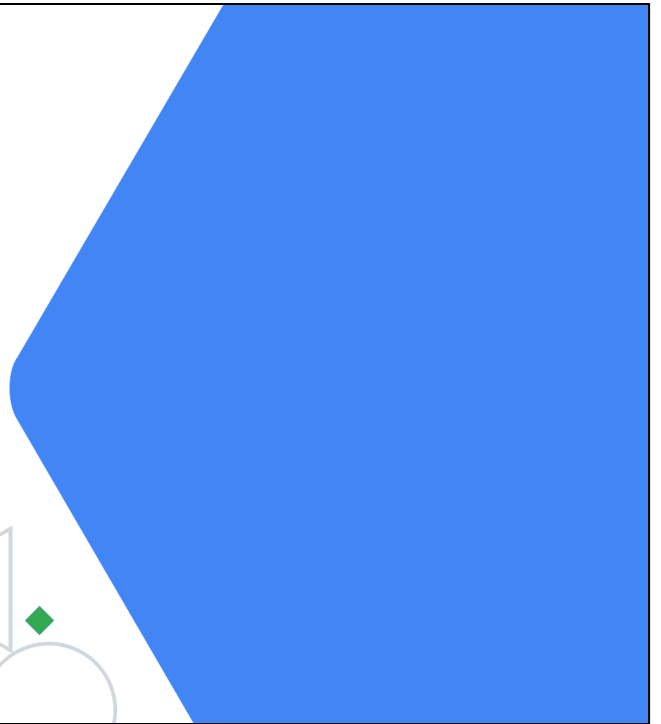
Identify the organization name, environment name, API proxy name, and revision by providing their values in the request path parameters.

You can also optionally set the trace session's timeout and set a filter condition in the body of the request.



## Demo

Tracing in Apigee Hybrid



---

# Agenda

Build and Deploy API Proxies

Tracing and Debugging

**Developer Portal Solutions**

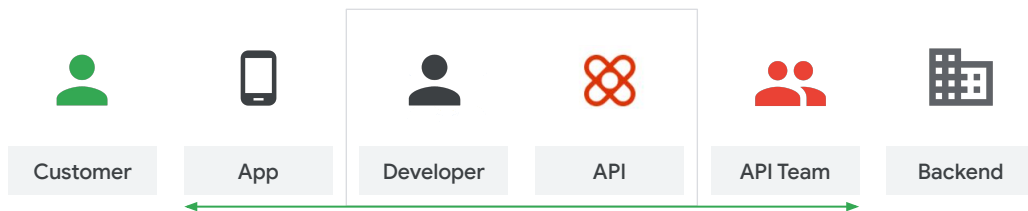
Managing Environments

Lab



In this lecture, we will discuss the developer portal solutions you can use with Apigee hybrid.

## App developers are customers of your APIs



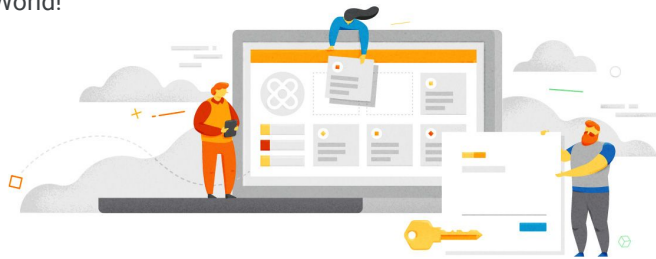
Application developers can be internal or external to your organization, or they can belong to partner organizations.

Design your APIs based on what app developers need. App developers are customers of your APIs.

Your APIs should have a consistent interface, be well-documented, and enable app developers to build great applications.

## Developer portals enable app developers to...

- Learn about your service offerings.
- Learn how to use your APIs by reviewing comprehensive documentation.
  - Including live documentation via SmartDocs.
- Register using a self-service process in order to build apps against your APIs.
- Minimize time to first "Hello, World!"



A developer portal enables app developers to discover your API offerings and teaches them how to use your APIs by providing comprehensive documentation.

It provides functionality to:

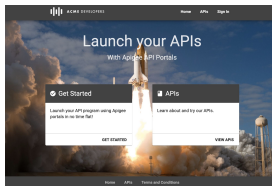
- a) easily self-register applications to use your API products and
- b) enables app developers to quickly build their applications.

# Types of developer portals

Apigee hybrid supports several developer portal solutions:

## Integrated Portal

- Self-service portal using JavaScript and CSS for development
- Hosted by Apigee



## Drupal 8 Portal

- Fully customizable self-service portal using Drupal, an enterprise-level content management system (CMS)
- Integrates with Apigee using Apigee-supported Drupal modules
- Highly flexible, but with a higher learning curve and effort required

## Do-It-Yourself (DIY) Custom Portal

- Build on any platform and integrate with Apigee using Apigee Management APIs
- Highest level of effort required

[Developer portal comparison](#)



Apigee hybrid supports multiple developer portal solutions.

The integrated developer portal enables simple self-service portal development and is hosted by Apigee in the management plane.

The Drupal 8 portal is a fully customizable portal that is based on the open source Drupal content management system and integrated with Apigee using modules.

The do-it-yourself portal is also fully customizable and implemented using Apigee APIs.

The Drupal and DIY developer portals are hosted by you and require a high level of effort to implement.

## Developer portal requirements

To provide a great developer experience, the developer portal must provide the following capabilities to your app developers:

- Seamless self-service and easy onboarding
- A well-cataloged list of API products
- Interactive API documentation
- Custom-branded developer experience
- Analytics and insights to understand usage patterns



A well-designed developer portal provides a great developer experience.

To accomplish this, a developer portal must have self-service registration features for developers and their applications, thus providing a seamless onboarding experience.

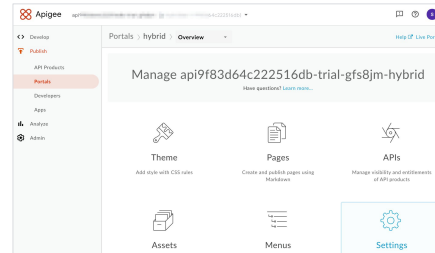
It should include a catalog of your available API products that the app developer can browse and view documentation in.

A developer portal can also include app analytics that help the developer view and analyze their application usage patterns.

# Building the integrated portal

Tools are available in the hybrid UI to build the integrated portal and enable you to:

- Develop portal content with the [Page editor](#) using markdown or HTML.
- Customize the look and feel using CSS with the [Theme editor](#).
- Publish your API product catalog.
- Autogenerate interactive documentation for your API products using [OpenAPI specifications](#).
- Manage your image files using the [Asset manager](#).
- Customize portal menus using the [Menu editor](#).
- Configure [Google Analytics](#) and custom analytics tracking.
- Extend your portal functionality using [custom scripts](#).



[Building an integrated developer portal](#)

The Apigee integrated portal includes a set of tools that enable you to:

Develop portal content in markdown or html.

Customize the look and feel using CSS stylesheets.

Publish your API product catalog.

Generate interactive documentation for your APIs using open API specifications.

Manage your images.

Customize the portal menus.

Configure Google analytics.

Extend functionality of the portal with custom scripts.

## Building the Drupal portal

- Apigee hybrid supports [Drupal 8](#), which is an open source, enterprise-level content management system (CMS).
- Using the Drupal portal tools, and Apigee-developed Drupal modules, you can build a fully customizable developer portal.
- [Apigee Developer Portal Kickstart](#) is a Drupal distribution that enables you to quickly create an Apigee developer portal using Drupal 8.
- With the D8 self-managed option, you are responsible for hosting and maintaining the portal site.
  - You can choose a provider, such as [Panthéon](#) or [Acquia](#), to manage and host your developer portal.



Using Drupal modules developed by Apigee, you can build a fully customizable developer portal based on Drupal 8.

The Apigee Developer Portal Kickstart is a Drupal distribution that enables you to quickly create an Apigee developer portal using Drupal 8.

You are responsible for developing, hosting, and managing the Drupal 8 developer portal.

<https://docs.apigee.com/api-platform/publish/drupal/open-source-drupal-8>

## Installation steps: Drupal portal

Steps	Details
Install prerequisites	<ul style="list-style-type: none"><li>• Install PHP 7.3</li><li>• Install Git and Zip packages</li><li>• Install PHP Composer</li></ul>
Install database	<ul style="list-style-type: none"><li>• Install PostgreSQL</li><li>• Create devportal database and devportal user</li></ul>
Install web server	<ul style="list-style-type: none"><li>• Install and configure Nginx web server</li><li>• Create a virtual host</li></ul>
Install Apigee Developer Portal Kickstart	<a href="https://www.drupal.org/docs/8/modules/apigee-developer-portal-kickstart">https://www.drupal.org/docs/8/modules/apigee-developer-portal-kickstart</a>
Configure Apigee modules	For self-service registration and other functionality

[Building a Drupal developer portal](#)



To install the Drupal portal, you must have certain prerequisite software packages installed.

The Apigee developer portal uses its own PostgreSQL database to store developer and application data, and other entities.

A webserver to serve the portal content is required.

The Apigee drupal modules must be configured for registration and other portal functionality.

Detailed installation instructions are on the Apigee developer portal documentation website.

---

# Agenda

Build and Deploy API Proxies

Tracing and Debugging

Developer Portal Solutions

Managing Environments

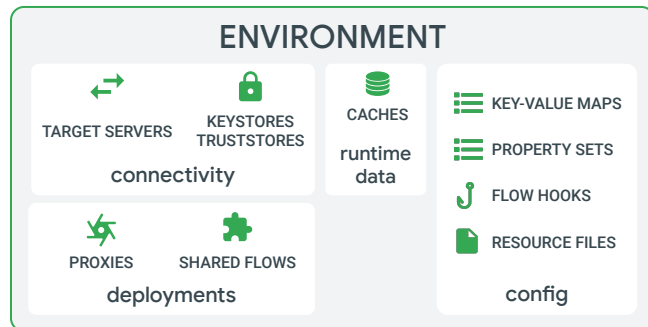
Lab



In this lecture, we will discuss managing environments in Apigee hybrid.

## Environment

- An environment provides a [runtime execution context](#) for API proxies.
- A proxy revision must be deployed to an environment before the proxy can process 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.



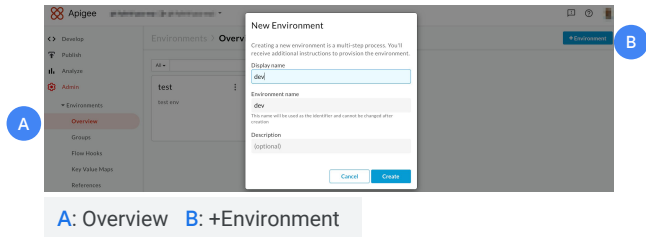
An environment in Apigee hybrid provides a runtime execution context for API proxies.

In order for an API proxy to receive and process API requests, it must be deployed to an environment.

Environments provide you with a flexible way to group your proxies, such as by functional domain or API development lifecycle.

## Managing environments

- Manage your environments with a combination of the [Apigee hybrid UI](#) and the hybrid runtime configuration file.
- An environment must be a member of an environment group before it can be used.
- To create a new environment:
  - Navigate to [Admin > Environments > Overview](#) in the Apigee UI.
- After the environment is created, it is in a [pending provisioning](#) state until fully provisioned.



- The environment name should not contain any spaces or other special characters. It can have only lowercase letters, numbers, and hyphens, with a maximum of 255 characters.



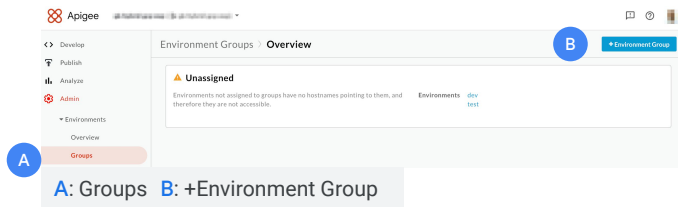
You create an environment by using the Apigee hybrid UI or Apigee API and configuring the cluster in the runtime plane.

You must assign an environment to an environment group before it can be used.

You can deploy API proxies to the environment after it is assigned to a group and the runtime plane is configured.

## Configuring an environment

- After an environment is created, it must be assigned to at least one environment group.
- To add an environment to a group:
  - Navigate to [Admin > Environments > Groups](#) in the Apigee UI.
- Create a new environment group to add unassigned environments.
- After creating the environment group, edit it to add an environment to it. You can add an environment to an existing group.



After you create an environment, you must assign it to an environment group.

You add an environment to a group using the Apigee UI or Apigee API.

You can add an environment to an existing group or to a new group by first creating the group.

## Configuring an environment in the cluster

After an environment is created via the UI or Apigee API, it must be configured in the runtime plane cluster.

To do this:

- Add the new environment definition to the `envs[]` property in your overrides file.
- Apply the override file changes to the cluster:

```
$ apigeectl apply -f overrides.yaml  
--all-envs
```

### overrides.yaml

```
namespace: my-namespace  
org: my-organization  
...  
envs:  
  - name: test  
    serviceAccountPaths:  
      synchronizer: "sa/synchronizer-sa.json"  
      udca: "sa/udca-sa.json"  
  
  - name: dev  
    serviceAccountPaths:  
      synchronizer: "sa/synchronizer-sa.json"  
      udca: "sa/udca-sa.json"  
  
...
```



After you create an environment using the UI or Apigee API, you configure it in the runtime plane using the `overrides.yaml` file.

The environment definition comprises the name of the environment and the service accounts of the synchronizer and UDCA components.

The override file changes must then be applied to the cluster.

## Configuring an environment group

- After an environment group is created, you configure it to add or modify hostaliases (domain names) for the group.
- You can also add additional environments to the group.
- Edit your runtime plane overrides.yaml file to configure a virtualhost for the environment group.
- Apply the override file changes with [apigeectl](#):  
`$ apigeectl apply -f overrides.yaml`

The screenshot shows the 'Environment Groups > devgrp' page in the Google Cloud Apigee console. The 'Environment group details' section includes a 'Group name' field with the value 'devgrp' and a 'Hostnames' field with the value 'example.devgrp'. Below these fields, a note states 'Enter one hostname per line. At least one is required.' and a link 'Configure Google Load Balancer with these domains.' is provided. At the bottom, the 'Environments' section shows a table with one entry: 'dev'.

Environment group details	
Group name	devgrp
Hostnames	example.devgrp

Enter one hostname per line. At least one is required.  
[Configure Google Load Balancer with these domains.](#)

Environments	
dev	



An environment group must be configured to add or modify host aliases or domain names that are used to route requests to API proxies deployed to environments in the group.

You can configure the group with more than one environment.

The environment group is mapped to a virtual host property in the overrides.yaml file. You then apply the changes to the overrides file to the runtime plane cluster.

## Creating an environment via the API

- You can create an environment by issuing a POST request to the [/organizations/\\* /environments](#) Apigee API.
- The response is a long-running operation object (LRO).
- Call the [organizations.operations.get](#) API at regular intervals to get the status of the LRO (and environment creation).
- View the full details of the create environment API here: [organizations.environments.create](#).

Details on creating an environment are documented [here](#).

### Sample Request:

```
$curl -X POST -H "Authorization: Bearer $TOKEN" -d "{\"name\": \"test\",...}" https://apigee.googleapis.com/v1/organizations/{org}/environments
```

### Sample Response:

```
{ "name": "XXXX", "metadata": { ... }, "done": boolean, "error": { "code": XXX, "message": "yyy", "details": [ ... ] }, "response": { ... } }
```



You can also create an environment by using the Apigee API.

Issue a POST request to the `/organizations/environments` API and pass in the name of your organization as a path parameter in the request.

The environment name, description, and other attributes are passed in the body of the request.

The create operation is asynchronous and returns a long-running operation object.

You must then make a GET request to the `/organizations/operations` API and pass in the value of the operation ID that was received in the response from the create operation.

The response from the operations API contains information about the status of the operation to create the environment.

You may need to make multiple calls to the operations API until the status indicates that the operation has completed.

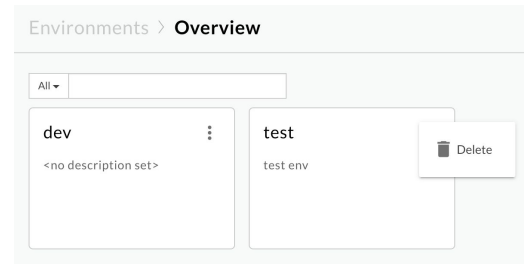
## Deleting an environment

Deleting an environment is a multi-step process:

1. Delete the environment from all associated environment groups using the Apigee UI or [API](#).
2. Apply config changes to your cluster using an [overrides.yaml](#) that contains the environment to be removed.
  - a. Do a dry-run to test the removal of the runtime components:

```
$apigeectl delete -f env-delete.yaml --env={env} --dry-run=client
```
  - b. If there are no errors, remove the runtime components:

```
$apigeectl delete -f env-delete.yaml --env={env}
```
3. Delete the environment from the management plane using the Apigee UI or [API](#).
4. Edit your overrides file to remove the configuration properties of the environment that was deleted.



**Note:** You must undeploy proxies from the environment before deleting it.

Details on deleting an environment are documented [here](#).

To delete an environment, you must first delete it from all associated environment groups. You can do this using the Apigee UI or API.

You then apply your overrides.yaml configuration that contains the environment to be deleted to the cluster, using the apigeectl delete command.

This will remove the runtime components associated with the environment that is being deleted.

It is a best practice to do a dry run first to check for any errors before deleting the environment.

Next, delete the environment from the management plane using the Apigee hybrid UI or API.

Finally, edit your overrides file to remove the deleted environment configuration.

## Deleting an environment via the API

- You can delete an environment by issuing a DELETE request to the [/organizations/\\*/environments](#) Apigee API.
- The response is a long-running operation object (LRO).
- Call the [organizations.operations.get](#) API at regular intervals to get the status of the LRO (and environment deletion).
- View the full details of the delete environment API here: [organizations.environments.delete](#).

### Sample Request:

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

### Sample Response:

```
{ "name": "XXXX", "metadata":
{...}, "done": boolean,
"error": {"code": XXX, "message":
"yyy", "details": [...]}, "response":
{...}
}
```



To delete an environment using the Apigee API, issue a DELETE request to the `/organizations/environments` API and pass in the name of your organization and environment as path parameters in the request.

The delete operation is asynchronous and returns a long-running operation object.

You must then make a GET request to the `/organizations/operations` API and pass in the value of the operation ID that was received in the response from the delete operation.

The response from the operations API contains information about the status of the operation to delete the environment.

You may need to make multiple calls to the operations API until the status indicates that the operation has completed.

## Updating an environment via the UI or API

- You can change the [display name](#) and [description](#) of an environment from the hybrid UI or Apigee API.
- To use the API, issue a PUT request to the [/organizations/\\*/environments](#) Apigee API.
- You must pass all existing properties to the API, even if they are not being changed. If you omit properties from the payload, the properties are removed.
- View the full details of the update environment API here: [organizations.environments.update](#).

### Sample Request:

```
$curl -X PUT -H "Authorization: Bearer $TOKEN" -d '{"name": \"test\", \"displayName\": \"testEnv\", \"description\": \"test environment\" }' https://apigee.googleapis.com/v1/organizations/{org}/environments/{env}
```

### Sample Response:

```
{ "name": "XXXX", ..., "properties": { ... }, ... }
```



You can use the Apigee hybrid UI or API to update the display name and description of an environment.

Issue a PUT call to the `/organizations/environments` API and pass in the name of your organization and environment as path parameters in the request.

The body of the request must contain all the existing environment properties, in addition to the properties being updated.

If this process is successful, the response from the API contains the updated environment object.

# Environment configuration management

To manage multiple environments in hybrid and maintain a well-organized set of overrides.yaml config files, follow these best practices:

- Use a [common.yaml](#) file that includes cassandra, metrics, and logger configuration.
- Use an [org.yaml](#) for all organization-scoped configuration.
- Use an [{envname}-env.yaml](#) for environment-specific configuration; one per environment.
- Run the [apigeectl init](#) and [apigeectl apply](#) commands in order as shown when applying these override configuration files to the cluster.

```
# setup system components
$apigeectl init -f common.yaml

# setup common components
$apigeectl apply -f common.yaml
--datastore --telemetry

# setup org level components
$apigeectl apply -f org.yaml --org

# setup environments for each env
$apigeectl apply -f {envname}-env.yaml
--env {envname}
```



As you operate and manage your hybrid runtime cluster, you may have to maintain multiple override.yaml configuration files over time.

A good practice is to use a structure of multiple override files, where each file contains configuration information for a functional resource or set of resources used in hybrid.

For example, you can have a common.yaml file that includes configuration for cassandra, metrics, and logger resources.

An organization.yaml file for any organization-scoped resources and configuration.

An environment.yaml file for each hybrid runtime environment that contains configuration for the environment being added or updated in the cluster.

You can initialize the cluster using the common.yaml file, and then apply the individual override config files, starting with the common resources, organization-level resources, and environments.

---

## Agenda

Build and Deploy API Proxies

Tracing and Debugging

Developer Portal Solutions

Managing Environments

Lab



We will now add an environment to your Apigee hybrid installation in a lab.

# Lab

## Managing Environments in Apigee Hybrid



In this lab, you add a new environment to your hybrid runtime cluster in Google Kubernetes Engine.

You create the environment using the Apigee hybrid UI and the required environment configuration in an `overrides.yaml` file, and apply that configuration to the cluster.

You verify that the environment is successfully provisioned by deploying a test API proxy to the environment and testing that the proxy works as expected.

# Lab review

Managing Environments  
in Apigee Hybrid



In this lab, you successfully created a new environment in your Apigee hybrid installation using the hybrid UI.

You then configured the environment in your runtime plane in a cluster in Google Kubernetes Engine.

You verified its operation by deploying and testing an API proxy in that environment.

You also verified that the previously configured environment continues to operate as expected.



## **Review: Deployment and Environment Management**

Hansel Miranda

Course Developer, Google Cloud



In this module, you learned how to deploy and manage API proxies in Apigee hybrid.

You learned about the deployment process and how to check the status of a proxy deployment.

You also learned about the various developer portal solutions available in Apigee hybrid.

You learned about environment management, configuration, and how to use the hybrid UI and Apigee API to perform create, update, and delete operations on the environment.

A lab to add a new environment was completed in this module.

In the next module, you will learn about hybrid security, RBAC, and how the hybrid runtime components are secured.