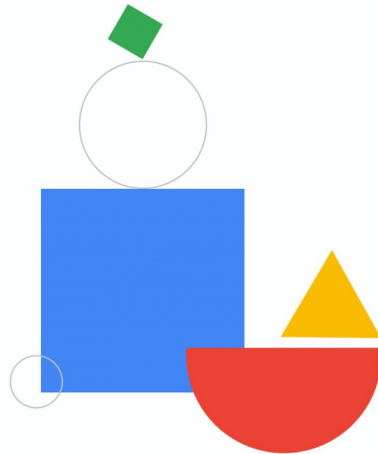


# Installation and Platform Operation



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

You will also learn about the tools to manage the hybrid runtime components and configure your cluster resources.

We will also discuss the process to backup and restore your hybrid runtime data.

# Agenda

01	Installation Process
02	Lab: Installing and Managing Apigee Hybrid
03	Platform Operation
04	Backup and Restore
05	Quiz



Let's start with the installation process. This process guides you through the high-level steps needed to install the Apigee hybrid runtime plane.

In this lecture, we will review these steps and the contents of the hybrid software distribution package.

We will also discuss the strategy used to configure the hybrid runtime plane.

## Process overview

To install Apigee hybrid, you will generally follow the steps below:

- Set up your Google Cloud account, project, and organization.
- Enable APIs and create an Apigee organization, environment group, and environment.
- Download [apigeectl](#), the hybrid software package to be installed.
- Calculate the approximate size of the Kubernetes clusters, and create and configure them.
- Install the cert-manager required by Apigee hybrid.
- For each cluster, create an overrides.yaml file with your hybrid-specific configuration.
- Apply the configuration.
- Expose the Apigee ingress gateway.
- Create a simple API proxy to verify operation.



The process to install Apigee hybrid is as follows:

- Set up your Google cloud account, project, and organization.
- Enable relevant APIs and create an Apigee organization, environment group, and environment.
- Download apigeectl, the hybrid software package for the version you plan to install.
- Determine the size of your Kubernetes clusters based on your API proxy workload and performance requirements, and then create the clusters.
- Install the cert-manager required by Apigee hybrid.
- The hybrid software includes configuration for most resources deployed in a cluster. You use an overrides file to override some of the configuration and provide any required properties.
- The configuration specified in this file is then applied to your hybrid clusters.
- Expose the Apigee ingress gateway.
- When all the hybrid services are running, deploy a simple API proxy to verify the installation.

## Distribution

The Apigee hybrid distribution is hosted by Google Cloud and publicly available.

- The current version of the distribution is maintained in a text file and available on Google Cloud.

Location of the current version of the text file:

<https://storage.googleapis.com/apigee-release/hybrid/apigee-hybrid-setup/current-version.txt?ignoreCache=1>

- Location of the current version of the hybrid distribution:

[https://storage.googleapis.com/apigee-release/hybrid/apigee-hybrid-setup/<version>/apigeectl\\_<platform>\\_<bit>.tar.gz](https://storage.googleapis.com/apigee-release/hybrid/apigee-hybrid-setup/<version>/apigeectl_<platform>_<bit>.tar.gz)

**version:** the value from the current-version.txt file. Example: 1.10.2

**platform:** mac or linux

**bit:** 32 or 64

You download the current or a specific version of the hybrid software distribution from a public storage location on Google Cloud.

The location URL contains the version number of the distribution and the platform on which you plan to install hybrid.

You can get the list of version numbers from the Apigee hybrid release website at [docs.apigee.com/release](https://docs.apigee.com/release).

## Package structure

The Apigee hybrid distribution package uses the file structure shown.

```
apigeectl_<ver>...
|
|--- apigeectl
|
|--- config
|    |--- values.yaml
|
|--- examples
|    |--- overrides-small.yaml
|    |--- overrides-medium.yaml
|    |--- ...
|
|--- plugins
|    |--- apigee-operators
|    |--- ...
|
|--- templates
|    |--- 0_init.yaml
|    |--- 1_apigee-datastore.yaml
|    |--- ...
|--- tools
|    |--- create-service-account
|    |--- ...
```

The hybrid distribution package contains a top-level directory with a name that begins with *apigeectl*, followed by the version and platform.

Lets review the contents of this directory...

## Directory contents

Path	Purpose
<a href="#">apigeectl</a>	Command-line interface (CLI) for installing and managing hybrid in a Kubernetes cluster.
<a href="#">/config</a>	Contains <b>deployment spec</b> configuration for the runtime plane with primary default config values.yaml that will be overridden by local configurations (or accepted as defaults).
<a href="#">/examples</a>	Examples of overrides.yaml files for <b>small, medium, and large deployments</b> that override certain properties in the <i>values.yaml</i> spec and are used as jumpstart configurations.
<a href="#">/plugins</a>	Contains runtime initialization components for <a href="#">apigee deployments</a> , Runtime Ingress, etc.
<a href="#">/templates</a>	Hybrid <b>configuration templates</b> that define runtime service components like MP, MART, Synchronizer, and Cassandra, which are merged into the values.yaml file by apigeectl.
<a href="#">/tools</a>	Contains the <i>create-service-account</i> and other tools used for hybrid <b>management tasks</b> .

**apigeectl** is a command-line executable that is used to install and manage Apigee hybrid in a Kubernetes cluster.

It is a wrapper for kubectl, which is a tool that lets you manage Kubernetes clusters. You must have kubectl installed and set up on your Kubernetes administration machine to use the apigeectl command.

The **config** subdirectory contains the deployment specification of the Apigee hybrid runtime resources, and their configuration in YAML file format.

For example, this file contains the replicaCountMin and replicaCountMax properties of the runtime message processor component installed in your cluster.

You usually override some of these configuration properties in separate files when installing and configuring Apigee hybrid.

The **examples** subdirectory contains sample override.yaml files for small-, medium-, and large-sized deployments of Apigee hybrid.

These files can be used as initial configurations for your type of deployment and should be updated as the requirements of your hybrid deployment change over time. The directory also contains a sample private overrides.yaml file that is used for hybrid runtime images from a different repository.

The **plugins** subdirectory contains deployment specifications for Apigee hybrid system components that are installed in the cluster.

This includes the apigee-operators, apigee-resources, and apigee-envoyfilter configuration files.

These system components manage the creation and release of hybrid components in the cluster, provide certificate management for hybrid components, and customize the Runtime Ingress gateway configuration.

The **templates** subdirectory contains template deployment specifications for the hybrid runtime components.

These templates are used to generate the values of the configuration properties for each component during installation of the hybrid runtime plane.

The hybrid runtime component configuration is organized into template files based on their scope of operation: organization, environment, and cluster.

The **tools** subdirectory contains the create-service-account tool that is used to create various Google Cloud service accounts.

These service accounts are used by the runtime components to securely access the Apigee hybrid management plane and Google Cloud services.

Other tools included in this directory enable you to pull and push runtime images to a user defined repository, clean up cassandra pods, and dump pod logs from your Kubernetes cluster.

## Setup

The recommended setup tasks to install Apigee hybrid:

1. Create the Kubernetes cluster.



After you download the hybrid software distribution, follow these steps to install the runtime plane:

First, based on your sizing requirements, create the cluster on your Kubernetes platform.



## Setup

The recommended setup tasks to install Apigee hybrid:

1. Create the Kubernetes cluster.
2. Install [cert-manager](#) in the cluster.



Apigee hybrid uses cert-manager to manage certificates used by the various components in the runtime plane.

Cert-manager is a native Kubernetes certificate management controller that helps with issuing and renewing of certificates.

You install cert-manager from a GitHub repository using the kubectl apply command.

## Setup

The recommended setup tasks to install Apigee hybrid:

1. Create the Kubernetes cluster.
2. Install [cert-manager](#) in the cluster.
3. Download and unpack the distribution package to a suitable directory on your machine:

```
apigeectl_<platform>_<bit>.tar.gz → ${HOME}/apigee/apigeectl.<ver>
```



Next, unpack the hybrid software distribution in a suitable directory on your administration machine.

In the example shown, the distribution is unpacked in the apigee directory under the user's home directory.

## Setup

The recommended setup tasks to install Apigee hybrid:

1. Create the Kubernetes cluster.
2. Install [cert-manager](#) in the cluster.
3. Download and unpack the distribution package to a suitable directory on your machine:

```
apigeectl_<platform>_<bit>.tar.gz → ${HOME}/apigee/apigeectl.<ver>
```

4. Add environment variables to your shell.



Set up shell environment variables for your Google Cloud project ID, region, zone, hybrid organization, etc.

These variables are used for cluster authentication and to run other hybrid installation commands.

## Setup

The recommended setup tasks to install Apigee hybrid:

1. Create the Kubernetes cluster.
2. Install [cert-manager](#) in the cluster.
3. Download and unpack the distribution package to a suitable directory on your machine:  
`apigeectl_<platform>_<bit>.tar.gz → ${HOME}/apigee/apigeectl.<ver>`
4. Add environment variables to your shell.
5. Create a directory structure for certificates, service accounts, and override files.

```
$mkdir ${HOME}/apigee/hybrid-files
```



Next, set up your administration machine with a recommended directory structure on the file system.

Create a subdirectory named `hybrid-files` under the `HOME/apigee` directory. You can give the directory any name you want.

This is used to store files for TLS certificates, service accounts, override configuration, and hybrid tools, which are all used in the hybrid runtime installation process.

## Setup

The recommended setup tasks to install Apigee hybrid:

6. Create symbolic links to point to the distribution sub-directories like tools and config.



Create symbolic links to the tools, config, templates, and plugins subdirectories from within the hybrid-files directory.

These links enable you to run the apigeectl command line tool from there.

## Setup

The recommended setup tasks to install Apigee hybrid:

6. Create symbolic links to point to the distribution sub-directories like tools and config.
7. Create the necessary certificates, keys, and service accounts.



A TLS certificate and key are needed for the runtime ingress gateway that handles API proxy traffic. For trial installations of Apigee hybrid, you can use a self-signed certificate.

Components in the hybrid runtime plane use Google Cloud service accounts to communicate with the management plane and with other Google Cloud services.

You can create these service accounts by using the create-service-account tool that is included in the Apigee hybrid distribution package.

## Setup

The recommended setup tasks to install Apigee hybrid:

6. Create symbolic links to point to the distribution sub-directories like tools and config.
7. Create the necessary certificates, keys, and service accounts.
8. Create an overrides file to configure the cluster.



The Apigee hybrid installer: `apigeectl` uses defaults for most cluster configuration settings for the hybrid runtime plane components.

However, a few settings do not have defaults, and you can override others.

You provide values for these settings in an `overrides.yaml` configuration file.

You can use one of the sample `overrides.yaml` files provided in the hybrid distribution as a starting point for your cluster configuration.

The `overrides.yaml` file is usually created in a subdirectory named “overrides” in the `hybrid-files` directory.

## Setup

The recommended setup tasks to install Apigee hybrid:

6. Create symbolic links to point to the distribution sub-directories like tools and config.
7. Create the necessary certificates, keys, and service accounts.
8. Create an overrides file to configure the cluster.
9. Enable synchronizer to authenticate against the management plane.



After the `overrides.yaml` file is created, you enable the required permissions for the synchronizer runtime component to authenticate against the hybrid management plane.



## Setup

The recommended setup tasks to install Apigee hybrid:

6. Create symbolic links to point to the distribution sub-directories like tools and config.
7. Create the necessary certificates, keys, and service accounts.
8. Create an overrides file to configure the cluster.
9. Enable synchronizer to authenticate against the management plane.
10. Install the Apigee hybrid runtime.



Run the `apigeectl` command from within the `hybrid-files` directory to install Apigee hybrid.

You first initialize the hybrid system namespaces by running the `apigeectl init` command. This installs the hybrid system components in the cluster.

After all the pods are running, you run the `apigeectl apply` command. The `apply` command installs the Apigee hybrid runtime components in your cluster.

You can use the `apigeectl check-ready` command to test whether all the pods in the cluster are ready.

## Setup

The recommended setup tasks to install Apigee hybrid:

6. Create symbolic links to point to the distribution sub-directories like tools and config.
7. Create the necessary certificates, keys, and service accounts.
8. Create an overrides file to configure the cluster.
9. Enable synchronizer to authenticate against the management plane.
10. Install the Apigee hybrid runtime.
11. Expose the Apigee ingress gateway.



Finally, set up the Kubernetes service for your Apigee ingress gateway. The Kubernetes service is needed for each ingress deployment to obtain an IP address that can be exposed. Client calls to proxies will invoke a hostname that will resolve to this IP address.

There are two options for providing a Kubernetes service to assign the IP address. The recommended option for production environments is to create a custom Kubernetes service for each Apigee ingress gateway. For non-production environments or to test initial traffic through the Apigee ingress gateway, Apigee hybrid provides a default Kubernetes service for each ingress deployment.

## Overrides strategy

```
gcp:
  region: gcp-region
  projectID: gcp-project-id

k8sCluster:
  name: cluster-name
  region: cluster-region

org: org-name

instanceID: "unique-id"

virtualhosts:
- name: env-group-name
  selector:
    app: apigee-ingressgateway
    ingress_name: INGRESS_NAME
  sslCertPath: ./certs/cert-name.pem
  sslKeyPath: ./certs/key-name.key

envs:
- name: environment-name
  serviceAccountPaths:
    synchronizer: ./sa/sync-sa.json
    udca: ./sa/udca-sa.json
    runtime: ./sa/runtime-sa.json
...
```

Apigee's install tool ([apigeectl](#)) uses an overrides strategy:

- Default values are provided in the [config/values.yaml](#) file, which should never be directly edited.
- The default values can be overridden by configurations supplied by you in a yaml file called "overrides.yaml."
- There can be more than one "overrides" file. It is a best practice to version your override files in source control.
- Name your override files to convey the use case for which they are created.

For example: "official-prod.yaml" or "experimental-dev.yaml"

- Overrides may describe one or more components of a configuration or the entire configuration.

Sample overrides.yaml file fragment

Google Cloud

When you install Apigee hybrid, you provide configuration properties that are used to configure hybrid components in your cluster.

The properties override the default values that come with the hybrid software distribution.

Using this strategy, you can have multiple override files that configure your cluster with different resource configurations, based on certain criteria. For example, you may have different config settings for a production cluster versus a non-production cluster.

You can also have separate override files where each file configures all the components needed for a single runtime environment in the cluster.

## Override file versioning

- Over time, you will accumulate multiple versions of the apigee tools and changes to override files.
- It is a best practice to maintain versions of the tool and overrides files in source control.
- If rollbacks become necessary, you can apply the previous version of the files to your cluster.
- It is recommended that you test and practice your installation, upgrade, and rollback procedures before production installation.



It is a best practice to maintain your `override.yaml` files in source control.

This allows versioning of your configuration and makes it easier to roll back to a previous version if needed.

# Namespaces

## Kubernetes Namespaces used by Apigee Hybrid

Namespace	Purpose
<a href="#">apigee</a>	Contains runtime plane components like runtime, cassandra, and UCDA.
<a href="#">apigee-system</a>	Used to deploy and maintain <a href="#">ApigeeDeployment</a> (AD) resources and validate deployment configurations.
<a href="#">cert-manager</a>	Provides PKI certificate management for runtime plane components.
<a href="#">default</a>	Always present in Kubernetes; the namespace given to services if one isn't supplied.

The Apigee hybrid installation process creates multiple namespaces in your Kubernetes cluster.

These namespaces logically group the runtime components by the type of functionality they provide.

The apigee-system, and cert-manager namespaces contain system components needed for creating the hybrid workload components including the Apigee ingress gateway, and for certificate management.

All of the hybrid workload components are deployed in the apigee namespace in your cluster.

## Verifying the installation

To verify the installation, create a basic API proxy that tests the connectivity between the management plane, runtime plane components, and backend.

If you don't have your own test backend, consider [httpbin.org](http://httpbin.org).

The basic API proxy should cover the following use cases:

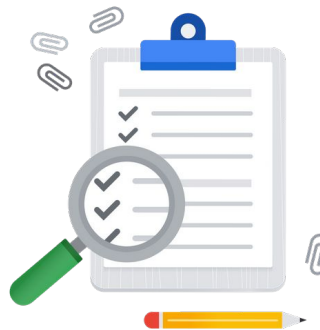
- API key verification
- Backend connectivity
- Quota and/or spike arrest

Verify the following in the management plane via the hybrid UI:

- Performance tab
- Trace
- Analytics

You can also verify the following in the Google Cloud Console:

- Log entries in Cloud Logging
- Metrics in Cloud Monitoring



After Apigee hybrid is installed, you can verify that it is operating as expected.

Create a test environment in the hybrid UI and configure this environment in the hybrid runtime plane.

Then create or upload a simple API proxy using the hybrid UI and deploy it to the test environment.

This test API proxy can use a mock backend as the target of the API.

Send multiple requests to the API. You can use curl, Postman, or any REST client to generate HTTP requests.

Confirm that the API proxy is returning the expected response from the backend.

You can add API key verification or quota policies to the test API proxy to verify that the runtime datastore is functioning as expected. This will confirm that the MART, Message Processor, and Cassandra runtime components are functioning as expected.

In addition, you can start Trace sessions to debug your proxy and view Analytics in the hybrid UI.

Using the Google Cloud Console, you can view log messages to troubleshoot the

hybrid runtime components and view metrics to track resource usage of the cluster components.

# Agenda

01	Installation Process
02	<a href="#">Lab: Installing and Managing Apigee Hybrid</a>
03	Platform Operation
04	Backup and Restore
05	Quiz



We will now do a lab: Installing and Managing Apigee Hybrid.



## Lab Intro

Installing and Managing  
Apigee Hybrid



Google Cloud

In this lab, you install and configure the Apigee hybrid runtime plane in a Kubernetes cluster on GKE. You then create Apigee hybrid environments and an environment group using the Apigee hybrid UI.

Next, you deploy simple API proxies to the Apigee hybrid environments and validate the environments by testing the API proxies.

Finally, you view API proxy analytics in the Apigee hybrid UI.

# Agenda

01	Installation Process
02	Lab: Installing and Managing Apigee Hybrid
03	Platform Operation
04	Backup and Restore
05	Quiz



In this lecture we discuss the tools used to operate the Apigee hybrid runtime platform.

## Using apigeectl

- [apigeectl](#) is a command-line interface (CLI) for installing and managing hybrid in a Kubernetes cluster. You must have [kubectl](#) installed and set up on your Kubernetes administration machine to use the [apigeectl](#) command.
- The tool is located at `/apigeectl` under the base installation directory of the hybrid distribution.
- Usage takes the format: `apigeectl [command] [flags]`

Command	Description
<a href="#">init</a>	Installs Apigee Operators and CRDs in the apigee-system namespace.
<a href="#">apply</a>	Applies configurations for Apigee hybrid runtime components to your Kubernetes cluster. <b>apigeectl apply</b> without flags will apply changes to all components in the cluster.
<a href="#">check-ready</a>	Checks the status of the hybrid component pods. When all component pods are ready, the message "All containers ready" is output.
<a href="#">delete</a>	Deletes hybrid components from the cluster.

For the full list of commands, see the [apigeectl CLI reference](#)

The *apigeectl* executable is included in the distribution package and installed in your `apigeectl` directory.

You use this tool to initialize and configure your Kubernetes cluster with the hybrid runtime system components and workloads.

When running the tool, provide the command name and any optional flags on the command line.

The command specifies the operation you want to perform, for example: `init`, `apply`, or `check-ready`. Flags specify command parameters.

To initialize the cluster with the hybrid system components, use the `init` command.

You use the `apply` command to apply your configuration and install the Apigee hybrid workloads into the cluster.

The `check-ready` command can be used to check whether all of the components in the cluster are running.

The complete list of commands is documented on the [CLI reference page](#) on the Apigee documentation website at [cloud.google.com](https://cloud.google.com/apigee).

## apigeectl flags

Flags allow you to specify optional parameters to the [apigeectl](#) command. Some flags allow you to control the scopes (and components) that the command applies configuration changes to.

Flag	Scope	Components
<a href="#">--datastore</a>	Storage	Cassandra
<a href="#">--env</a> <a href="#">--all-envs</a>	Environment	Runtime Synchronizer UDCA
<a href="#">--org</a>	Organization	Apigee Connect Agent MART Watcher
<a href="#">--telemetry</a>	Reporting	Logger Metrics
<a href="#">--redis</a>	In-memory storage	Redis

You can control the scope of the command used with `apigeectl` by using optional flags.

The flags shown here determine the set of runtime components that the `apigeectl` command applies the configuration to.

The `datastore` flag applies the configuration changes to the Cassandra component in the cluster.

The `env` flag applies the configuration changes to the Runtime, Synchronizer, and UDCA cluster components for the specified environment.

The `all-envs` flag applies the configuration changes to these components for all environments under the organization specified in your overrides config file.

The `org` flag applies the configuration changes to the Apigee Connect, MART, and Watcher components for the specified organization.

The `telemetry` flag applies the configuration changes to the logger and metrics components in the cluster.

The `redis` flag applies the configuration to the in-memory data storage scope (Redis).

## More apigeectl flags

Flag	Description
<code>--all</code>	Used with the <a href="#">delete</a> command. Deletes the entire Apigee hybrid installation from your cluster.
<code>--dry-run</code>	Executes the specified command without changing the cluster. Use with <code>--print-yaml</code> to output the rendered object specification to a file.
<code>-f</code> <code>--file-override-config</code>	YAML file that contains custom config properties for the hybrid deployment. The default value is <code>./overrides.yaml</code> . An overrides file is required for the <a href="#">apply</a> , <a href="#">check-ready</a> , <a href="#">delete</a> , and <a href="#">init</a> commands. You must specify the full path with this flag.
<code>--print-yaml</code>	Prints the configuration template output to stdout. See <a href="#">Print the configuration to a file</a> .
<code>-s, --settings</code> <code>virtualhosts</code>	Use this flag to apply changes to the virtualhosts property.

For the full list of flags, see the [apigeectl CLI reference](#).

The all flag is used with the delete command and deletes the entire Apigee hybrid installation from your cluster.

The dry-run flag allows you to check for any errors in the configuration without applying them to your cluster.

When used with the print-yaml flag, the complete cluster configuration for your hybrid runtime plane can be output to a file. You can then save the file in source control.

The f flag allows you to provide a specific override yaml configuration file for your cluster components. If the file is not specified, the command looks for a file named overrides.yaml in the current directory.

The settings flag is used to apply changes to the virtualhosts property to the cluster.

The complete list of flags is documented on the [CLI reference](#) page on the Apigee documentation website at [cloud.google.com](https://cloud.google.com/apigee).

## Runtime component management: apigeectl

To make a configuration change to a hybrid runtime plane component, edit your overrides file and apply the changes with [apigeectl](#).

For example, to change the replica count on the runtime MP component:

- Edit the overrides.yaml file to update the [replicaCountMin/replicaCountMax](#) entries.
- Use apigeectl to apply the change to the cluster:  

```
$ apigeectl apply --env {env_name} -f  
./my_overrides.yaml
```
- Check the readiness status of a hybrid runtime component using this command:  

```
$ apigeectl check-ready -f ./my_overrides.yaml
```

```
overrides.yaml  
  
...  
runtime:  
  replicaCountMin: 2  
  replicaCountMax: 4  
  resources:  
    cpu: 1000m  
    memory: 1Gi  
...
```

Here is an example of how to use the apigeectl command.

In this example, you would like to change the minimum and maximum number of replicas for the runtime message processor component in your cluster.

You first change the overrides.yaml configuration file on your file system.

Then, run the apigeectl apply command, passing in the environment name, and the name and location of the overrides config file.

You can then use the check-ready command to determine whether the component pods are up and running with the new configuration.

The **apigeectl** command uses **kubectl** behind the scenes to do its work. The **apigeectl** command options (**apply** and **delete**, and the use of YAML configuration files, are similar to their kubectl counterparts.

To learn more about how objects are managed in a Kubernetes cluster, see the following topics:

- [Imperative Management of Kubernetes Objects Using Configuration Files](#)
- [Overview of kubectl](#)

## Runtime component management: kubectl

- [kubectl](#) commands can be used to fetch information about the status of various objects in the cluster:
  - To get a list of all pods in the *apigee* namespace:

```
$ kubectl get pods -n apigee
```
  - For more detailed information about a given pod:

```
$ kubectl describe pods {POD_NAME} -n apigee
```

where {POD\_NAME} is obtained from the get pods command above.
  - To get a shell for an apigee-runtime pod:

```
$ kubectl exec -it -n apigee {runtime_pod_name} --container apigee-runtime  
-- /bin/bash
```
- Details on kubectl and the list of commands are on the [Kubernetes docs page](#).

apigeectl uses the Kubernetes command line tool: kubectl.

Kubectl enables you to manage and control Kubernetes clusters.

You can use kubectl to list and describe cluster resources, like pods, and check on their current status.

You can also start command shells to access the containers running in those pods for debugging purposes.

# ApigeeDeployment

- Apigee hybrid uses a Kubernetes [Custom Resource Definition \(CRD\)](#) called [ApigeeDeployment \(AD\)](#) to define, update, and release stateless Apigee hybrid components in the cluster.
- ApigeeDeployments can be managed using kubectl commands. To get a list of all the ADs in your cluster, run:

```
$kubectl get ad -n {namespace}
```

NAME	AGE
apigee-mart-my_org	4h
apigee-runtime-my_org-my_env	4h
apigee-synchronizer-my_org-my_env	4h
apigee-udca-my_org-my_env	4h

- Hybrid uses a custom controller called [apigee-operators \(ao\)](#), along with the CRD, to maintain the desired state of hybrid resources. The controller:
  - Validates the ApigeeDeployment configuration using an [admission webhook](#) before persisting it in the cluster.
  - Implements the release of message processors in the cluster.

The Kubernetes [CustomResourceDefinition](#) API resource allows you to define custom resources in your cluster.

Using a CRD object, you can create a custom resource with a name and schema that you specify.

Apigee hybrid uses a CustomResourceDefinition called ApigeeDeployment.

The ApigeeDeployment objects can be managed by using kubectl commands.

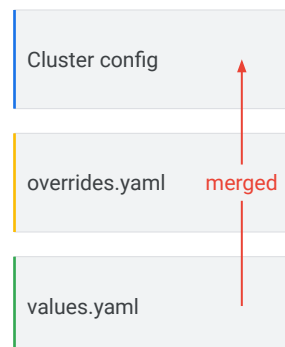
Along with the Apigee Operators custom controller, they validate deployment configurations and deploy hybrid runtime resources in the cluster.

The **apigee-deployment-admissionhook** is a HTTP callback used for validating requests for changes to ApigeeDeployment resources.



## Configuring hybrid

- Apigee hybrid maintains its **default component configuration** in the file:  
`{hybrid_base_install_dir}/config/values.yaml`
- As part of the installation process, you create an `overrides.yaml` file that typically includes only a **subset** of the configuration properties in `values.yaml`.
- When you apply a configuration to a cluster, your overrides are merged with the defaults to create the complete Kubernetes cluster configuration.
- **Not all properties in the `values.yaml` file are editable**, and some overrides are **mandatory**, such as `gcpProjectID`.
- If you want to modify a component's defaults, first copy its YAML from `values.yaml` into your overrides file, and modify it there. **DO NOT** modify the `values.yaml` file directly.



Apigee hybrid uses a layered approach to configure your cluster.

The default configuration is stored in a file called `values.yaml` in the `config` subdirectory under the `apigeectl` directory in the hybrid distribution.

To install Apigee hybrid, you create an overrides configuration file that includes mandatory properties and other properties that override the default values in the `values.yaml` file.

Not all default properties can be overridden. For those that can and should be modified, the best practice is to provide the new values in the `overrides.yaml` config file instead of directly modifying the `values.yaml` file. This approach preserves your modified values during product upgrades.

## Using overrides

- You can create as many overrides files as you want, where each one serves a specific requirement.
  - For example, you might have an overrides file that tunes your cluster for production, and another for creating a testing cluster. You can then maintain these files in your source control system.
- Use the `-f` option with `apigeectl` to specify the location and name of an overrides file:

```
$apigeectl apply -f ./test_env_override.yaml
```
- The hybrid installation contains sample overrides files to help set up your hybrid deployment. They are located in the directory: `{hybrid_base_install_dir}/examples`
- It is a good practice to copy and modify the overrides file that most closely matches your installation requirements.
- See the full [configuration properties](#) documentation for reference.

It is also a best practice to maintain multiple override files in source control.

You can have one overrides file for your production clusters and a separate one for a non-production cluster.

You can also have separate override files for each hybrid environment in the cluster.

A good practice is to maintain a set of override files where each file contains only the configuration required for each group of components.

For example, you can have a `common.yaml` overrides file for components like `cassandra`, `logging`, and `metrics`.

Another file can contain the configuration properties for the hybrid organization and resources like `mart` and `apigee connect`.

Environment-level properties for the runtime, synchronizer, and `udca` components can be in their own overrides file, one file per hybrid environment.

Virtual host configuration can also be in its own overrides configuration file.

Using this strategy, you can control changes to a subset of components in the cluster by updating and applying only the specific overrides file.

## Checking for configuration errors

Check for errors in your hybrid cluster configuration using options with the `apigeectl` command:

- Use the `--dry-run` flag to check for errors in your `overrides.yaml` file without modifying your cluster.  

```
$ apigeectl apply -f ./my_overrides.yaml --dry-run=true (kubectl v1.17.x)
$ apigeectl apply -f ./my_overrides.yaml --dry-run=client (kubectl v1.18.x)
```
- Use the `--print-yaml` and `dry-run` flags to print the cluster configuration to a file.
  - This is useful for debugging and enables you to store the configuration in source control for reference.
  - Be aware that the output contains password and secrets that you might not want to store.

```
$ apigeectl apply -f ./my_overrides.yaml --dry-run=client --print-yaml >
apigee-hybrid.yaml
```

The *apigeectl* tool contains useful command line flags called `dry-run` and `print-yaml`.

You can use the `dry-run` flag with the `apply` command to check for errors in your configuration without making any changes to the cluster.

When the `dry-run` flag is combined with the `print-yaml` flag, you can generate your entire cluster configuration.

You can then save this configuration to a file and maintain it in source control for reference.

## Automated issue surfacing

Automated issue surfacing (AIS) provides quick information about ongoing issues detected within your Hybrid cluster.

- Runtime watcher scans to determine if there are any configuration issues.
- AIS links to documentation to help with troubleshooting and resolution.
- AIS looks for known, common, system detectable issues.
- AIS cannot detect all the issues within a cluster.
- You can check for any existing issues with the `kubectl get apigeeissues` command.

```
kubectl -n apigee get apigeeissues
```

Automated issue surfacing (AIS) provides quick information about ongoing issues detected within your Hybrid cluster.

Starting with Apigee hybrid v1.10, the Apigee runtime watcher component automatically scans the control plane and Kubernetes API server state to determine if there are any configuration issues. By default, the scanning happens every 60 seconds. You can change the interval or disable scanning if you prefer.

When AIS detects an issue, it creates a new instance of `ApigeeIssue` within the Kubernetes API server. These instances contain information about the issues and links to documentation on the specific issues.

When you resolve issues, they will automatically be deleted from the Kubernetes API server when the scanning determines they are no longer occurring.

Automated issue surfacing only looks for known, common, system detectable issues and cannot detect all the issues within a cluster.

You can check for any existing issues with the `kubectl get apigeeissues` command.

## Support for custom annotations

You can create custom annotations for the following properties:

- **cassandra**
- **connectAgent**
- **logger**
- **mart**
- **metrics**
- **runtime**
- **synchronizer**
- **udca**
- **watcher**

overrides.yaml

```
...
runtime:
  annotations:
    businessunit:"bu1"
...
```

Annotations are key/value maps used to attach metadata to Apigee hybrid Kubernetes pods.

You can create custom annotations for the following properties:

- **cassandra**
- **connectAgent**
- **logger**
- **mart**
- **metrics**
- **runtime**
- **synchronizer**
- **udca**
- **watcher**

To add a custom annotation, add a stanza to the **OVERRIDES.yaml** file for the respective component.

# Agenda

01	Installation Process
02	Lab: Installing and Managing Apigee Hybrid
03	Platform Operation
04	Backup and Restore
05	Quiz



In this lecture, we discuss procedures to perform backups of your hybrid runtime plane datastore.

We will also discuss how to restore your cluster from a backup.

## Runtime Cassandra

- Apigee hybrid uses a Cassandra ring installed in the runtime plane to provide data persistence for KMS, KVM, Quota, and other data needed by API proxies.
- Cassandra is a replicated database that is configured to have at least 3 copies of your data in each cluster.
- Cassandra uses streaming replication to maintain the data replicas in each region.
- In hybrid, Cassandra backups are **not** enabled by default. It is a good practice, however, to [enable Cassandra backups](#) in case your data is accidentally deleted.



Apigee hybrid uses a Cassandra database to store your hybrid runtime data.

Cassandra is a replicated database that is configured to have at least 3 copies of your data in each cluster. Cassandra uses streaming replication and read repairs to maintain the data replicas at any given point.

The Cassandra runtime data store is deployed to your Kubernetes cluster as a StatefulSet. The Cassandra pod reads and writes data using a PersistentVolume. If the pod is deleted and re-created, it will continue to have access to the PersistentVolume.

In hybrid, Cassandra backups are **not** enabled by default. It is a good practice, however, to enable Cassandra backups in case your data is accidentally deleted.

## Backup overview

There are two methods to configure backups:

1. Using Apigee hybrid CSI backup and restore, which uses Kubernetes CSI (Container Storage Interface) snapshots.
2. Using non-CSI hybrid backup and restore, which copies schema and other data to backup files.

For both methods, you'll configure backup in your `overrides.yaml` file.



You can use one of two methods to configure backups.

Using Apigee hybrid CSI backup and restore, which uses Kubernetes CSI (Container Storage Interface) snapshots. This method is recommended for hybrid instances hosted in Google Cloud, AWS, or Azure.

Using non-CSI hybrid backup and restore, which copies schema and other data to backup files. This is the recommended method for on-prem installations.

For both methods, you'll configure backup in your `overrides.yaml` file. Apigee recommends you make a copy of the `overrides.yaml` file, so that you can reuse it during the recovery process.

Let's take a look at using the Apigee hybrid CSI backup and restore method.

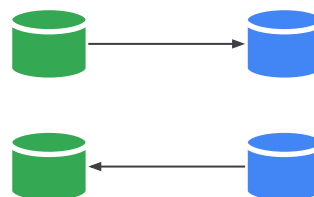


## Cassandra CSI backup and restore

Starting with Apigee hybrid 1.9, you can backup and restore your hybrid data using CSI (Container Storage Interface) snapshots.

- CSI backup generates disk snapshots and stores them as encrypted data in cloud storage.
- CSI backup does not need a Google Cloud Storage bucket or a remote server to store backup data.
- CSI backup is recommended for hybrid instances hosted in Google Cloud, AWS, or Azure.

For more information, see [Cassandra CSI backup and restore](#).



Starting with Apigee hybrid 1.9, you can backup and restore your hybrid data using CSI (Container Storage Interface) snapshots. CSI backup generates disk snapshots and stores them as encrypted data in cloud storage. CSI backup does not need a Google Cloud Storage bucket or a remote server to store backup data.

CSI backup is recommended for hybrid instances hosted in Google Cloud, AWS, or Azure.

For more information, see [Cassandra CSI backup and restore](#).

Now let's take a look at how to schedule backups in Cloud Storage and in a remote server, using non-csi backup and restore methods.

## Cassandra non-CSI backups

### Entities that are backed up:

- Cassandra schema including the user schema (Apigee keyspace definitions)
- Cassandra partition token information per node
- A daily snapshot of the Cassandra data

### Backup data storage:

- Backed-up data is stored in a Cloud Storage bucket that you must create.
- You can also store backups on a server file system.

Cassandra non-CSI backups are best suited to schedule backups in Cloud Storage and in a remote server.

The entities that are backed up in this process include the Cassandra schema, partition token information per node, and a daily snapshot of the data.

The backup is stored in a Cloud Storage bucket that you must create before enabling the Cassandra backup.

You can also back up your datastore to a server file system. This topic is covered in later slides in this section.

## Scheduling backups

Backups are scheduled as cron jobs in the runtime plane. To schedule Cassandra backups:

1. [Create a service account](#) with the standard [roles/storage.objectAdmin](#) role. This role allows you to write backup data to Cloud Storage.

Use the create-service-account tool that is part of the hybrid distribution in the <base-install-dir>/tools directory.

```
$ ./tools/create-service-account --env prod \  
  --profile apigee-cassandra \  
  --name my-cassandra-backup-sa --dir ./service-accounts
```

To schedule backups, you must perform a set of steps.

First, create a service account with the storage.objectAdmin role. This service account role allows you to back up data to Cloud Storage.

To create this service account, use the create-service-account tool in the tools directory in the hybrid distribution.

Name the service account apigee-cassandra, and store the service account json key file in the service-accounts subdirectory.

## Scheduling backups

Backups are scheduled as cron jobs in the runtime plane. To schedule Cassandra backups:

1. [Create a service account](#) with the standard [roles/storage.objectAdmin](#) role. This role allows you to write backup data to Cloud Storage.

Use the create-service-account tool that is part of the hybrid distribution in the <base-install-dir>/tools directory.

```
$ ./tools/create-service-account --env prod \  
--profile apigee-cassandra \  
--name my-cassandra-backup-sa --dir ./service-accounts
```

2. [Create a Cloud Storage bucket](#). Specify a reasonable data [retention policy](#) (15 days) for the bucket.

Create a storage bucket in Cloud Storage.

Specify a data retention policy for the bucket. 15 days is a reasonable period.

## Scheduling backups

Backups are scheduled as cron jobs in the runtime plane. To schedule Cassandra backups:

1. [Create a service account](#) with the standard [roles/storage.objectAdmin](#) role. This role allows you to write backup data to Cloud Storage.

Use the create-service-account tool that is part of the hybrid distribution in the <base-install-dir>/tools directory.

```
$ ./tools/create-service-account --env prod \  
--profile apigee-cassandra \  
--name my-cassandra-backup-sa --dir ./service-accounts
```

2. [Create a Cloud Storage bucket](#). Specify a reasonable data [retention policy](#) (15 days) for the bucket.
3. [Edit the overrides.yaml](#) file to configure Cassandra backup:

overrides.yaml

```
cassandra:  
  ...  
  backup:  
    enabled: true  
    serviceAccountPath:  
      "/service-accounts/apigee-cassandra.json"  
    dbStorageBucket: "gs://cassandra-bkup"  
    schedule: "45 23 * * 6"  
    cloudProvider: "GCP"  
  ...
```

Google Cloud

Update your overrides.yaml file and add the backup configuration properties to the cassandra stanza.

These properties enable the backup, set the path to the service account file and the Cloud Storage bucket, and configure a cron schedule to run the backup.

For example, the configuration shown will schedule a backup using the service account json file (generated from the create-service-account command above, and stored in the filepath ./service-accounts/apigee-cassandra.json).

The data is backed up to the Cloud Storage bucket at gs://cassandra-bkup. The cron schedule to run the backup is as configured.

## Scheduling backups

Backups are scheduled as cron jobs in the runtime plane. To schedule Cassandra backups:

1. [Create a service account](#) with the standard [roles/storage.objectAdmin](#) role. This role allows you to write backup data to Cloud Storage.

Use the create-service-account tool that is part of the hybrid distribution in the <base-install-dir>/tools directory.

```
$ ./tools/create-service-account --env prod \
  --profile apigee-cassandra \
  --name my-cassandra-backup-sa --dir ./service-accounts
```

2. [Create a Cloud Storage bucket](#). Specify a reasonable data [retention policy](#) (15 days) for the bucket.
3. [Edit the overrides.yaml](#) file to configure Cassandra backup:
4. [Apply the changes to the runtime cluster](#):

```
$ apigeectl apply --datastore -f overrides.yaml
```

When you apply the backup configuration, Kubernetes recreates the Cassandra nodes, so allow for enough time before backup starts.

### overrides.yaml

```
cassandra:
  ...
  backup:
    enabled: true
    serviceAccountPath:
      "/service-accounts/apigee-cassandra.json"
    dbStorageBucket: "gs://cassandra-bkup"
    schedule: "45 23 * * 6"
    cloudProvider: "GCP"
  ...
```

Google Cloud

Use the *apigeectl* command to apply the override configuration changes to the cluster.

Pass in the datastore flag that indicates the cassandra component and the path to the overrides.yaml file that contains your backup config properties.

Allow for sufficient time in the cron schedule for the backup to begin after the configuration changes are applied to the cluster.

## Verify the backup job

5. Run the following command to verify the backup job:

```
$kubectl get cronjob -n apigee
```

NAME	SCHEDULE	SUSPEND	ACTIVE	LAST SCHEDULE	AGE
apigee-cassandra-backup	33 * * * *	False	0	<none>	94s

To verify the backup job, run the following command: `kubectl get cronjob -n apigee`.

## Viewing backup logs

- When a backup job is scheduled and running, you can view progress by monitoring the [backup pod log](#).
- Get a list of the pods to determine the backup pod name:  
`$kubectl get pods -n <namespace>`

NAME	READY	STATUS	RESTARTS	AGE
apigee-cassandra-backup-1554515580-pff6s	0/1	Running	0	54s

- Run the `kubectl logs` command to view the pod log. Partial output follows:  
`$kubectl logs -f apigee-cassandra-backup-1554515580-pff6s`

```
...
INFO: Backup 20190406190808 completed
waiting result
to get schema from 10.32.0.28
INFO: /tmp/schema.cql has been generated
...
```

```
...
Copying from <TDIN>...
/ [1 files][ 0.0 B/ 0.0 B]
Operation completed over 1 objects.
INFO: backup created tarball and
transferred the file to
gs://cassandra-bkup/apigeecluster/dc-1
finished uploading schema.cql
```

The backup job creates a `schema.cql` file and uploads it to the Cloud Storage bucket. It then instructs the Cassandra node to back up the data and waits for the data to be uploaded.

You can monitor a backup job that has started based on your configured cron schedule.

First, use the `kubectl get pods` command to determine the name of the pod running the backup job.

Run the `kubectl logs` command, passing in the name of the pod running the backup job.

You will see output in the pod log as the backup job executes.

Note the backup snapshot timestamp. It is used to configure your override properties when your backup is restored.



## Restoring Cassandra in a single region

In a single region deployment, Apigee hybrid is deployed in a single data center or a region.

The Cassandra backup can reside either on Cloud Storage or on a remote server.

To restore Cassandra backups:

1. Verify the hybrid version to ensure that the version is the same that created the backup files.

```
$apigeectl version
```



In a single region deployment, Apigee hybrid is deployed in a single data center or a region. If you have multiple Apigee organizations in your deployment, the restore process restores data for all the organizations. In a multi-organization setup, you cannot restore a specific organization.

In your configuration, the Cassandra backup can reside either on Cloud Storage or on a remote server.

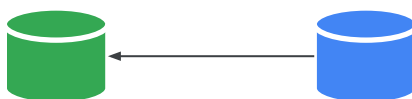
Ensure that the version of Apigee hybrid is the same version that created the backup files.

## Restoring Cassandra in a single region

2. Confirm that the Kubernetes cluster you are restoring to does not have a prior Apigee hybrid installation. This command will delete the existing Apigee hybrid installation:

```
$apigeectl delete -f overrides.yaml
```

```
$kubectl -n apigee get apigeedatastore,apigeeredis,apigeetelemetry,org,env,arc $apigeectl delete  
--all -f overrides.yaml
```



Confirm that the Kubernetes cluster you are restoring to does not have a prior Apigee hybrid installation. If you are restoring to the existing cluster, use the commands shown to delete the existing Apigee hybrid installation.

## Restoring Cassandra in a single region

3. Modify the overrides-restore.yaml file:
  - a. Specify the `namespace` element.
  - b. Add the `cassandra.restore` properties.
    - `snapshotTimestamp` is the timestamp associated with the backup you are restoring.

### overrides.yaml

```
namespace: your-restore-namespace
...
cassandra:
  storage:
    type: gcepd
    capacity: 500Gi
  gcepd:
    replicationType: regional-pd
  nodeSelector:
    key: cloud.google.com/gke-nodepool
    value: apigee-data
  ...
  restore:
    enabled: true
    snapshotTimestamp: "20190417002207"
    serviceAccountPath:
      "/service-accounts/apigee-cassandra.json"
    dbStorageBucket: "gs://cassandra-bkup"
    cloudProvider: "GCP"
  ...
```

Modify the overrides yaml file to specify the namespace element.

Add the restore configuration properties to the cassandra stanza.

Specify the timestamp of the backup snapshot to restore, the path to the service account file, and the Cloud Storage bucket.

## Restoring Cassandra in a single region

4. Create a new hybrid runtime deployment. This will create a new Cassandra cluster and begin restoring the backup data into the cluster:

```
$apigeectl init -f ./overrides-restore.yaml  
$apigeectl check-ready -f overrides/overrides.yaml  
$apigeectl apply -f ./overrides-restore.yaml  
$apigeectl check-ready -f overrides/overrides.yaml
```



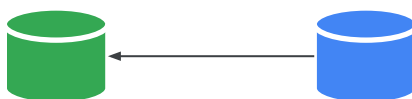
Create a new hybrid runtime deployment. This will create a new Cassandra cluster and begin restoring the backup data into the cluster.

## Restoring Cassandra in a single region

5. Verify the restoration job progress and confirm that apigeeds and all the other pods are up:

```
$kubectl get apigeeds -n apigee
```

```
$kubectl get pods -n apigee
```



Verify the restoration job progress and confirm that apigeeds and all the other pods are up. Upon successful completion of the restore and confirmation that the runtime components are healthy, it is recommended to configure a backup on the cluster.

## Viewing restore logs

- You can check the restore job logs to search for any errors:

```
$kubectl get pods -n <namespace>
```

NAME	READY	STATUS	RESTARTS	AGE
apigee-cassandra-restore-b4lhf	0/1	Completed	0	51m

- Run the kubectl logs command to view the pod log. Partial output follows:

```
$kubectl logs -f apigee-cassandra-restore-b4lhf -n <namespace>
```

```
...
INFO: download successfully extracted the
backup files from
gs://cassandra-bkup/apigeecluster/dc-1
    finished downloading schema.cql
    to create schema from 10.32.0.28
...
```

```
...
INFO: Restore 20190405011309 completed
INFO:
./apigee/data/cassandra/data/ks1/user-9fbae960571411e9
9652c7b15b2db6cc restored successfully
INFO: Restore 20190405011309 completed
waiting on waiting nodes $pid to finish 106
Restore finished
```

To monitor the progress of the restore job, first use the kubectl get pods command to determine the name of the pod running the job.

Run the kubectl logs command, passing in the name of the pod running the restore job.

You will see output in the pod log as the restore job executes and completes.

## Configuring file system backups

You can store backups of your Cassandra database to compressed files in the file system of a server you specify.

Follow these [steps](#) to configure file system backups:

1. Set up the server and configure SSH.
  - o After setup and configuration, verify that the server can be reached from the apigee-cassandra pods in the runtime plane over SSH.
2. Set up the backup schedule and destination.



Google Cloud

You can store backups of your Cassandra database to compressed files in the file system of a server you specify.

Backups occur on a schedule that you specify in your overrides file. The connection to the server is by secure SSH.

To configure a file system backup, you must first set up a server and install and configure SSH. As part of this process, you create a key pair.

The public key is copied to the server, and the location of the private key is configured in your overrides config file.

Test the SSH connection between your Cassandra pods in the runtime cluster and your backup server.

Next, you configure your overrides.yaml file with the backup configuration.

## Configuring file system backups - overrides.yaml

To set up the backup schedule and destination:

1. Edit your overrides.yaml config file to add entries for the cassandra backup. Specify:

- o Path to the SSH private key file
- o File system server IP address
- o Storage location on the file system to contain the backup  
cloudProvider property with value "HYBRID"
- o Cron schedule for the backup

2. Apply the changes to your cluster using apigeectl:

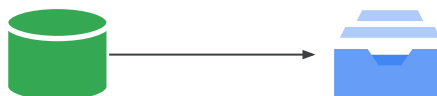
```
$ apigeectl apply --datastore -f overrides.yaml
```

3. Verify the backup job:

```
kubect1 get cronjob -n apigee
```

### overrides.yaml

```
cassandra:
  ...
  backup:
    enabled: true
    keyFile: "../keys/ssh_private.key"
    server: "34.56.78.90"
    storageDirectory:
      "/home/apigee/cassbackup"
    cloudProvider: "HYBRID"
    schedule: "45 23 * * 6"
  ...
```



Google Cloud

Edit your overrides.yaml configuration file to add the backup properties.

Set the enabled property to true, and provide the path to the SSH private key file and the backup server IP address.

Set the directory location where the backup files are to be stored on the backup server.

Set the required *cloudProvider* property to HYBRID, and set the cron schedule for the backup.

Apply the configuration changes to your cluster using the *apigeectl* command with the *datastore* flag.

Finally, verify the backup job.

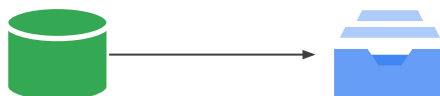


## Configuring file system restore

- The Cassandra backup can also reside on a remote server.
- The process for restoring is the same as Cloud Storage backups, except the overrides.yaml is configured differently.
- Modify the overrides-restore.yaml file:
  - a. Specify the `namespace` element.
  - b. Add the `cassandra.restore` properties.
    - `snapshotTimestamp` is the timestamp associated with the backup you are restoring.

### overrides.yaml

```
namespace: your-restore-namespace
...
cassandra:
  ...
  restore:
    enabled: true
    keyFile: "./keys/ssh_private.key"
    server: "34.56.78.90"
    storageDirectory:
      "/home/apigee/cassbackup"
    cloudProvider: "HYBRID"
    snapshotTimestamp: "20190417002207"
  ...
```



Google Cloud

The Cassandra backup can also reside on a remote server, and the process to restore is the same as mentioned in the previous slides on restoring cassandra in a single region.

For non-cloud storage restores, modify the overrides.yaml file to specify the namespace element.

Add the restore configuration properties to the cassandra stanza, for example, the path to the SSH private key file, the file system server IP address, the storage location on the file system to contain the backup, and the cloudProvider property with value "HYBRID".

Specify the timestamp of the backup snapshot to restore and the other properties used to configure the backup.

Then continue the restore process as you would for restoring Cassandra in a single region.

# Agenda

01	Installation Process
02	Lab: Installing and Managing Apigee Hybrid
03	Platform Operation
04	Backup and Restore
05	Quiz



# Question #1

## Question

Which of the following properties cannot be configured in the overrides.yaml file used in the hybrid installation process?

- A. Google Cloud project ID
- B. Namespace
- C. Service account files
- D. API proxy definition

## Question #1

### Answer

Which of the following properties cannot be configured in the overrides.yaml file used in the hybrid installation process?

- A. Google Cloud project ID
- B. Namespace
- C. Service account files
- D. API proxy definition



### Explanation:

- A. Incorrect. The Google Cloud project id must be configured in the overrides.yaml file to identify the project for sending log and metrics data.
- B. Incorrect. The namespace property identifies the Kubernetes cluster where the Apigee hybrid components are installed. It is optional and defaults to apigee.
- C. Incorrect. Service accounts files are required to be configured for most of the hybrid runtime components.
- D. Correct! API proxy definitions are created via the hybrid UI or Apigee API in the management plane and downloaded by the synchronizer to the runtime plane.

## Question #2

### Question

Which of the following methods of using the apigeectl command is invalid?

- A. `apigeectl apply -f ./overrides.yaml`
- B. `apigeectl init -f ./overrides.yaml -c environment`
- C. `apigeectl delete -f ./overrides.yaml`
- D. `apigeectl check-ready -f ./overrides.yaml`

## Question #2

### Answer

Which of the following methods of using the apigeectl command is invalid?

- A. `apigeectl apply -f ./overrides.yaml`
- B. `apigeectl init -f ./overrides.yaml -c environment`
- C. `apigeectl delete -f ./overrides.yaml`
- D. `apigeectl check-ready -f ./overrides.yaml`



### Explanation:

- A. Incorrect. This command applies the configuration specified in the `overrides.yaml` file to the hybrid runtime cluster.
- B. Correct! `-c` is not a valid flag to the `apigeectl` command.
- C. Incorrect. This command deletes the hybrid components in the cluster using the specified `overrides.yaml` file.
- D. Incorrect. This command checks the readiness state of the hybrid runtime component pods in the cluster.

## Question #3

### Question

Which of the following is a true statement regarding backup of the Cassandra datastore in the hybrid runtime plane?

- A. Cassandra backup is enabled by default.
- B. The Cassandra backup process does not need authentication to perform the backup.
- C. Apigee hybrid can only back up Cassandra to a Cloud Storage bucket.
- D. The Cassandra backup schedule is configured in the overrides configuration file using standard crontab syntax.

## Question #3

### Answer

Which of the following is a true statement regarding backup of the Cassandra datastore in the hybrid runtime plane?

- A. Cassandra backup is enabled by default.
- B. The Cassandra backup process does not need authentication to perform the backup.
- C. Apigee hybrid can only back up Cassandra to a Cloud Storage bucket.
- D. The Cassandra backup schedule is configured in the overrides configuration file using standard crontab syntax.

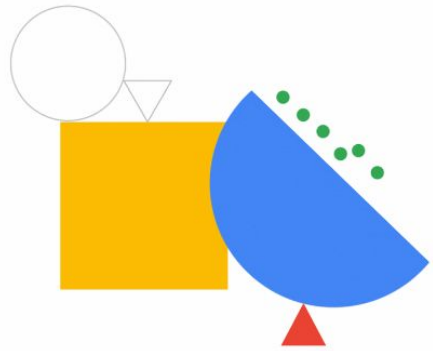


### Explanation:

- A. Incorrect. Cassandra backup is not enabled by default. It is a good practice to enable backups using the overrides configuration file.
- B. Incorrect. A service account with storage.objectAdmin role is required to perform the backup to Cloud Storage. Backups to the file system use SSH keys.
- C. Incorrect. You can configure Cassandra backups to an external file system.
- D. Correct! You configure the backup schedule using the schedule property in the overrides configuration file.



## Review: Installation and Platform Operation



In this module you learned how to install and configure Apigee hybrid on a Kubernetes platform. You installed hybrid on Google Kubernetes Engine in a lab.

You also learned how to use the *apigeectl* tool to initialize and configure the hybrid runtime cluster.

Finally, you learned about the backup and restore process for the runtime datastore component used in hybrid.