Table of Contents

[Section 1- Apigee Hybrid Installation 2](#_Toc75620832)

[Part 1 – Project and Organization Setup 2](#_Toc75620833)

[1. Crete a Google Account 2](#_Toc75620834)

[2. Create Google Cloud Project 2](#_Toc75620835)

[3. Enable APIs 2](#_Toc75620836)

[4. Create an Organization 3](#_Toc75620837)

[5. Create an environment group 9](#_Toc75620838)

[Part 2: - Hybrid runtime setup 17](#_Toc75620839)

[0.Before you Begin 17](#_Toc75620840)

[**0.1 Prerequisites** 17](#_Toc75620841)

[**0.2 Apigee hybrid: Supported platform** 18](#_Toc75620842)

[Step1-Create a cluster 19](#_Toc75620843)

[**1. Cluster Management** 19](#_Toc75620844)

[1.1 Minimum Cluster requirement for Apigee Cluster 19](#_Toc75620845)

[1.2 Configuring dedicated node pools 19](#_Toc75620846)

[1.3 Configure ports and set up firewalls 20](#_Toc75620847)

[**1.3.1 Internal Connection** 20](#_Toc75620848)

[**1.3.2 External connections** 23](#_Toc75620849)

[**1.3.3 Securing the runtime installation** 25](#_Toc75620850)

[**1.3.4 Data encryption** 26](#_Toc75620851)

[Step2-Install Cert Manager 26](#_Toc75620852)

[Step3- ASM Installation Steps 28](#_Toc75620853)

[**2.2** **Supported ASM versions** 28](#_Toc75620854)

[**2.3** **Installing Anthos Service Mesh on GKE** 29](#_Toc75620855)

[**2.4** **Required Google APIs-** 29](#_Toc75620856)

[**2.5** **Requirement** 30](#_Toc75620857)

[**2.6** **Restrictions** 31](#_Toc75620858)

[**2.7** **Choosing a certificate authority** 31](#_Toc75620859)

[**2.8** **Installing required tools** 31](#_Toc75620860)

[**2.9** **Running the script** 32](#_Toc75620861)

[**2.10** **Deploying and redeploying workloads** 37](#_Toc75620862)

[**2.11** **Enabling auto-injection** 38](#_Toc75620863)

[**2.12** **Perform final hybrid configuration and install ASM** 39](#_Toc75620864)

# Section 1- Apigee Hybrid Installation

<https://cloud.google.com/apigee/docs/hybrid/v1.4/precog-gcpaccount>

# **Part 1** – Project and Organization Setup

## Crete a Google Account

## Create Google Cloud Project

## Enable APIs

**Apigee API**

* 1. Apigee API
  2. Apigee Connect API
  3. Cloud Pub/Sub API
  4. Cloud Resource Manager API

**GKE API**

1. Compute Engine API
2. Google Kubernetes API

**The APIs you just added are displayed in the list of enabled APIs:**

* Apigee API
* Apigee Connect API
* Cloud Pub/Sub API
* Cloud Resource Manager API
* Compute Engine API
* Kubernetes Engine API

## **Create an Organization**

### **4.1 Prerequisite**

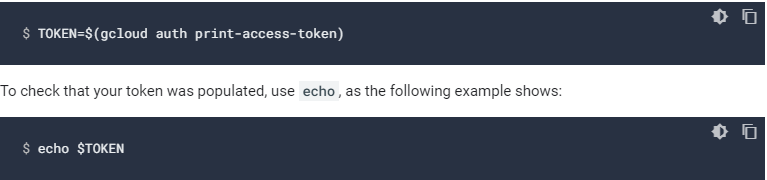
To create an organization, you must meet one of the following conditions:

* Have an evaluation account. Evaluation accounts expire after 60 days. At that time, the organization will be deleted.
* Have a paid account

If neither of these is true, then you must contact [Apigee Sales](https://pages.apigee.com/contact-sales-reg.html) before you can continue.

### **4.2 To create a new organization and provision it:**

1. On the command line, get your gcloud authentication credentials, as the following example shows:



*[root@centos7 ~]# TOKEN=$(gcloud auth print-access-token)*

*[root@centos7 ~]#*

*[root@centos7 ~]# echo $TOKEN*

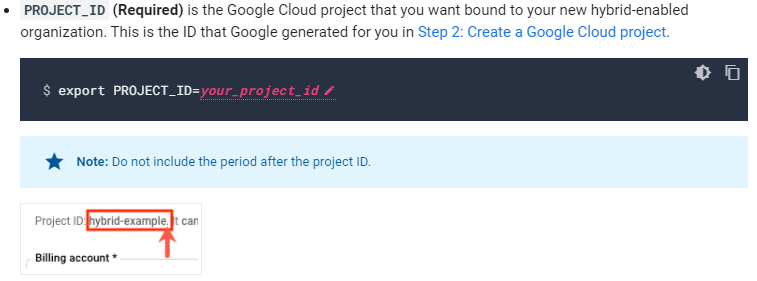
*ya29.a0AfH6SMDtFrt0NIvbZ2xNt4rl9vP8kSmRXrDaukBKxg-UC8ZJXFt6zHqMxWGFW595-\_eaQPpdjvkCZ8rJIfSWKV1S\_DxN2XpLS2jnLCSyBAdD1KEq5eApyvW\_gaP2Q1Y6PEughskfNTAuNof7SVMWXoFsFw3FmHvA2SUK*

*[root@centos7 ~]#*

1. Create the following environment variables for the elements of the organization. You will use these in the command to create the org.

* **PROJECT\_ID** **(Required)** is the Google Cloud project that you want bound to your new hybrid-enabled organization. This is the ID that Google generated for you in [Step 2: Create a Google Cloud project](https://cloud.google.com/apigee/docs/hybrid/v1.4/precog-gcpproject).

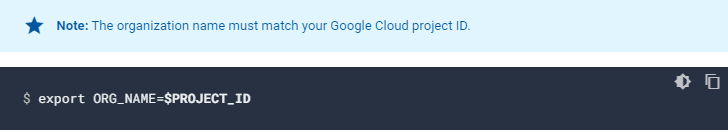
$ export PROJECT\_ID=**your\_project\_id**



* **ORG\_NAME** **(Required)** is the programmatic ID that you want for your hybrid-enabled organization.**Note:**

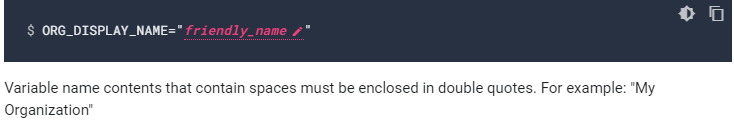
The organization name must match your Google Cloud project ID.

$ export ORG\_NAME=**$PROJECT\_ID**



* **ORG\_DISPLAY\_NAME** (Optional) is the user-friendly name for your organization. This value does not need to be unique, and can include spaces and special characters. For example, "My Hybrid Organization".

$ ORG\_DISPLAY\_NAME="**friendly\_name**"



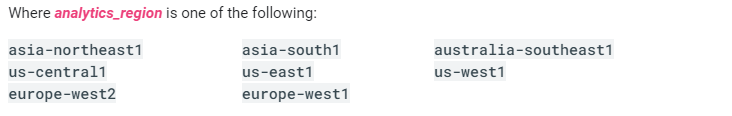
Variable name contents that contain spaces must be enclosed in double quotes. For example: "My Organization"

* **ORGANIZATION\_DESCRIPTION** (Optional) is information about the organization that you'd like to use as a reminder of its purpose. For example, "My first organization". $ ORGANIZATION\_DESCRIPTION="**description\_text**"



* **ANALYTICS\_REGION** **(Required)** is the primary region for analytics data storage. $ export ANALYTICS\_REGION=**analytics\_region**

Where **analytics\_region** is one of the following:



Choose a region that is geographically close or one that satisfies your organization's storage requirements.

* **RUNTIMETYPE** **(Required)** is the runtime type of the Apigee organization, where HYBRID is user-managed Apigee hybrid runtime.

**$** export RUNTIMETYPE=HYBRID



*export PROJECT\_ID=psp-apigee-hybrid*

*export ORG\_NAME=psp-apigee-hybrid*

*export ORG\_DISPLAY\_NAME="PSP Hybrid Org"*

*export ORGANIZATION\_DESCRIPTION="PSP Hybrid Org"*

*export ANALYTICS\_REGION=australia-southeast1*

*export RUNTIMETYPE=HYBRID*

1. Send an authenticated POST request to the [Create organizations API](https://cloud.google.com/apigee/docs/reference/apis/apigee/rest/v1/organizations/create)

curl -H "Authorization: Bearer **$TOKEN**" -X POST -H "content-type:application/json" \

-d '{

"name":"'"**$ORG\_NAME**"'",

"displayName":"'"**$ORG\_DISPLAY\_NAME**"'",

"description":"'"**$ORGANIZATION\_DESCRIPTION**"'",

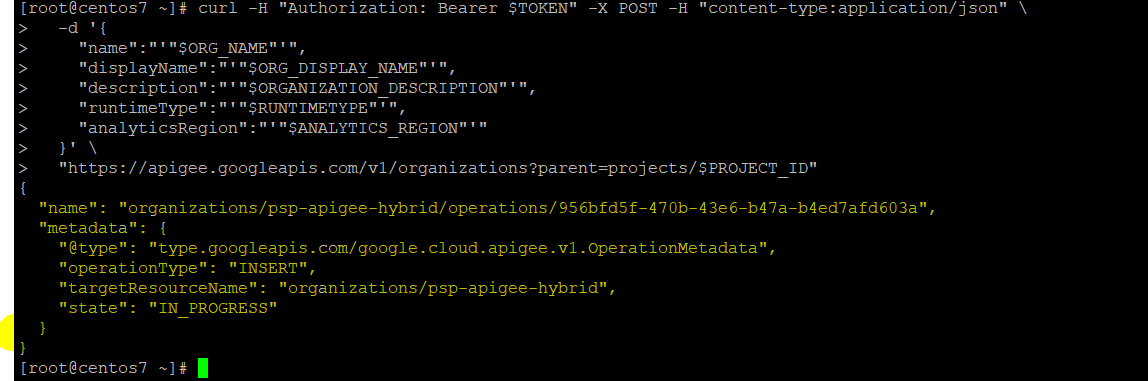
"runtimeType":"'"**$RUNTIMETYPE**"'",

"analyticsRegion":"'"**$ANALYTICS\_REGION**"'"

}' \

"https://apigee.googleapis.com/v1/organizations?parent=projects/**$PROJECT\_ID**"

On a successful creation request, the organizations API should respond with a message similar to the following:

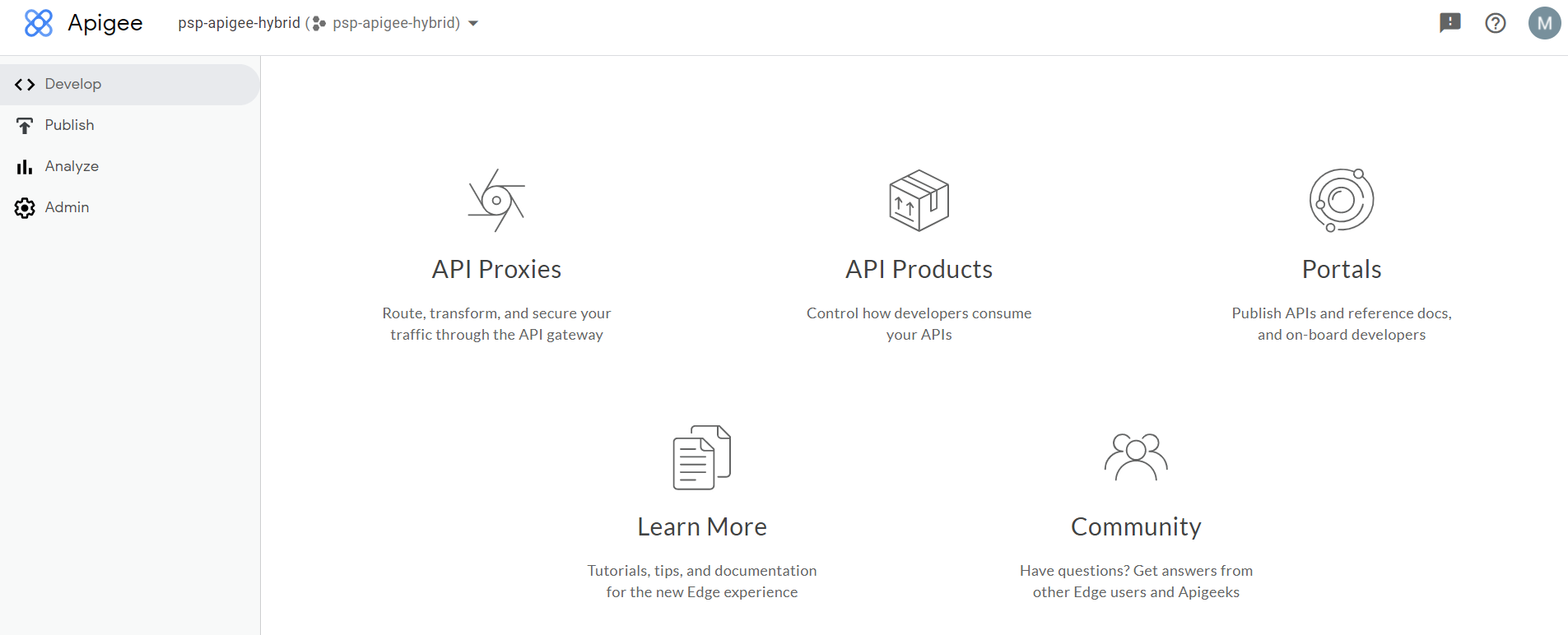


On a successful creation request, the organizations API should respond with a message similar to the following:

{  
  "name": "organizations/**org\_name**/operations/**long\_running\_operation\_ID**",  
  "metadata": {  
    "@type": "type.googleapis.com/google.cloud.apigee.v1.OperationMetadata",  
    "operationType": "INSERT",  
    "targetResourceName": "organizations/**org\_name**",  
    "state": "IN\_PROGRESS"  
  }  
}

Where:

* + **long\_running\_operation\_ID** is the UUID of an asynchronous, long-running operation. You can use this ID to check the status of your organization creation request (described in step 5).
  + **org\_name** is the ID of your new organization that is currently being created.
  + As the state property in the response indicates, Apigee has started to create the new organization, so its state is IN\_PROGRESS. This process can take several minutes.
  + If you get an error, see [Troubleshooting organization creation](https://cloud.google.com/apigee/docs/hybrid/v1.4/precog-provision#errors).
* *Your Apigee Hybrid Organization is setup now.*



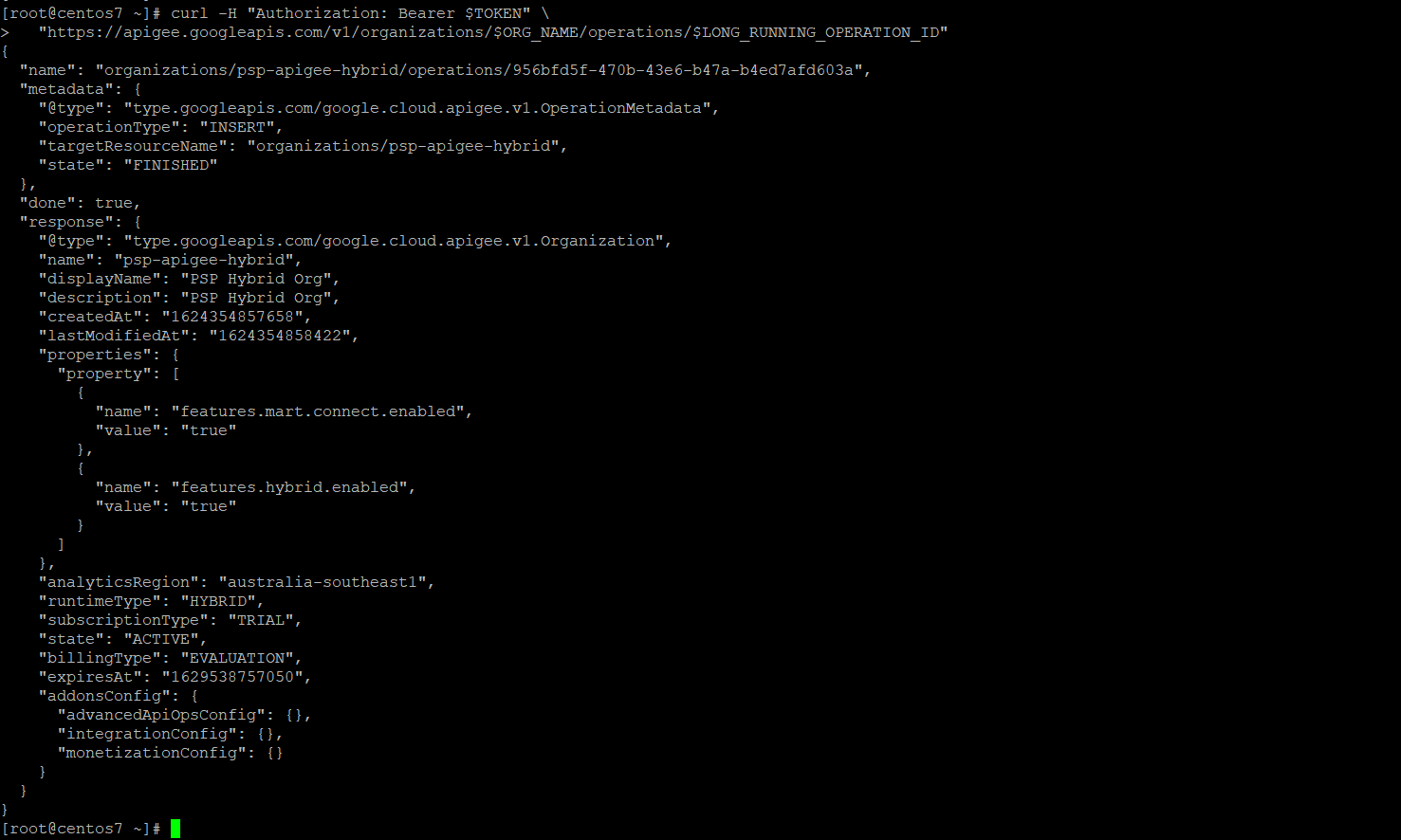
1. Save the long-running operation ID to an environment variable. It will be useful for future management tasks.

$ export LONG\_RUNNING\_OPERATION\_ID=956bfd5f-470b-43e6-b47a-b4ed7afd603a

1. You can check the status of the long-running operation whose ID Apigee returned on your initial creation request. To do this, use the operations API. For example:

$ curl -H "Authorization: Bearer **$TOKEN"** \

[https://apigee.googleapis.com/v1/organizations/**$ORG\_NAME**/operations/**$LONG\_RUNNING\_OPERATION\_ID**](https://apigee.googleapis.com/v1/organizations/$ORG_NAME/operations/$LONG_RUNNING_OPERATION_ID)"



**Congratulations!**

You have created a new organization and it is ready for you to use. Continue to [Step 5: Add a domain and reserve a static IP](https://cloud.google.com/apigee/docs/hybrid/v1.4/precog-add-dns).

## [**5: Add a domain and reserve a static IP**](https://cloud.google.com/apigee/docs/hybrid/v1.4/precog-add-dns)**.**

### **Reserve a static IP**

Follow the steps below to create a static IP address for the runtime load balancer. This address exposes the hybrid runtime to API proxy requests coming from outside the cluster.

**Note:**It's important to create a static IP address, because when pods are restored or when you upgrade, it's possible that the external IP will change, causing an interruption in API traffic. By creating a static IP, the address exposed outside the cluster address will never change unless you release it.

1. Open the [Google Cloud Console](https://console.cloud.google.com/) and log in with the account you created in [Step 1: Create a Google Cloud account](https://cloud.google.com/apigee/docs/hybrid/v1.4/precog-gcpaccount).
2. Select the project that you created in [Step 2: Create a Google Cloud project](https://cloud.google.com/apigee/docs/hybrid/v1.4/precog-gcpproject).
3. If you have not already done so, enable the **Cloud DNS API**. See [Enabling APIs](https://cloud.google.com/apis/docs/getting-started#enabling_apis).
4. Create a static IP address:

**Note:**The static IP address will be the IP address that is exposed outside the cluster, so give it a name that makes sense to you for that purpose. For example: **apigee-loadbalancer**

* If you are on GKE, follow the instructions in [Reserving a static external IP address](https://cloud.google.com/compute/docs/ip-addresses/reserve-static-external-ip-address#reserve_new_static) to create a static IP address.
* If you are on GKE on-prem (Anthos), follow the instructions in the [Installing using static IP addresses](https://cloud.google.com/anthos/gke/docs/on-prem/how-to/install-static-ips) to create a static IP address.

1. Copy the external IP address you just reserved. For example: 34.56.78.90. You will need it in a later step.

### **Get a registered domain**

If you already have a registered domain, you can use it for the Apigee hybrid installation. If not, you can register a domain name through [Google Domains](https://domains.google/) or another domain registrar of your choice.

### **Configure your DNS**

Create a new A type record in a zone in your DNS server as follows:

* Specify a DNS zone subdomain name. For example, suppose your zone's parent domain is acme.com, the subdomain name might be myapi. In this example, the address would be: myapi.acme.com.
* Set the A record address to the numeric IP address of your **static IP**, in dotted decimal format. For example: 34.56.78.90.

If you do not have an existing domain available to use, we provide a quickstart example that shows how to use Google Domains and Google Cloud DNS to create a domain and set up a DNS zone. See [DNS quickstart](https://cloud.google.com/apigee/docs/hybrid/v1.4/dns-quickstart).

**View Organization Details**

You can view the metadata details for the organization you created using an Apigee API. You can also use an API to list all of the organizations to which your Google Cloud account has access. To perform these actions, you use the [organizations API](https://cloud.google.com/apigee/docs/reference/apis/apigee/rest/v1/organizations).

Before trying the APIs, refresh your authorization token:

TOKEN=$(gcloud auth print-access-token)

Send a GET request (with no body) to the following [Get organizations API](https://cloud.google.com/apigee/docs/reference/apis/apigee/rest/v1/organizations/get) endpoint:

https://apigee.googleapis.com/v1/organizations/**org\_name**

[root@centos7 ~]# curl -H "Authorization: Bearer $TOKEN" \

> "https://apigee.googleapis.com/v1/organizations/psp-apigee-hybrid"

{

"name": "psp-apigee-hybrid",

"displayName": "PSP Hybrid Org",

"description": "PSP Hybrid Org",

"createdAt": "1624354857658",

"lastModifiedAt": "1624354858422",

"properties": {

"property": [

{

"name": "features.mart.connect.enabled",

"value": "true"

},

{

"name": "features.hybrid.enabled",

"value": "true"

}

]

},

"analyticsRegion": "australia-southeast1",

"runtimeType": "HYBRID",

"subscriptionType": "TRIAL",

"projectId": "psp-apigee-hybrid",

"state": "ACTIVE",

"billingType": "EVALUATION",

"expiresAt": "1629538757050",

"addonsConfig": {

"advancedApiOpsConfig": {},

"integrationConfig": {},

"monetizationConfig": {}

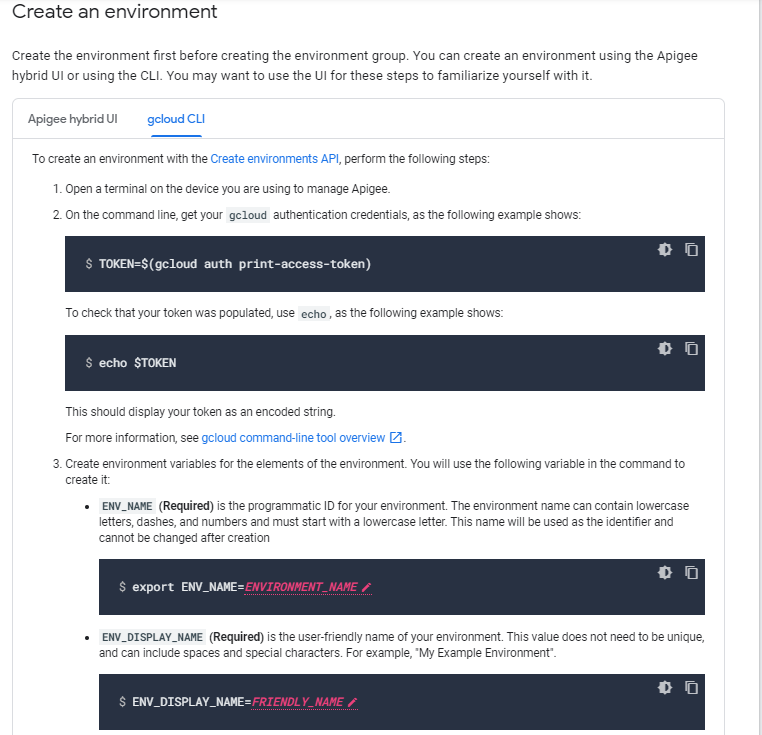
}

}

[root@centos7 ~]#

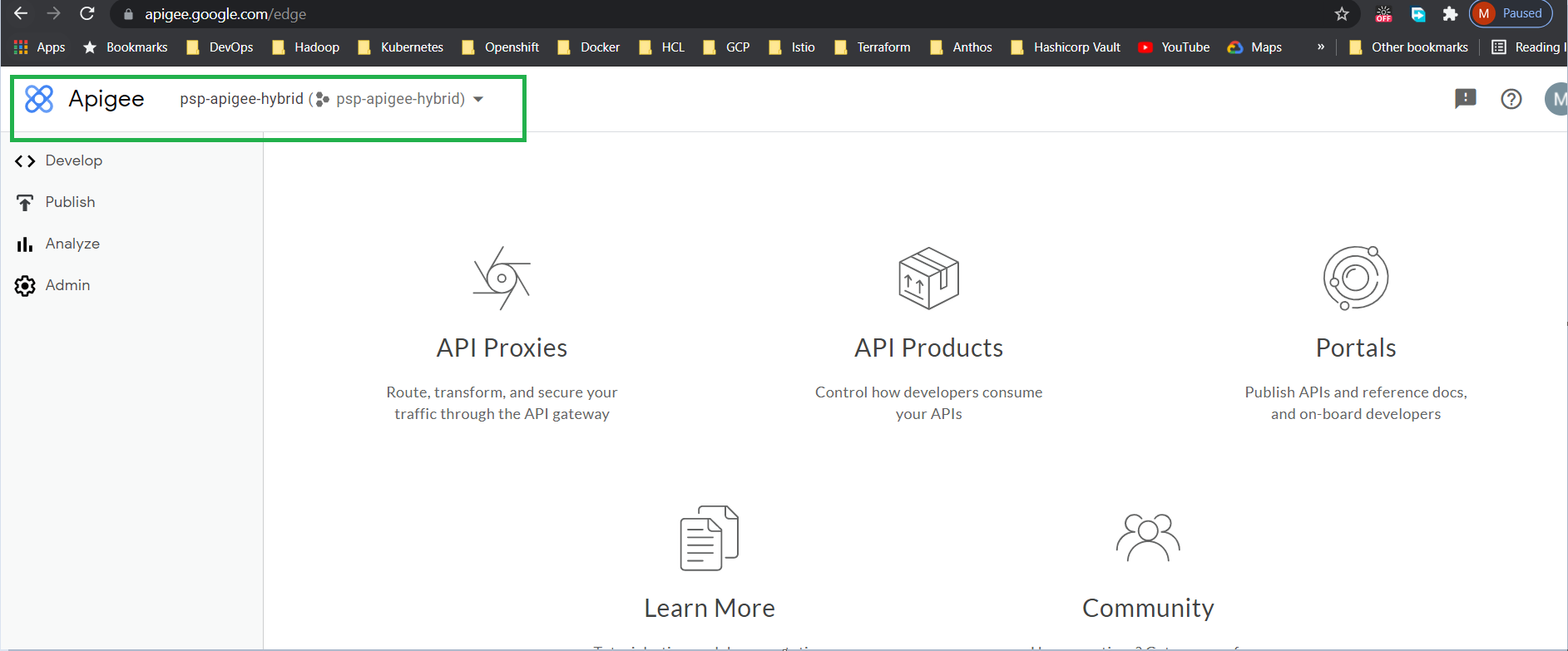
## **Create an environment group**

**Using gcloud CLI**



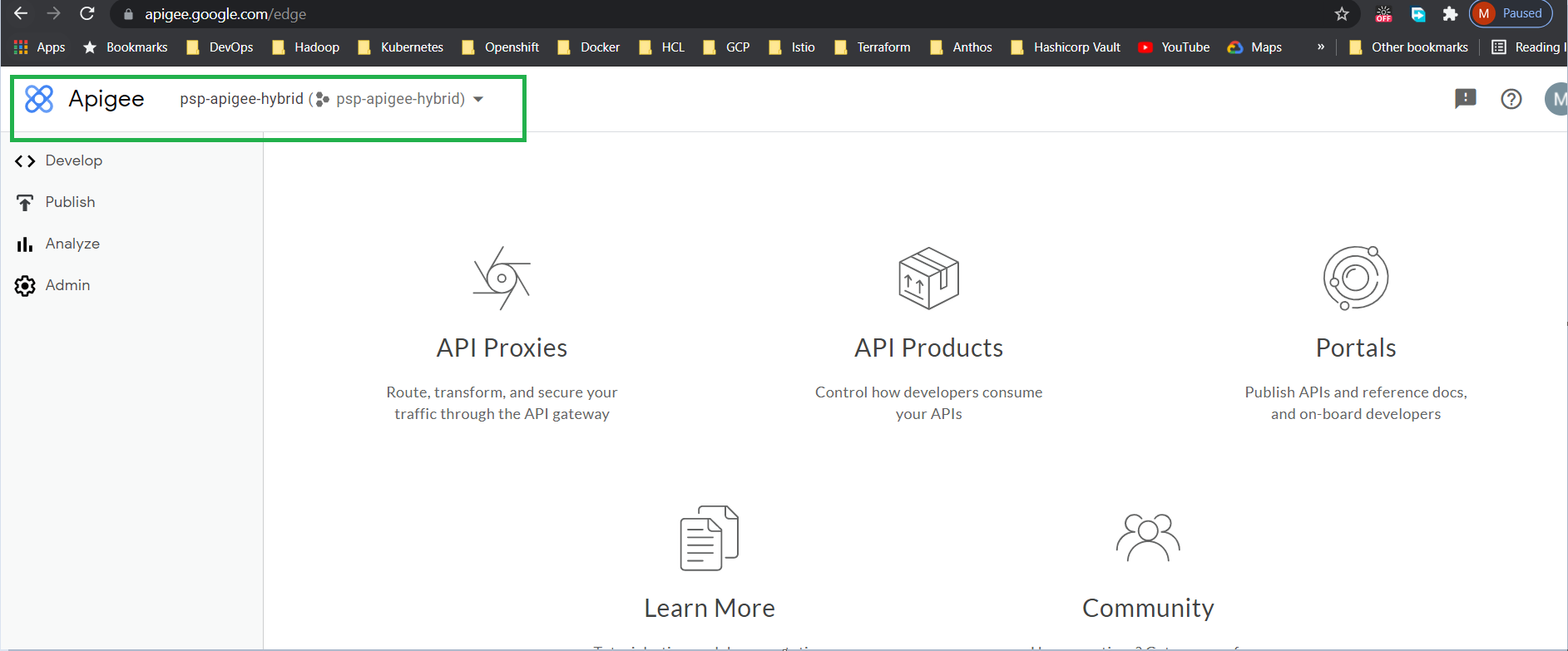


1. Open the [Apigee UI](https://apigee.google.com/). If this is the first time you are logging in, hybrid displays a consent dialog. If you are prompted to select from more than one account, choose the account that owns the project you created in [Step 2: Create a Google Cloud project](https://cloud.google.com/apigee/docs/hybrid/v1.4/precog-gcpproject). Then click **Allow**.

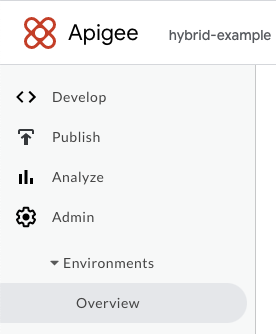


1. Ensure that your hybrid-enabled organization is selected from the organization drop-down

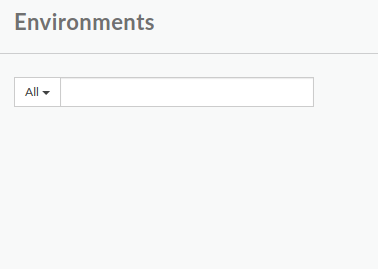
list. If it is not selected, select it from the drop-down list, as the following example shows:



3.Click **Admin > Environments > Overview**.

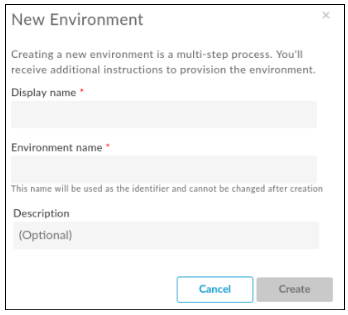


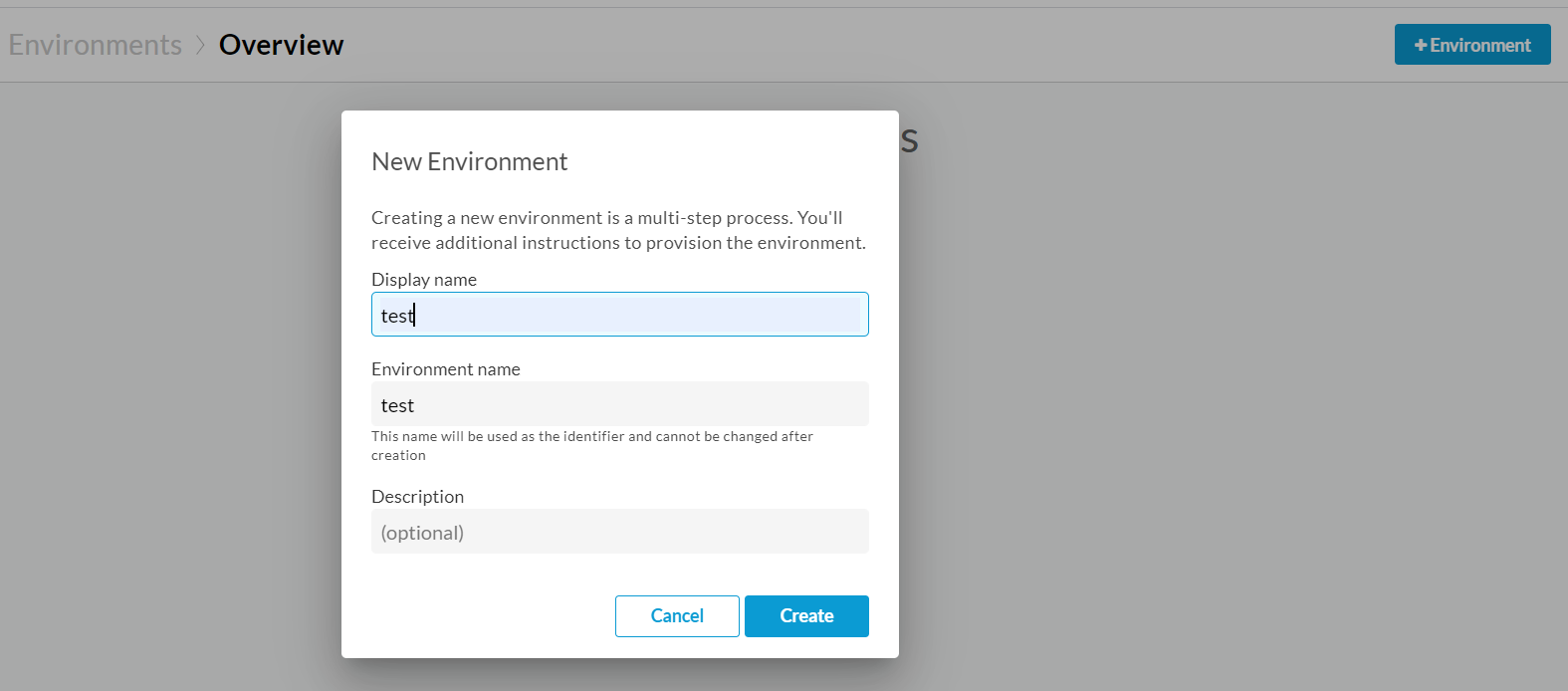
The **Environments** view is displayed, with no environments (yet):



4.Click **+Environment**.

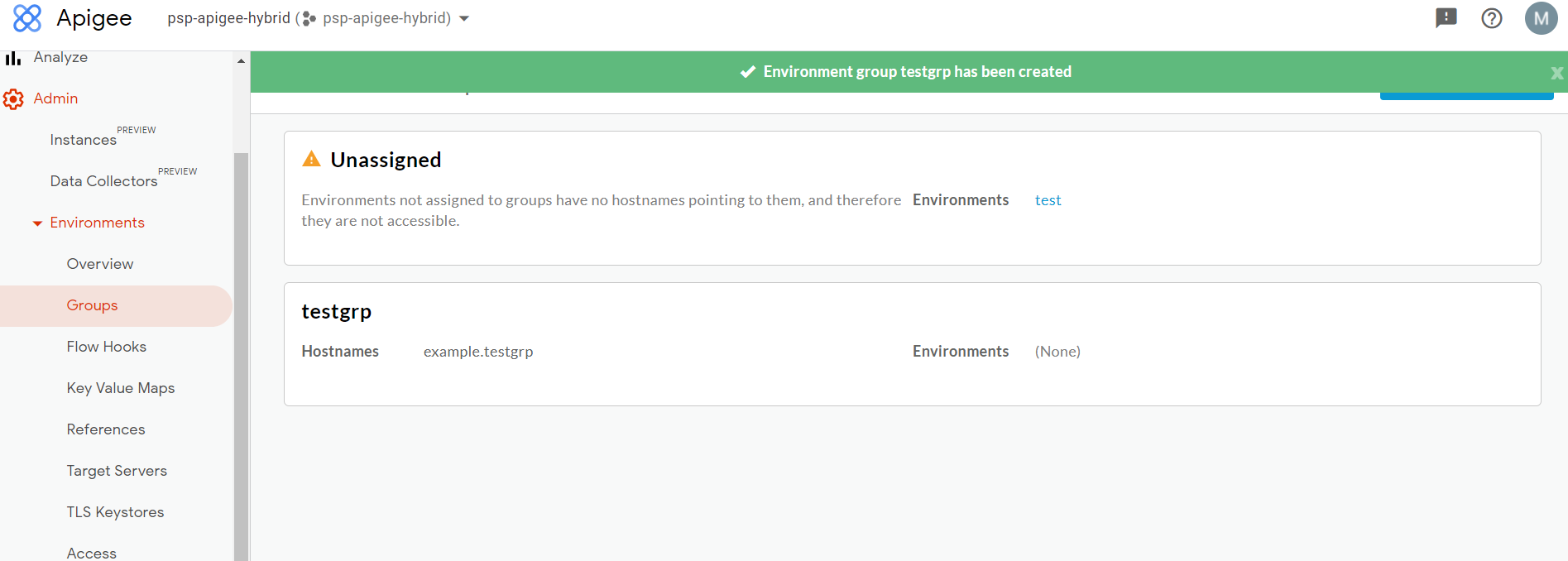
The **New environment** dialog is displayed:





**Create an environment group**

Now create an environment group and assign the environment you just created to it. Environment groups allow you to group environments together, and provide the hostnames for routing the proxies deployed to the environments within the group. You must create at least one environment group, and you must assign at least one hostname to the group. For an overview of environment groups, see [About environments and environment groups](https://cloud.google.com/apigee/docs/api-platform/fundamentals/environments-overview).



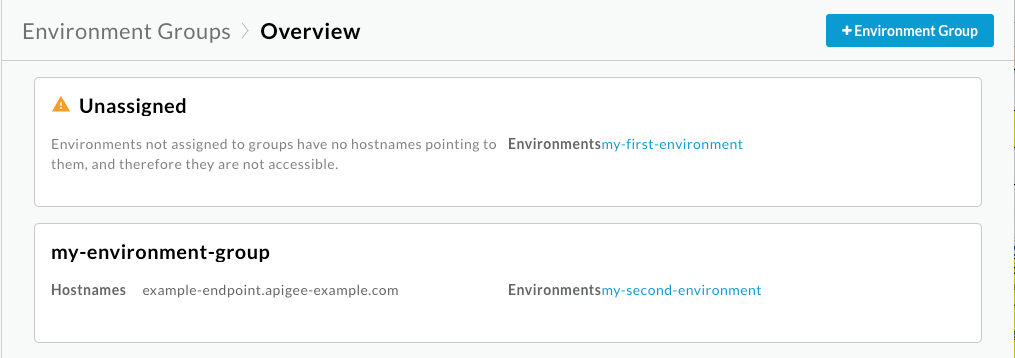
To create an environment group using the Apigee UI, perform the following steps:

1. Open the [Apigee UI](https://apigee.google.com/).

Your browser navigates to the hybrid UI main landing page.

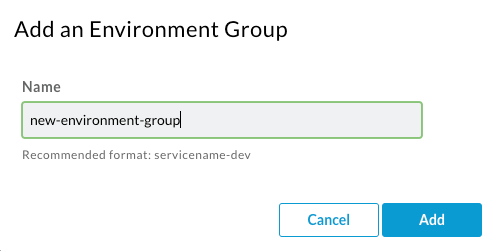
1. Ensure that your hybrid-enabled organization is selected from the organization drop-down list. If it is not selected, select it from the drop-down list.
2. Click **Admin > Environments > Groups**.

The **Environment Groups Overview** view is displayed, with any created environment groups:



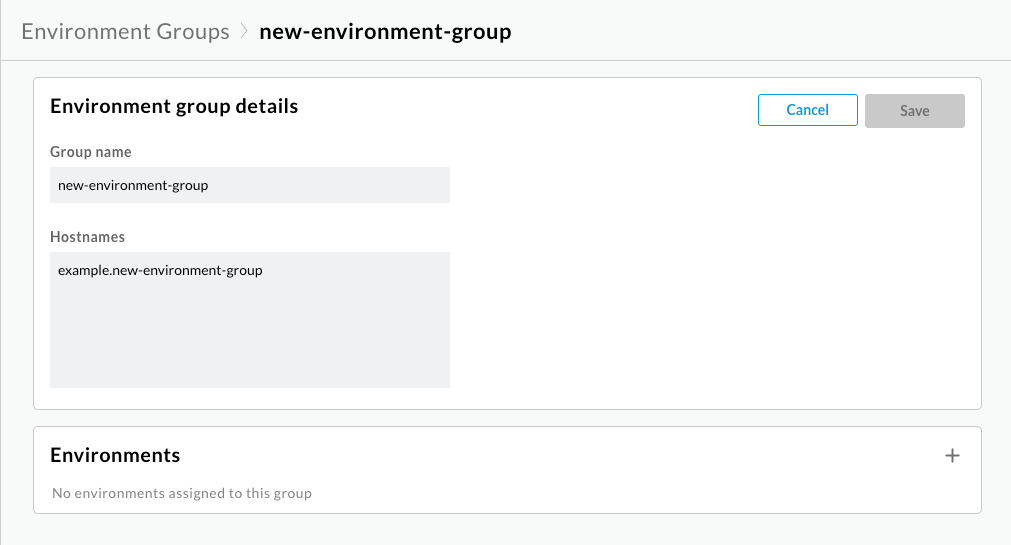
1. Click **+Environment Group**.

The **Add an Environment Group** dialog is displayed.

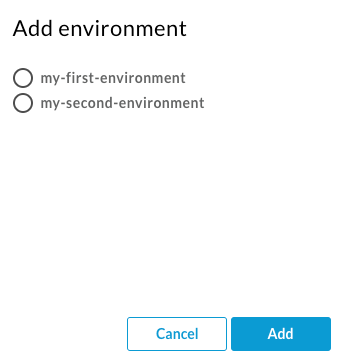


1. Enter a name and then click **Add**.
2. Hold the pointer over the newly-created environment group and then click **Edit** edit.

Add a hostname that all proxies deployed to environments within this group will use. This should be a domain you have management access to. If you need to register a domain, see a domain registrar like [Google Domains](https://domains.google/).



1. Click **Add (+)**.
2. The **Add environment** dialog displays.



1. Select an environment from the list and then click **Add**.

<https://cloud.google.com/apigee/docs/hybrid/v1.4/2-1-install-create-cluster>

# **Part 2: - Hybrid runtime setup**

## 0.Before you Begin

Each supported platform has its own permission requirements for creating a cluster. After the cluster is created. As cluster owner, you can proceed to install the Apigee-specific components (including Apigee, ASM, and cert-manager) into the cluster. However, if you want to delegate to another user the installation of the runtime components into the cluster, you can manage the necessary permissions through Kubernetes [authn-authz](https://kubernetes.io/docs/reference/access-authn-authz/rbac/).

To install the hybrid runtime components into the cluster, a non-cluster-owner user should have CRUD permission on these resources:

* ClusterRole
* Webhooks (ValidatingWebhookConfiguration and MutatingWebhookConfiguration)
* PriorityClass
* ClusterIssuer
* CustomerResourceDefinitions
* StorageClass (optional, if the default StorageClass is not used)

### **0.1 Prerequisites**

Some basic prerequisites which should be completed before we proceed for Apigee Hybrid installation.

 Complete the Google Cloud and UI setup steps in [Configure Google Cloud services and hybrid UI](https://cloud.google.com/apigee/docs/hybrid/v1.4/precog-overview).

 Have a domain name you can use for your Apigee hybrid installation. This should be a domain you have management access to. If you need to register a domain, see a domain registrar like [Google Domains](https://domains.google/).

### **0.2 Apigee hybrid: Supported platform**

Apigee hybrid supports the following versions of these platforms:



## Step1-Create a cluster

### 1. Cluster Management

These steps assume you have created environment variables in the previous steps. Check them with the following commands:

echo $PROJECT\_ID

echo $ANALYTICS\_REGION

echo $ORG\_NAME

echo $ENV\_NAME

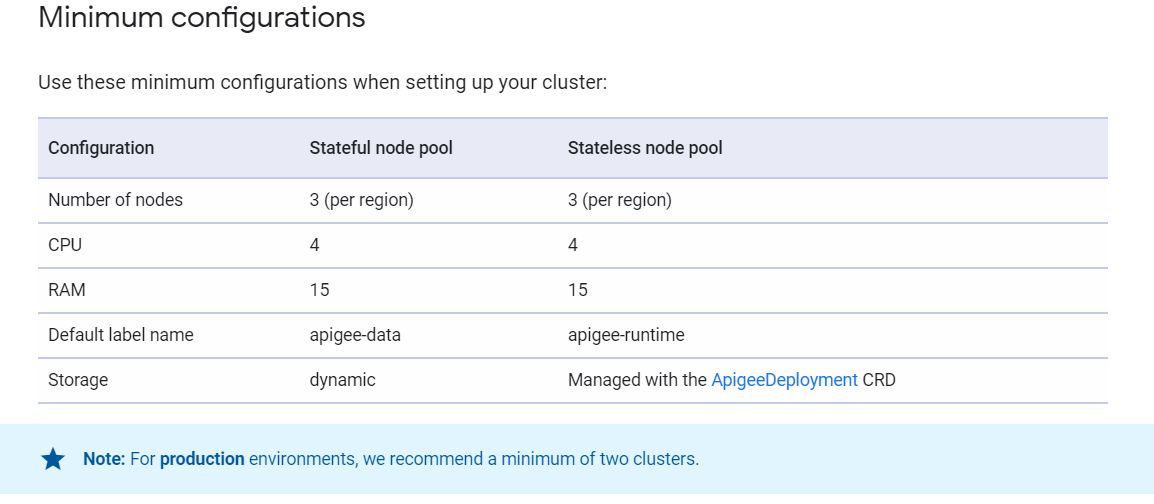
#### 1.1 Minimum Cluster requirement for Apigee Cluster

A node pool is a group of nodes within a cluster that all have the same configuration. By

default, hybrid assigns all pods to the default node pool; however, you can create dedicated

node pools and assign hybrid components to them as a way of distributing resources.

Typically, you define dedicated node pools when you have pods with differing resource requirements. For example, the apigee-cassandra pods require persistent storage, while the other Apigee hybrid pods do not. For this reason, we recommend that you create a stateful node pool for Cassandra and a stateless node pool for the rest of the hybrid runtime services. See [Configure dedicated node pools](https://cloud.google.com/apigee/docs/hybrid/v1.4/configure-dedicated-nodes) for details



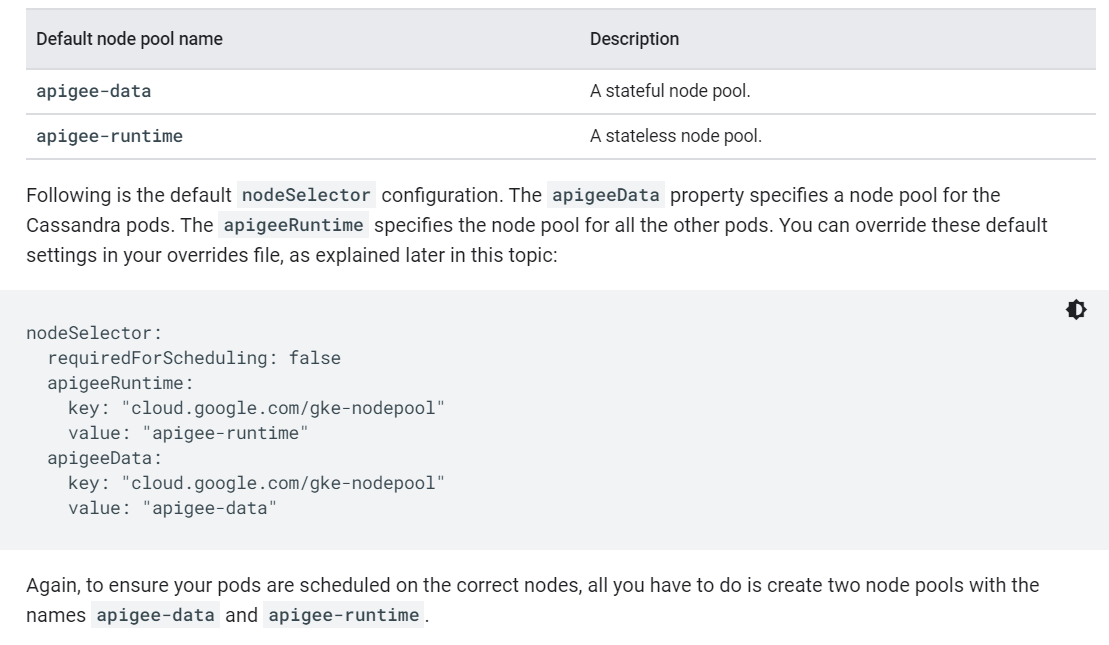
#### **1.2 Configuring dedicated node pools**

A [node pool](https://cloud.google.com/kubernetes-engine/docs/concepts/node-pools) is a group of nodes within a cluster that all have the same configuration. *Typically, you define separate node pools when you have pods with differing resource requirements*. For example, the apigee-cassandra pods require persistent storage, while the other Apigee hybrid pods do not.

This topic discusses how to configure dedicated node pools for a hybrid installation.

Using the default nodeSelectors

The best practice is to set up two dedicated node pools: one for the Cassandra pods and one for all the other runtime pods. Using default [nodeSelector](https://kubernetes.io/docs/concepts/configuration/assign-pod-node/" \l "nodeselector) configurations, the installer will assign the Cassandra pods to a *stateful* node pool named apigee-data and all the other pods to a *stateless* node pool named apigee-runtime. All you have to do is create node pools with these names, and Apigee hybrid handles the pod scheduling details for you:

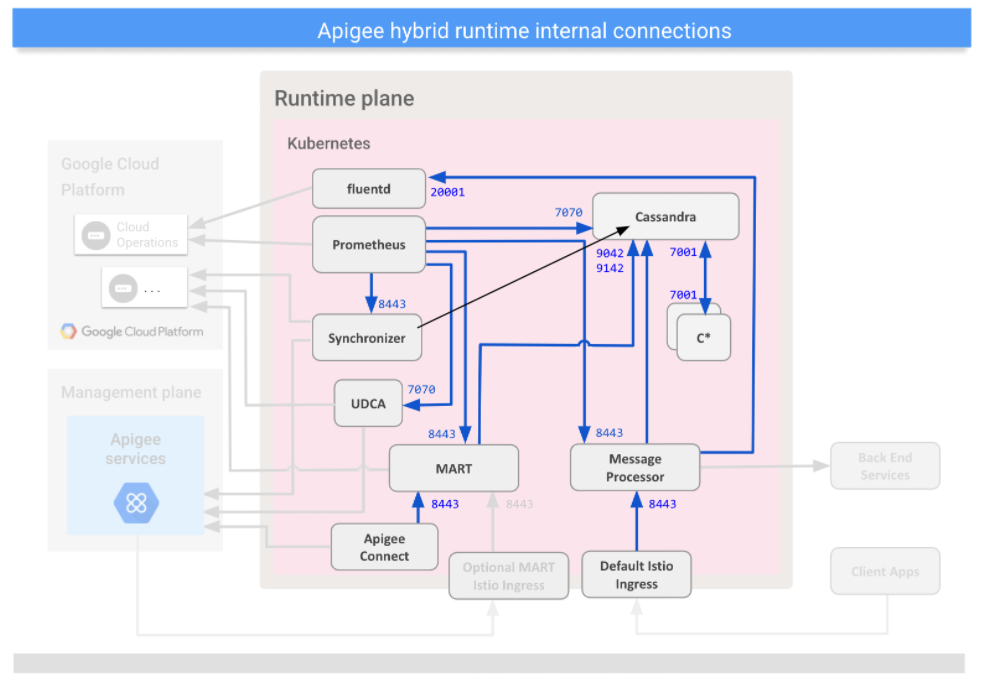


#### **1.3 Configure ports and set up firewalls**

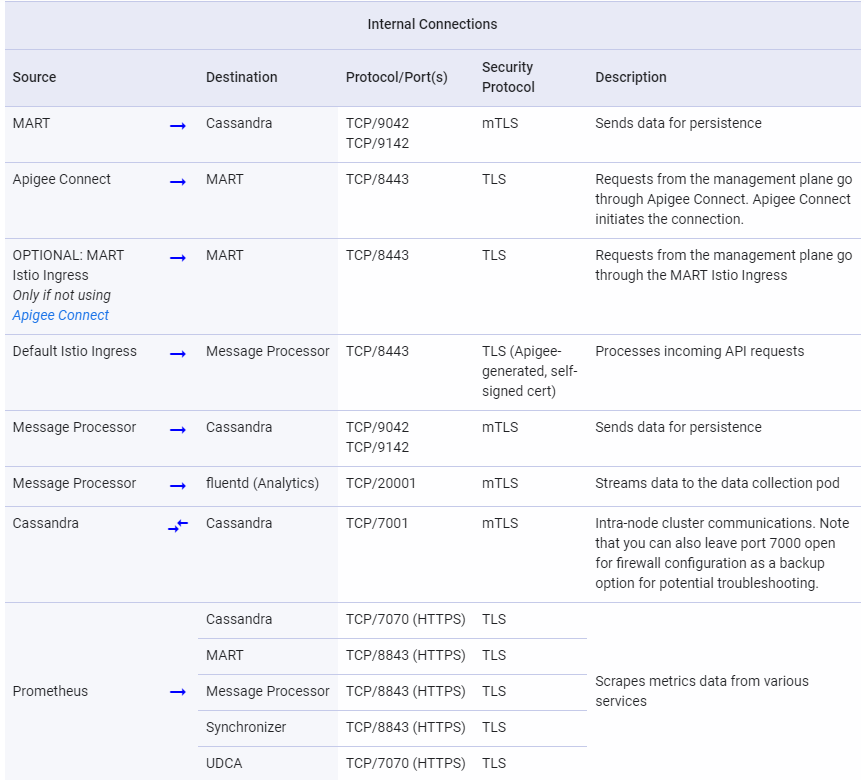
Understanding which ports the hybrid runtime plane uses is important for enterprise implementations

##### **1.3.1 Internal Connection**

Communication between the runtime plane and management plane is secured with TLS 1-way and OAuth 2.0. Individual services use different protocols, depending on which service they are communicating with.



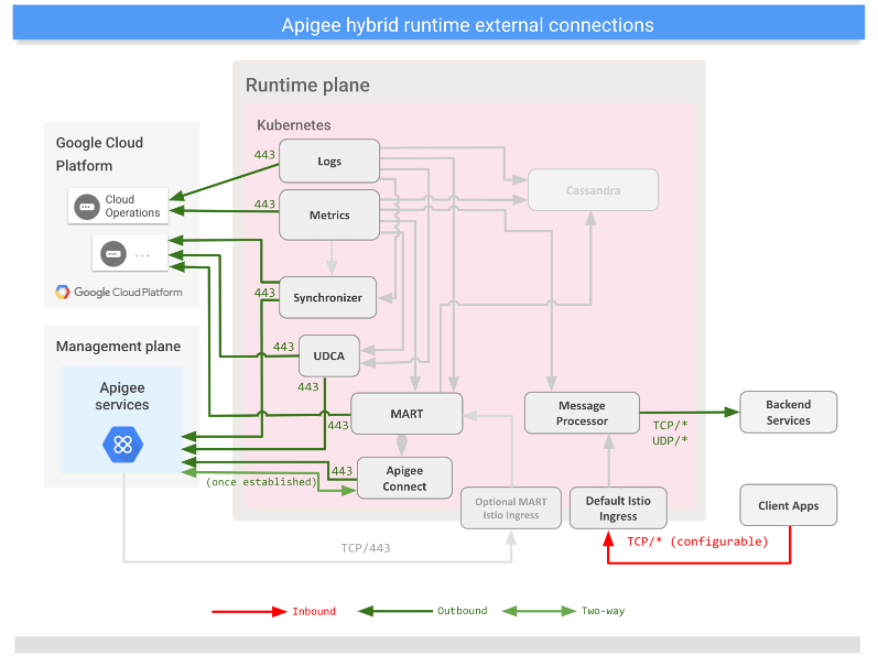
The following table describes the ports and communications channels within the hybrid runtime plane:



##### **1.3.2 External connections**

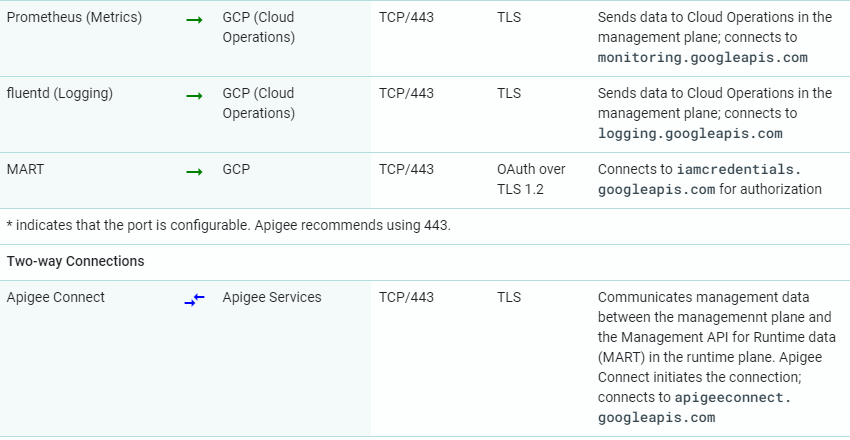
To appropriately configure your network firewall, you should know the inbound and outbound ports used by hybrid to communicate with external services.

The following image shows the ports used for external communications with the hybrid runtime plane:



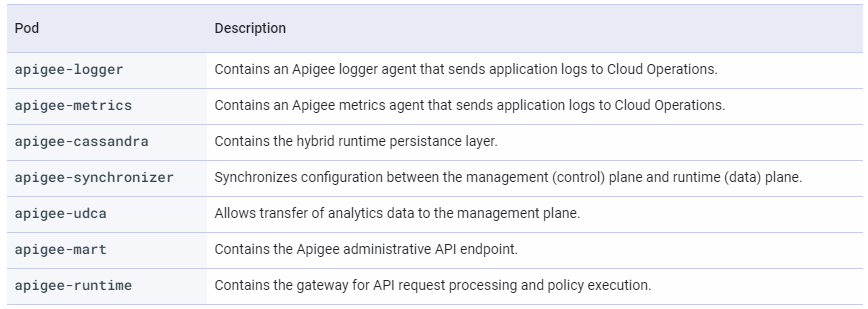
The following table describes the ports used for external communications with the hybrid runtime plane:





##### **1.3.3 Securing the runtime installation**

A typical Apigee hybrid installation is made of multiple pods, as listed in the following table. Each of these pods require specific access to ports, and not every pod needs to communicate with every other pod. For a detailed map of these internal connections and the security protocols they employ, see [Internal connections](https://cloud.google.com/apigee/docs/hybrid/v1.4/ports#internal).



##### **1.3.4 Data encryption**

By default, the following data is stored *encrypted* in the hybrid runtime plane:

* Key management system (KMS) data
* Key-value map (KVM) data
* Cache data

Data encryption does not require any special configuration on your part. However, if for some reason you want to use your own encryption keys (replacing the default ones) you can do so, as explained in this topic.

By default, Apigee hybrid provides a set of Base64-encoded keys that are used to encrypt KVM, KMS, and cache data. The Apigee hybrid installer stores the keys in the runtime plane as [Kubernetes Secrets](https://kubernetes.io/docs/concepts/configuration/secret/), and uses them to encrypt your data with AES-128 standard encryption. The keys are under your control; the hybrid management plane is never aware of them at any time.

Refer - <https://cloud.google.com/apigee/docs/hybrid/v1.4/key-encryption>

## **Step2-Install Cert Manager**

Use the following command to install cert-manager v0.14.2 from GitHub

kubectl apply --validate=false -f <https://github.com/jetstack/cert-manager/releases/download/v0.14.2/cert-manager.yaml>

[root@centos7 Downloads]# kubectl apply --validate=false -f https://github.com/jetstack/cert-manager/releases/download/v0.14.2/cert-manager.yaml

customresourcedefinition.apiextensions.k8s.io/certificaterequests.cert-manager.io created

customresourcedefinition.apiextensions.k8s.io/certificates.cert-manager.io created

customresourcedefinition.apiextensions.k8s.io/challenges.acme.cert-manager.io created

customresourcedefinition.apiextensions.k8s.io/clusterissuers.cert-manager.io created

customresourcedefinition.apiextensions.k8s.io/issuers.cert-manager.io created

customresourcedefinition.apiextensions.k8s.io/orders.acme.cert-manager.io created

namespace/cert-manager created

serviceaccount/cert-manager-cainjector created

serviceaccount/cert-manager created

serviceaccount/cert-manager-webhook created

clusterrole.rbac.authorization.k8s.io/cert-manager-cainjector created

clusterrole.rbac.authorization.k8s.io/cert-manager-view created

clusterrole.rbac.authorization.k8s.io/cert-manager-controller-ingress-shim created

clusterrole.rbac.authorization.k8s.io/cert-manager-controller-issuers created

clusterrole.rbac.authorization.k8s.io/cert-manager-controller-certificates created

clusterrole.rbac.authorization.k8s.io/cert-manager-controller-orders created

clusterrole.rbac.authorization.k8s.io/cert-manager-controller-challenges created

clusterrole.rbac.authorization.k8s.io/cert-manager-controller-clusterissuers created

clusterrole.rbac.authorization.k8s.io/cert-manager-edit created

clusterrolebinding.rbac.authorization.k8s.io/cert-manager-cainjector created

clusterrolebinding.rbac.authorization.k8s.io/cert-manager-controller-clusterissuers created

clusterrolebinding.rbac.authorization.k8s.io/cert-manager-controller-orders created

clusterrolebinding.rbac.authorization.k8s.io/cert-manager-controller-certificates created

clusterrolebinding.rbac.authorization.k8s.io/cert-manager-controller-ingress-shim created

clusterrolebinding.rbac.authorization.k8s.io/cert-manager-controller-issuers created

clusterrolebinding.rbac.authorization.k8s.io/cert-manager-controller-challenges created

role.rbac.authorization.k8s.io/cert-manager-cainjector:leaderelection created

role.rbac.authorization.k8s.io/cert-manager:leaderelection created

rolebinding.rbac.authorization.k8s.io/cert-manager-cainjector:leaderelection created

rolebinding.rbac.authorization.k8s.io/cert-manager:leaderelection created

service/cert-manager created

service/cert-manager-webhook created

deployment.apps/cert-manager-cainjector created

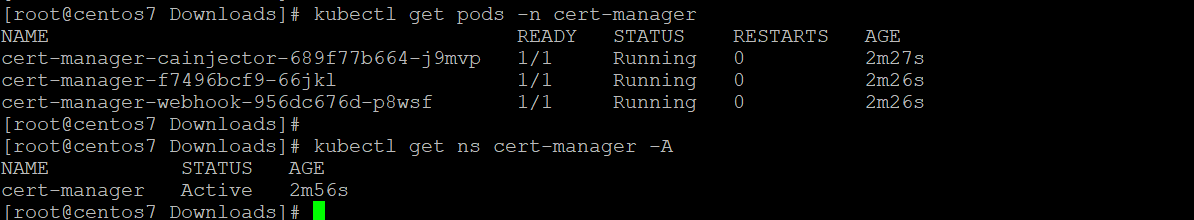
deployment.apps/cert-manager created

deployment.apps/cert-manager-webhook created

mutatingwebhookconfiguration.admissionregistration.k8s.io/cert-manager-webhook created

validatingwebhookconfiguration.admissionregistration.k8s.io/cert-manager-webhook created

[root@centos7 Downloads]#



**Pod image name in cert-manager-**

-------Pod Name: cert-manager-cainjector-689f77b664-j9mvp-------

Image ID: quay.io/jetstack/cert-manager-cainjector@sha256:6af7f9b167f6a7613d763cd4b1e00ca6b49668639fe64e40bb7f3d085f121000

-------Pod Name: cert-manager-f7496bcf9-66jkl-------

Image ID: quay.io/jetstack/cert-manager-controller@sha256:21e6a1fbdf8a82c63c40f03f62d80406b03c9ebca48f10ad89b23d7dfbba998c

-------Pod Name: cert-manager-webhook-956dc676d-p8wsf-------

Image ID: quay.io/jetstack/cert-manager-webhook@sha256:63c15ac43360287b471b534508a30e9507d4885d33e98792c227b01fd19b0c44

[root@centos7 Downloads]#

## **Step3- ASM Installation Steps**

* + 1. Follow the ASM setup and configuration steps
    2. When you have completed the ASM setup and config steps, go to the next section to complete the hybrid configuration and ASM installation steps.
    3. Add (or update) the spec:components stanza in the istio-operator.yaml file below the meshConfig: section and immediately above values:, where **reserved\_static\_ip** is an IP address your runtime ingress gateway can use. If you do not have a reserved static IP address, for this Quick Start, you can leave the LoadBalancerIP property out.
    4. Return now to the ASM documentation you used previously, and complete ASM installation (install or apply the istio-operator.yaml file to the cluster). If you are on ASM 1.6 and given a choice, choose **PERMISSIVE mTLS**.

### **3.1 Supported ASM versions**

* For new hybrid version 1.4.4 installations, install ASM version 1.7.x.
* For upgrades from previous versions of hybrid, install ASM version 1.6.x.

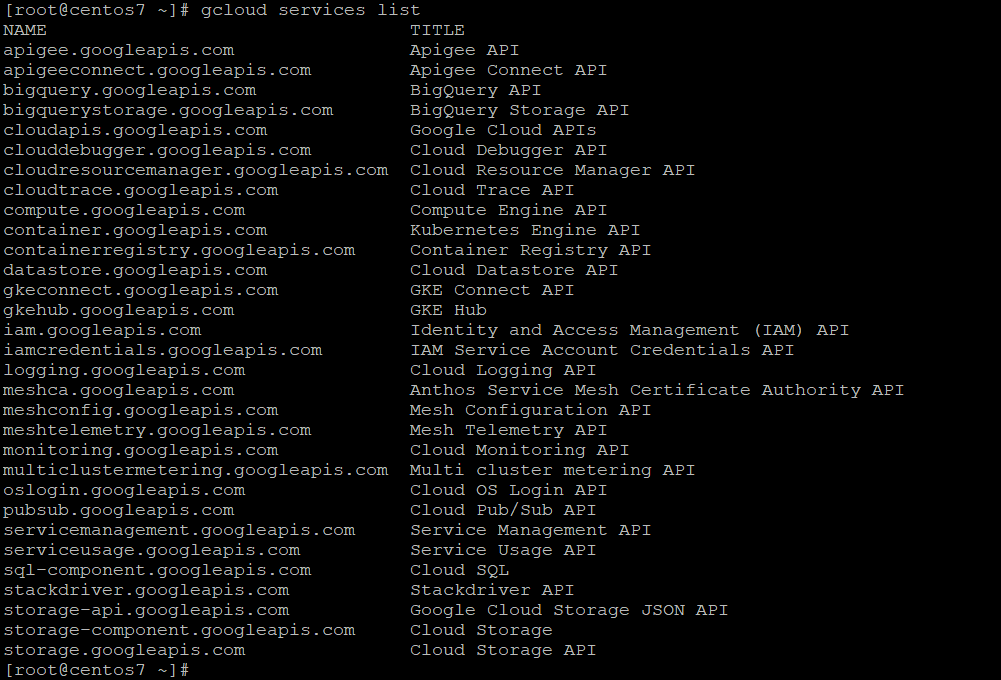
<https://cloud.google.com/service-mesh/v1.7/docs/scripted-install/gke-asm-onboard-1-7#new_installation>

### **3.2** **Installing Anthos Service Mesh on GKE**

This guide explains how to install, migrate, or upgrade to Anthos Service Mesh version 1.7.8 for a mesh containing one or more GKE clusters that are in the same project. You use a Google-provided script, which configures your project and cluster, and then installs Anthos Service Mesh.

### **3.3 Required Google APIs-**

container.googleapis.com  
compute.googleapis.com  
monitoring.googleapis.com  
logging.googleapis.com  
cloudtrace.googleapis.com  
meshca.googleapis.com  
meshtelemetry.googleapis.com  
meshconfig.googleapis.com  
iamcredentials.googleapis.com  
gkeconnect.googleapis.com  
gkehub.googleapis.com  
cloudresourcemanager.googleapis.com  
stackdriver.googleapis.com



### **3.4 Requirement**

* Your GKE cluster must meet the following requirements:
* A machine type that has at least four vCPUs, such as e2-standard-4. If the machine type for your cluster doesn't have at least four vCPUs, change the machine type as described in [Migrating workloads to different machine types](https://cloud.google.com/kubernetes-engine/docs/tutorials/migrating-node-pool).
* The minimum number of nodes depends on your machine type. Anthos Service Mesh requires at least eight vCPUs. If the machine type has four vCPUs, your cluster must have at least two nodes. If the machine type has eight vCPUs, the cluster only needs one node. If you need to add nodes, see [Resizing a cluster](https://cloud.google.com/kubernetes-engine/docs/how-to/resizing-a-cluster).
* The script enables [Workload Identity](https://cloud.google.com/kubernetes-engine/docs/how-to/workload-identity) on your cluster. Workload Identity is the recommended method of calling Google APIs. Enabling Workload Identity changes the way calls from your workloads to Google APIs are secured, as described in [Workload Identity limitations](https://cloud.google.com/kubernetes-engine/docs/how-to/workload-identity#limitations).
* Optional but recommended, enroll the cluster in a [release channel](https://cloud.google.com/kubernetes-engine/docs/concepts/release-channels). We recommend that you enroll in the Regular release channel because other channels might be based on a GKE version that isn't supported with Anthos Service Mesh 1.7.8. For more information, see [Supported environments](https://cloud.google.com/service-mesh/v1.7/docs/supported-features#supported_environments). Follow the instructions in [Enrolling an existing cluster in a release channel](https://cloud.google.com/kubernetes-engine/docs/concepts/release-channels#enrolling_an_existing_cluster_in_a_release_channel) if you have a static GKE version.
* To be included in the service mesh, service ports must be named, and the name must include the port's protocol in the following syntax: name: **protocol**[-**suffix**] where the square brackets indicate an optional suffix that must start with a dash. For more information, see [Naming service ports](https://cloud.google.com/service-mesh/v1.7/docs/naming-service-ports).
* If you are installing Anthos Service Mesh on a private cluster, you must open port 15017 in the firewall to get the webhook used with [automatic sidecar injection](https://cloud.google.com/service-mesh/v1.7/docs/proxy-injection) to work properly. For more information, see [Opening a port on a private cluster](https://cloud.google.com/service-mesh/v1.7/docs/private-cluster-open-port).
* If you have created a [service perimeter](https://cloud.google.com/vpc-service-controls/docs/service-perimeters) in your organization, you might need to add the Mesh CA service to the perimeter. See [Adding Mesh CA to a service perimeter](https://cloud.google.com/service-mesh/v1.7/docs/security/set-service-perimeter) for more information.
* For migrations, istiod must be installed in the istio-system namespace, which is typically the case.

### **3.5 Restrictions**

* A Google Cloud project can only have one mesh associated with it.

### **3.6 Choosing a certificate authority**

For both new installations and migrations, you can use Anthos Service Mesh certificate authority (Mesh CA) or [Citadel](https://istio.io/v1.7/docs/concepts/security/#high-level-architecture) (now incorporated in istiod) as the certificate authority (CA) for issuing [mutual TLS (mTLS)](https://cloud.google.com/service-mesh/v1.7/docs/security-overview#mutual_tls) certificates.

We generally recommend that you use Mesh CA for the following reasons:

* Mesh CA is a highly reliable and scalable service that is optimized for dynamically scaled workloads on Google Cloud.
* With Mesh CA, Google manages the security and availability of the CA backend.
* Mesh CA lets you rely on a single root of trust across clusters.

For new installations of Anthos Service Mesh, by default, the script enables Mesh CA.

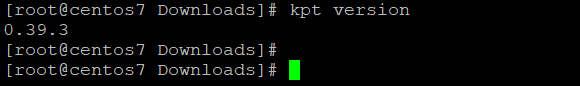
### **3.7 Installing required tools**

You can run the script on [Cloud Shell](https://cloud.google.com/shell/docs/launching-cloud-shell) or on your local machine running Linux. Cloud Shell pre-installs all the required tools. Note that macOS isn't supported because it comes with an old version of bash.

To run the script locally:

1. **Make sure you have the following tools installed:**
   * 1. The [Cloud SDK](https://cloud.google.com/sdk/docs/quickstarts) (the gcloud command-line tool)
     2. The standard command-line tools: awk, curl, grep, sed, sha256sum, and tr
     3. [git](https://git-scm.com/downloads)
     4. [kpt](https://github.com/GoogleContainerTools/kpt) [ version must be older one <= 0.39 ]

wget <https://storage.googleapis.com/kpt-dev/latest/linux_amd64/kpt>



* + 1. [kubectl](https://kubernetes.io/docs/tasks/tools/install-kubectl/)
    2. [jq](https://stedolan.github.io/jq/)

1. **Authenticate with the Cloud SDK:**

gcloud auth login

1. **Update the components:**

gcloud components update

1. **Make sure that git is in your path so that kpt can find it.**

### **3.8 Running the script**

This section describes how to download the script, set the required and optional parameters, and run the script. For a detailed explanation of what the script does, see [Understanding the script](https://cloud.google.com/service-mesh/v1.7/docs/scripted-install/gke-asm-onboard-1-7#understanding_the_script).

1. Download the version of the script that installs Anthos Service Mesh 1.7.8 to the current working directory:

curl https://storage.googleapis.com/csm-artifacts/asm/install\_asm\_1.7 > install\_asm

Although you download the script from a secure Cloud Source Repositories location, the script is also available on GitHub in the [anthos-service-mesh-packages](https://github.com/GoogleCloudPlatform/anthos-service-mesh-packages/tree/release-1.7-asm/scripts/asm-installer) repository so that you can see what it does before you download it. The version of the install\_asm script on the release-1.7-asm branch installs Anthos Service Mesh 1.7.8.

<https://github.com/GoogleCloudPlatform/anthos-service-mesh-packages/tree/release-1.7-asm/scripts/asm-installer>

1. Download the SHA-256 of the file to the current working directory:

curl https://storage.googleapis.com/csm-artifacts/asm/install\_asm\_1.7.sha256 > install\_asm.sha256

1. With both files in the same directory, verify the download:

sha256sum -c --ignore-missing install\_asm.sha256

1. Make the script executable:

chmod +x install\_asm

1. Set the options and specify the flags to run the script. You always include the following options: project\_id, cluster\_name, cluster\_location, and mode. Depending on the mode, you might need to include the ca option.
   * The **project\_id, cluster\_name, and cluster\_location** options identify the cluster on which to install Anthos Service Mesh.
   * The mode is either **install, migrate, or upgrade.**
   * The ca specifies the Certificate Authority to either **mesh\_ca or citadel**.
2. To complete setting up Anthos Service Mesh, you need to enable automatic sidecar injection and [deploy or redeploy workloads](https://cloud.google.com/service-mesh/v1.7/docs/scripted-install/gke-asm-onboard-1-7#deploying_and_redeploying_workloads).

**Only validate**

./install\_asm \

--project\_id **PROJECT\_ID** \

--cluster\_name **CLUSTER\_NAME** \

--cluster\_location **CLUSTER\_LOCATION** \

--mode install \

--output\_dir **DIR\_PATH** \

--only\_validate

./install\_asm \

--project\_id **psp-apigee-hybrid** \

--cluster\_name psp-gke-apigee \

--cluster\_location australia-southeast1-a \

--mode install \

--output\_dir **asm** \

--only\_validate

[root@centos7 Downloads]# ./install\_asm --project\_id psp-apigee-hybrid --cluster\_name psp-gke-apigee --cluster\_location australia-southeast1-a --mode install --output\_dir asm --only\_validate --verbose

install\_asm: Setting up necessary files...

install\_asm: Generating a new kubeconfig...

install\_asm: Checking installation tool dependencies...

install\_asm: Checking for project psp-apigee-hybrid...

install\_asm: Confirming cluster information for psp-apigee-hybrid/australia-southeast1-a/psp-gke-apigee...

install\_asm: Confirming node pool requirements for psp-apigee-hybrid/australia-southeast1-a/psp-gke-apigee...

install\_asm: [WARNING]: ASM requires you to have at least 8 vCPUs in node pools whose

machine type is at least 4 vCPUs.

australia-southeast1-a/psp-gke-apigee does not meet this requirement. ASM

may not function as expected.

install\_asm: Fetching/writing GCP credentials to kubeconfig file...

install\_asm: [WARNING]: Running: 'gcloud container clusters get-credentials psp-gke-apigee --project=psp-apigee-hybrid --zone=australia-southeast1-a'

install\_asm: [WARNING]: -------------

Fetching cluster endpoint and auth data.

kubeconfig entry generated for psp-gke-apigee.

install\_asm: Verifying connectivity (20s)...

install\_asm: kubeconfig set to psp-apigee-hybrid/australia-southeast1-a/psp-gke-apigee...

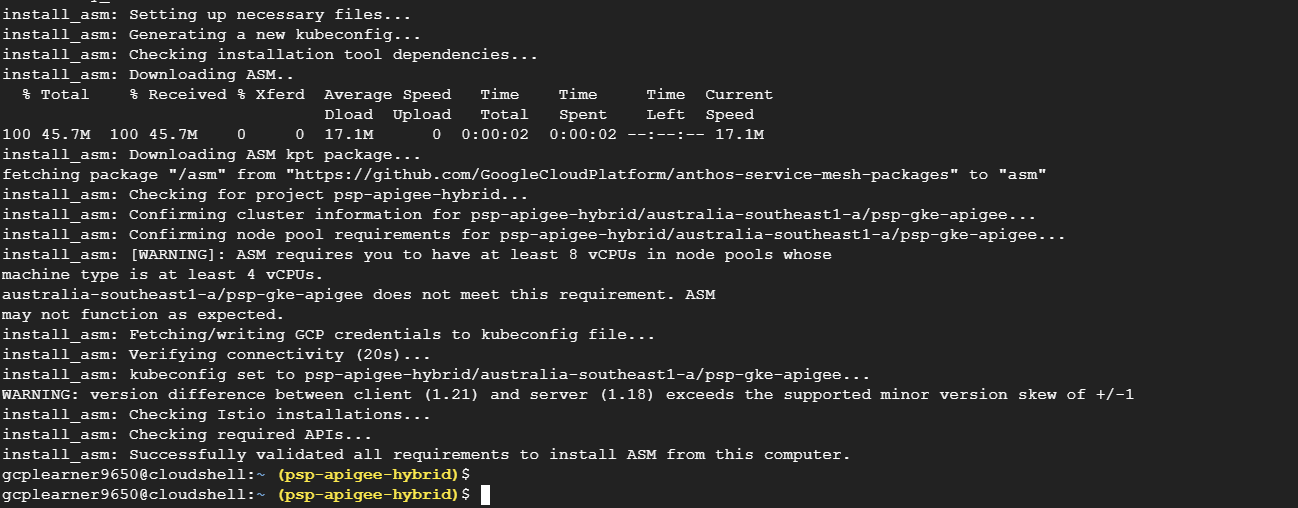
WARNING: version difference between client (1.21) and server (1.18) exceeds the supported minor version skew of +/-1

install\_asm: Checking Istio installations...

install\_asm: Checking required APIs...

install\_asm: Successfully validated all requirements to install ASM from this computer.

[root@centos7 Downloads]#



**Installation**

The following command runs the script for a new installation, enables Mesh

CA (the default CA for new installs, so you don't need the ca option in this

case), and allows the script to enable the required Google APIs.

./install\_asm \

--project\_id **PROJECT\_ID** \

--cluster\_name **CLUSTER\_NAME** \

--cluster\_location **CLUSTER\_LOCATION** \

--mode install \

--enable\_apis

./install\_asm \

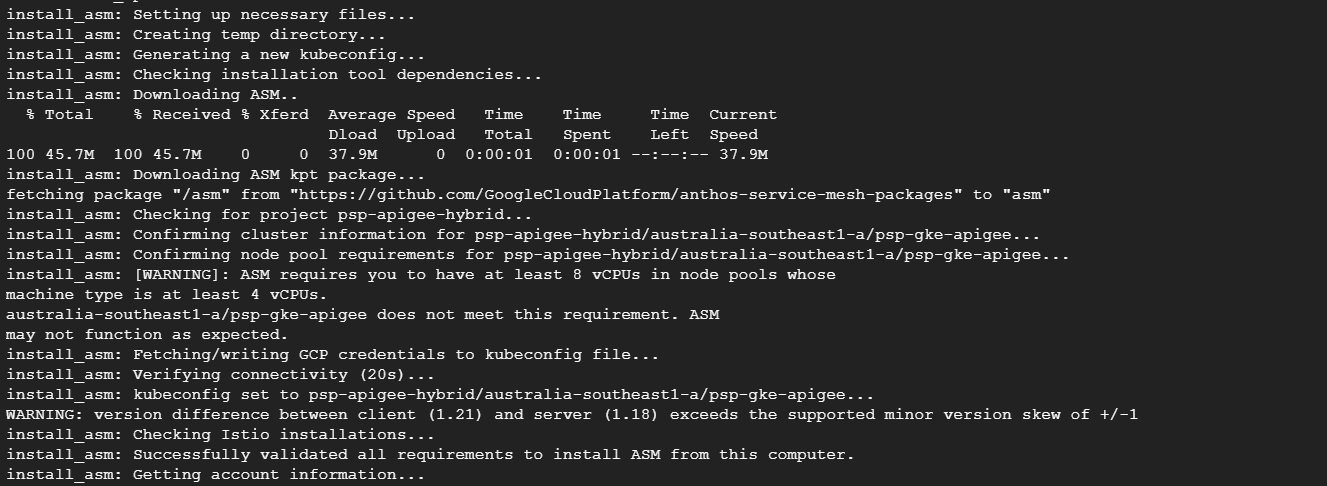
--project\_id **psp-apigee-hybrid** \

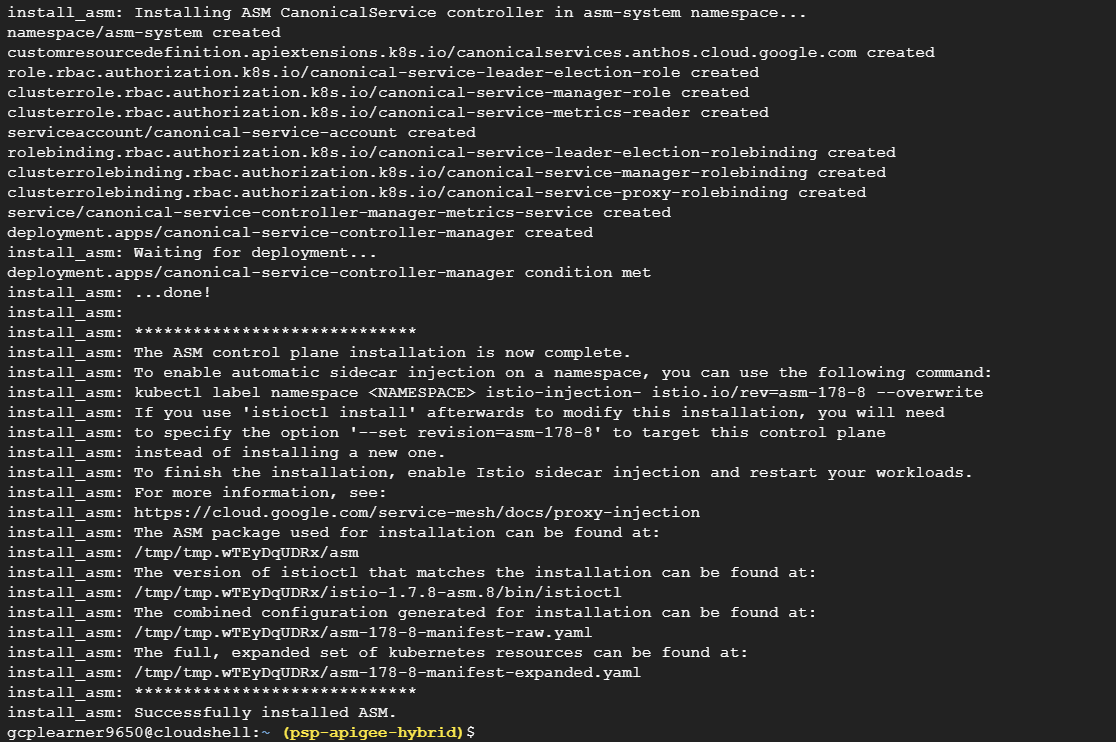
--cluster\_name psp-gke-apigee \

--cluster\_location australia-southeast1-a \

--mode install \

--enable\_apis





* To be included in the service mesh, service ports must be named, and the name must include the port's protocol in the following syntax: name: **protocol**[-**suffix**] where the square brackets indicate an optional suffix that must start with a dash. For more information, see [Naming service ports](https://cloud.google.com/service-mesh/v1.7/docs/naming-service-ports).
* For upgrades from previous versions of hybrid, install ASM version 1.6.x.

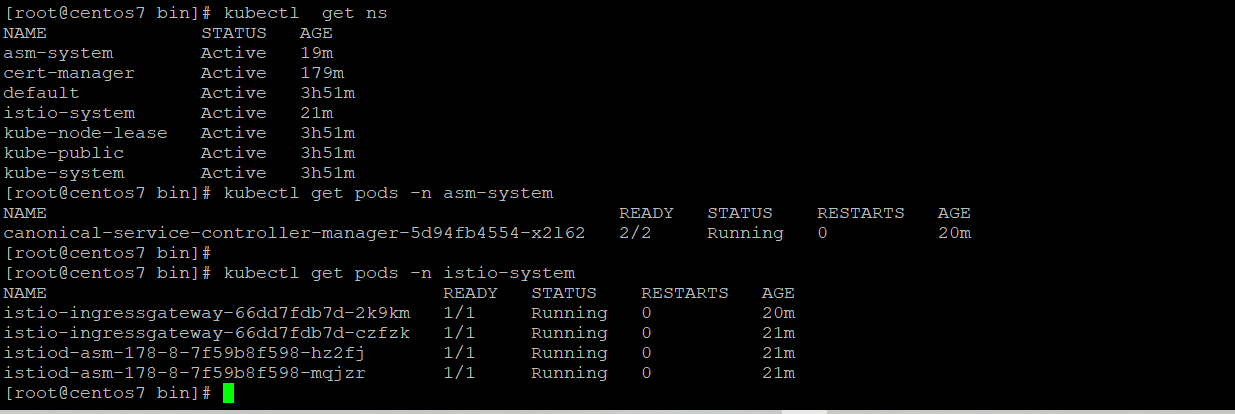
<https://cloud.google.com/service-mesh/v1.7/docs/scripted-install/gke-asm-onboard-1-7#new_installation>

This guide explains how to install, migrate, or upgrade to Anthos Service Mesh version 1.7.8 for a mesh containing one or more GKE clusters that are in the same project. You use a Google-provided script, which configures your project and cluster, and then installs Anthos Service Mesh.

You can use this guide and the script for the following use cases:

* New installations of Anthos Service Mesh.

**ASM Validation-**



### **3.9 Deploying and redeploying workloads**

The installation isn't complete until you enable automatic sidecar proxy injection (auto-injection).

* For new installations, you need to enable auto-injection and restart the Pods for any workloads that were running on your cluster before you installed Anthos Service Mesh.
* Before you deploy new workloads, make sure to enable auto-injection so that Anthos Service Mesh can monitor and secure traffic.

To enable auto-injection, you get the revision label that the script applied to istiod and label your namespaces with the revision label. The following sections provide details.

**Get the revision label**

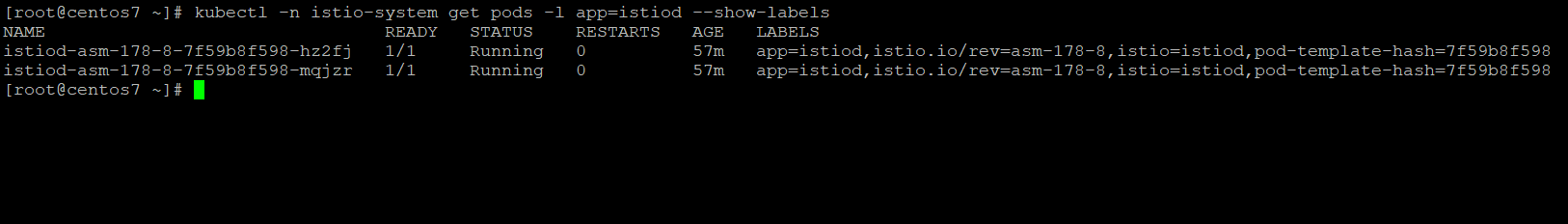
The script adds a revision label in the format istio.io/rev=asm-178-8 to istiod. To enable auto-injection, you add a matching revision label to your namespace(s). The revision label is used by the sidecar injector webhook to associate injected sidecars with a particular istiod revision. After adding the label, any existing Pods in the namespace must be restarted for sidecars to be injected.

**1.Set the current context for kubectl:**

$ gcloud container clusters get-credentials psp-gke-apigee --zone australia-southeast1-a --project psp-apigee-hybrid

1. **Display the labels on istiod to get the revision label set by the script:**

$ kubectl -n istio-system get pods -l app=istiod --show-labels



[root@centos7 ~]# kubectl -n istio-system get pods -l app=istiod --show-labels

NAME READY STATUS RESTARTS AGE LABELS

istiod-asm-178-8-7f59b8f598-hz2fj 1/1 Running 0 66m app=istiod,istio.io/rev=asm-178-8,istio=istiod,pod-template-hash=7f59b8f598

istiod-asm-178-8-7f59b8f598-mqjzr 1/1 Running 0 66m app=istiod,istio.io/rev=asm-178-8,istio=istiod,pod-template-hash=7f59b8f598

[root@centos7 ~]#

### **3.10 Enabling auto-injection**

* + 1. [Get the value in the revision label](https://cloud.google.com/service-mesh/v1.7/docs/scripted-install/gke-asm-onboard-1-7#get_the_revision_label) for istiod.
    2. Add the revision label to a namespace and remove the istio-injection label. In the following command, change **REVISION** to the value that matches the revision on istiod.

$ kubectl label namespace **NAMESPACE** istio.io/rev=**REVISION** istio-injection- --overwrite

* + 1. Restart the Pods to trigger re-injection.

$ kubectl rollout restart deployment -n **NAMESPACE**

* + 1. Verify that your Pods are configured to point to the new version of istiod.

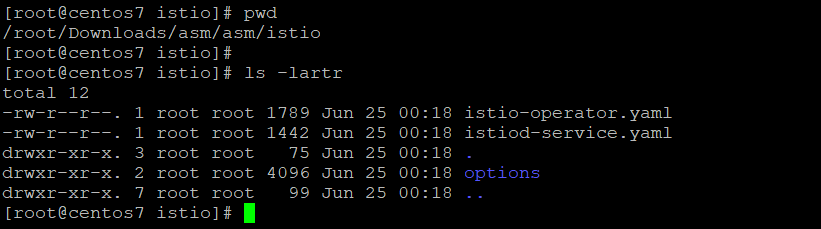
$ kubectl get pods -n **NAMESPACE** -l istio.io/rev=**REVISION**

* + 1. Test your application to verify that the workloads are working correctly.
    2. If you have workloads in other namespaces, repeat the steps to label the namespace and restart Pods.

### **3.11 Perform final hybrid configuration and install ASM**

Finally, add hybrid-specific configurations to the istio-operator.yaml file and install ASM.

* + 1. Ensure that you're in the ASM installation's root directory. For example: 1.7.3-asm.6.



* + 1. Open the istio-operator.yaml file in an editor.

In **ASM 1.7.x**: ./asm/istio/istio-operator.yaml

In **ASM 1.6.x**: ./asm/cluster/istio-operator.yaml

* + 1. Add (or update) the spec:components stanza in the istio-operator.yaml file below the meshConfig: section and immediately above values:, where **reserved\_static\_ip** is an IP address your runtime ingress gateway can use. If you do not have a reserved static IP address, for this Quick Start, you can leave the LoadBalancerIP property out.

<https://cloud.google.com/apigee/docs/hybrid/v1.4/2-3-install-asm#text-to-copy>

components:

ingressGateways:

- name: istio-ingressgateway

enabled: true

k8s:

service:

type: LoadBalancer

loadBalancerIP: static\_ip **# If you do not have a reserved static IP, leave this out.**

ports:

- name: status-port

port: 15021 # for ASM 1.7.x and above, else 15020

targetPort: 15021 # for ASM 1.7.x and above, else 15020

- name: http2

port: **80**

targetPort: **8080**

- name: https

port: **443**

targetPort: **8443**

**ASM Configuration yaml files.**

**Deployments**.



**Service and Ingress**

****

## **Step4-Install apigeectl**

This step explains how to download and install [apigeectl](https://cloud.google.com/apigee/docs/hybrid/v1.4/cli-reference), set up the installation directories, and create Google Cloud service accounts, and TLS credentials that are required for Apigee hybrid to operate.

### **4.1** **Download and install**apigeectl

apigeectl is the command-line interface (CLI) for installing and managing Apigee hybrid in a Kubernetes cluster.

The following steps describe how to get apigeectl:

* 1. Store the latest version number in a variable using the following command:

$export VERSION=$(curl -s \

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

* 1. Check that the variable was populated with a version number using the following command. If you want to use a different version, you can save that in an environment variable instead.

echo $VERSION

1.4.0

* 1. Download the release package for your operating system using the following command:

curl -LO \

<https://storage.googleapis.com/apigee-release/hybrid/apigee-hybrid->setup/**$VERSION**/apigeectl\_linux\_64.tar.gz

* 1. Create a directory on your system to serve as the base directory for the Apigee hybrid installation.
  2. Extract the downloaded gzip file contents into the base directory you just created using the following command:

$ tar xvzf filename.tar.gz -C path-to-base-directory

* 1. Change directory to the base directory using the cd command.
  2. The tar contents are, by default, expanded into a directory with the version and platform in its name. For example: ./apigeectl\_1.4.0-d591b23\_linux\_64. Rename that directory to apigeectl using the following command:

$ mv ***apigeectl\_1.4.0-d591b23\_linux\_64*** apigeectl

* 1. Change to the directory using the following command:

$ cd ./apigeectl

This directory is the apigeectl home directory. It is wherethe apigeectl executable command is located.

* 1. Create an environment variable to hold this home directory path using the following command:

$export APIGEECTL\_HOME=$PWD

* 1. Verify that the variable holds the correct path using the following command:

echo $APIGEECTL\_HOME

[root@centos7 apigeectl]# export APIGEECTL\_HOME=/root/Downloads/apigeectl

### **4.2 Set up the project directory structure**

The following directory structure is a suggested approach. It separates Apigee hybrid release software from configuration files that you must create. Through the use of the $APIGEECTL\_HOME variable and symbolic links that you create, you can easily switch to a new software version if you choose to. See also [Upgrading Apigee hybrid](https://cloud.google.com/apigee/docs/hybrid/v1.4/upgrade).

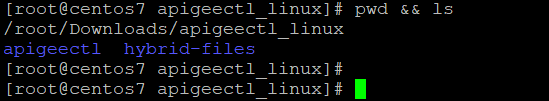
* + 1. Be sure you are in the base directory (the directory where the apigeectl directory is located) by using the following command:

$ cd $APIGEECTL\_HOME/..

* + 1. Create a new folder called hybrid-files using the following command. You can give the directory any name you wish, but in the docs, the name hybrid-files is used consistently. Later, you will store configuration files, service account keys, and TLS certificates in this folder. This folder lets you keep your config files separate from the apigeectl software installation.

$ mkdir hybrid-files

* + 1. The current directory structure now looks like the following



* + 1. Change directory into the hybrid-filesfolder using the following command:

$ cd hybrid-files

* + 1. Inside the hybrid-files directory, create the following subdirectories to organize files that you will create later:

$ mkdir overrides

$ mkdir service-accounts

$ mkdir certs

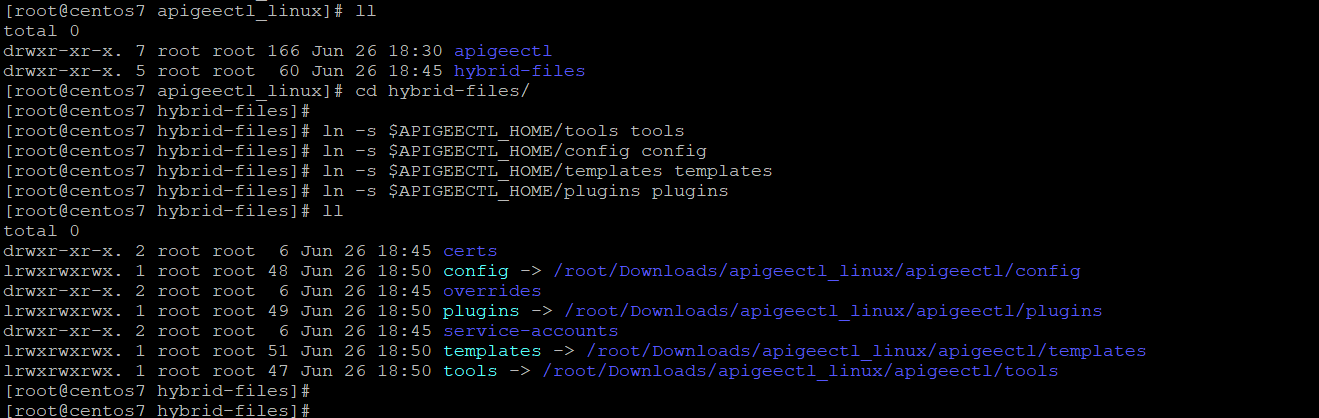
* + 1. Inside the hybrid-files directory, create the following symbolic links to $APIGEECTL\_HOME. These links allow you to run the apigeectl command from inside the hybrid-files directory as the following example shows:

ln -s $APIGEECTL\_HOME/tools tools

ln -s $APIGEECTL\_HOME/config config

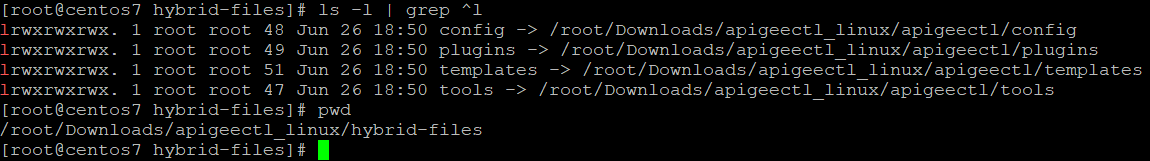
ln -s $APIGEECTL\_HOME/templates templates

ln -s $APIGEECTL\_HOME/plugins plugins



* + 1. To check that the symlinks were created correctly, execute the following command and make sure the link paths point to the correct locations:

$ ls -l | grep ^l



## **Step 5: Create service accounts and credentials**

This step explains how to create the Google Cloud service accounts and TLS credentials that are required for Apigee hybrid to operate.

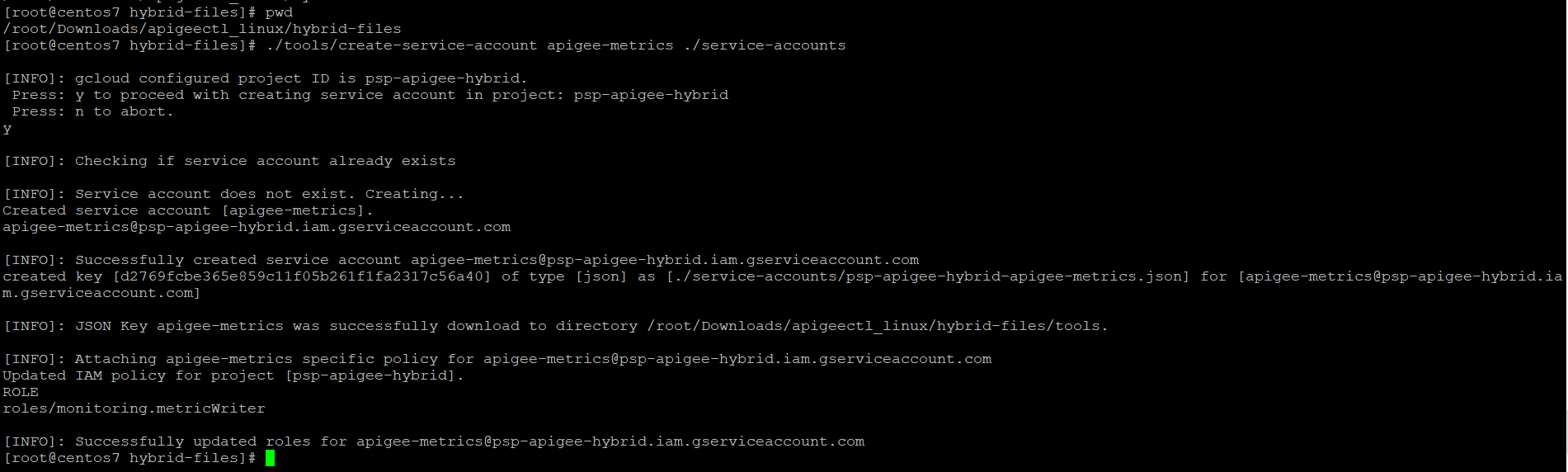
### **5.1 Create the service accounts**

* Apigee hybrid uses Google Cloud [service accounts](https://cloud.google.com/iam/docs/service-accounts) to allow hybrid components to communicate by making authorized API calls.
* In this step, you use an Apigee hybrid command-line tool to create a set of service accounts and download the service account private key files.
* Apigee provides a tool, create-service-account, that creates the service accounts, assigns the roles to the service accounts, and creates and downloads the key files for the service account in a single command.

1. Be sure that you are in the **base\_directory**/hybrid-files directory you configured in [Set up the project directory structure](https://cloud.google.com/apigee/docs/hybrid/v1.4/2-5-install-service-accounts#setup-directory).

2. Execute the following command from inside the **hybrid-files** directory. This command creates a service account for the apigee-metrics component and places the downloaded key in the ./service-accounts directory.

$ ./tools/create-service-account apigee-metrics ./service-accounts



1. Now, create the rest of the service accounts using the following commands. The create-service-account command is interactive and requires a response for each account.

./tools/create-service-account apigee-synchronizer ./service-accounts

./tools/create-service-account apigee-udca ./service-accounts

./tools/create-service-account apigee-mart ./service-accounts

./tools/create-service-account apigee-cassandra ./service-accounts

./tools/create-service-account apigee-logger ./service-accounts

./tools/create-service-account apigee-watcher ./service-accounts

./tools/create-service-account apigee-distributed-trace ./service-accounts

We can automate the above task using below command in a shell script

yes | ./tools/create-service-account apigee-synchronizer ./service-accounts

yes | ./tools/create-service-account apigee-udca ./service-accounts

yes | ./tools/create-service-account apigee-mart ./service-accounts

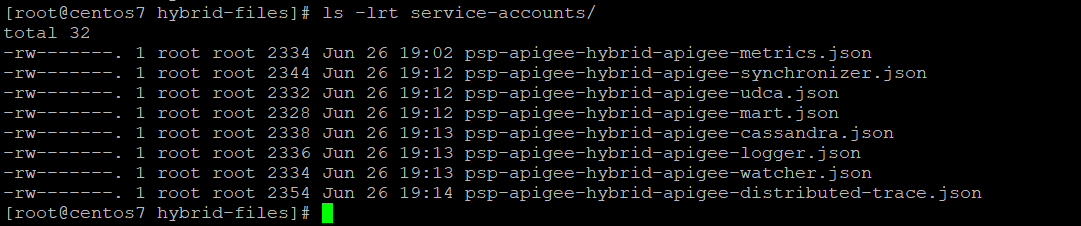
yes | ./tools/create-service-account apigee-cassandra ./service-accounts

yes | ./tools/create-service-account apigee-logger ./service-accounts

yes | ./tools/create-service-account apigee-watcher ./service-accounts

yes | ./tools/create-service-account apigee-distributed-trace ./service-accounts

1. Verify that the service account keys were created using the following command. You are responsible for storing these private keys securely. The key filenames are prefixed with the name of your Google Cloud project.

$ ls ./service-accounts 

### **5.12 Create TLS certificates**

* You are required to provide TLS certificates for the runtime ingress gateway in your Apigee hybrid configuration.
* For the purpose of this quickstart (a non-production trial installation), the runtime gateway can accept self-signed credentials. In the following steps, [openssl](https://www.openssl.org/) is used to generate the self-signed credentials.

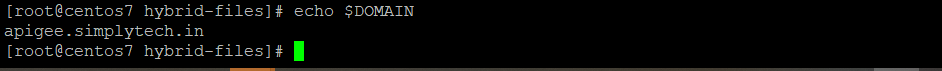
In this step, you will create the TLS credential files and add them to

the **base\_directory**/hybrid-files/certs directory. In [Step 6: Configure the cluster](https://cloud.google.com/apigee/docs/hybrid/v1.4/2-6-install-configure-cluster), you

will add the file paths to the cluster configuration file.

1. Be sure that you are in the **base\_directory**/hybrid-files directory you configured in [Set up the project directory structure](https://cloud.google.com/apigee/docs/hybrid/v1.4/2-5-install-service-accounts#setup-directory).
2. Make sure to save a domain name to the DOMAIN environment variable using the following command:

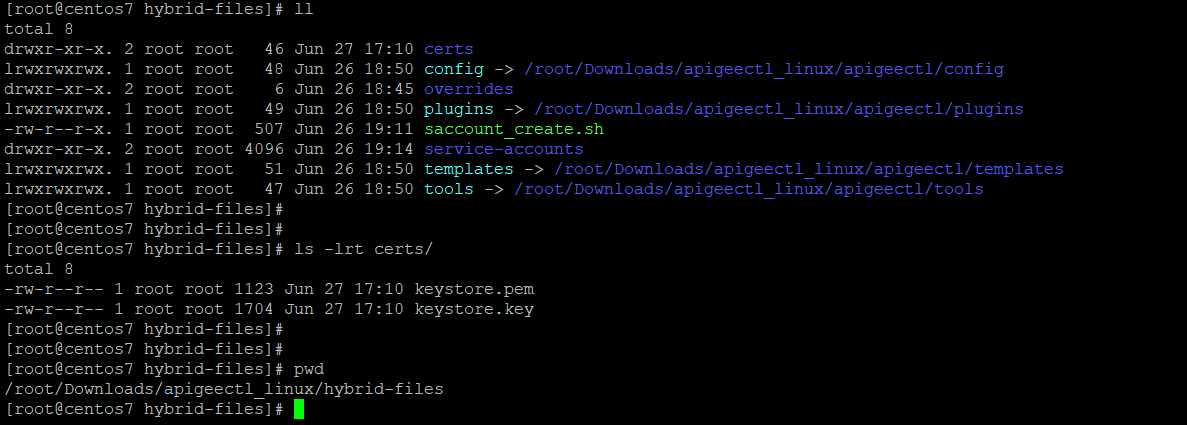
$ echo $DOMAIN



1. Execute the following command from inside the hybrid-files directory:

$ openssl req -nodes -new -x509 -keyout ./certs/keystore.key -out \

./certs/keystore.pem -subj '/CN='$DOMAIN'' -days 3650



Where **DOMAIN** is the same one you used for your environment in [Part 1, Step 5: Create an environment group](https://cloud.google.com/apigee/docs/hybrid/v1.4/precog-add-environment).

This command creates a self-signed certificate/key pair that you can use for the quickstart installation.

1. Check to make sure the files are in the ./certs directory using the following command

$ ls ./certs

keystore.pem

keystore.key

Where keystore.pem is the self-signed TLS certificate file and keystore.key is the key file.

You now have the service accounts and credentials needed to manage Apigee hybrid in your Kubernetes cluster. Next, you will create a file that is used by Kubernetes to deploy the hybrid runtime components to the cluster.

## **Step 6: Configure the hybrid runtime cluster**

### **6.1 Specify configuration overrides**

The Apigee hybrid installer uses defaults for many settings; however, there are a few settings that do not have defaults. You must provide values for these settings, as explained next.

<https://cloud.google.com/apigee/docs/hybrid/v1.4/2-6-install-configure-cluster#syntax>

1. Be sure you are in the **hybrid-base-directory**/hybrid-files/overrides/ directory.

$ cd **hybrid-base-directory**/hybrid-files/overrides

1. Create a new file named ***overrides*.yaml** in your favorite text editor. For example:

$vi **overrides**.yaml

The **overrides**.yaml provides the configuration for your unique Apigee hybrid

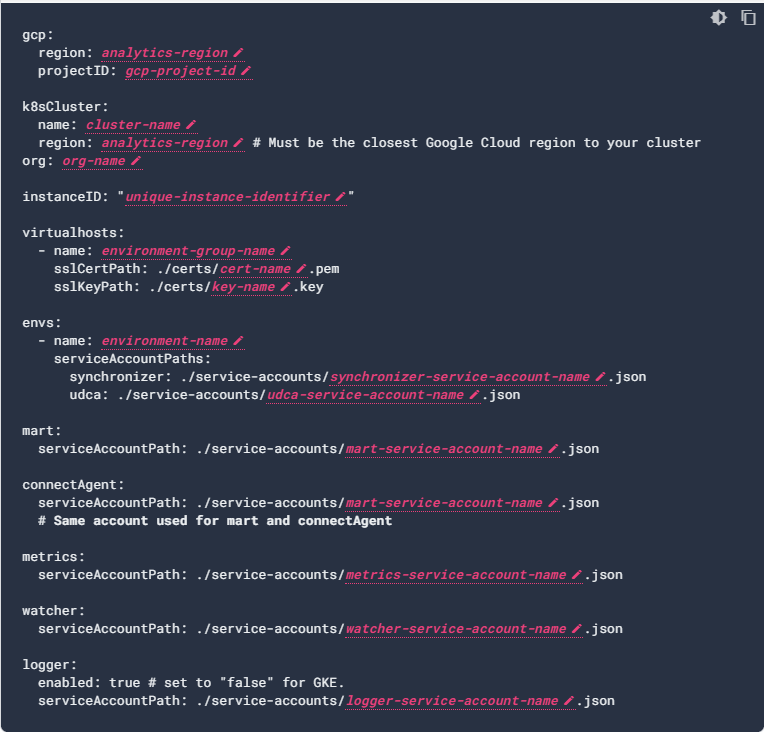
installation. The overrides file in this step provides a basic configuration for a small-

footprint hybrid runtime installation, suitable for your first installation.

1. In **overrides**.yaml, add the required property values, shown below. A detailed description of each property is also provided below:

The following example shows a completed overrides file with example property values

added:



1. When you are finished, save the file.

## **Step 7: Install hybrid runtime**

### **7.1 Enable synchronizer access**

**To enable synchronizer access:**

1. Create a Google Cloud service account and add the **Apigee Organization Admin** role to it. This service account will be used to authenticate an API call that you will make in a later step. An easy way to create the service account is through the GCP console. For instructions, see [Creating and managing service accounts](https://cloud.google.com/iam/docs/creating-managing-service-accounts#iam-service-accounts-create-gcloud) in the GCP documentation.

For example, the following gcloud commands will create the service account and assign the **Apigee Organization Admin** to it:

**Create a service account**

$ gcloud iam service-accounts create apigee-org-admin

--display-name="apigee-org-admin"

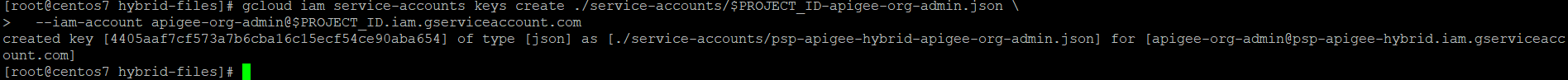
Where **apigee-org-admin** is the name of the service account you are creating. "apigee-org-admin" is recommended for this tutorial.

$ Assign the **Apigee Org Admin** role to the service account:

$ gcloud projects add-iam-policy-binding $PROJECT\_ID \

--member="serviceAccount:apigee-org-admin@$PROJECT\_ID.iam.gserviceaccount.com" \

--role="**roles/apigee.admin**"



**Where:**

**$PROJECT\_ID** is the name of your Google Cloud project that you created in [Step 2: Create a Google Cloud project](https://cloud.google.com/apigee/docs/hybrid/v1.4/precog-gcpproject).

**apigee-org-admin** is the name of the service account you just created.

**roles/apigee.admin** is the **Apigee Org Admin** role.

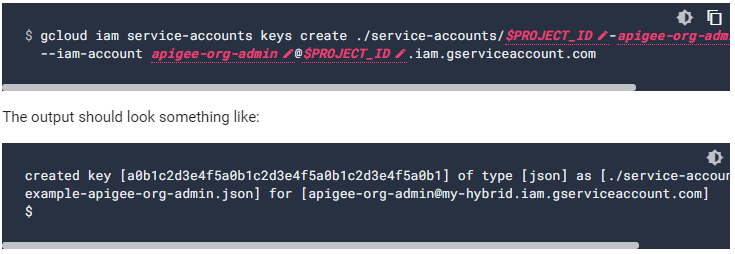
1. Download the service account key to your system. Use the following command to make download the key into your service-accounts/ directory. For more information see the instructions in [Creating service account keys](https://cloud.google.com/iam/docs/creating-managing-service-account-keys#iam-service-account-keys-create-gcloud) in the GCP documentation.
2. Make sure you are in the **/hybrid-base-directory**/hybrid-files/ directory.

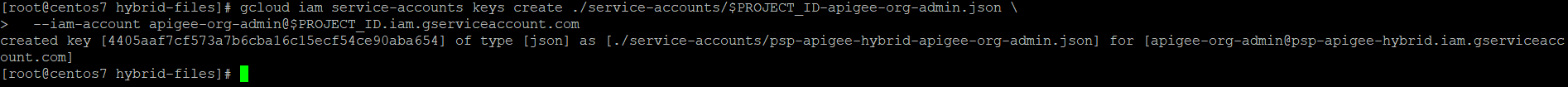
B. Download the key:

gcloud iam service-accounts keys create ./service-

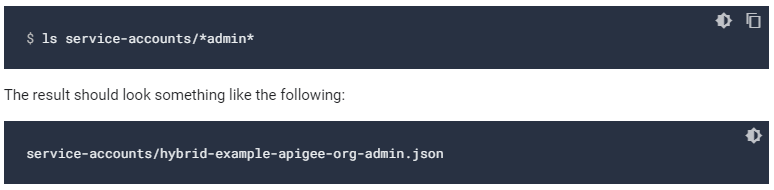
accounts/$PROJECT\_ID-apigee-org-admin.json \

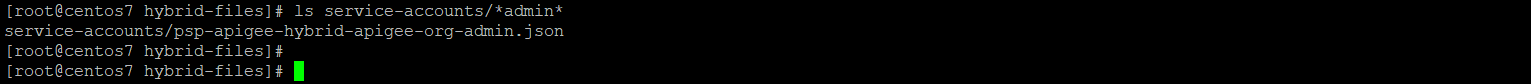
--iam-account apigee-org-admin@$PROJECT\_ID.iam.gserviceaccount.com





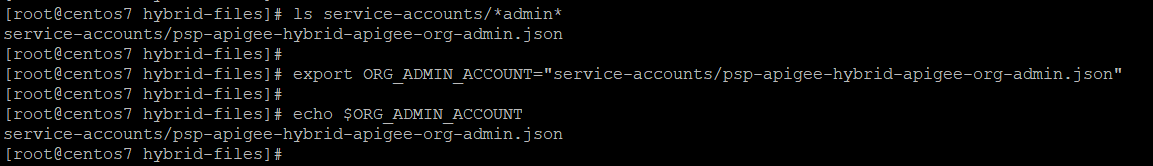
1. Verify the path to the Apigee Org Admin service account key with the following command:



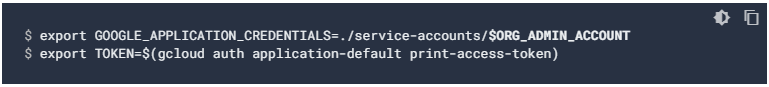


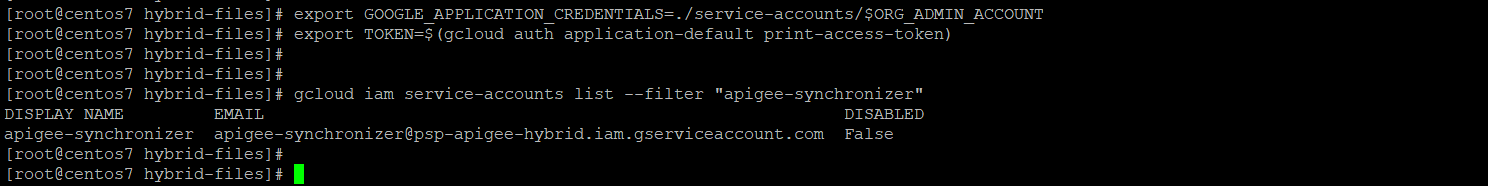
1. Create an **ORG\_ADMIN\_ACCOUNT** environment variable with the name of the key file. For example:



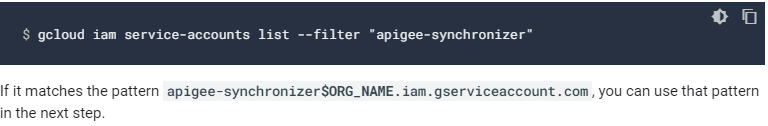


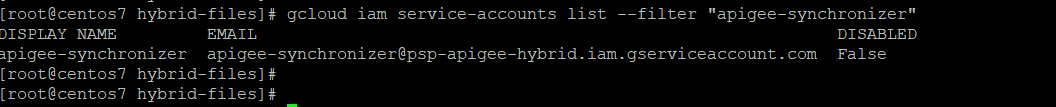
1. Execute the following commands to get a token:



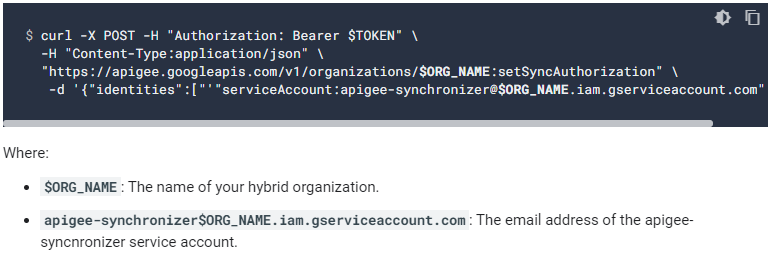


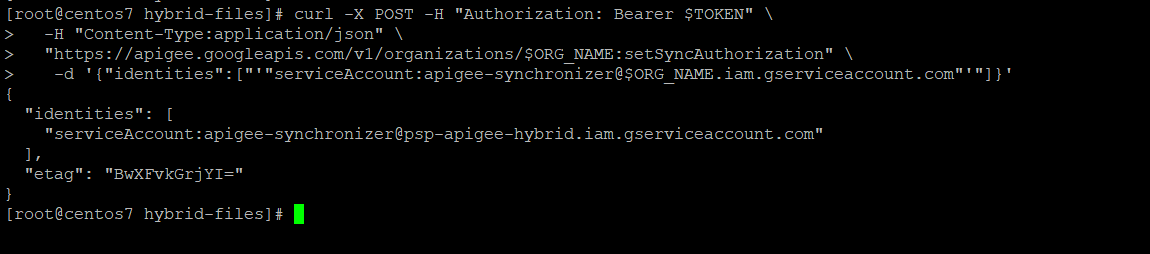
1. Get the email address for your apigee-synchronizer service account with the following command:



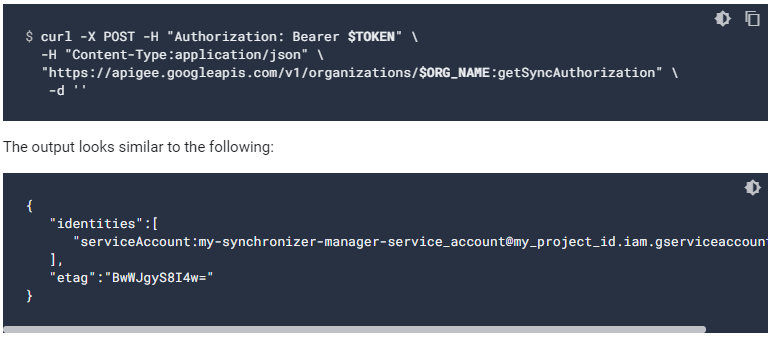


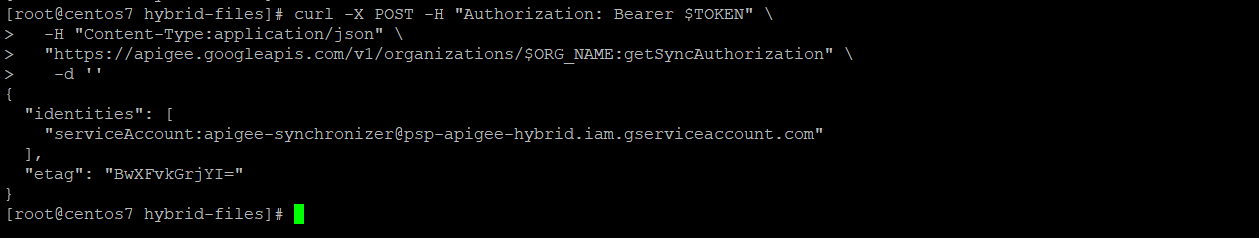
1. Call the [setSyncAuthorization](https://cloud.google.com/apigee/docs/reference/apis/apigee/rest/v1/organizations/setSyncAuthorization) API to enable the required permissions for Synchronizer using the following command:



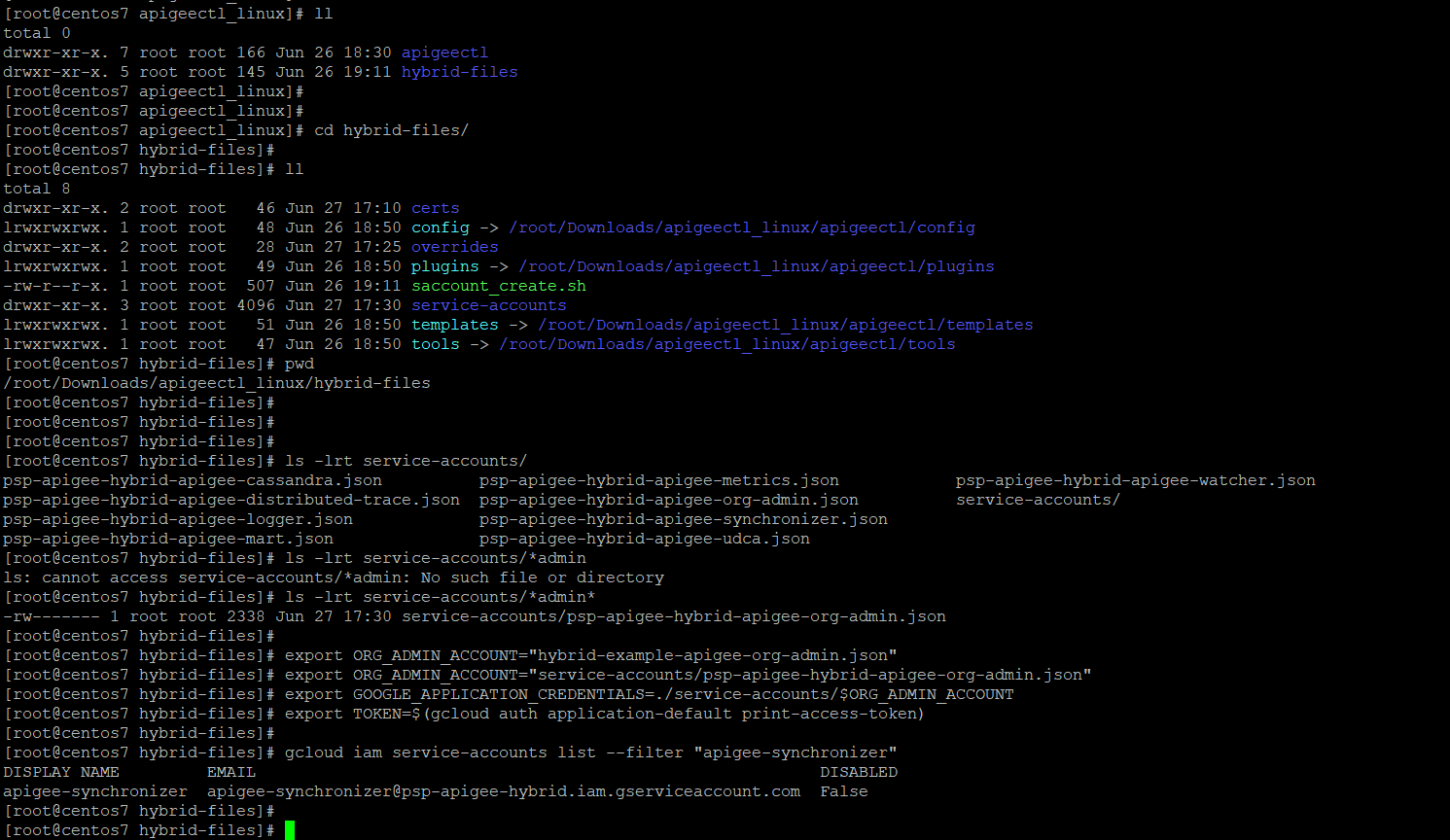


1. To verify that the service account was set, use the following command to call the API to get a list of service accounts:









### **7.2 Apply the configuration to the cluster**

Use the following steps to install Apigee hybrid into your cluster:

1. Be sure that you are in the ***hybrid-base-directory***/hybrid-files directory.
2. Verify that kubectl is set to the correct context using the following command. The current context should be set to the cluster to which you are deploying Apigee hybrid.

$ kubectl config get-contexts

1. For AWS on GKE, EKS, and GKE on prem platforms only, Verify that the KUBECONFIG variable is set using the following command.

$ echo $KUBECONFIG

1. Do a *dry run* initialization. Execute the init command with the --dry-run flag. Doing a dry run lets you check for any errors before any changes are made to the cluster.

In hybrid version 1.4.4, the syntax of the --dry-run flag depends on the version of kubectl you are running. Check the version of kubectl with the following command:

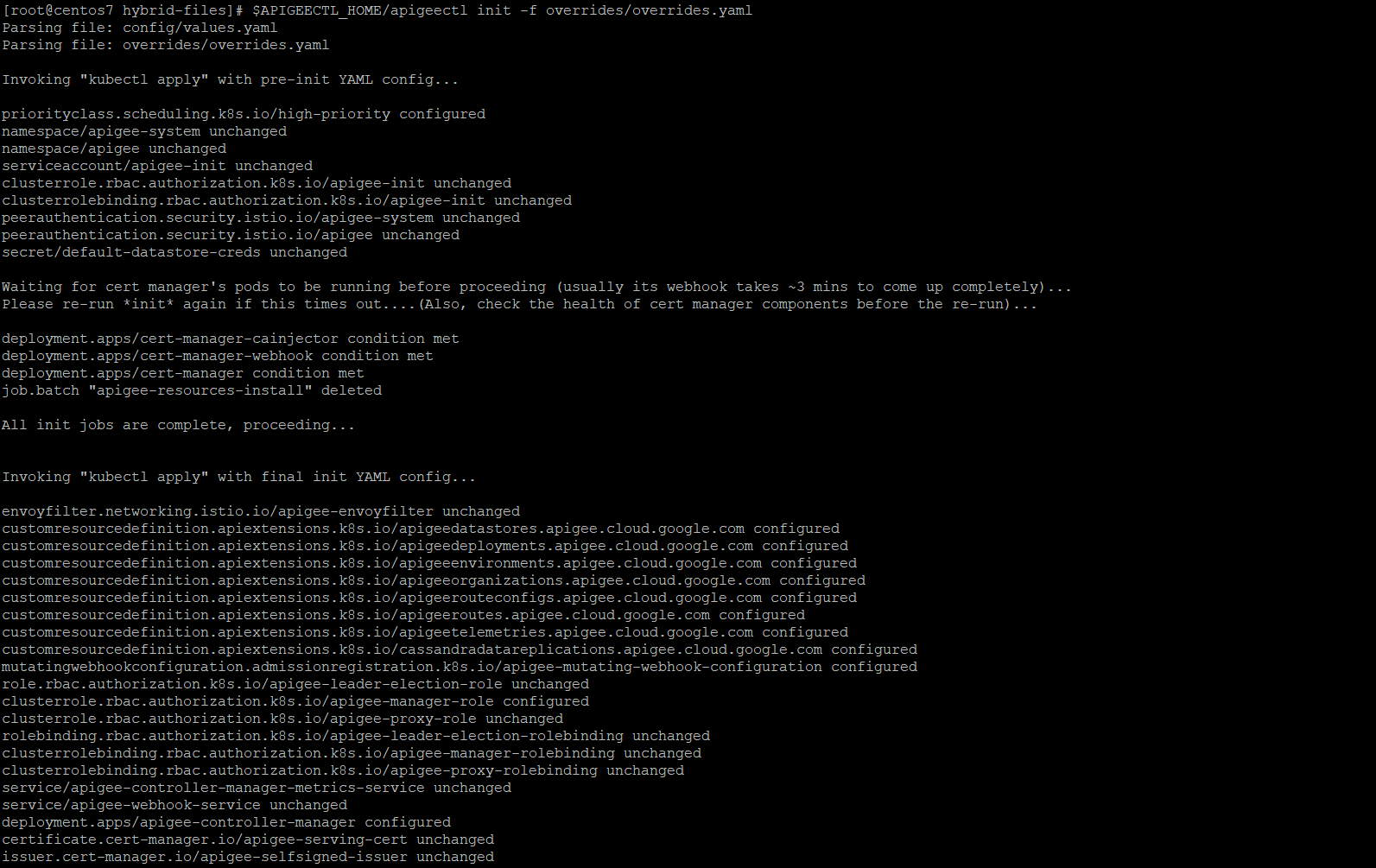
$ kubectl version

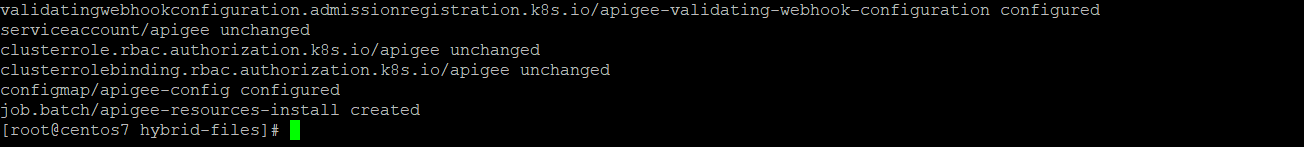
1. If there are no errors, execute the init command as follows

$ **$APIGEECTL\_HOME**/apigeectl init -f overrides/**overrides**.yaml

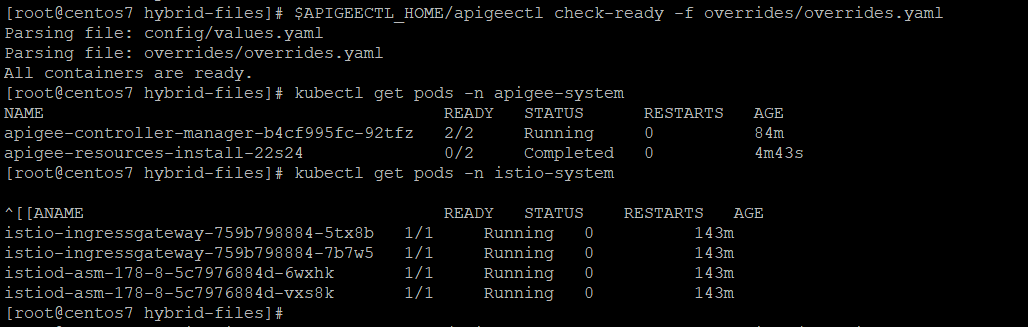
The init command installs the [Apigee deployment services](https://cloud.google.com/apigee/docs/hybrid/v1.4/apigee-deployment-services) Apigee Deployment

Controller and Apigee Admission Webhook.





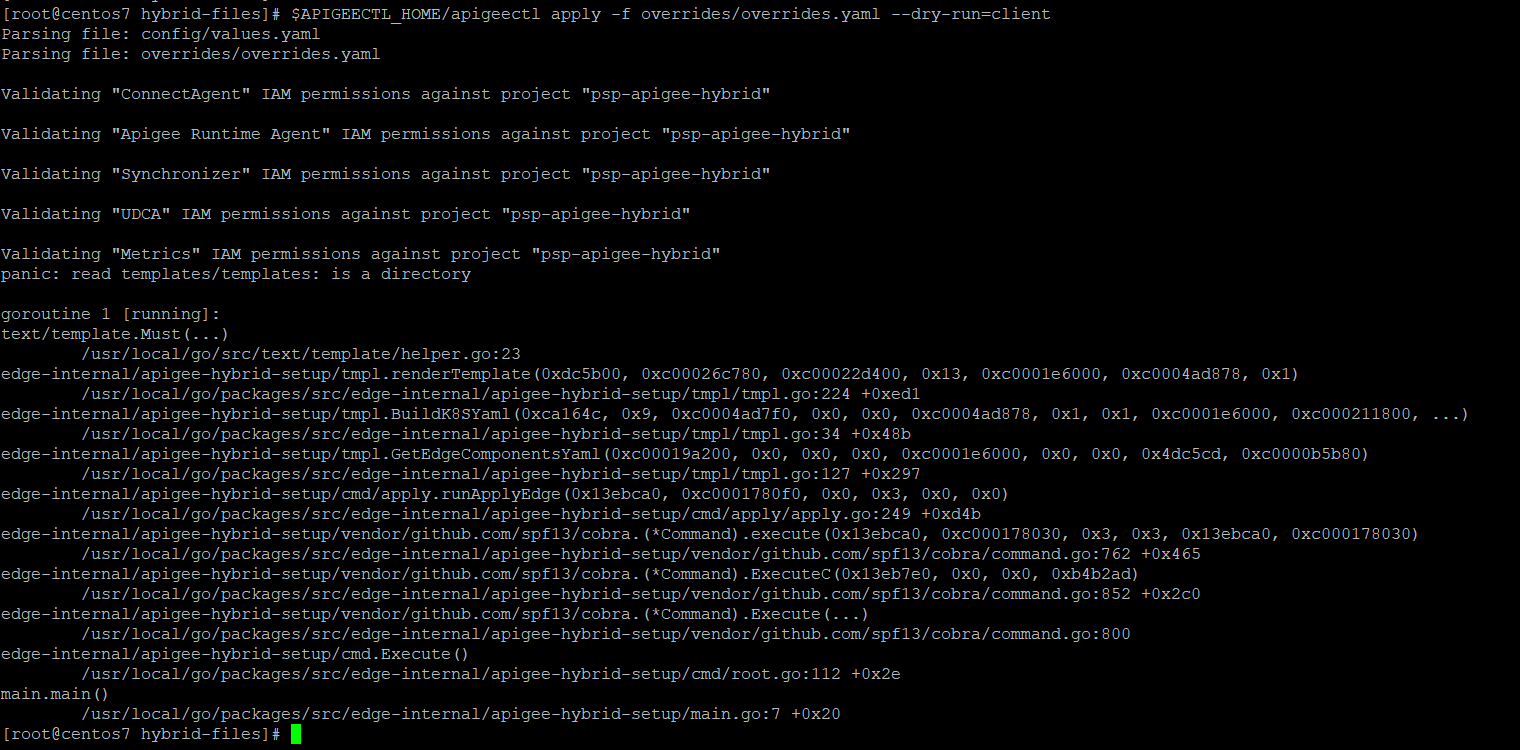
1. To check the status of the deployment, you can use the following commands:



1. Do a dry run install. Execute the apply command with the --dry-run flag.

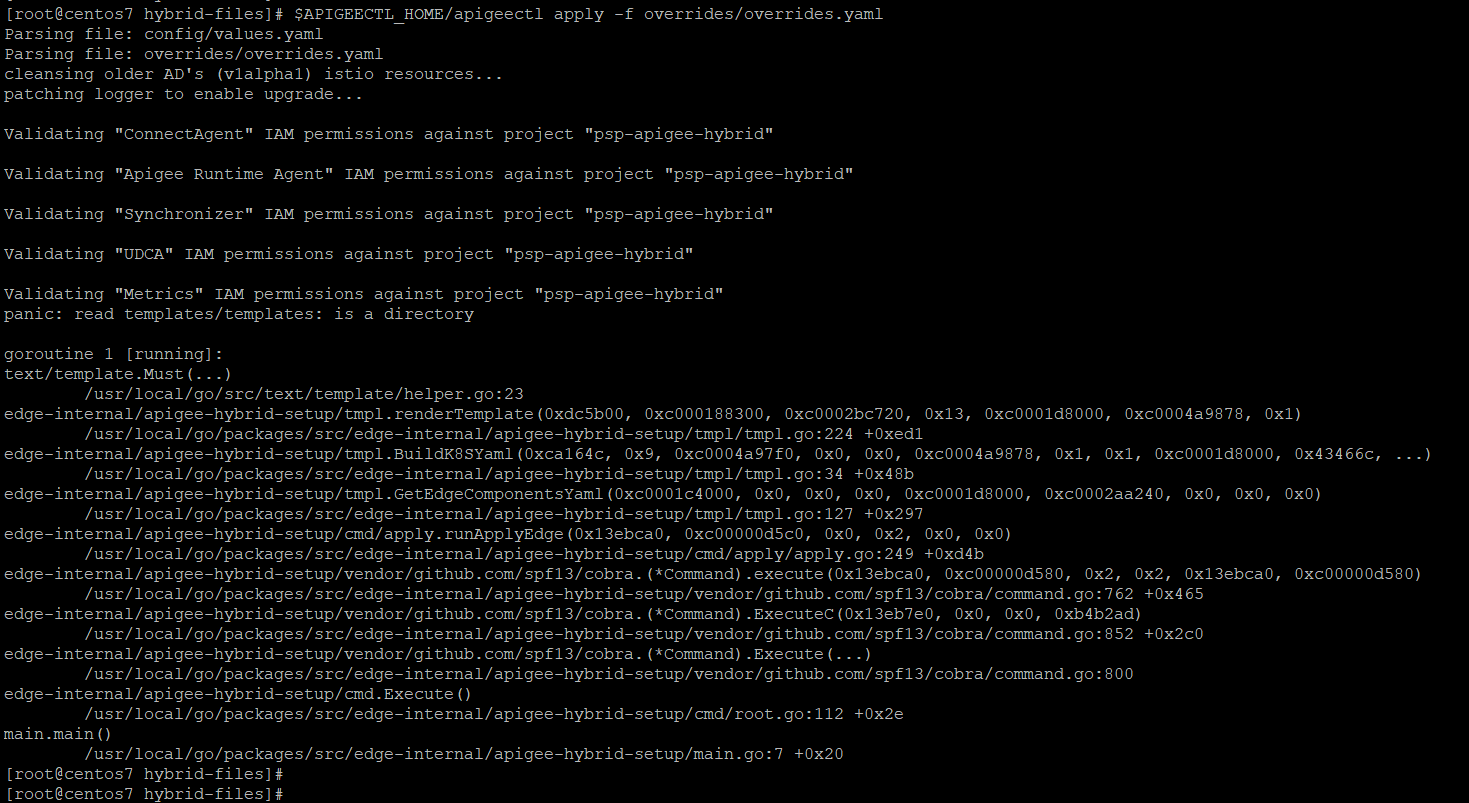
$ kubectl version 1.18 and newer:

$ **$APIGEECTL\_HOME**/apigeectl apply -f overrides/overrides.yaml --dry-run=client

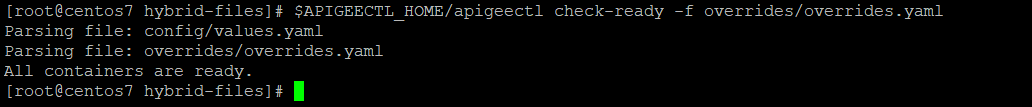


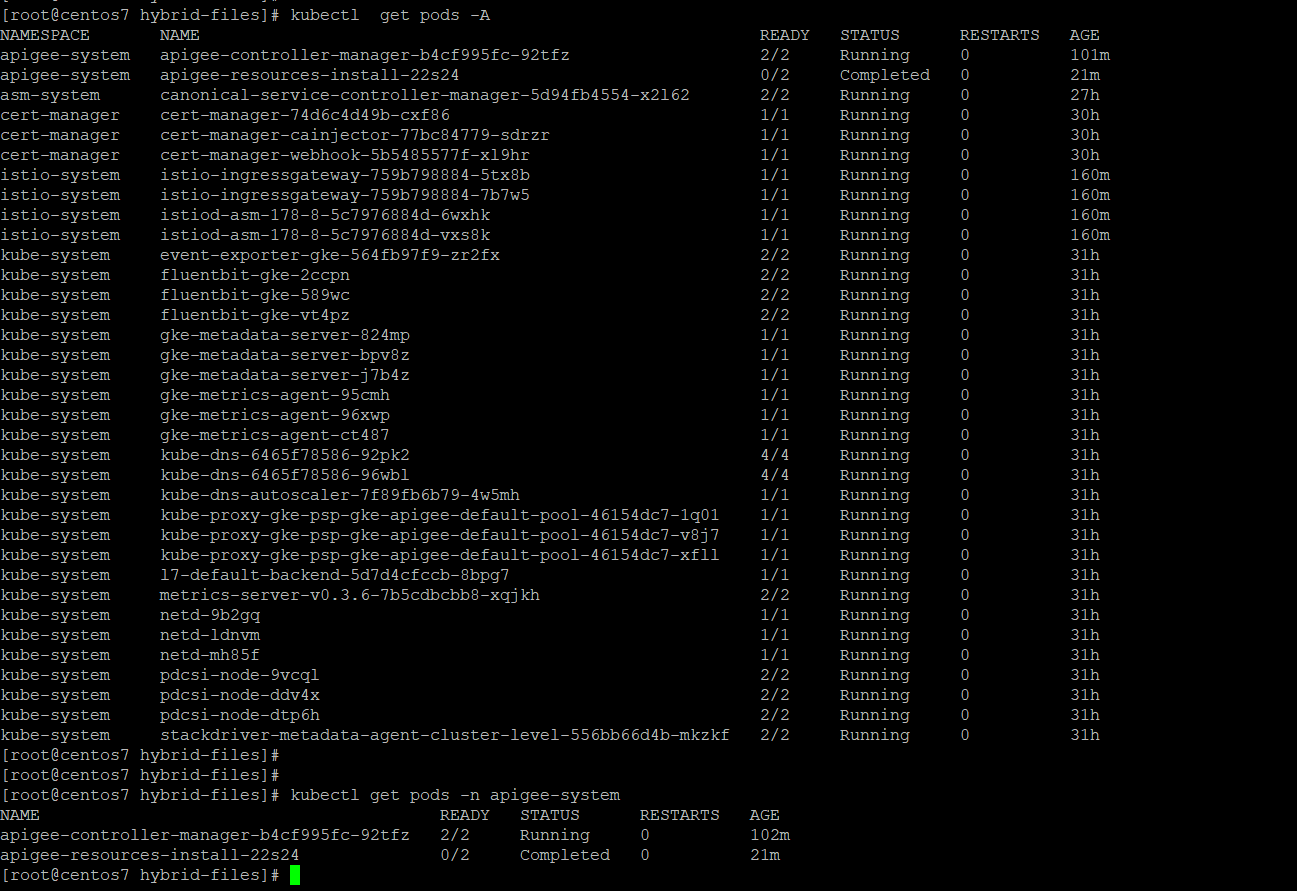
1. If there are no errors, you can apply the [Apigee-specific runtime components](https://cloud.google.com/apigee/docs/hybrid/v1.4/what-is-hybrid#about-the-runtime-plane) to the cluster with the following command:

$ **$APIGEECTL\_HOME**/apigeectl apply -f overrides/**overrides**.yaml



1. To check the status of the deployment, run the following command





# **Part3. After Install and Test**

## 3.1- Create and deploy a new API proxy

Now that you've configured Google Cloud and the hybrid UI, and installed and configured the runtime, you're ready to see how it all works together.

This section walks you through the following:

1. [**Create a new API proxy**](https://cloud.google.com/apigee/docs/hybrid/v1.4/test-new-proxy#create-proxy) in the Apigee UI using the API Proxy Wizard
2. [**Deploy your new proxy**](https://cloud.google.com/apigee/docs/hybrid/v1.4/test-new-proxy#deploy) to your cluster with the UI

### **3.1.1 Create a new API proxy using the hybrid UI**

This section describes how to create a new API proxy in the UI by using the API Proxy Wizard.

**To create a simple API proxy using the API Proxy Wizard:**

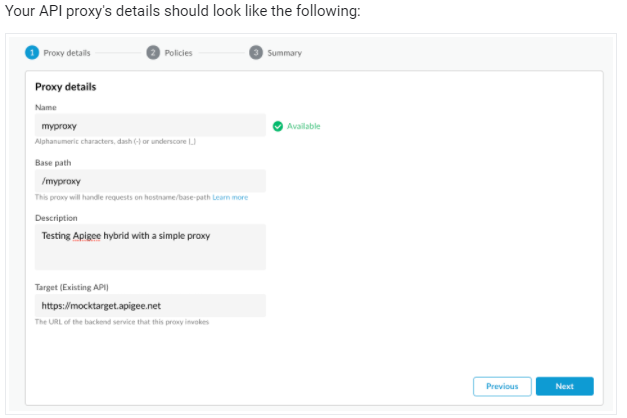
1. Open [Apigee UI](https://apigee.google.com/) in a browser.
2. Select **Develop > API Proxies** in the main view.
3. From the **Environment** drop-down list, select the environment in which you want to create a new API proxy. This section assumes the name of the environment is "test". You created at least one environment in [Step 5: Add an environment](https://cloud.google.com/apigee/docs/hybrid/v1.4/precog-add-environment).

The hybrid UI displays a list of API proxies for that environment. If you haven't created any proxies yet, the list is empty.

1. Click **+Proxy** in the upper right. The API Proxy Wizard starts.
2. Select **Reverse proxy** (most common), and click **Next**.

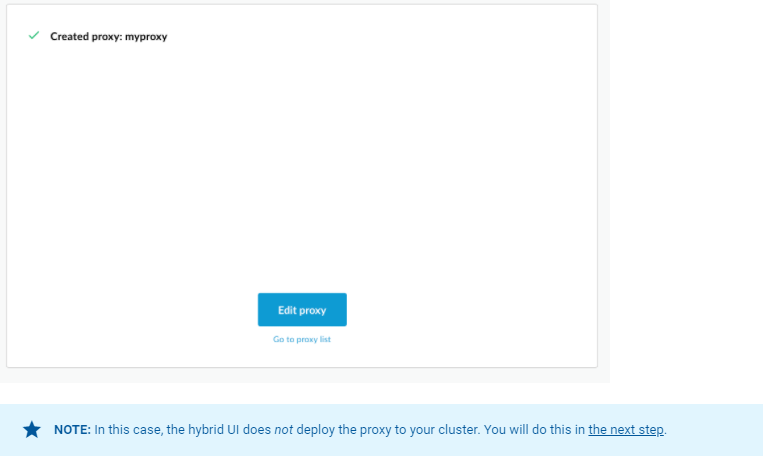
The **Proxy details** view is displayed.

1. Configure your proxy with the following settings:
   * **Proxy Name:** Enter "myproxy". The remaining steps in this section assume that this is your proxy's ID.
   * **Proxy Base Path:** Automatically set to "/myproxy". The **Proxy Base Path** is part of the URL used to make requests to your API. Edge uses the URL to match and route incoming requests to the proper API proxy.
   * (Optional)**Description:** Enter a description for your new API proxy, such as "Testing Apigee hybrid with a simple proxy".
   * **Target (Existing API):** Enter "https://mocktarget.apigee.net". This defines the target URL that Apigee invokes on a request to the API proxy. The mocktarget service is hosted at Apigee and returns simple data. It requires no API key or access token.



1. Click **Next**.
2. On the **Policies** screen, select **Pass through (no authorization)** as the security option.
3. Click **Next**.
4. On the **Summary** screen, click **Create**.

Hybrid generates the proxy (sometimes referred to as *the proxy bundle*):

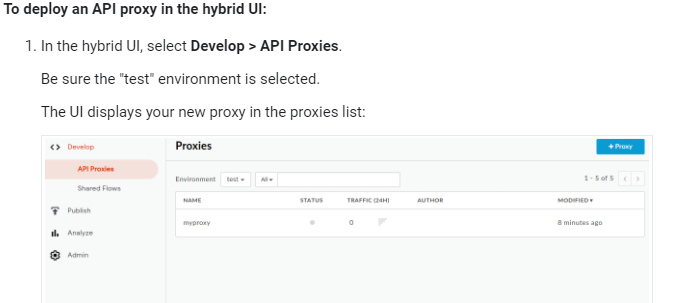


11. Click **Go to proxy list**.

Hybrid displays the **Proxies** view, which displays a list of API proxies. The new proxy should be at the top of the list, with a gray status indicator, meaning that it has not yet been deployed.

### **3.1.2 Deploy your proxy to the cluster using the hybrid UI**

After creating a new proxy, you must deploy it so that you can try it out. This section describes how to deploy your new proxy using the hybrid UI.



2. Click on the "myproxy" proxy.

The UI displays the **API Proxies Overview** tab for that proxy.

Notice that under **Deployments**, the **Revision** column shows "Not deployed".

1. In the **Revision** column, expand the drop-down selector to choose the revision to deploy.

The drop down list displays only "1" and "Undeploy".

1. Select "1"—the revision that you want to deploy—from the drop down list.

The UI prompts you to confirm the deployment:



1. Click **Deploy**

The UI begins the process of deploying revision 1 of your new proxy to the cluster.

Note that deployment is not an instantaneous process. Hybrid's "eventually consistent" deployment model means that a new deployment will be rolled out to the cluster over a short period of time and not immediately.

While there are [several ways](https://cloud.google.com/apigee/docs/api-platform/deploy/ui-deploy-overview#viewing-deployment-status) to check the deployment status of a proxy in the UI, the next two steps explain how to call the API proxy you just deployed and how to check the deployment status with a call to the [Apigee APIs](https://cloud.google.com/apigee/docs/reference/apis/apigee/rest).

### **3.1.3 Call the API proxy**

When the [UI indicates your proxy is deployed](https://cloud.google.com/apigee/docs/api-platform/deploy/ui-deploy-overview#viewing-deployment-status), you can try calling it using cURL or the REST client of your choice. For example:



If you used a self-signed certificate and you get a TLS/SSL error, try these steps instead:



## 3.2 Check your API proxy's deployment status using the API

This section describes how to check the deployment status of an API proxy using the [Apigee APIs](https://cloud.google.com/apigee/docs/reference/apis/apigee/rest).

1. Deploy a test API proxy as explained in [Create and deploy a new API proxy](https://cloud.google.com/apigee/docs/hybrid/v1.4/test-new-proxy).

2. Locate the JSON file with the Apigee Organization Admin service account key. This service account and key was created in [Enable synchronizer access](https://cloud.google.com/apigee/docs/hybrid/v1.4/install-apply-hybrid#enable-synchronizer-access).

3. Execute these two commands to get a token

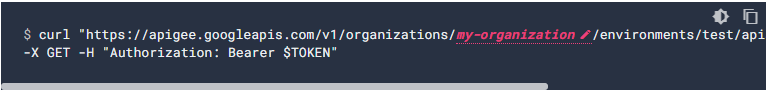
export GOOGLE\_APPLICATION\_CREDENTIALS=org-admin-service-account-file  
export TOKEN=$(gcloud auth application-default print-access-token)

Where **org-admin-service-account-file** is the path on your system to the service account key you downloaded with the **Apigee Organization Admin** role.

4. Call the revisions API, with the following parts:

* **Base URL:** https://apigee.googleapis.com/v1
* **Endpoint URL:** /organizations/my-organization/environments/test/apis/myproxy/revisions/1/deployments
* **Protocol:** HTTPS
* **Method:** GET
* **Headers:** "Authorization: Bearer $TOKEN"

The following example calls the deployment revisions API with these settings using curl:



You should receive a response similar to the following:



This example response shows the API proxy's status is READY, indicating a successful deployment.

If you get an empty response or an error, check that:

* You used the correct base URL. Note that the hybrid base URL is not the same as the Edge API's base URL. Use https://apigee.googleapis.com/v1.
* You used the correct endpoint URL. Note that the revision is "1" and the endpoint is /organizations/my\_organization/environments/test/apis/myproxy/revisions/1/deployments. If you specify a revision that doesn't exist, the request results in an empty response like the following:

