


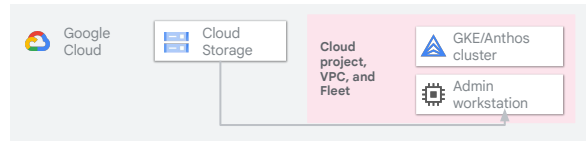
# Today's agenda



- 01 Introduction to Anthos Service Mesh
- 02 Lab: Anthos Service Mesh Walkthrough
- 03 Lab review
- 04 Architecture
- 05 [Installation](#)
- 06 Life of a request in the mesh
- 07 Mesh telemetry and instrumentation
- 08 Anthos Service Mesh dashboards
- 09 Anthos Service Mesh pricing and support
- 10 Lab: Observing Anthos Services

# Anthos Service Mesh installation prerequisites

- There is an Anthos or GKE cluster with at least 16 vCPUs and 15 GB of memory.
- Enable GKE Workload Identity.
- Connect the clusters to a fleet.
- If installing ASM on a private cluster, open port 15017 in the firewall to get the webhooks used for automatic sidecar injection and configuration validation to work.
- Download `asmcli` to a machine with direct access to your cluster.



Download `asmcli`

```
curl
https://storage.googleapis.com/csm-artifacts/asm/asmcli\_version# > asmcli
chmod +x asmcli
```

Google Cloud

Before installing Anthos Service Mesh, you must fulfil certain requirements. You must have created a GKE or an Anthos cluster with at least 16 vCPUs and 15 GB of memory. Also, your clusters must have GKE Workload Identity enabled and be registered to a fleet. When deploying Anthos clusters on VMware, bare metal, or AWS, the enablement of GKE workload Identity and the registration in a fleet takes place automatically.

Additionally, if you want to install Anthos Service Mesh on a private cluster, you must open port 15017 in the firewall to get the webhooks used for automatic sidecar injection and configuration validation to work.

Once those prerequisites are fulfilled, download the `asmcli` utility from Cloud Storage into a machine with direct access to your cluster in order to perform the installation.

# Anthos Service Mesh

## Managed Control Plane

- Managed Anthos Service Mesh is a Google-managed service mesh that you can enable.
- Google handles the reliability, upgrades, scaling, and security for you in a backward-compatible manner.
- If you need more control over the mesh - for example, you want to use a specific Anthos Service Mesh version - you can install and manage it yourself.

Google Cloud

Google recommends using Managed Anthos Service Mesh wherever possible. If you need to use a specific version of Anthos Service Mesh, then you instead install and use the self-managed Anthos Service Mesh.

Self-managed Anthos Service Mesh is covered later in this lecture.

## Use the fleet API to enable Anthos Service Mesh

When you enable managed Anthos Service Mesh using the fleet API:

- Google applies the recommended control plane configuration.
- Google enables automatic data plane management.
- Your cluster is enrolled in an Anthos Service Mesh release channel based on the GKE cluster release channel.
- Google enables endpoint discovery and cross cluster load balancing throughout your service mesh with default settings
  - You must create firewall rules.

Google Cloud

The fleet API is `mesh.googleapis.com`. The control plane and the data plane are kept up to date with each new release.

For more information, refer to [Provision managed Anthos Service Mesh](#) in the Google Cloud documentation.

# Anthos Service Mesh installation

## Self-managed

- Anthos Service Mesh uses the `asmcli` utility to perform the ASM control plane installation on a cluster.
- Notice some of the parameters used on the installation command:
  - `--enable_all`
    - Grant required IAM permissions.
    - Enable the required Google APIs.
    - Label the cluster identifying the mesh.
    - Register the cluster to fleet.
  - `--ca`
    - `mesh_ca`: uses the Google-managed Mesh CA.
    - `gcp_cas`: uses Mesh CA together with the CA Service.
    - `citadel`: uses the in-cluster Istio CA.

```
./asmcli install \  
  --project_id PROJECT_ID \  
  --cluster_name NAME \  
  --cluster_location LOCATION \  
  --fleet_id FLEET_PROJECT_ID \  
  --output_dir DIR_PATH \  
  --enable_all \  
  --ca mesh_ca  
  --custom_overlay OVERLAY_FILE
```

- `--custom_overlay`
  - Envoy access logs
  - Cloud Tracing

Google Cloud

Anthos Service Mesh uses the `asmcli` utility to perform the ASM control plane installation on a cluster.

Let's take a look at some of the flags available with this command.

Enable all, performs all steps required before installing Anthos Service Mesh. Those include:

- Granting required IAM permissions.
- Enabling the required Google APIs.
- Labeling the cluster identifying the mesh.
- And registering the cluster to a fleet.

The CA flag, enables you to choose your CA of choice. The options include:

- `mesh_ca`, which uses the Google-managed Mesh CA.
- `gcp_cas`, which uses Mesh CA together with the CA Service.
- And `citadel`, which uses the in cluster Istio CA.

Finally, you can add additional features by including a `custom_overlay`. For instance, you can configure the mesh to collect Envoy access logs in Cloud Logging and traces in Cloud Tracing.

# Anthos Service Mesh sidecar injection

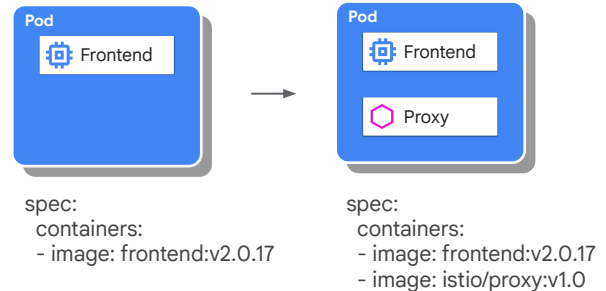
For an application to start using the mesh:

- Manually inject the Envoy sidecar proxies.

```
istioctl kube-inject -f app.yaml | \
kubectl apply -f -
```

- Configure automatic sidecar injection in a namespace.

```
kubectl create namespace $APP_NAMESPACE
kubectl label namespace $APP_NAMESPACE \
  istio.io/rev=asm-1112-17 --overwrite
kubectl apply -n APP_NAMESPACE -f ./app.yaml
```



Google Cloud

Great, at this point, we have the mesh created but applications are not using it yet.

For an application to start using the mesh, we need to either:

- Manually inject the Envoy sidecar proxies with the “istioctl kube-inject” command. This command injects the sidecar proxy configuration in your application YAML and it applies the new configuration.
- Or configure automatic sidecar injection in a namespace by labeling the namespace with the same asm revision as the control plane. Sidecars are automatically added to your pods using a mutating webhook admission controller.

As soon as the sidecar proxies are running, they will intercept traffic, add mTLS when possible, and collect telemetry data which will be accessible from the Anthos Service Mesh console. No configuration required.

# Anthos Service Mesh sidecar injection

For an application to start using the mesh:

- Manually inject the Envoy sidecar proxies.

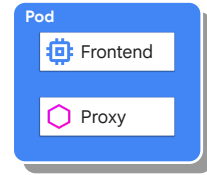
```
istioctl kube-inject -f app.yaml | \
kubectl apply -f -
```

- Configure automatic sidecar injection in a namespace.

```
kubectl create namespace $APP_NAMESPACE
kubectl label namespace $APP_NAMESPACE \
  istio.io/rev=asm-1410-1 --overwrite
kubectl apply -n APP_NAMESPACE -f ./app.yaml
```



spec:  
containers:  
- image: frontend:v2.0.17



spec:  
containers:  
- image: frontend:v2.0.17  
- image: istio/proxy:v1.0

In-cluster control plane with Istiod:  
- asm-141--1: version of Anthos Service Mesh  
Google-managed control plane:  
- asm-managed for the regular channel  
- asm-managed-rapid for the rapid channel  
- asm-managed-stable for the stable channel

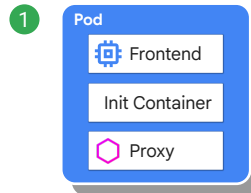
Google Cloud

Notice that here we are using an in-cluster istiod with ASM 14.1. If we were using the Google-managed control plane with managed upgrades, we could use the revision to specify the cadence at which Anthos Service Mesh gets upgraded. Use the revision `asm-managed` for the regular channel, `asm-managed-rapid` for the rapid channel, and `asm-managed-stable` for the stable channel. In open-source Istio, you would use a different label called `istio-injection` and set it to `enabled`.

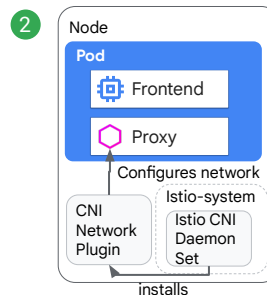
# Anthos Service Mesh sidecar network configuration

Depending on your mesh configuration:

1. An Init Container configures the IP Tables of the pod.
  - Requires Kubernetes RBAC permissions to deploy pods with the NET\_ADMIN and NET\_RAW.
  - Elevated Kubernetes RBAC permissions is problematic for security compliance for some organizations.
2. Istio Container Network Interface (CNI) plugin configures traffic redirection in the network setup phase of the Kubernetes pod lifecycle.
  - Elevated Kubernetes RBAC permissions are not needed.



spec:  
containers:  
- image: frontend:v2.0.17  
- image: istio/proxy:v1.0  
initContainers:  
- image: istio-init



spec:  
containers:  
- image: frontend:v2.0.17  
- image: istio/proxy:v1.0

Google Cloud

You might still be wondering, how do the sidecars configure the pod's networking to intercept all requests?

There are two options depending on your mesh configuration:

- By default an Init Container configures the IP Tables of the pod.
  - In this configuration, the user or service-account deploying pods to the mesh must have Kubernetes RBAC permissions to deploy pods with the NET\_ADMIN and NET\_RAW.
  - Elevated Kubernetes RBAC permissions is problematic for some organizations' security compliance.
- Alternatively, you can enable the Istio CNI plugin, which configures traffic redirection in the Kubernetes pod lifecycle's network setup phase.
  - This option does not require elevated Kubernetes RBAC permissions.