Service Mesh/Istio

This chapter contains the following topics:

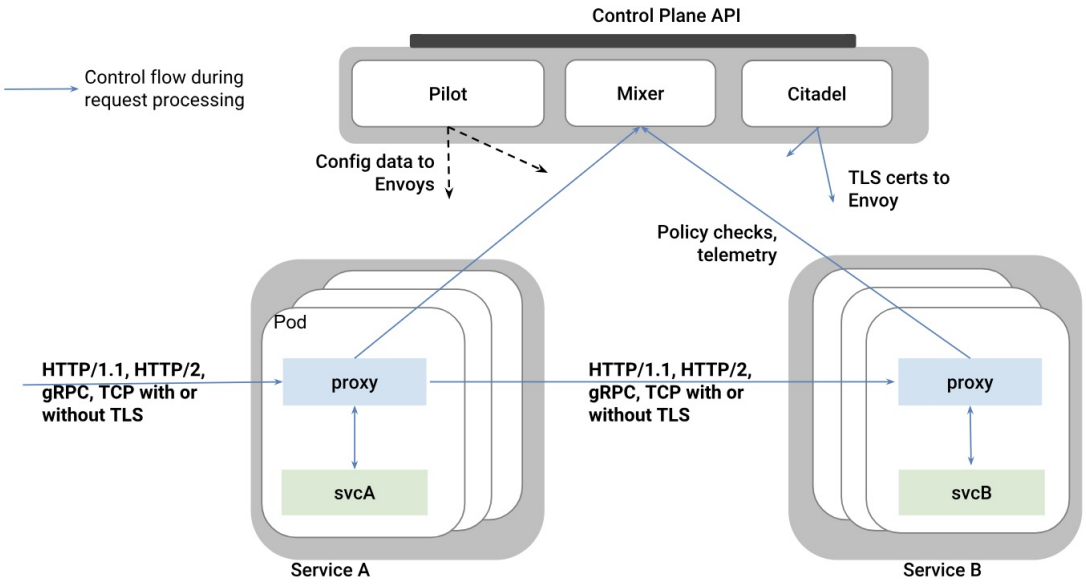
Introduction to Service meshes and Istio

Cisco Container Platform includes support for Istio service meshes. An Istio service mesh is logically split into a data plane and a control plane. The data plane is composed of a set of intelligent proxies (Envoy) and the control plane provides a reliable Istio framework. The term Istio is sometimes also used as a synonym to refer to the entire service mesh stack that includes the Control Plane and the Data Plane components (although strictly istio is the control plane and a proxy like envoy is the data plane).

The Service mesh technology allows you to construct North-South and East-West L4 and L7 application traffic meshes. It provides containerized applications a language-independent framework that removes several common tasks related to L4 and L7 application networking from the actual application code while enhancing operational capabilities including monitoring, security, load balancing and troubleshooting for these applications. These tasks include L4 and L7 service routing and load balancing, support for polyglot environments in a language-independent manner and advanced telemetry. You can deploy a service mesh in a multi-cloud topology allowing these functions to operate with applications that run across multiple separate cloud deployments.

The following figure shows a high-level architecture summary of an Istio based service mesh architecture. In Cisco Container Platform, the components of Istio and Envoy are supported in the upstream Istio community. The Control and Data Plane components of the solution, including Pilot, Mixer, Citadel and the data plane Envoy proxy for both North-South and East-West load balancing, are supported on Cisco Container Platform.

For more information on these technologies, refer to the upstream community documentation pages for istio and envoy.



Istio Architecture

Note Currently, this feature is marked as a Technology Preview feature and uses the Istio community version v0.8.0. You need to contact your service representative for support details for the version of Cisco Container Platform that you happen to be running.

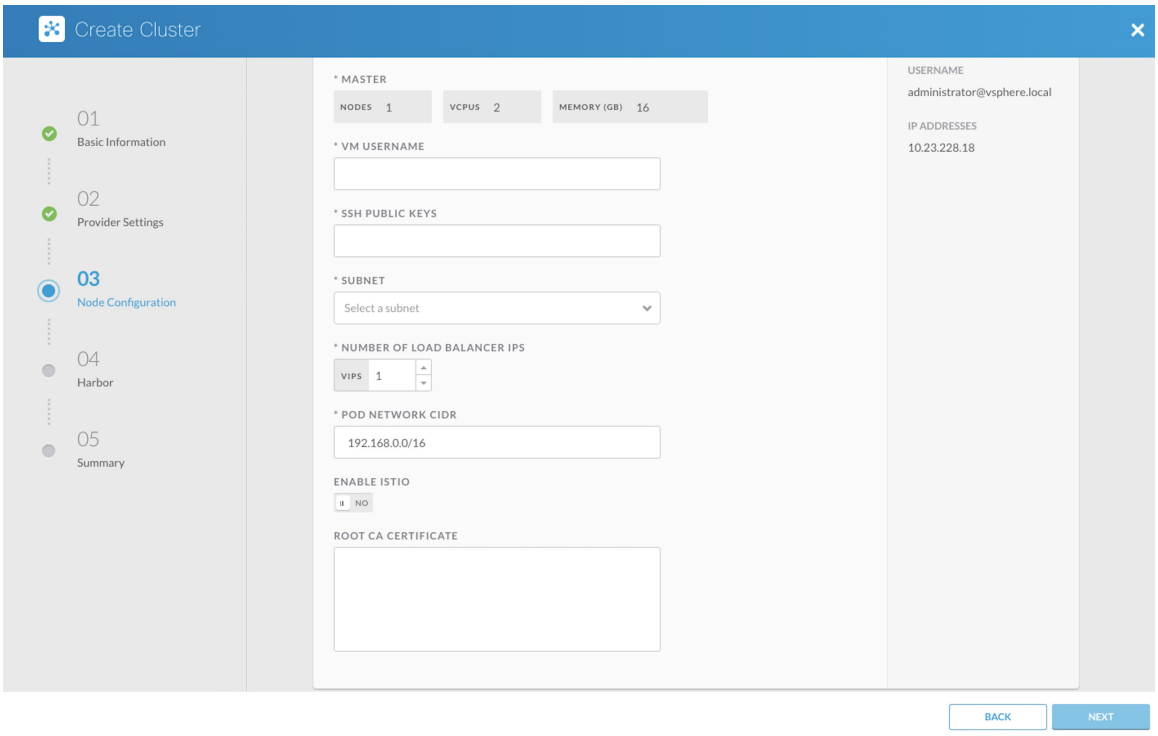
Configuring Service meshes/ Istio in Cisco Container Platform

An Istio service mesh is a configurable feature on the Cisco Container Platform. You can configure a separate instance of the service mesh stack on each tenant cluster. Support for Istio needs to be configured at the time of creation of a tenant kubernetes cluster. You can perform this configuration using APIs or the Cisco Container Platform web user interface.

Each instance of the Istio service mesh consumes an IP address from the Virtual IP address pool that is associated with the tenant cluster. Consequently, you need to ensure that there is sufficient number of IP addresses free and available in the VIP pool before enabling Istio. Typically at least three IP addresses are required–1 each for the kubernetes api, kubernetes Ingress, and Istio ingress gateway. This number may change in future when additional features consume some more Virtual IP addresses.

For more information on the required number of Virtual IP addresses for a given software version of Cisco Container Platform, refer to the Virtual IP address section.

The following figure shows a screen capture from the GUI that shows how this feature can be enabled on a tenant cluster of the Cisco Container Platform.



Note that in this version of software, there is just a single boolean flag to enable an istio based service mesh in a tenant cluster of Cisco Container Platform. If enabled, a pre-determined configuration of an istio based service mesh (with envoy as the data plane) is configured in the tenant kubernetes cluster. An internal instance of a service loadbalancer is automatically configured and a Virtual IP address is automatically allocated for use by the ingress gateway function of istio.

Monitoring Service meshes

On Cisco Container Platform, the Istio control plane is deployed in a special istio-system namespace of a tenant kubernetes cluster. This is similar to how other add-on services such as Prometheus based monitoring or NGINX based Kubernetes ingress are provided to end users. In a production deployment, a tenant cluster admin will typically grant individual application teams read-write access to their own development namespaces but not to namespaces of system add-on services such as istio thereby protecting the control plane of such services from getting over-written accidentally or maliciously by end-user application containers.

The following is a checklist of monitoring and troubleshooting steps when using istio on the Cisco Container Platform:

1. If tenant cluster fails to deploy with istio enabled, in addition to the usual troubleshooting steps for Cisco Container Platform, also check to ensure there were a sufficient number of Virtual IP addresses available in the pool configured for this tenant cluster. In v1.4 version of Cisco Container Platform, at least 3 IP addresses need to be free and available for a tenant cluster that also has istio enabled.
2. Confirm that all pods are running in the istio-system namespace of the tenant cluster. The following figure shows a sample CLI output indicating all istio control pods are running correctly in a tenant cluster. If one or more pods continuously fails to run, use "kubectl describe pod <name_of_pod>" to try and isolate the issue.

```
ccpuser@vhosakot-istio14-master5ebb31962c:~$ kubectl get pods -n istio-system
NAME                                      READY     STATUS      RESTARTS   AGE
grafana-5b977b576f-2r5gs                  1/1       Running     0          20h
istio-citadel-5ff4f56f56-lk6wz            1/1       Running     0          20h
istio-egressgateway-6567bc7ffb-84tj8      1/1       Running     0          20h
istio-ingressgateway-5dfb78f45b-c6jxc     1/1       Running     0          20h
istio-mixer-post-install-w56cx            0/1       Completed   0          20h
istio-pilot-6ddc9b5b49-hl5nd              2/2       Running     0          20h
istio-policy-f67cb98b5-n2q2m              2/2       Running     0          20h
istio-sidecar-injector-5545db64bf-tttc9   1/1       Running     0          20h
istio-statsd-prom-bridge-949999c4c-82spd  1/1       Running     0          20h
istio-telemetry-667d4c6765-2s9hj          2/2       Running     0          20h
istio-tracing-754cdfd695-2wd45            1/1       Running     0          20h
prometheus-86cb6dd77c-4cj77               1/1       Running     0          20h
servicegraph-ccd4d4859-sgcwc              1/1       Running     0          20h
```

3. Confirm that all istio services are running in the istio-system namespace of the tenant cluster. The following figure shows a normal working CLI output for checking this.

```
ccpuser@vhosakot-istio14-master5ebb31962c:~$ kubectl get svc -n istio-system
NAME                       TYPE           CLUSTER-IP       EXTERNAL-IP    PORT(S)
grafana                    ClusterIP      10.98.223.200    <none>         3000/TCP
istio-citadel              ClusterIP      10.97.93.126     <none>         8060/TCP,9093/TCP
istio-egressgateway        ClusterIP      10.108.19.80     <none>         80/TCP,443/TCP
istio-ingressgateway       LoadBalancer   10.111.228.87    10.10.99.148   80:31380/TCP,443:31390/TCP,31400:31400/TCP
istio-pilot                ClusterIP      10.104.249.174   <none>         15003/TCP,15005/TCP,15007/TCP,15010/TCP,15011/TCP,8080/TCP,909
istio-policy               ClusterIP      10.108.75.85     <none>         9091/TCP,15004/TCP,9093/TCP
istio-sidecar-injector     ClusterIP      10.109.55.202    <none>         443/TCP
istio-statsd-prom-bridge   ClusterIP      10.107.183.156   <none>         9102/TCP,9125/UDP
istio-telemetry            ClusterIP      10.110.209.16    <none>         9091/TCP,15004/TCP,9093/TCP,42422/TCP
prometheus                 ClusterIP      10.101.6.183     <none>         9090/TCP
servicegraph               ClusterIP      10.105.53.151    <none>         8088/TCP
tracing                    LoadBalancer   10.101.62.116    <pending>      80:31960/TCP
zipkin                     ClusterIP      10.99.116.160    <none>         9411/TCP
ccpuser@vhosakot-istio14-master5ebb31962c:~$
```

4. Confirm that the ingress gateway service has an external IP address allocated and that this IP address is one of the previously available IP addresses in the Virtual IP address pool associated with this tenant cluster. An example of this CLI output can be seen in the prior figure.

5. If everything looks okay you should be able to deploy sample applications such as the well known *bookinfo* example application documented in the istio upstream community web site.

6. The istioctl cli utility is not deployed in the current version of the Cisco Container Platform. Most of the istio functionality is now available via regular kubectl based cli but if you should need to use istioctl, then the following steps can be used to deploy it on a tenant kubernetes cluster of the Cisco Container Platform.

```
export ISTIO_VERSION=0.8.0
curl -L https://git.io/getLatestIstio | sh -
chmod +x istio-${ISTIO_VERSION}/bin/istioctl
sudo mv istio-${ISTIO_VERSION}/bin/istioctl /usr/local/bin/
istioctl version
```

Since istio is an early and evolving project, you are encouraged to make use of upstream documentation for full details and operational guidelines.