Ingress Gateway in Istio

What is an Istio Gateway?

Pavan Kumar
Jul 31, 2020 · 4 min read

An Istio Gateway describes a LoadBalancer operating at either side of the service mesh. Istio Gateways are of two types.

Istio Ingress Gateway: Controlling the traffic coming inside the Mesh.

Istio Egress Gateway: Controlling the traffic going outside the Mesh.

Now let us understand this thing with an example.

I have 2 versions of my application running in my cluster version:v1 and version:v2. The version v1 is available at http://<ingress-ip>/v1 and the second version is available at http://<ingress-ip>/v2. The same scenario can also be achieved by using Kubernetes Ingress, but when we use Istio Gateways we can take advantage of the rich Istio Traffic Management and Security features like Request-Routing, Traffic Mirroring, Circuit breaking, etc.

Now Deploy the Pod having the webpage serving the version:v1 contents

```
apiVersion: v1
kind: Pod
metadata:
  labels:
    app: httpd
    version: v1
  name: httpd-v1
  namespace: default
spec:
  containers:
  - image: httpd
    name: httpd
    resources: {}
```

```
    resources: {}
    volumeMounts:
    - mountPath: /usr/local/apache2/htdocs
      name: index-html
  dnsPolicy: ClusterFirst
  initContainers:
  - command:
    - sh
    - -c
    - mkdir /usr/local/apache2/htdocs;( echo '<html> <body> <h1>This is version V1!</h1>
      </body></html>' ) > /usr/local/apache2/htdocs/index.html
    image: busybox
    name: busybox
    volumeMounts:
    - mountPath: /usr/local/apache2/htdocs
      name: index-html
  restartPolicy: Always
  volumes:
  - emptyDir: {}
    name: index-html
```

podv1.yaml hosted with ♡ by GitHub view raw

Now deploy a new pod with the new version:v2. The pod would be picked by the web-service service selector.

```
apiVersion: v1
kind: Pod
metadata:
  labels:
    app: httpd
    version: v2
  name: httpd-v2
  namespace: default
spec:
  containers:
  - image: httpd
    name: httpd
    resources: {}
    volumeMounts:
    - mountPath: /usr/local/apache2/htdocs
      name: index-html
  dnsPolicy: ClusterFirst
  initContainers:
  - command:
    - sh
    - -c
    - "mkdir /usr/local/apache2/htdocs;(
```

```
echo '<html>
<body>
<h1>This is version V2!</h1>
</body></html>'
) > /usr/local/apache2/htdocs/index.html"
    name: busybox
    image: busybox
    volumeMounts:
    - mountPath: /usr/local/apache2/htdocs
      name: index-html
  restartPolicy: Always
  volumes:
  - emptyDir: {}
    name: index-html
```

podv2.yaml hosted with ♡ by GitHub view raw

Expose your pod via ClusterIP service as we would be using Istio Ingress Gateway to expose our services to the outside world.

Now if you curl the web service you could see the requests round robins to version v1 and v2

> [root@master istio-medium]# **kubectl run curl-test — image=odise/busybox-curl — rm -it — /bin/sh -c “while true; do curl web-service; sleep 1; done”**
> <html> <body> <h1>This is version V1!</h1> </body></html>
> <html> <body> <h1>This is version V2!</h1> </body></html>
> <html> <body> <h1>This is version V1!</h1> </body></html>
> <html> <body> <h1>This is version V2!</h1> </body></html>
> <html> <body> <h1>This is version V1!</h1> </body></html>
> <html> <body> <h1>This is version V2!</h1> </body></html>
> <html> <body> <h1>This is version V1!</h1> </body></html>

But we want our version:v1 application to be served at “http://<ingress-ip>/v1” and our version:v2 to be served at “http://<ingress-ip>/v2”. This could be achieved by using the VirtualService CRD by Istio. But first, let's define a Gateway(Load-Balancer) for our application. This is done by using the Gateway resource in Istio.

Gateway:

a) name: Specifies the name of the Gateway

b) selector: These are the labels of the gateway on which the configuration should be applied.

c) servers: This specifies the list of server specifications.

d) port.number: The port number on which the gateway should listen.

e) port.name: The name that should be given to the port.

f) port.protocol: The protocol exposed on the port.

g) hosts: The hosts exposed by this gateway.

Unlike Kubernetes Ingress resources the Gateway configuration doesn't include traffic routing configurations. Instead, the traffic configuration is made in Istio CRD's like VirtualService and DestinationRules. Let us see how to configure the above scenario in VirtualService and DestinationRules.

Virtual Service:

a) name: Specifies the name of the Virtual Service

b) hosts: The destination hosts to which traffic is being sent.

c) http: It is the list of routing rules for HTTP traffic

d) **gateway.name**: The name of the gateway to which this configuration should be applied. This should match the name given in the Gateway resource.

e) http.match.uri.prefix: URI to match for prefix-based match.

f) rewrite.uri: Target URI where the traffic must be redirected. This is similar to the annotation **nginx.ingress.kubernetes.io/rewrite-target** in nginx-ingress controller.

f) subset: The name of the subset that the traffic should be directed to which is defined in the corresponding Destination Rule

Destination Rule:

a) name: The name of the Destination Rule

b) hosts: The host to which traffic is sent. Here the host is the DNS name of our Kubernetes Service

c) subsets: Named set that represents the Individual version of the service

d) subsets.name: Name of the subset

e) labels: Map of the labels that are used to select the pods

Now let's apply the gateway and the corresponding VirtualService and DestinationRules.

And now curl http://<ingress_ip><ingress_node_port>/v1 for first version and http://<ingress_ip><ingress_node_port>/v2 for second version

Ingress gateway in Istio

Let us visualize the same using the kiali dashboard. We would try to access only the version:v1 using the prefix "/v1". This should route the requests only to the version:v1.

> ***istioctl dashboard kiali***

Traffic is routed to only version:v1 based on the user request

Hurrah!! We successfully implemented Ingress gateway in Istio and we also implemented path-based routing using Istio Ingress gateway.

Prerequisites

Introduction to Istio Service Mesh

What is Istio Service Mesh?

medium.com

How to Install Istio using istioctl

As discussed in my earlier article we would be working on a few real-time scenarios where we would be able to...

medium.com

Recommended

Mirroring of Live Traffic in Kubernetes using Istio Traffic Mirroring

Why is traffic mirroring needed?

medium.com

MTLS in Istio

Setup Mutual TLS in Istio Service Mesh by plugging in existing CA Certificates

medium.com

Weighted routing in Kubernetes using Istio

Let us suppose version:v1 of your application is being used by your customers. A new version:v2 of an application is...

medium.com

Istio Service Mesh Istio Kubernetes Ingress Microservices