# Istio

# Prerequisites

# These steps require you to have a cluster running a compatible version of Kubernetes, preferably a multi-node Kubernetes

# 

# Installation

# $ curl -L https://istio.io/downloadIstio | sh -

# $ cd istio-1.5.1

# $ export PATH=$PWD/bin:$PATH

# Install istio with demo profile. This is ideal for getting started as it is configured with default settings and have monitoring tools like Prometheus, Grafana, Jaeger and Kiali installed by default. In production, it is not recommended and should use “default”

# $ istioctl manifest apply --set profile=demo

Detected that your cluster does not support third party JWT authentication. Falling back to less secure first party JWT

- Applying manifest for component Base...

✔ Finished applying manifest for component Base.

………

………

✔ Installation complete

Instruct Istio to automatically inject Envoy sidecar proxies on default namespace. This will ensure istio inject the side car proxies in all application we deploy on to default namespace

# $ kubectl label namespace default istio-injection=enabled

# Traffic Routing

# $ cd hello-world/routing

# Deploy the hello-world versions V1 and V2. A Cluster IP service shall also be created by the yaml file.

# $ kubectl create -f helloworld-both.yaml

# $ kubectl get svc

# 

# helloworld ClusterIP 10.100.102.17 <none> 5000/TCP 1h

# $ curl <http://10.100.102.17:5000/hello>

# Hello version: v1, instance: helloworld-v1-578dd69f69-qq8nb

# Create a Istio Gateway by applying below yaml file.

# $ kubectl create -f helloworld-gateway.yaml

# Now send request to the service and see the traffic getting routed to the both V1 and V2. You can see the graph using Kiali dashboard.

# $ watch curl -s -o /dev/null <http://10.100.102.17:5000/hello>

# Connecting to Kiali

# $ kubectl get svc -n istio-system -o wide

# $ kubectl edit svc -n istio-system kiali

# Modify the service to use the type as NodePort (search for type = ClusterIP and update to NodePort) and save it.

# 

# $ kubectl get svc -n istio-system -o wide

# kiali NodePort 10.111.82.175 <none> 20001:32215/TCP

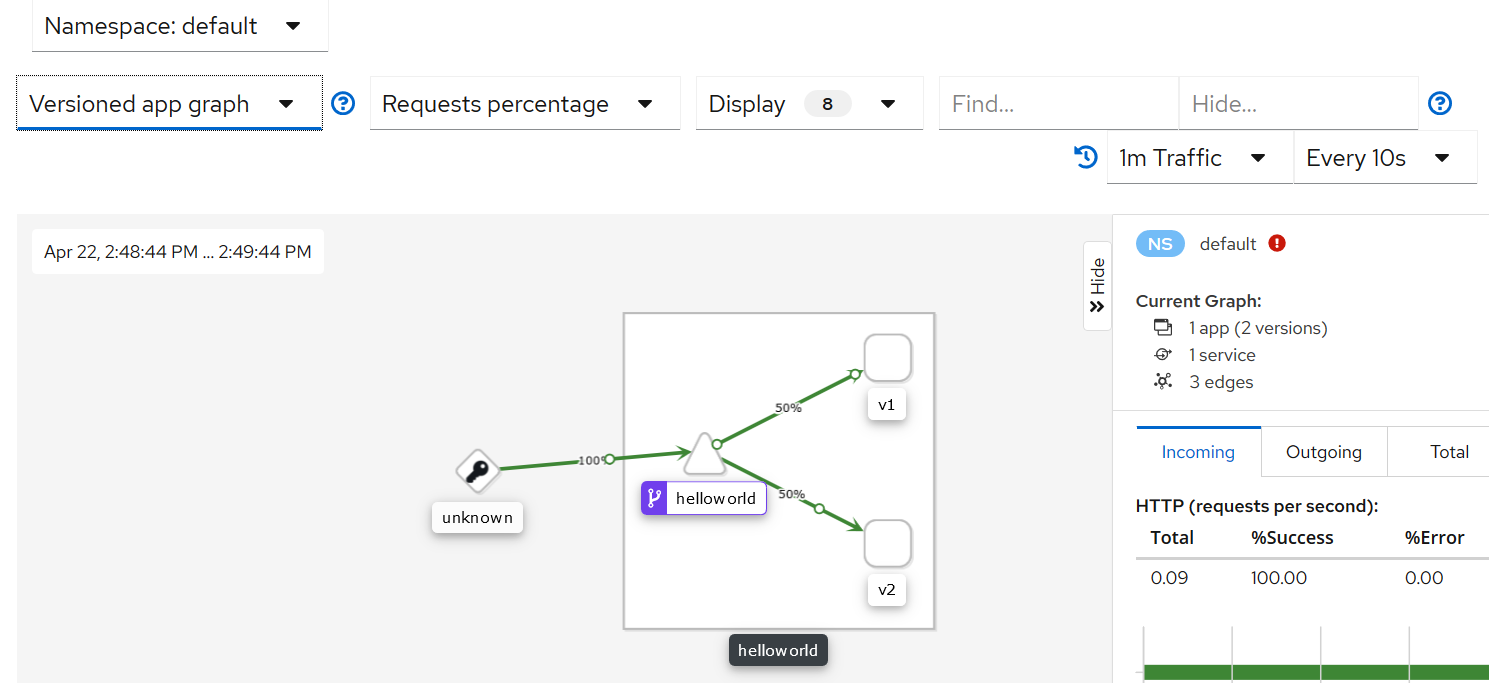
# Use the NodePort IP (In my cluster it is pointing to 32215) to connect Kilai from any of the Node port from the browser

# <http://172.42.42.100:32215/>

# Username/password admin/admin

# Select the Graph Menu and Choose the namespace as default. The helloworld service traffic routing can be analyzed from the Kiali dashboard

# 



# Security

# $ cd hello-world/auth

# $ kubectl create -f echo-server.yaml

# $ kubectl get svc -o wide

# NAME TYPE CLUSTER-IP EXTERNAL-IP PORT(S) AGE SELECTOR

# echoserver ClusterIP 10.100.12.225 <none> 8080/TCP 8m3s app=echoserver

# $ curl 10.100.12.225:8080

# The output shall response from EchoServer; ie details about the request and headers

# Hostname: echoserver-v1-59b7f9c5c7-qvhwq

# Pod Information:

# -no pod information available-

# Server values:

# server\_version=nginx: 1.13.3 - lua: 10008

# Request Information:

# client\_address=127.0.0.1

# method=GET

# real path=/

# query=

# request\_version=1.1

# request\_scheme=http

# request\_uri=http://10.100.12.225:8080/

# Request Headers:

# accept=\*/\*

# content-length=0

# host=10.100.12.225:8080

# user-agent=curl/7.29.0

# x-b3-sampled=1

# x-b3-spanid=f137b146f69729c6

# x-b3-traceid=3608b8208bd22ae3f137b146f69729c6

# x-forwarded-proto=http

# x-request-id=d4c5f17a-baae-9b09-905a-048307e71051

# Request Body:

# -no body in request-

# Deploy an nginx server and see if we can access the EchoServer from the Nginx pod

# $ kubectl create -f nginx.yaml

# $ kubectl get svc,pod -o wide

# Make sure the Nginx service and POD are running. Connect to Nginx POD using exec

# $ kubectl exec -it nginx-v1-698b4f6679-p6cbx /bin/sh

# If curl is not available, install it using apk

# # apk update

# # apk add curl

# # curl 10.100.12.225:8080

# # exit

If curl command from Nginx POD works, that indicate the communication between the PODs are working fine.

# # apk add curl

# MTLS Policy

1. **USE\_PEER**

# $ kubectl create -f mtls-peer-auth.yaml

# policy.authentication.istio.io/echoserver created

$ kubectl get svc -o wide

# NAME TYPE CLUSTER-IP EXTERNAL-IP PORT(S) AGE SELECTOR

# echoserver ClusterIP 10.100.12.225 <none> 8080/TCP 8m3s app=echoserver

Update the echoserver service to change to NodePort so that we can try to access it from outside the cluster.

$ kubectl edit svc echoserver

Update the type to NodePort and save it.

$ kubectl get svc -o wide

echoserver NodePort 10.100.12.225 <none> 8080:31007/TCP 4h56m app=echoserver

Now try to access the URL from outside the cluster using your browser. You will not able to access the service now.

http://<NodePort IP>:<NodePort>

For Exp:- http://172.42.42.100:31007/

$ kubectl exec -it nginx-v1-698b4f6679-p6cbx /bin/sh

Defaulting container name to nginx.

/ # curl 10.100.12.225:8080

Hostname: echoserver-v1-59b7f9c5c7-qvhwq

Pod Information:

-no pod information available-

……

/ #exit

1. **ISTIO\_MUTUAL**

# $ kubectl create -f mtls-mutual-auth.yaml

# policy.authentication.istio.io/echoserver created

$ kubectl get svc -o wide

# NAME TYPE CLUSTER-IP EXTERNAL-IP PORT(S) AGE SELECTOR

# echoserver ClusterIP 10.100.12.225 <none> 8080/TCP 8m3s app=echoserver

Update the echoserver service to change to NodePort so that we can try to access it from outside the cluster.

$ kubectl edit svc echoserver

Update the type to NodePort and save it.

$ kubectl get svc -o wide

echoserver NodePort 10.100.12.225 <none> 8080:31007/TCP 4h56m app=echoserver

Now try to access the echoserver URL from the Nginx pod.

$ kubectl exec -it nginx-v1-698b4f6679-p6cbx /bin/sh

Defaulting container name to nginx.

/ # curl 10.100.12.225:8080

Hostname: echoserver-v1-59b7f9c5c7-qvhwq

Pod Information:

-no pod information available-

……