

Universidad de Guadalajara.

Computación Tolerante a fallas.



Diego Ivan Becerra Gonzalez.

Sección D06.

## Objetivo.

Responder a las preguntas y generar un ejemplo utilizando Istio.

## Desarrollo.

¿Qué es Istio?

Istio es una plataforma de servicio de malla (service mesh) de código abierto que se utiliza para conectar, asegurar, controlar y observar microservicios en una infraestructura de nube.

Istio proporciona una capa de abstracción entre los servicios y la red subyacente, lo que permite una mayor visibilidad, seguridad y control sobre las comunicaciones entre servicios. Esto se logra mediante la inyección de un proxy Envoy en cada contenedor de servicio, que gestiona el tráfico de red y proporciona una amplia gama de características de seguridad y monitorización.

Istio también ofrece características como el balanceo de carga, el control de tráfico basado en reglas, la gestión de autenticación y autorización, la telemetría y la trazabilidad de los servicios. Estas características ayudan a los desarrolladores y administradores de infraestructuras a gestionar mejor los entornos de microservicios, lo que les permite mejorar la seguridad, la confiabilidad y el rendimiento de sus aplicaciones.

Ahora seguimos con la ejecución del tutorial.

Primero abrimos Docker.

Después ejecutamos el siguiente comando en la consola para instalar istio.


```
C:\Users\diego>istioctl install --set profile=demo -y
✓ Istio core installed
✓ Istiod installed
✓ Egress gateways installed
✓ Ingress gateways installed
✓ Installation complete
M
making this installation the default for injection and validation.


Thank you for installing Istio 1.17. Please take a few minutes to tell
us about your install/upgrade experience! https://forms.gle/hMHGiwZHPU7UQRWe9
```


Después corremos el siguiente comando para que Istio inyecte en todos los pods del namespace sus sidecars, así podrá escuchar la información entre pods.

```
C:\Users\diego>kubectl label namespace default istio-injection=enabled
namespace/default labeled
```

Después descargamos los siguientes tres archivos del repositorio de GitHub del video:

 01-hello-app-v1

 02-hello-app-v2

 04-nginx-proxy

Y montamos las imágenes en Docker.

```
C:\istio-1.17.2>kubectl apply -f 01-hello-app-v1.yaml
deployment.apps/hello-v1 created
service/hello-v1-svc created

C:\istio-1.17.2>kubectl apply -f 02-hello-app-v2.yaml
deployment.apps/hello-v2 created
service/hello-v2-svc created

C:\istio-1.17.2>kubectl apply -f 04-nginx-proxy.yaml
deployment.apps/nginx created
configmap/nginx-config created
service/nginx-svc created
```

Tenemos que descargar los demás componentes como Grafana, Prometheus, Kiali y Jaeger, ya que a pesar de que en el video menciona que instalando Istio se instala toda la otra paquetería, en realidad no es así. Esto se hace con los siguientes comandos:

```
C:\istio-1.17.2>kubectl apply -f https://raw.githubusercontent.com/istio/istio/release-1.15/samples/addons/kiali.yaml
serviceaccount/kiali created
configmap/kiali created
clusterrole.rbac.authorization.k8s.io/kiali-viewer created
clusterrole.rbac.authorization.k8s.io/kiali created
clusterrolebinding.rbac.authorization.k8s.io/kiali created
role.rbac.authorization.k8s.io/kiali-controlplane created
rolebinding.rbac.authorization.k8s.io/kiali-controlplane created
service/kiali created
deployment.apps/kiali created

C:\istio-1.17.2>kubectl apply -f https://raw.githubusercontent.com/istio/istio/release-1.15/samples/addons/prometheus.yaml
serviceaccount/prometheus created
configmap/prometheus created
clusterrole.rbac.authorization.k8s.io/prometheus created
clusterrolebinding.rbac.authorization.k8s.io/prometheus created
service/prometheus created
deployment.apps/prometheus created

C:\istio-1.17.2>kubectl apply -f https://raw.githubusercontent.com/istio/istio/release-1.15/samples/addons/jaeger.yaml
deployment.apps/jaeger created
service/tracing created
service/zipkin created
service/jaeger-collector created
```

```
C:\istio-1.17.2>kubectl apply -f https://raw.githubusercontent.com/istio/istio/release-1.15/samples/addons/grafana.yaml
serviceaccount/grafana created
configmap/grafana created
service/grafana created
deployment.apps/grafana created
configmap/istio-grafana-dashboards created
configmap/istio-services-grafana-dashboards created
```

Verificamos que la instalación haya sido correcta.

```
C:\istio-1.17.2>kubectl -n istio-system get pods
```

NAME	READY	STATUS	RESTARTS	AGE
grafana-56bdf8bf85-bh2pb	1/1	Running	0	75s
istio-egressgateway-85649899f8-v76h8	1/1	Running	0	8m19s
istio-ingressgateway-f56888458-5lzh	1/1	Running	0	8m19s
istiod-64848b6c78-xtgw2	1/1	Running	0	8m33s
jaeger-c4fdf6674-zrl5w	1/1	Running	0	83s
kiali-5ff49b9f69-pg4np	1/1	Running	0	114s
prometheus-85949fddb-sqgmc	2/2	Running	0	95s

Y ejecutamos la interfaz de Kiali:

```
C:\istio-1.17.2>istioctl dashboard kiali
http://localhost:20001/kiali
```

El comando anterior lo que hará es abrir la interfaz grafica de kiali en el navegador. Aquí podemos ver los diferentes pods y servicios en nuestro cluster, están funcionando, pero no hay comunicación entre ellos.



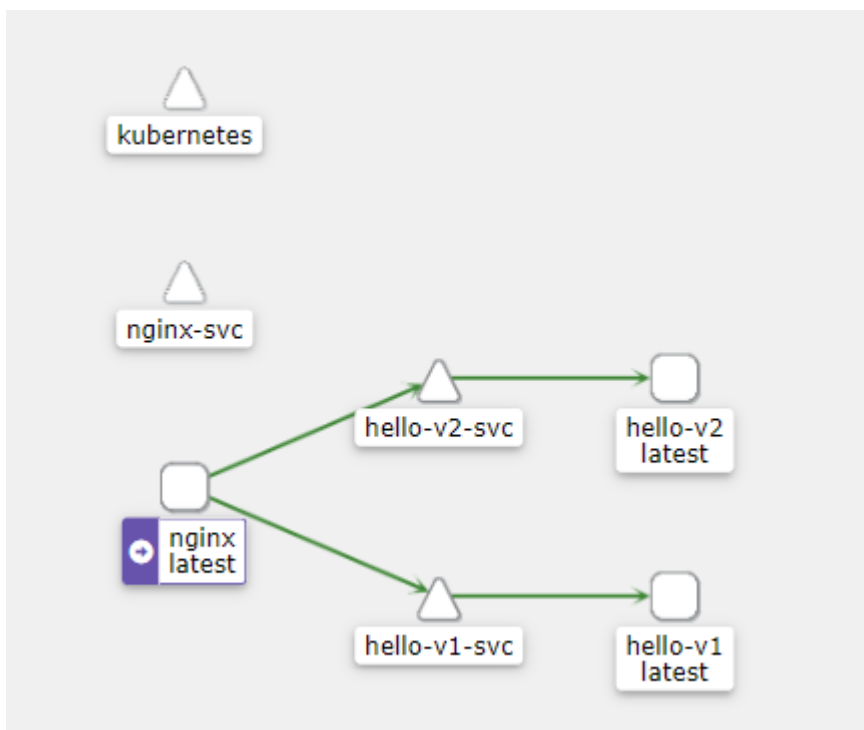
Para comenzar la comunicación de prueba entre pods tenemos que ejecutar el siguiente comando:

```
PS C:\Users\diego> kubectl port-forward svc/nginx-svc 8080:80
```

Y en una terminal de Linux(en este caso se aprovecho que estaba instalado GitBash) se ejecuta el siguiente comando:

```
diego@LAPTOP-ECVOPM04 MINGW64 ~  
$ while sleep 1; do curl localhost:8080/v1 && curl localhost:8080/v2;done
```

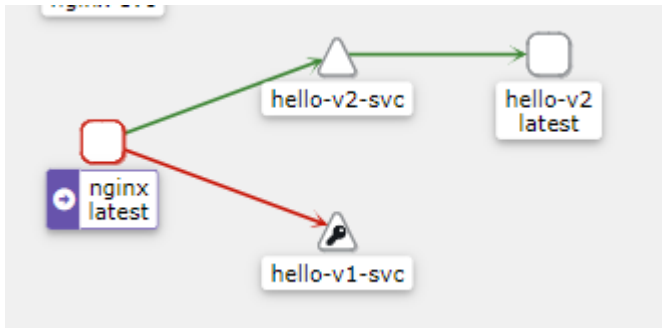
Esto lo que hará es que el pod de nginx mande peticiones a los pods hello-v1 y hello-v2.



Y para probar como se ve el trafico cuando hay un error en alguno de los pods, hay que ejecutar el siguiente comando:

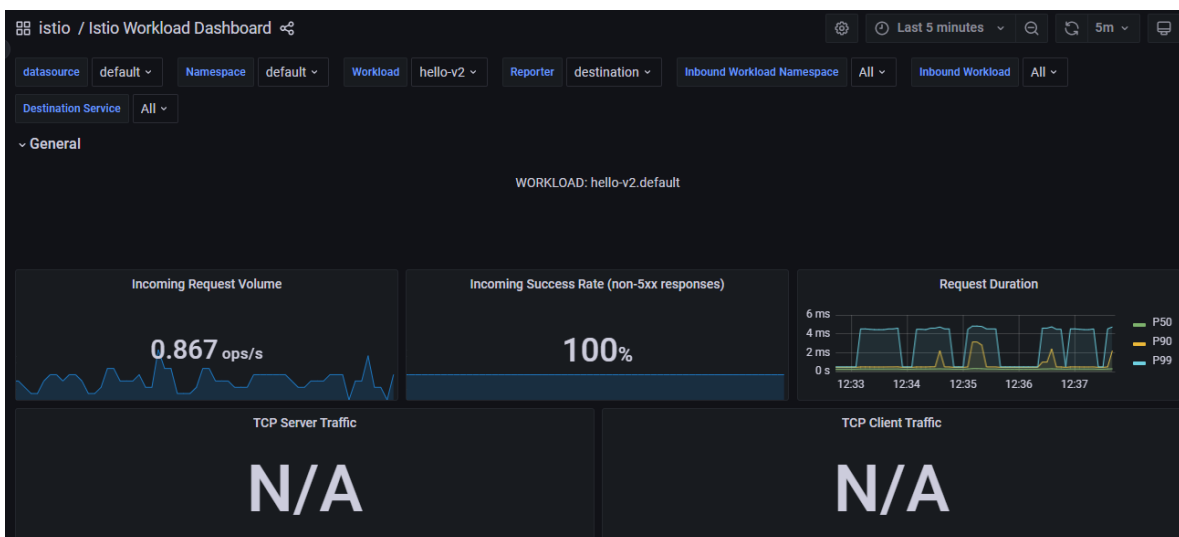
```
PS C:\istio-1.17.2> kubectl delete deployment hello-v1  
deployment.apps "hello-v1" deleted
```

Así es como se ve el trafico cuando el servicio hello-v1 no funciona:



Finalmente ejecutamos el siguiente comando para ver la interfaz de Grafana:

```
PS C:\Users\diego> istioctl dashboard grafana
http://localhost:3000
```



## Conclusión.

Istio es una plataforma de malla de servicios que proporciona funciones de gestión para aplicaciones en contenedores y microservicios. Sus funciones básicas incluyen la gestión de tráfico, la seguridad, la observabilidad y las políticas. Istio nos sirve para administrar aplicaciones de manera más eficiente en entornos de contenedores y microservicios, proporcionando funciones avanzadas para mejorar el rendimiento, la seguridad y la visibilidad. Sin embargo, al ser una herramienta tan completa, si

puede llegar a abrumar su complejidad en un principio. Considero que igual que ha pasado con otras aplicaciones, es una herramienta muy poderosa para ejemplos tan sencillos, por el lado de que no aprovechan todo su potencial.

## Bibliografía.

Paradigma Digital. (2019, 13 marzo). *[Meetup] Microservicios con Istio en OpenShift* [Video]. YouTube. [https://www.youtube.com/watch?v=Qete\\_HitzgQ](https://www.youtube.com/watch?v=Qete_HitzgQ)

Pelado Nerd. (2021, 16 febrero). *Introducción a ISTIO / Service Mesh* [Video]. YouTube. <https://www.youtube.com/watch?v=ofJ5swfP2kQ>