



Ejercicio 11: ISTIO

Materia: Computación tolerante a fallas

Universidad de Guadalajara

Profesor: Michel Emanuel López Franco

6/5/2024

José Maximiliano Díaz Méndez

Introducción

¿Qué es z?

Istio es un service mesh de código abierto que se utiliza para simplificar la gestión de servicios y mejorar la observabilidad y seguridad de las aplicaciones distribuidas.

¿Qué es Service Mesh?

Un service mesh es una infraestructura dedicada que se añade a las aplicaciones distribuidas, como microservicios, para agregar capacidades como gestión de tráfico, observabilidad y seguridad sin modificar el código de la aplicación.

Instalar ISTIO en Mac OS

1. Usando homebrew en Mac OS basta con ejecutar el siguiente comando

```
Maxwell~/projects/computacion-tolerante-a-fallas/actividad-11 o c554cb7|master
> brew install istioctl
==> Downloading https://ghcr.io/v2/homebrew/core/istioctl/manifests/1.21.2
Already downloaded: /Users/max021311/Library/Caches/Homebrew/downloads/089d84fc9f73264e9f3cd6ab4e0afc0fd212939414aa2b2ddab8fe66bea5a851--istioct
l-1.21.2.bottle_manifest.json
==> Fetching istioctl
==> Downloading https://ghcr.io/v2/homebrew/core/istioctl/blobs/sha256:60b9d222dc8b951b558b13c55b374b593759067d261e7bfbbe50423a1cd3dc68
Already downloaded: /Users/max021311/Library/Caches/Homebrew/downloads/9f1b27dc9437edb0818310f8724b322361e6bafaeb708be1dfcb6da45fc3cc62--istioct
l-1.21.2.arm64_sonoma.bottle.tar.gz
==> Pouring istioctl--1.21.2.arm64_sonoma.bottle.tar.gz
==> Caveats
zsh completions have been installed to:
  /opt/homebrew/share/zsh/site-functions
==> Summary
📦 /opt/homebrew/Cellar/istioctl/1.21.2: 96 files, 93.2MB
==> Running 'brew cleanup istioctl'...
Disable this behaviour by setting HOMEBREW_NO_INSTALL_CLEANUP.
Hide these hints with HOMEBREW_NO_ENV_HINTS (see 'man brew').
```

2. Ahora para instalarlo en el cluster de Kubernetes sería con el siguiente comando

```
Maxwell~/projects/computacion-tolerante-a-fallas/actividad-11 o c554cb7|master
> istioctl install --set profile=demo -y
✓ Istio core installed
✓ Istiod installed
✓ Egress gateways installed
✓ Ingress gateways installed
✓ Installation complete
Made this installation the default for injection and validation.
```

3. Comprobamos que se haya instalado obteniendo los namespaces del Cluster. Donde podemos observar que existe el namespace istio-system.

```
Maxwell~/projects/computacion-tolerante-a-fallas/actividad-11 o c554cb7|master
> kubectl get ns
NAME                STATUS    AGE
default              Active    9m17s
ingress-nginx        Active    9m12s
istio-system         Active    72s
kube-node-lease      Active    9m17s
kube-public          Active    9m17s
kube-system          Active    9m17s
kubernetes-dashboard Active    9m7s
```

4. Habilitamos la inyección de ISTIO a los pods con el siguiente comando que añade la etiqueta istio-injection con el valor enabled al namespace por defecto.

```
Maxwell~/projects/computacion-tolerante-a-fallas/actividad-11 o c554cb7|master
> kubectl label namespace default istio-injection=enabled
namespace/default labeled
```

5. Podemos observar la inyección de ISTIO en los pods con el siguiente comando que nos muestra que por cada pod hay dos contenedores, uno siendo el propio contenedor del pod y el otro el inyectado por ISTIO para monitorizar.

```
Maxwell~/projects/computacion-tolerante-a-fallas/actividad-11 o c554cb7|master
> kubectl get pods
NAME                                READY    STATUS    RESTARTS    AGE
nodejs-app-bb588dbd-bxs9l          0/2     PodInitializing    0            1s
nodejs-app-bb588dbd-d7rhj          0/2     PodInitializing    0            1s
nodejs-app-bb588dbd-pzd82          0/2     PodInitializing    0            1s
```

6. Para verlo con mas detalle podemos describir cualquiera de los pods anteriormente listados y ver los contenedores que esta ejecutando usando el comando `kubectl describe pod nodejs-app-bb588dbd-bxs9l`.

```
Containers:
  nodejs-app:
    Container ID: docker://c15e61d8535ed910f90aa5fd12ccda62b13b
    Image: jose9348/node-app:1.0
    Image ID: docker-pullable://jose9348/node-app@sha256:90
    Port: 3000/TCP
    Host Port: 0/TCP
    State: Running
      Started: Sun, 05 May 2024 21:10:26 -0600
    Ready: True
    Restart Count: 0
    Environment:
      PREFIX: /api
    Mounts:
      /var/run/secrets/kubernetes.io/serviceaccount from kube-api
  istio-proxy:
    Container ID: docker://57b022e6fe956f783f66c2c33571420af0791
    Image: docker.io/istio/proxyv2:1.21.2
    Image ID: docker-pullable://istio/proxyv2@sha256:15f2457
    Port: 15090/TCP
    Host Port: 0/TCP
    Args:
```

7. Instalar e iniciar el dashboard Kiali para ISTIO.

```
Maxwell~/projects/computacion-tolerante-a-fallas/actividad-11 o c554cb7|master ⚡
> kubectl apply -f https://raw.githubusercontent.com/istio/istio/release-1.21/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
```

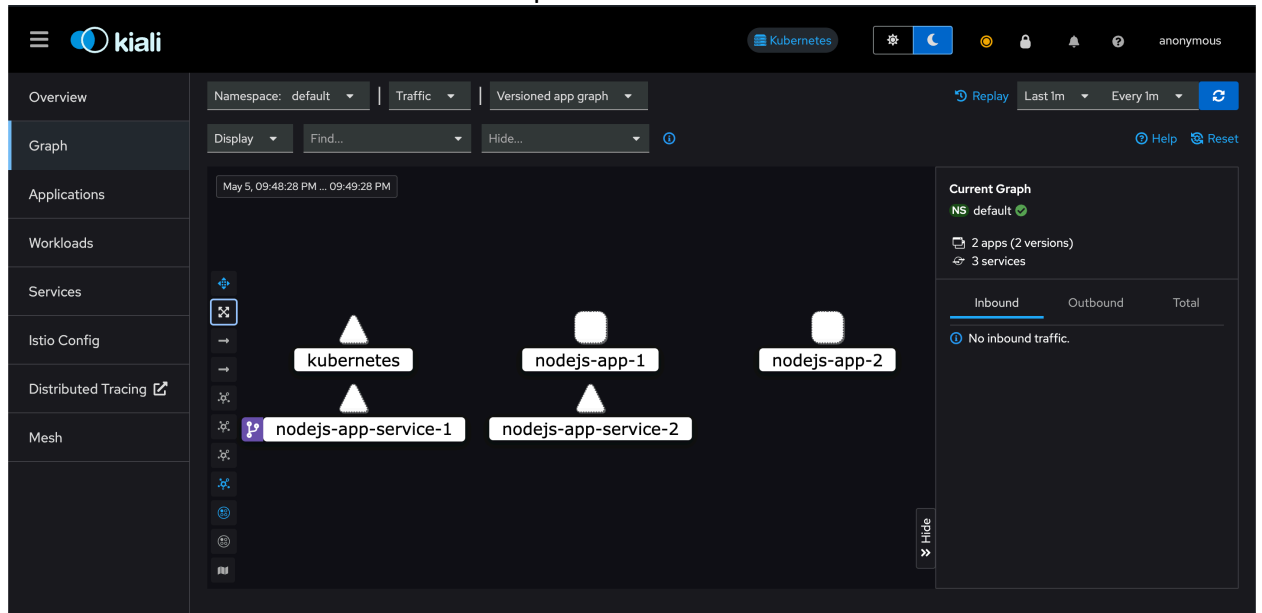
8. Instalar e iniciar Prometheus.

```
Maxwell~/projects/javascript/ctf-final-project o 107df7f|master ⚡
> kubectl apply -f https://raw.githubusercontent.com/istio/istio/release-1.21/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
```

9. Abrir el dashboard de Kiali con el siguiente comando que abra una pestaña automaticamente en el navegador.

```
deployment.apps/kiali created
Maxwell~/projects/computacion-tolerante-a-fallas/actividad-11 o c554cb7|master ⚡
> istioctl dashboard kiali
http://localhost:20001/kiali
```

10. En el dashboard de Kiali si nos vamos a la sección Graph podemos ver un gráfico del tráfico de los servicios en el cluster. Que por ahora esta en blanco



11. Obtener puertos de los servicios.

```
Maxwell~/projects/computacion-tolerante-a-fallas/actividad-11 o c554cb7|master ⚡
> kubectl get service
```

NAME	TYPE	CLUSTER-IP	EXTERNAL-IP	PORT(S)	AGE
kubernetes	ClusterIP	10.96.0.1	<none>	443/TCP	46m
nodejs-app-service-1	ClusterIP	10.97.16.30	<none>	80/TCP	6m40s
nodejs-app-service-2	ClusterIP	10.105.31.152	<none>	80/TCP	6m40s

12. Acceder a un contenedor de alguno de los servicios.

```
Maxwell~/projects/computacion-tolerante-a-fallas/actividad-11 o c554cb7|master ⚡
> kubectl get pods
```

NAME	READY	STATUS	RESTARTS	AGE
nodejs-app-1-85bd7b7994-nhffr	2/2	Running	0	15m
nodejs-app-2-564f979bd-x6jtx	2/2	Running	0	6m25s

```
Maxwell~/projects/computacion-tolerante-a-fallas/actividad-11 o c554cb7|master ⚡
> kubectl exec -it nodejs-app-1-85bd7b7994-nhffr sh
kubectl exec [POD] [COMMAND] is DEPRECATED and will be removed in a future version
/usr/src/app #
```

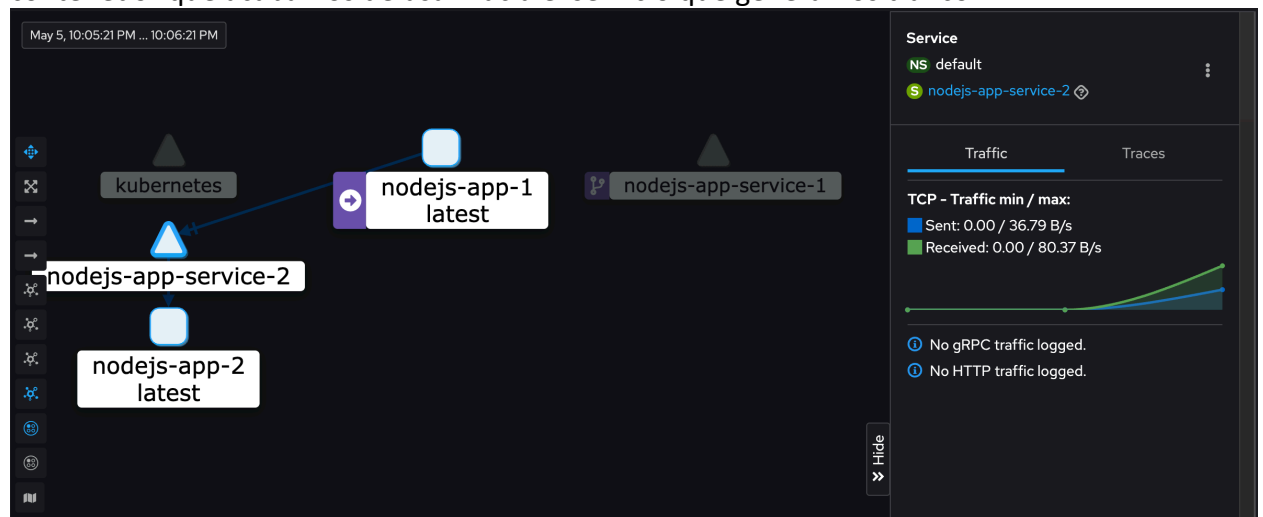
13. Generar tráfico desde un contenedor de un servicio hacia otro servicio.

```

Maxwell~/projects/computacion-tolerante-a-fallas/actividad-11 ○ c554cb7|master ⚡
> kubectl exec -it nodejs-app-1-85bd7b7994-nhffr sh
kubectl exec [POD] [COMMAND] is DEPRECATED and will be removed in a future version.
/usr/src/app # apk add --update curl
fetch https://dl-cdn.alpinelinux.org/alpine/v3.19/main/aarch64/APKINDEX.tar.gz
fetch https://dl-cdn.alpinelinux.org/alpine/v3.19/community/aarch64/APKINDEX.tar.gz
OK: 15 MiB in 25 packages
/usr/src/app # curl -w '\n' http://nodejs-app-service-2:80/api/v2/ok
Hi from nodejs-app-2-564f979bd-x6jtx
/usr/src/app #

```

14. Si regresamos al gráfico de tráfico en Kiali podemos observar como hay tráfico desde el contenedor que acabamos de usar hacia el servicio que generamos tráfico.



15. Reemplazar Ingress con Gateway de ISTIO. Para lo que primero necesitamos crear un Gateway el cuál recibirá el tráfico al que después vincularemos un servicio virtual que gestionara el ruteo del tráfico a los servicios.

```
apiVersion: networking.istio.io/v1alpha3
kind: VirtualService
metadata:
  name: my-virtual-service
spec:
  hosts:
  - "*"
  gateways:
  - my-gateway
  http:
  - match:
    - uri:
        prefix: /api/v1
      route:
      - destination:
          host: nodejs-app-service-1
          port:
            number: 80
    - match:
      - uri:
          prefix: /api/v2
        route:
        - destination:
            host: nodejs-app-service-2
            port:
              number: 80
  ---
apiVersion: networking.istio.io/v1alpha3
kind: Gateway
metadata:
  name: my-gateway
spec:
  # The selector matches the ingress gateway pod labels.
  # If you installed Istio using Helm following the standard
  # documentation, this would be "istio=ingress"
  selector:
    istio: ingressgateway # use istio default controller
  servers:
  - port:
      number: 8080
      name: http
      protocol: HTTP
    hosts:
    - "*"

```

16. Probar el funcionamiento de Gateway

```
Maxwell~/projects/computacion-tolerante-a-fallas/actividad-11 ○ c554cb7|master ⚡  
> curl http://127.0.0.1/api/v2/ok  
Hi from nodejs-app-2-564f979bd-x6jtx%  
Maxwell~/projects/computacion-tolerante-a-fallas/actividad-11 ○ c554cb7|master ⚡  
> curl http://127.0.0.1/api/v1/ok  
Hi from nodejs-app-1-85bd7b7994-nhffr%
```


Conclusión

Me costo mas de lo que esperaba poder hacer funcionar correctamente ISTIO ya que no sabía que no venía preinstalado Kiali ni Prometheus para poder monitorizar el tráfico. Por lo demás fue fácil con la experiencia previa con Kubernetes de las actividades pasadas.

Referencias

The Istio service mesh. (s. f.). Istio. <https://istio.io/latest/about/service-mesh/>