Ejercicio 11: ISTIO

Materia: Computación tolerante a fallas

Universidad de Guadalajara

Profesor: Michel Emanuel López Franco

6/5/2024

José Maximiliano Díaz Méndez

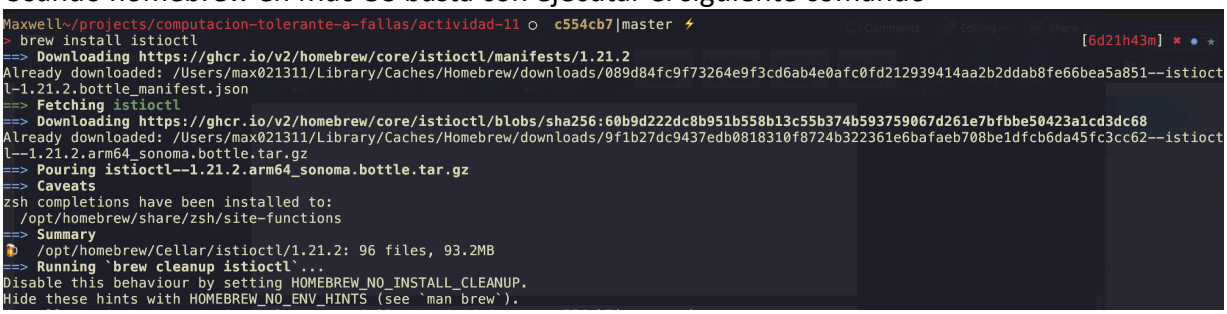
Introducción

¿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.

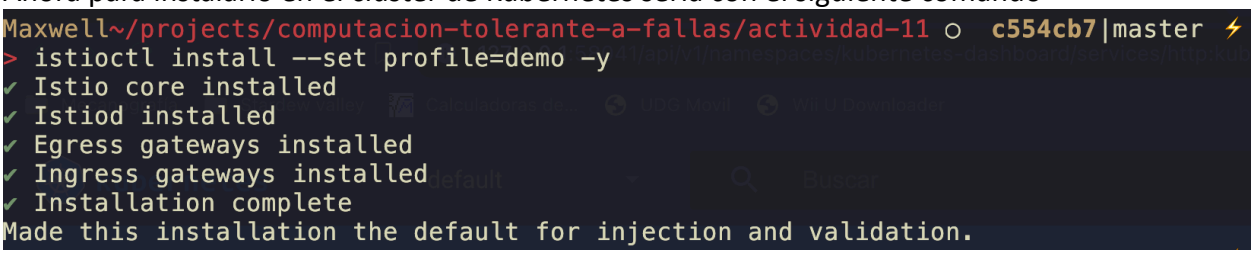
¿Qué es Service Mesh?
Un service mesh es una infraestructura dedicada que se añade a las aplicaciones distribuidas, como microservicios, para agregar capacidades como gestión de tráfico, observabilidad y seguridad sin modificar el código de la aplicación.

Instalar ISTIO en Mac OS

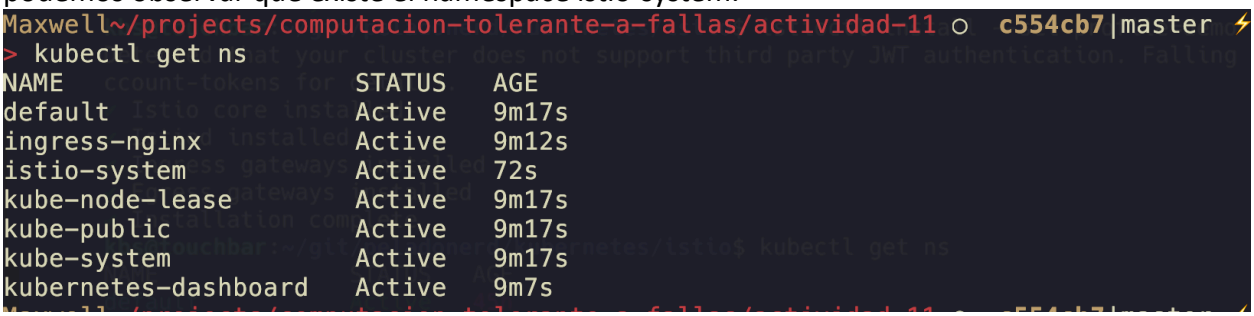
1. Usando homebrew en Mac OS basta con ejecutar el siguiente comando



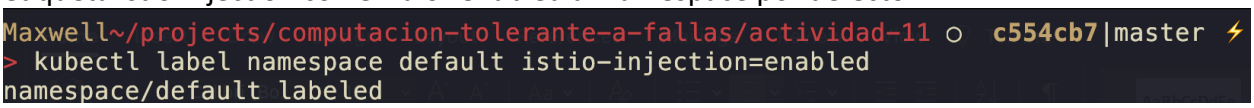
2. Ahora para instalarlo en el cluster de Kubernetes sería con el siguiente comando



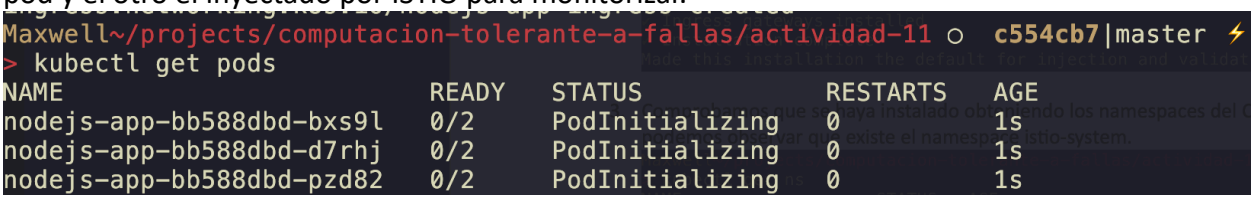
3. Comprobamos que se haya instalado obteniendo los namespaces del Cluster. Donde podemos observar que existe el namespace istio-system.



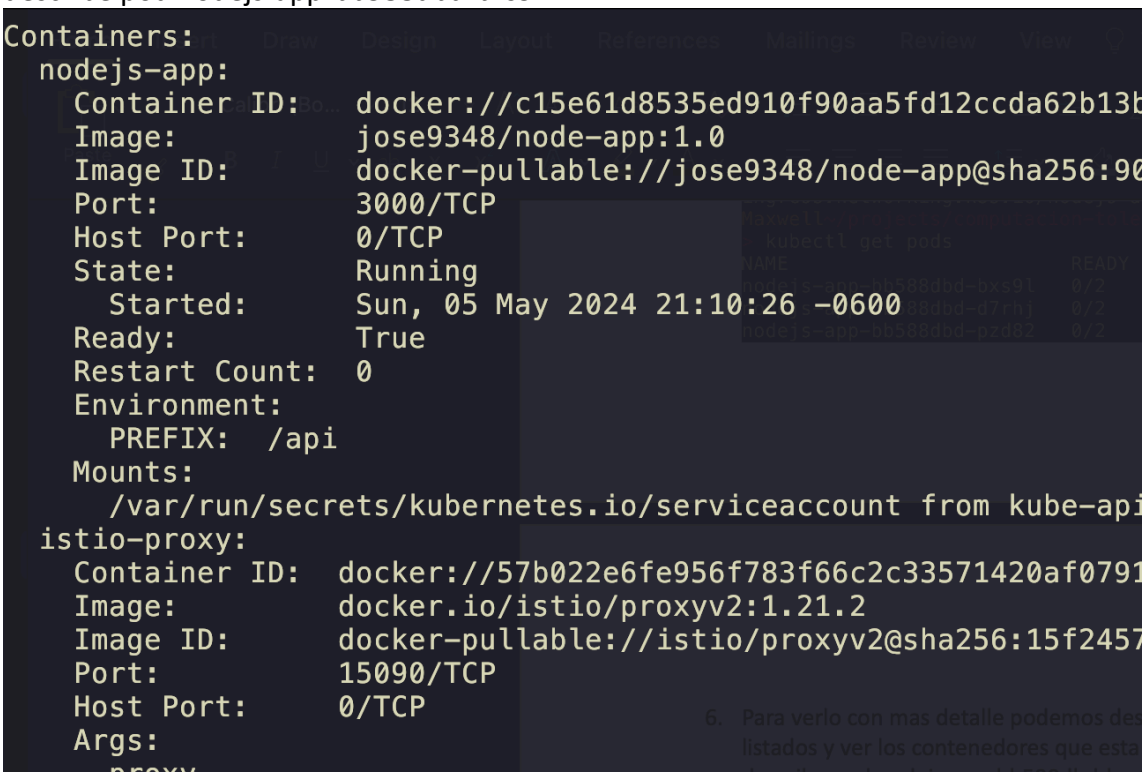
4. Habilitamos la inyección de ISTIO a los pods con el siguiente comando que añade la etiqueta istio-injection con el valor enabled al namespace por defecto.



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.

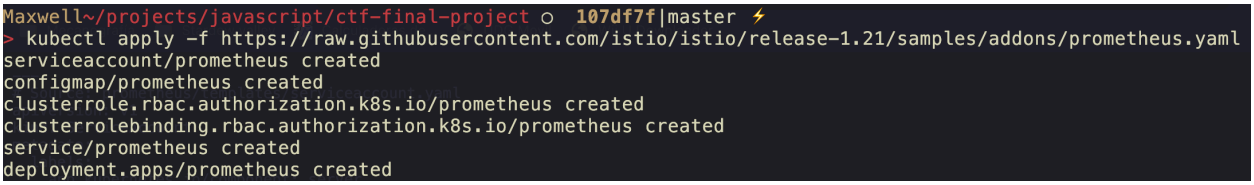


6. Para verlo con mas detalle podemos describir cualquiera de los pods anteriormente listados y ver los contenedores que esta ejecutando usando el comando `kubectl describe pod nodejs-app-bb588dbd-bxs9l`.



7. Instalar e iniciar el dashboard Kiali para ISTIO.

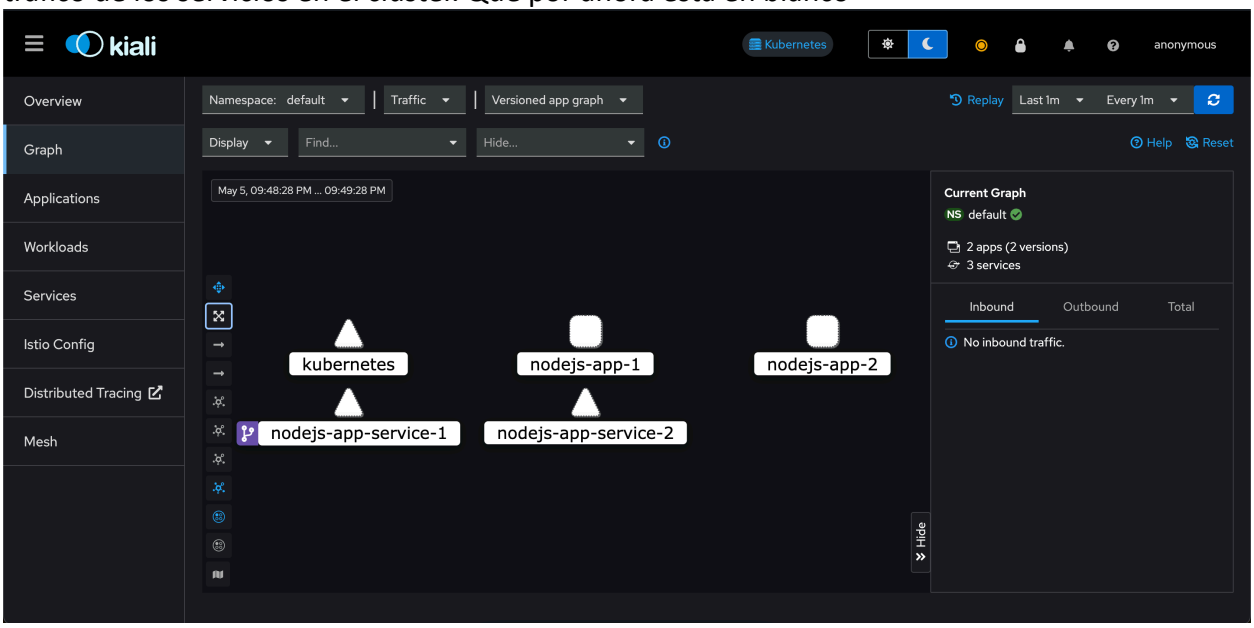
8. Instalar e iniciar Prometheus.



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

```
Maxwell~/projects/computacion-tolerante-a-fallas/actividad-11 ○  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 ○  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
```

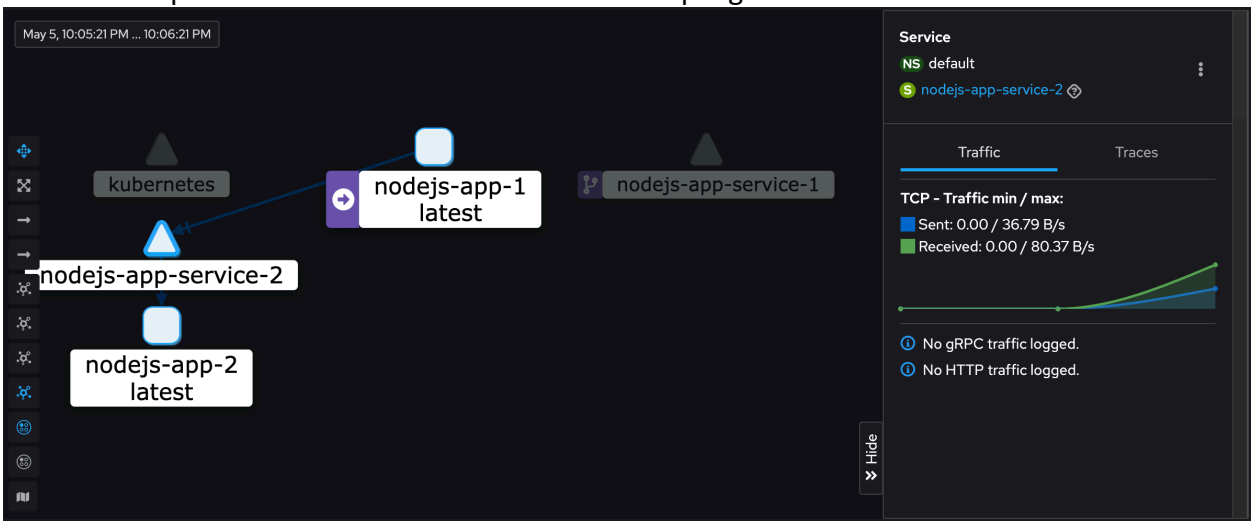
12. Acceder a un contenedor de alguno de los servicios.

```
Maxwell~/projects/computacion-tolerante-a-fallas/actividad-11 ○  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 ○  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 regresamo 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 recibira el tráfico al que después vincularemos un servicio virtual que gestionara el ruteo del tráfico a los servicios.

```yaml
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/