



Universidad de Guadalajara
Centro Universitario de Ciencias Exactas e Ingenierías
División de Tecnologías para la Integración Ciber-Humana
Departamento de Ciencias Computacionales
Ingeniería en Computación



ISTIO

Ejemplo de uso de Istio.

Alumno

Hernández Cortez Kevin Uriel.

217734547.

Materia

Computación Tolerante a Fallas.

D06

Profesor

Lopez Franco Michel Emanuel.

Fecha de entrega

Lunes 06 de mayo de 2024.

Introducción

En esta tarea, se podrá ver el cómo desplegar una aplicación de Python en Kubernetes utilizando Istio como service mesh. Istio da una manera de conectar, asegurar y controlar el tráfico entre los microservicios de una aplicación, mejorando así la gestión de servicios en un entorno de contenedores.

Desarrollo

Códigos

```
- APP.PY
    from flask import Flask
    app = Flask(__name__)

    @app.route('/')
    def hello_world():
        return '¡Hola, Mundo!'

    if __name__ == '__main__':
        app.run(debug=True, host='0.0.0.0', port=5000)

- DOCKERFILE
    FROM python:3.9-slim

    WORKDIR /app

    COPY requirements.txt requirements.txt
    RUN pip install -r requirements.txt

    COPY . .

    CMD [ "python", "app.py" ]

- REQUIREMENTS.TXT
    flask

- APP.YAML
    apiVersion: apps/v1
    kind: Deployment
    metadata:
      name: app-flask
    spec:
      replicas: 1
      selector:
        matchLabels:
          app: app-flask
      template:
        metadata:
          labels:
            app: app-flask
        spec:
```

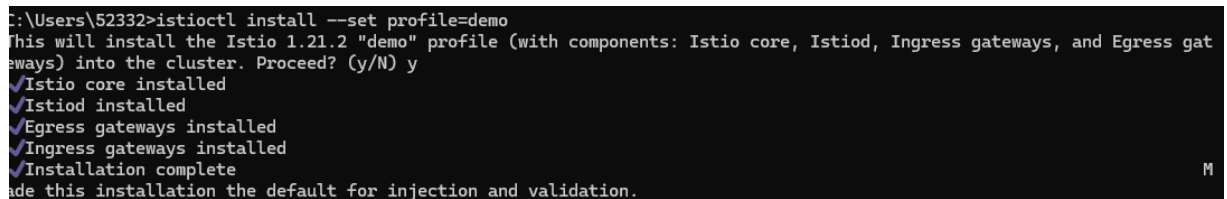
```

    containers:
      - name: app-flask
        image: tu_usuario/app-flask:latest
        ports:
          - containerPort: 5000
    ---
    apiVersion: v1
    kind: Service
    metadata:
      name: app-flask
    spec:
      ports:
        - port: 80
          targetPort: 5000
      selector:
        app: app-flask
- INGRESS.YAML
  apiVersion: networking.k8s.io/v1
  kind: Ingress
  metadata:
    name: app-ingress
  spec:
    rules:
      - host: tu-aplicacion.com
        http:
          paths:
            - path: /
              pathType: Prefix
          backend:
            service:
              name: app-flask
              port:
                number: 80

```

Explicación de programa

Instalación de ISTIO



```

C:\Users\52332>istioctl install --set profile=demo
This will install the Istio 1.21.2 "demo" profile (with components: Istio core, Istiod, Ingress gateways, and Egress gateways) into the cluster. Proceed? (y/N) y
✓ Istio core installed
✓ Istiod installed
✓ Egress gateways installed
✓ Ingress gateways installed
✓ Installation complete
To make this installation the default for injection and validation.

```

Primeramente, se instala Istio en mi clúster de Kubernetes ejecutando el comando *istioctl install --set profile=demo*

Despliegue de aplicación

```
app.py > ...
1  from flask import Flask
2  app = Flask(__name__)
3
4  @app.route('/')
5  def hello_world():
6      return '¡Hola, Mundo!'
7
8  if __name__ == '__main__':
9      app.run(debug=True, host='0.0.0.0', port=5000)
10
```

Se crea el archivo en Python de la implementación de la aplicación (con un ejemplo tipo “Hola Mundo”).

```
Dockerfile
1  FROM python:3.9-slim
2
3  WORKDIR /app
4
5  COPY requirements.txt requirements.txt
6  RUN pip install -r requirements.txt
7
8  COPY . .
9
10 CMD [ "python", "app.py" ]
11
```

Se crea el Dockerfile para contener la aplicación.

```
requirements.txt
1  flask
```

También se crea un txt para los requisitos de la aplicación Flask.

```
C:\Users\52332\Desktop\EjemploIstio>docker build -t kevinhc47/app-flask:latest .
[+] Building 0.7s (10/10) FINISHED
=> [internal] load build definition from Dockerfile                                docker:default
=> => transferring dockerfile: 198B                                              0.0s
=> [internal] load metadata for docker.io/library/python:3.9-slim                0.4s
=> [internal] load .dockerignore                                                 0.0s
=> => transferring context: 2B                                                  0.0s
=> [1/5] FROM docker.io/library/python:3.9-slim@sha256:44122e46edb1c3ae2a144778db3e01c78b6de3af20ddcc38d43032dec 0.0s
=> [internal] load build context                                                0.0s
=> => transferring context: 93B                                                0.0s
=> CACHED [2/5] WORKDIR /app                                                    0.0s
=> CACHED [3/5] COPY requirements.txt requirements.txt                          0.0s
=> CACHED [4/5] RUN pip install -r requirements.txt                             0.0s
=> CACHED [5/5] COPY . .                                                        0.0s
=> exporting to image                                                            0.0s
=> => exporting layers                                                            0.0s
=> => writing image sha256:c73112aef53edb6da3202cbdbcc554b2727d20745c7e3d860f776ebf24a65b4 0.0s
=> => naming to docker.io/kevinhc47/app-flask:latest                          0.0s

View build details: docker-desktop://dashboard/build/default/default/s287w0hcj3h0hhvks94vfb4x5

What's Next?
View a summary of image vulnerabilities and recommendations → docker scout quickview
```

kevinhc47 / app-flask

Contains: Image • Last pushed: 38 minutes ago

Security unknown ☆ 0 2 Public

Se crea y se sube a Docker Hub la imagen.

Configuración de Istio

```
C:\Users\52332\Desktop\EjemploIstio>kubectl label namespace default istio-injection=enabled
namespace/default labeled
```

Se aplica la inyección lateral de Istio al espacio de nombres

```
C:\Users\52332\Desktop\EjemploIstio>kubectl apply -f app.yaml
deployment.apps/app-flask created
service/app-flask created
```

Se aplica la configuración de Kubernetes.

```
C:\Users\52332\Desktop\EjemploIstio>kubectl apply -f https://raw.githubusercontent.com/istio/istio/release-1.11/samples/bookinfo/networking/bookinfo-gateway.yaml
gateway.networking.istio.io/bookinfo-gateway created
virtualservice.networking.istio.io/bookinfo created
```

Se crea una puerta de enlace Istio.

```
C:\Users\52332\Desktop\EjemploIstio>kubectl get svc istio-ingressgateway -n istio-system
NAME                                TYPE           CLUSTER-IP      EXTERNAL-IP      PORT(S)
istio-ingressgateway               LoadBalancer   10.98.114.251    <pending>        15021:32238/TCP,80:31686/TCP,443:31693/TCP,31400:31512/TCP,15443:30775/TCP
12m                                6m46s
```

Finalmente, se verifica la dirección IP externa del gateway.

Conclusión

ISTIO

Al finalizar esta tarea, he aprendido cómo configurar Istio para una aplicación, acceder a ella a través de Istio y aprovechar las características que Istio tiene para ofrecerme, como service mesh. Este conocimiento puede servirme para construir y gestionar aplicaciones más robustas y escalables en entornos de contenedores.

Bibliografía.

- Paradigma Digital. (2019, 13 marzo). *[Meetup] Microservicios con Istio en OpenShift* [Vídeo]. YouTube. https://www.youtube.com/watch?v=Qete_HitzgQ
- Pelado Nerd. (2021, 16 febrero). *Introducción a ISTIO / Service Mesh* [Vídeo]. YouTube. <https://www.youtube.com/watch?v=ofJ5swfP2kQ>
- *Kubernetes service external ip pending*. (s. f.). Stack Overflow.
<https://stackoverflow.com/questions/44110876/kubernetes-service-external-ip-pending>