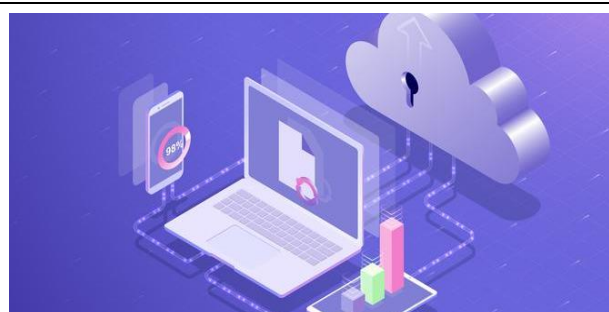




Universidad de Guadalajara
CUCEI – Centro Universitario de Ciencias Exactas e Ingenierías
Mtro. Michel Emanuel López Franco
Computación tolerante a fallas

Reporte 14



**Velasco Hernandez Victor
Manuel**

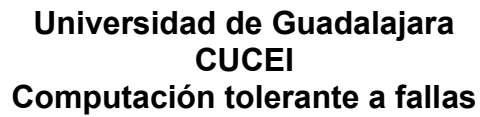
Código: 216598879

| D06 | 2022A |

| 26/Abril/2022 |

Istio





Victor Manuel Velasco Hernández
|Código: 216598879 | | 2022A |
|26/ Abril /2022 | |D06|



Contestar preguntas y generar ejemplo

Es un service mesh (interconexión de aplicaciones que comparten información con otras) que extiende Kubernetes para establecer una programación, redes compatibles con aplicaciones (application aware networking), usando el Envoy service proxy. Trabaja con Kubernetes y los tradicionales workloads, contiene un estándar de manejo de tráfico universal, telemetría y seguridad a complejos despliegues de aplicaciones. [1]-[3]

El manejo de servicios mesh brinda la posibilidad a un sistema de computo o clúster de poder manejar una gran cantidad de procesos de manera planificada e interconectada, lo cuál hace que sea posible la realización de procesos cada vez más complejos con mayor seguridad, lo cuál resulta ser una excelente herramienta para lograr una mejora en el manejo de operaciones realizadas dentro de un clúster.

Instalaremos Istio, con el cuál se correrá el ejemplo del siguiente tutorial: [Istio Setup in Kubernetes | Step by Step Guide to install Istio Service Mesh - YouTube](#) , para lo cuál, es indispensable tener Docker, Kubernetes y Minikube, recordemos que todo esto se debe hacer con permisos de administrador:

```

root@kali: ~/istio-1.13.3
File Actions Edit View Help
root@kali) ~)
# curl -L https://istio.io/downloadIstio | sh -

% Total    % Received % Xferd  Average Speed   Time    Time     Time  Cur
rent                                  Dload  Upload   Total    Spent    Left   Spa
ed
  0      0    0      0     0      0      0      0  0 --:--:-- --:--:-- --:--:--
100    101    0    101     0     0     255     0  0 --:--:-- --:--:-- --:--:--
255
  0      0    0      0     0      0      0      0  0 --:--:-- 00:00:01 --:--:--
  0      0    0      0     0      0      0      0  0 --:--:-- 00:00:02 --:--:--
100   4926   100   4926     0     0    1856     0  0 0:00:02 0:00:02 --:--:-- 4
848

```



Procedemos a configurar la capacidad del clúster de Kubernetes, con el comando “*minikube start --cpus 4 --memory 10000*” :

```
kali@kali: ~  
File Actions Edit View Help  
Removed all traces of the "minikube" cluster.  
  
(kali@kali)-[~]  
$ minikube start --cpus 4 --memory 10000  
minikube v1.25.2 on Debian kali-rolling (vbox/amd64)  
Automatically selected the docker driver  
Your cgroup does not allow setting memory.  
  More information: https://docs.docker.com/engine/install/linux-postinstall/#your-kernel-does-not-support-cgroup-s  
Starting control plane node minikube in cluster minikube  
Pulling base image ...  
Creating docker container (CPUs=4, Memory=10000MB) ...  
Preparing Kubernetes v1.23.3 on Docker 20.10.12 ...  
  kubelet.housekeeping-interval=5m  
  Generating certificates and keys ...  
  Booting up control plane ...  
  Configuring RBAC rules ...  
Verifying Kubernetes components ...  
  Using image gcr.io/k8s-minikube/storage-provisioner:v5  
Enabled addons: default-storageclass, storage-provisioner  
Done! kubectrl is now configured to use "minikube" cluster and "default" namespace by default
```

Ahora, con permisos de usuario estándar definimos la ruta de la carpeta “bin” del instalador de Istio para empezar a ejecutarlo con comando “*istioctl install*”, esto hará que se despliegue en contenedores, directamente orquestados por Kubernetes:

```
kali@kali: ~  
File Actions Edit View Help  
  
(kali@kali)-[~]  
$ export PATH=$PATH:/home/kali/Downloads/istio-1.13.3/bin/  
  
(kali@kali)-[~]  
$ istioctl install  
This will install the Istio 1.13.3 default profile with ["Istio core" "Istiod"  
✓ Istio core installed  
✓ Istiod installed  
✓ Ingress gateways installed  
✓ Installation complete  
Making this installation the default for injection and validation.  
  
Thank you for installing Istio 1.13. Please take a few minutes to tell us ab
```



Y con este comando “*kubectl get ns*” podemos ver los Nodos que Istio ha agregado a Kubernetes:

```
kali@kali: ~  
File Actions Edit View Help  
node-web-app-5f7b45b8f4-gmjp6 1/1 Running 2 (24m ago) 3d21h  
  
(kali@kali)-[~]  
$ kubectl get ns  
  
NAME STATUS AGE  
default Active 3d21h  
istio-system Active 16m  
kube-node-lease Active 3d21h  
kube-public Active 3d21h  
kube-system Active 3d21h  
kubernetes-dashboard Active 3d21h  
  
(kali@kali)-[~]  
$
```

Ahora, procederemos a introducir los Microservicios de nuestra aplicación en Kubernetes, esto con el comando “*kubectl apply -f <archivo>*” a nuestro manifiesto de Kubernetes:

The screenshot shows a file manager window with two YAML files: `istio-manifests.yaml` and `kubernetes-manifests.yaml`. The terminal window shows the command `kubectl apply -f kubernetes-manifests.yaml` being executed, resulting in the creation of various services and deployments.

```
(kali@kali)-[~/Documents/microservices-demo/release]  
$ kubectl apply -f kubernetes-manifests.yaml  
deployment.apps/emailservice created  
service/emailservice created  
deployment.apps/checkoutservice created  
service/checkoutservice created  
deployment.apps/recommendationservice created  
service/recommendationservice created  
deployment.apps/frontend created  
service/frontend created  
service/frontend-external created  
deployment.apps/paymentservice created  
service/paymentservice created  
deployment.apps/productcatalogservice created  
service/productcatalogservice created  
deployment.apps/cartservice created  
service/cartservice created  
deployment.apps/loadgenerator created  
deployment.apps/currencyservice created  
service/currencyservice created  
deployment.apps/shippingservice created  
service/shippingservice created  
deployment.apps/redis-cart created  
service/redis-cart created  
deployment.apps/adservice created  
service/adservice created
```




Lo siguiente es configurar los proxys de cada servicio desplegado, ya que no se encuentran, en la siguiente imagen se muestran los pods desplegados, esto gracias al comando “*kubecttl get pods*”, antes de configurar los proxys debemos de esperar a que los pods estén corriendo correctamente, esto puede tardar más de 10 minutos:

```
kali@kali: ~/ComputacionTolerante/7-Docker/app
File Actions Edit View Help
└─$ kubecttl get pods
NAME                                READY    STATUS    RESTARTS    AGE
adservice-75656d5f44-6tr7s          1/1      Running   1 (10m ago)  25m
cartservice-8c64564d4-lx5jh         1/1      Running   3 (97s ago)  26m
checkoutservice-5d45565464-5m7lg    1/1      Running   7 (5m20s ago)  26m
currencyservice-7dc56c8-kbdz5        1/1      Running   2 (11m ago)  26m
emailservice-67b75bf988-qbwtdg      1/1      Running   5 (11m ago)  26m
frontend-5db5d7b788-bj7zq           1/1      Running   1 (20m ago)  26m
loadgenerator-77bc9cbc96-lwvkz       1/1      Running   0            26m
paymentservice-6f69f8b58d-bps7f      1/1      Running   6 (5m15s ago)  26m
productcatalogservice-67f5c88476-cwl8r 1/1      Running   7 (5m19s ago)  26m
recommendationservice-7ddd87dccc-ndc67 1/1      Running   6 (10m ago)  26m
redis-cart-78746d49dc-nfz5v          1/1      Running   0            26m
shippingservice-55bd6c45bb-gwdzp      1/1      Running   3 (10m ago)  26m

(kali@kali)-[~/ComputacionTolerante/7-Docker/app]
└─$
```

Para activar los proxys necesitamos crear una etiqueta para “istio-injection” con el comando “*kubecttl label namespace default istio-injection=enabled*”:

```
kali@kali: ~
File Actions Edit View Help
paymentservice-6f69f8b58d-bps7f 1/1 Running 6 (67s ago) 21m
productcatalogservice-67f5c88476-cwl8r 0/1 CrashLoopBackOff 6 (71s ago) 21m
recommendationservice-7ddd87dccc-ndc67 1/1 Running 6 (6m37s ago) 21m
redis-cart-78746d49dc-nfz5v 1/1 Running 0 21m
shippingservice-55bd6c45bb-gwdzp 1/1 Running 3 (6m50s ago) 21m

(kali@kali)-[~]
└─$ kubecttl label namespace default istio-injection=enabled
namespace/default labeled

(kali@kali)-[~]
└─$ kubecttl get ns default --show-labels
NAME    STATUS    AGE    LABELS
default Active    3d23h  istio-injection=enabled,kubernetes.io/metadata.name=default

(kali@kali)-[~]
└─$
```



Tras esto, procedemos a eliminar los pods establecidos en el archivo YAML y los volvemos a cargar:

```
kali@kali: ~/Documents/microservices-demo/release
File Actions Edit View Help
(kali@kali)~[~/Documents/microservices-demo/release]
$ kubectl delete -f kubernetes-manifests.yaml
deployment.apps "emailservice" deleted
service "emailservice" deleted
deployment.apps "checkoutservice" deleted
service "checkoutservice" deleted
deployment.apps "recommendationservice" deleted
service "recommendationservice" deleted
deployment.apps "frontend" deleted
service "frontend" deleted
service "frontend-external" deleted
deployment.apps "paymentservice" deleted
service "paymentservice" deleted
deployment.apps "productcatalogservice" deleted
service "productcatalogservice" deleted
deployment.apps "cartservice" deleted
service "cartservice" deleted
deployment.apps "loadgenerator" deleted
deployment.apps "currencyservice" deleted
service "currencyservice" deleted
deployment.apps "shippingservice" deleted
service "shippingservice" deleted
deployment.apps "redis-cart" deleted
service "redis-cart" deleted
deployment.apps "adservice" deleted
service "adservice" deleted

(kali@kali)~[~/Documents/microservices-demo/release]
$ kubectl apply -f kubernetes-manifests.yaml
deployment.apps/emailservice created
service/emailservice created
deployment.apps/checkoutservice created
service/checkoutservice created
deployment.apps/recommendationservice created
```

Al utilizar el comando “*kubectl get pods*” nos podemos dar cuenta que para cada pod hay dos contenedores, hay algunos que no tienen todos listos o están pendientes, esto se debe a limitantes de la CPU del equipo de computo:

```
kali@kali: ~/Documents/microservices-demo/release
File Actions Edit View Help
(kali@kali)~[~/Documents/microservices-demo/release]
$ kubectl get pods
```

NAME	READY	STATUS	RESTARTS	AGE
adservice-75656d5f44-g67m5	0/2	Pending	0	3m
cartservice-8c64564d4-4pvh8	1/2	Running	3 (20s ago)	3m3s
checkoutservice-5d45565464-vhwzk	2/2	Running	1 (12s ago)	3m5s
currencyservice-7dc56c8-trjh8	2/2	Running	2 (65s ago)	3m2s
emailservice-67b75bf988-nsxp4	1/2	Running	1 (2m21s ago)	3m5s
frontend-5db5d7b788-qx4s4	2/2	Running	0	3m4s
loadgenerator-77bc9cbc96-r4kvb	2/2	Running	0	3m2s
paymentservice-6f69f8b58d-5md8p	1/2	Running	2 (7s ago)	3m4s
productcatalogservice-67f5c88476-5h9pf	2/2	Running	2 (50s ago)	3m3s
recommendationservice-7ddd87dccc-qzswj	1/2	Running	2 (108s ago)	3m4s
redis-cart-78746d49dc-vs42s	2/2	Running	0	3m1s
shippingservice-55bd6c45bb-vvtmd	2/2	Running	2 (10s ago)	3m2s



Lo siguiente, es agregar las herramientas de visualización y métricas para poder observar el funcionamiento gráfico de Istio, esto con el comando “*kubectl apply -f samples/addons*” dentro de la carpeta instaladora de Istio:

```
kali@kali: ~/Downloads/istio-1.13.3
File Actions Edit View Help
(kali@kali)-[~/Downloads/istio-1.13.3]
$ kubectl apply -f samples/addons/
serviceaccount/grafana created
configmap/grafana created
service/grafana created
deployment.apps/grafana created
configmap/istio-grafana-dashboards created
configmap/istio-services-grafana-dashboards created
deployment.apps/jaeger created
service/tracing created
service/zipkin created
service/jaeger-collector created
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
serviceaccount/prometheus created
configmap/prometheus created
clusterrole.rbac.authorization.k8s.io/prometheus created
```

Con el siguiente comando “*kubectl get pod -n istio-system*” podemos visualizar los cambios realizados dentro de los pods de Istio:

```
kali@kali: ~/Downloads/istio-1.13.3
File Actions Edit View Help
(kali@kali)-[~/Downloads/istio-1.13.3]
$ kubectl get pod -n istio-system
```

NAME	READY	STATUS	RESTARTS	AGE
grafana-67f5ccd9d7-tq2m8	1/1	Running	0	5m54s
istio-ingressgateway-6dc56fc9f9-56l9f	1/1	Running	0	29m
istiod-8488b9bdc7-9dtlz	1/1	Running	0	29m
jaeger-78cb4f7d4b-b8cq	1/1	Running	0	5m53s
kiali-c946fb5bc-zbvtm	1/1	Running	0	5m47s
prometheus-7cc96d969f-d6bj7	2/2	Running	0	5m44s



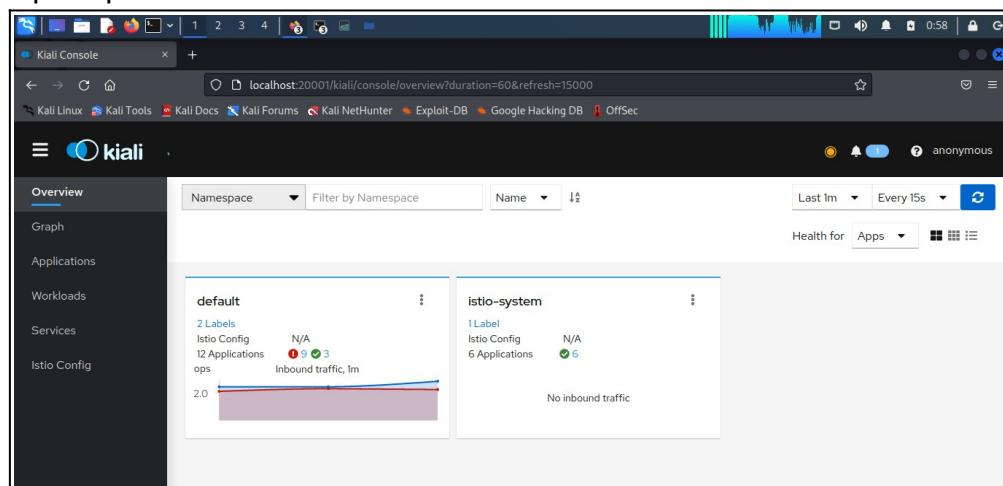
Lo siguiente es introducir el comando `"kubectl get svc -n istio-system"` para ver los puertos de los servicios de Istio, para después buscar el puerto del servicio Kiali, el cuál es el correspondiente al del Service Mesh de Istio, y después ponerlo en el comando `"kubectl port-forward svc/kiali -n istio-system 20001"`:

```
kali@kali: ~/Downloads/istio-1.13.3
File Actions Edit View Help

(kali@kali)-[~/Downloads/istio-1.13.3]
$ kubectl get svc -n istio-system
NAME                TYPE          CLUSTER-IP    EXTERNAL-IP    PORT(S)
grafana              ClusterIP     10.106.151.91 <none>         3000/TCP
istio-ingressgateway LoadBalancer  10.111.107.117 <pending>      15021:3161
istiod               ClusterIP     10.111.51.184 <none>         15010/TCP,
jaeger-collector     ClusterIP     10.101.232.236 <none>         14268/TCP,
kiali                ClusterIP     10.106.115.1  <none>         20001/TCP,
prometheus           ClusterIP     10.108.21.94  <none>         9090/TCP
tracing              ClusterIP     10.99.189.191 <none>         80/TCP,166
zipkin               ClusterIP     10.108.3.226  <none>         9411/TCP

(kali@kali)-[~/Downloads/istio-1.13.3]
$ kubectl port-forward svc/kiali -n istio-system 20001
Forwarding from 127.0.0.1:20001 -> 20001
Forwarding from [::1]:20001 -> 20001
```

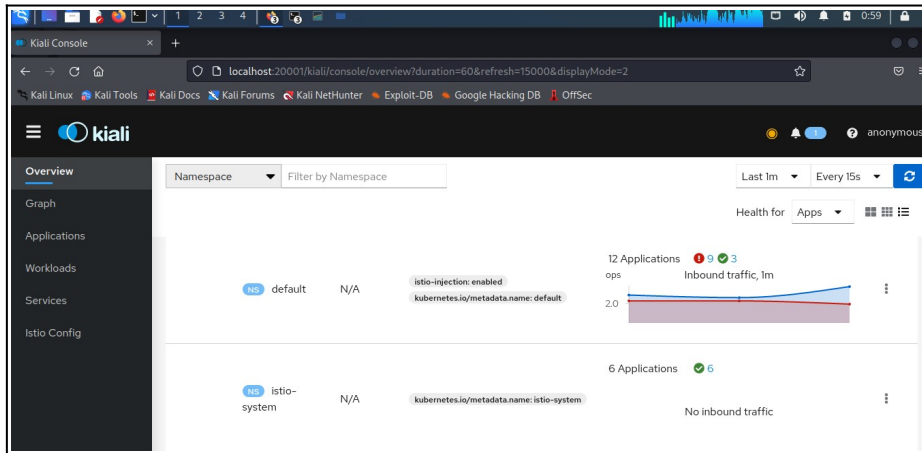
Finalmente, tenemos que ir al enlace de `"localhost:20001"` en el explorador para ver la interfaz de Kiali, como podemos observar, nuestro clúster necesita más recursos de CPU para poder correr todos los contenedores:



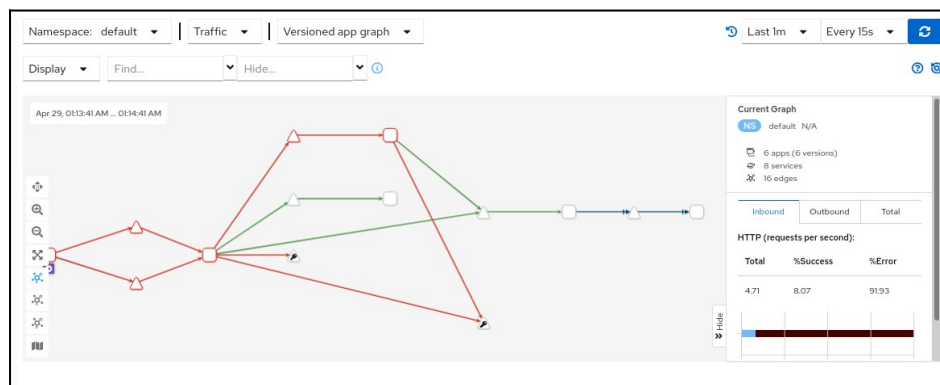
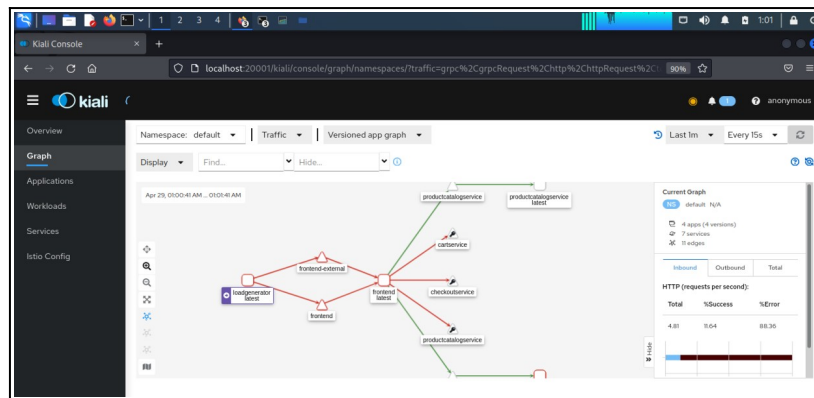


Universidad de Guadalajara
CUCEI
Computación tolerante a fallas

Victor Manuel Velasco Hernández
|Código: 216598879 | | 2022A |
|26/ Abril /2022 | |D06|



Y yendo al apartado de gráfico, se muestra la relación de los pods que tienen unos con otros, lo cuál deja en evidencia el funcionamiento del Service Mesh:



[4]



Universidad de Guadalajara
CUCEI
Computación tolerante a fallas

Victor Manuel Velasco Hernández
|Código: 216598879 | | 2022A |
|26/ Abril /2022 | |D06|

Conclusiones:

El uso de programas que hagan uso del service mesh y Kubernetes resulta ser muy útil para amplificar las capacidades de los orquestadores y poder realizar operaciones complejas dentro de un clúster, esto para desplegar procesos que queramos que se realicen en aplicaciones bajo demanda, ayuda a hacer un sistema tolerante a fallos, por lo que resulta ser fundamental conocer la naturaleza de estas herramientas para desarrollar cada vez, mejores funciones y aplicaciones que garanticen el mejor acceso a la información de los programas, y la presencia de ejecución sin tenerlo en el equipo, lo cual ayuda a tener una mejor comprensión de las eventualidades que se presentan en las computadoras, esto hace que podamos evitar posibles problemas en el futuro.

Link de repositorio:

<https://github.com/Victor012396/ComputacionTolerante.git>

Bibliografía:

- [1] "Istio", Istio. <https://istio.io/latest/> (consultado el 19 de abril de 2022).
- [2] "¿Qué es Istio?" <https://www.redhat.com/es/topics/microservices/what-is-istio> (consultado el 28 de abril de 2022).
- [3] "istio", el 6 de mayo de 2021. <https://www.ibm.com/mx-es/cloud/learn/istio> (consultado el 28 de abril de 2022).
- [4] TechWorld with Nana, Configuración de Istio en Kubernetes | Guía paso a paso para instalar Istio Service Mesh, (el 26 de febrero de 2021). Consultado: el 29 de abril de 2022. [En línea Video]. Disponible en: <https://www.youtube.com/watch?v=voAyroDb6xk>