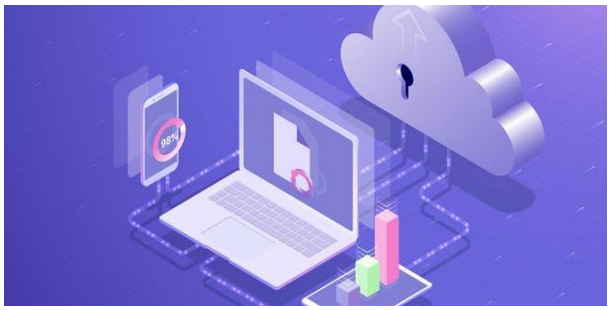
Universidad de Guadalajara

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

| | | |
|---|---|---|
| | **Universidad de Guadalajara<br>CUCEI<br>Computación tolerante a fallas** | **Victor Manuel Velasco Hernández<br>\|Código: 216598879 \| \| 2022A \|<br>\|26/ Abril /2022 \| \|D06\|** |

# Reporte 14-Istio

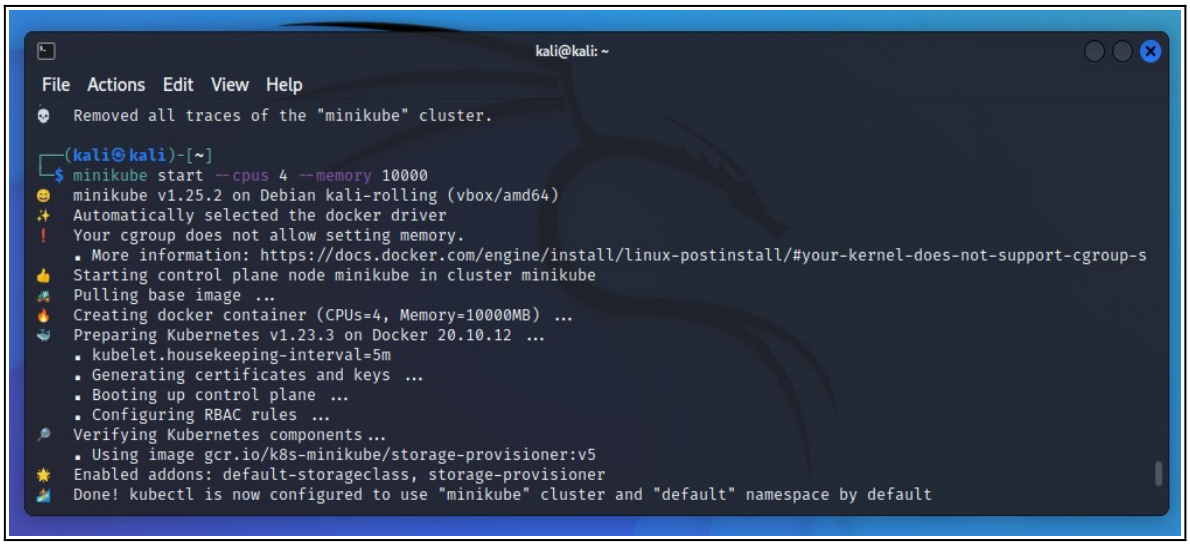
**Objetivo:**
Contestar preguntas y generar ejemplo

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

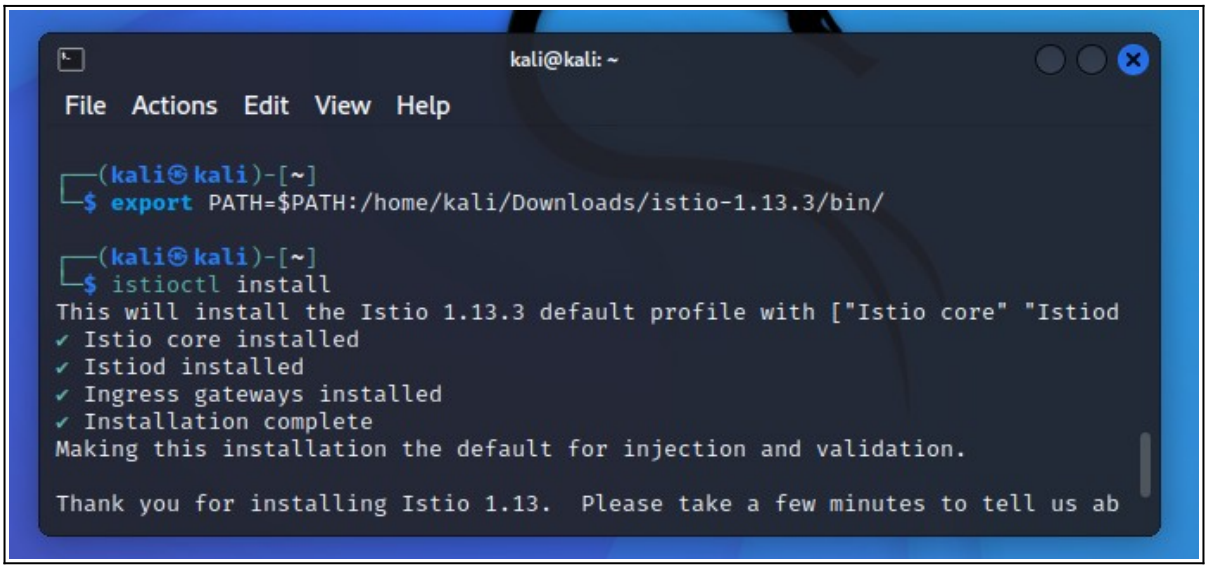
**Introducción:**
El manejo de servicios mesh brinda la posibilidad a un sistema de computo o clúster de poder manejar una gran cantidad de procesos de manera planificada e interconectada, lo cuál hace que sea posible la realización de procesos cada vez más complejos con mayor seguridad, lo cuál resulta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

**Desarrollo:**
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:

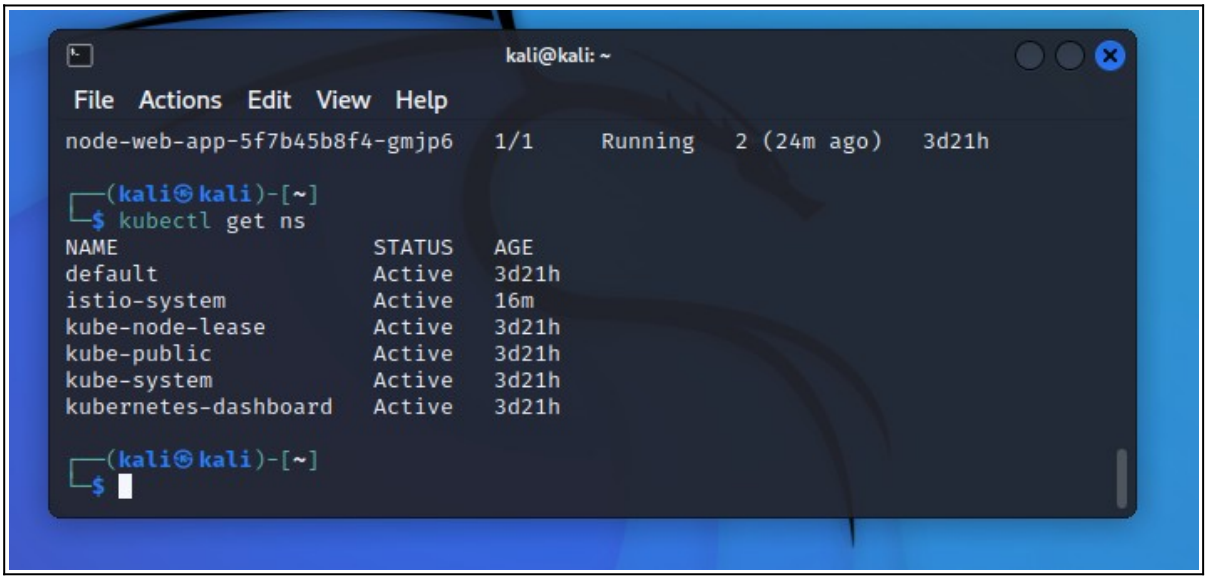
Procedemos a configurar la capacidad del clúster de Kubernetes, con el comando "*minikube start –cpus 4 –memory 10000*" :



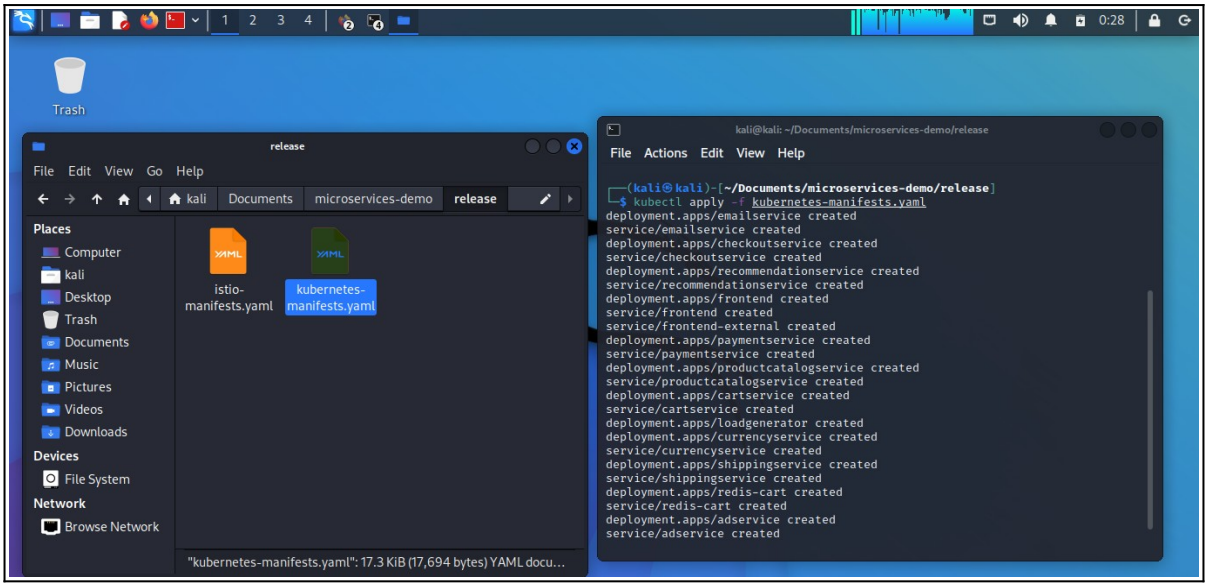

Ahora, con permisos de usuario estándar definimos la ruta de la carpeta "bin" del instalador de Istio para empezar a ejecutarlo con comando "*istioctl install*", esto hará que se despliegue en contenedores, directamente orquestados por Kubernetes:



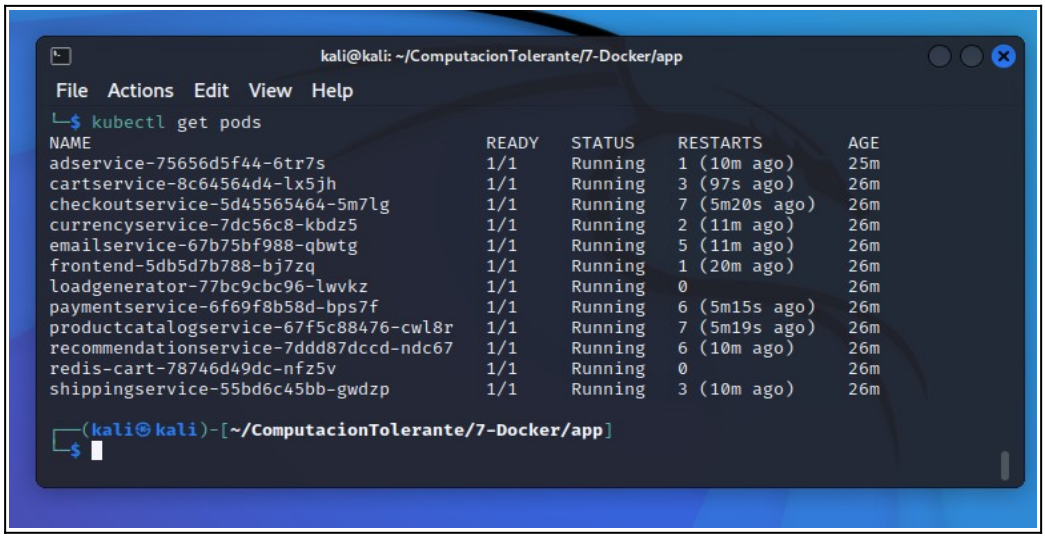
Y con este comando " *kubectl get ns*" podemos ver los Nodos que Istio ha agregado a Kubernetes:



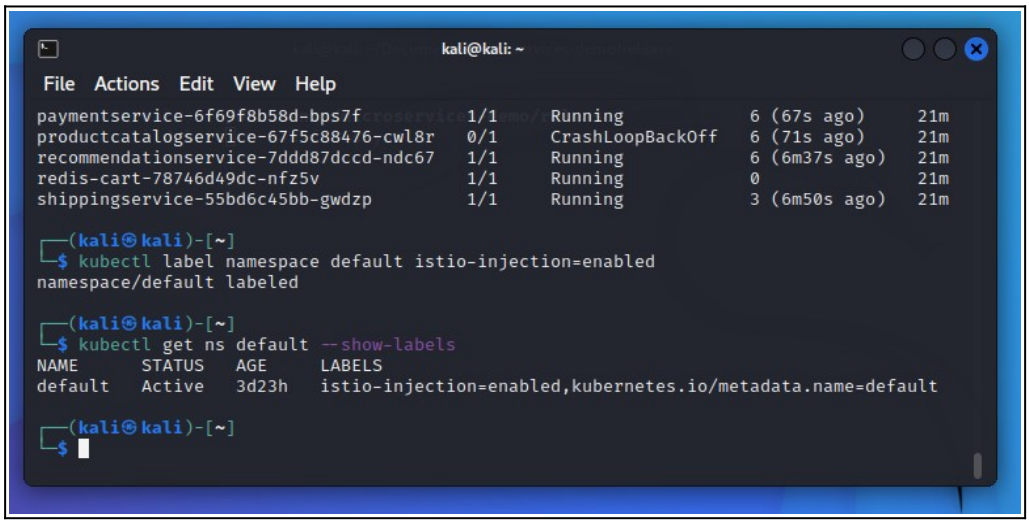

Ahora, procederemos a introducir los Microservicios de nuestra aplicación en Kubernetes, esto con el comando "*kubectl apply -f <archivo>*" a nuestro manifiesto de Kubernetes:



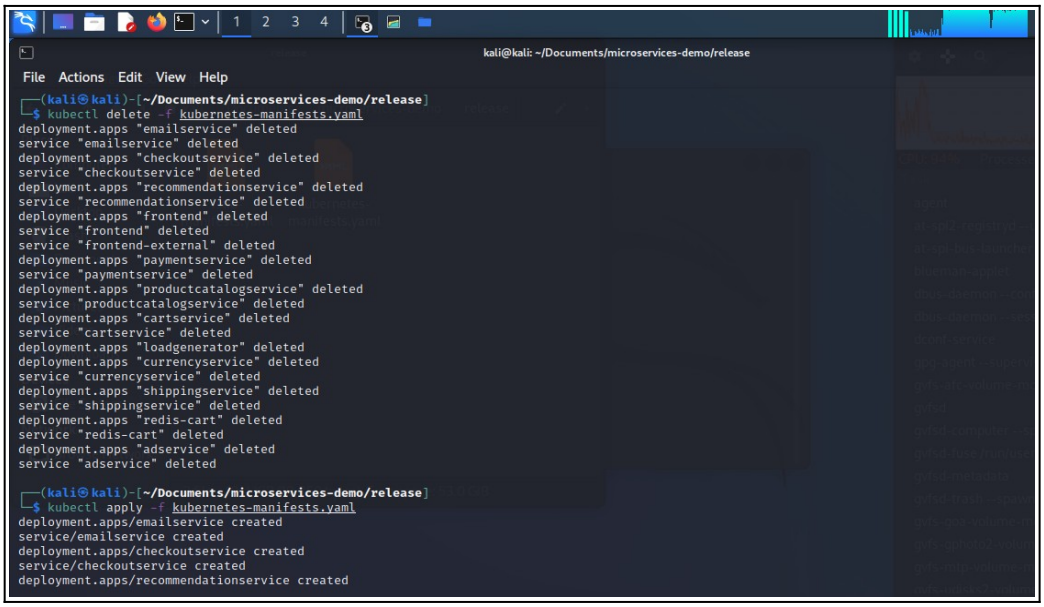
Lo siguiente es configurar los proxys de cada servicio desplegado, ya que no se encuentran, en la siguiente imagen se muestran los pods desplegados, esto gracias al comando "*kubectl get pods*", antes de configurar los proxys debemos de esperar a que los pods estén corriendo correctamente, esto puede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



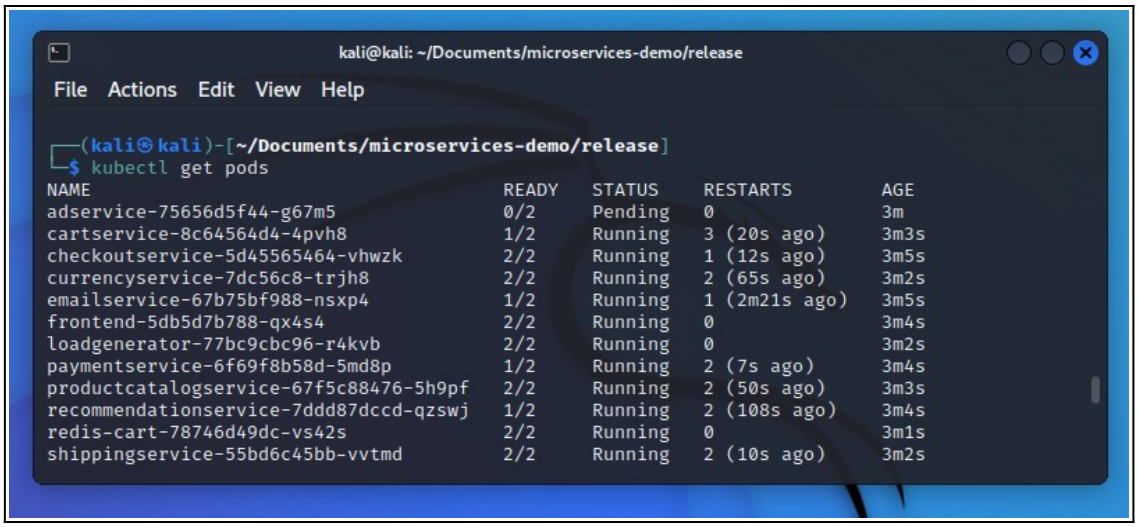

Para activar los proxys necesitamos crear una etiqueta para "istio-injection" con el comando "*kubectl label namespace default istio-injection=enabled*":



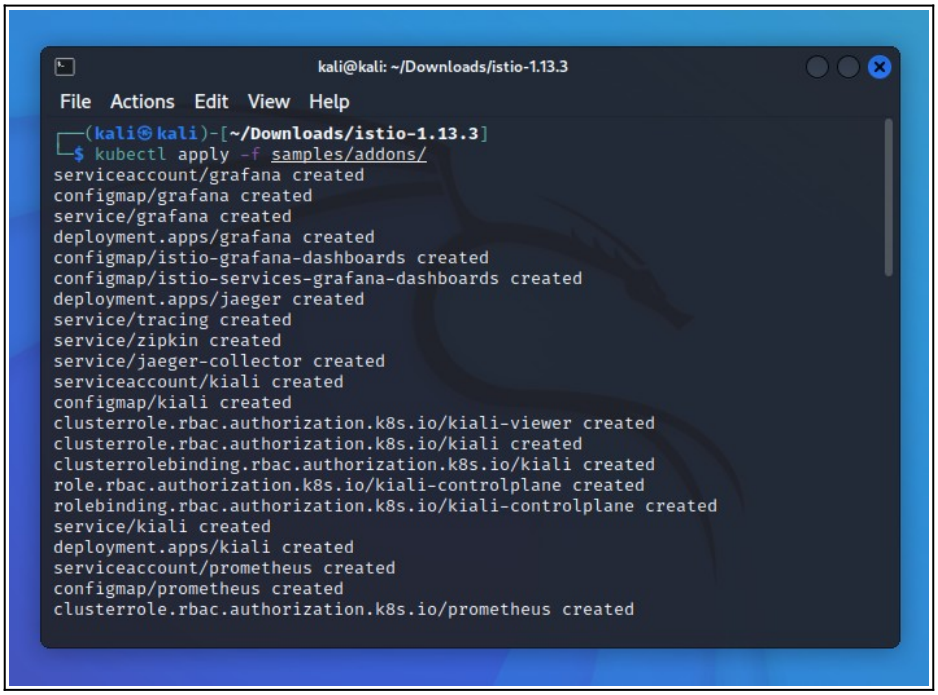
Tras esto, procedemos a eliminar los pods establecidos en el archivo YAML y los volvemos a cargar:



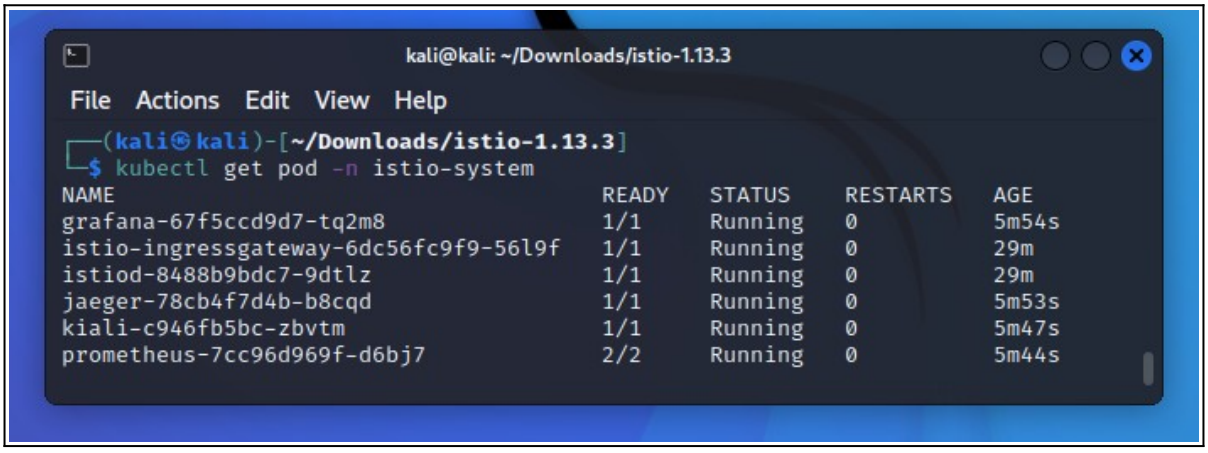

Al utilizar el comando "*kubectl get pods*" nos podemos dar cuenta que para cada pod hay dos contenedores, hay algunos que no tienen todos listos o están pendientes, esto se debe a limitantes de la CPU del equipo de computo:



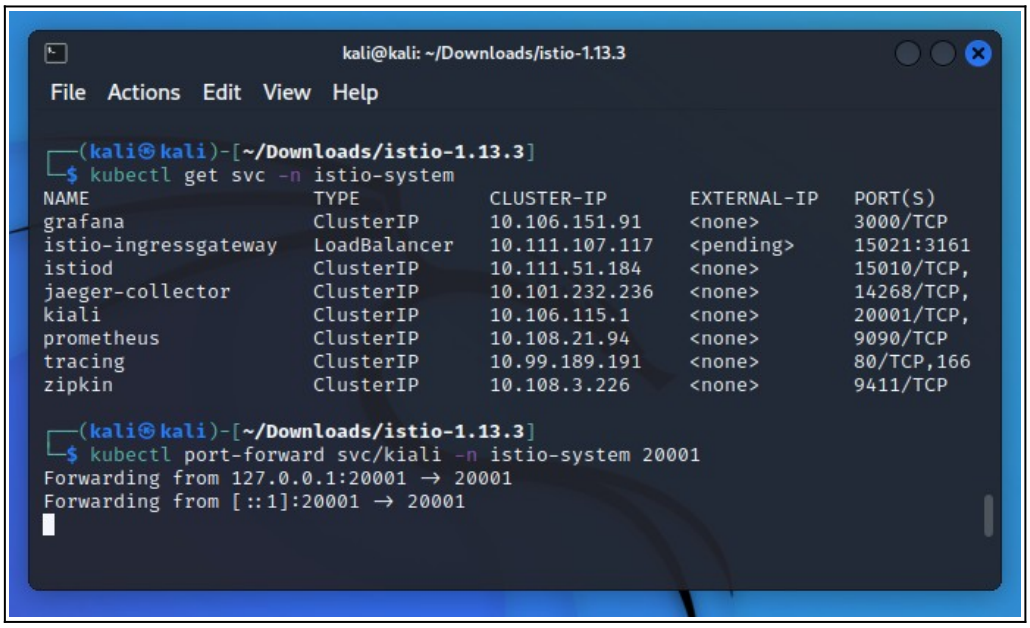
Lo siguiente, es agregar las herramientas de visualización y métricas para poder observar el funcionamiento gráfico de Istio, esto con el comando "*kubectl apply -f samples/addons/*" dentro de la carpeta instaladora de Istio:



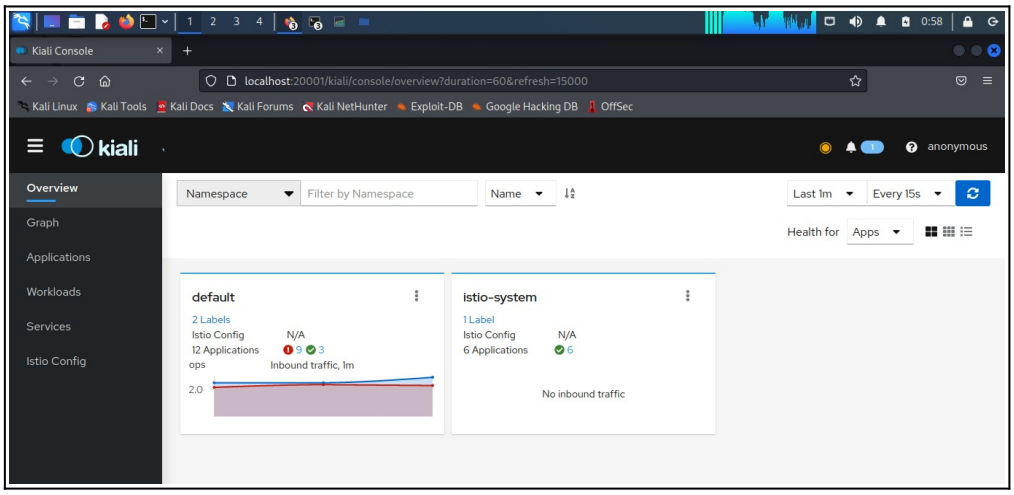

Con el siguiente comando "*kubectl get pod -n istio-system*" podemos visualizar los cambios realizados dentro de los pods de Istio:

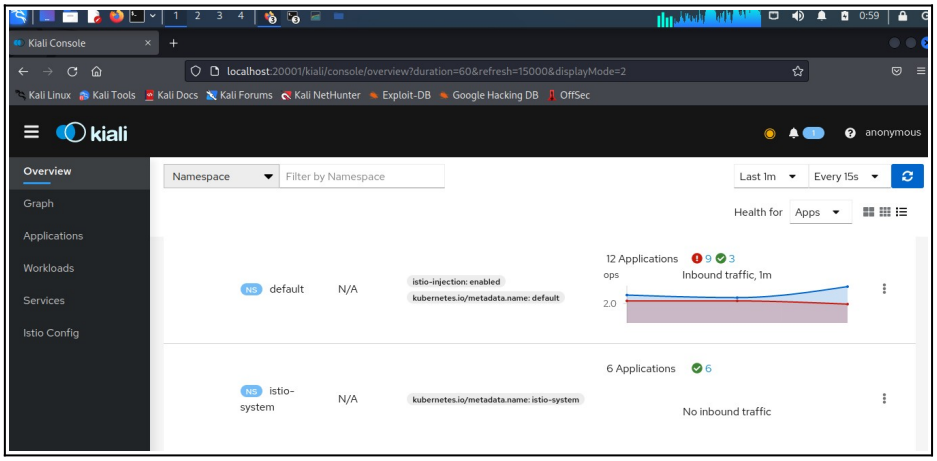


Lo siguiente es introducir el comando "*kubectl get svc -n istio-system*" para ver los puertos de los servicios de Istio, para después buscar el puerto del servicio Kiali, el cuál es el correspondiente al del Service Mesh de Istio, y después ponerlo en el comando "*kubectl port-forward svc/kiali -n istio-system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

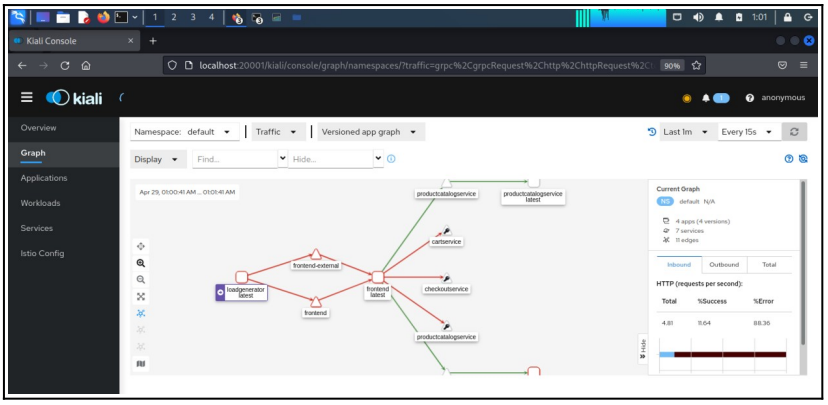


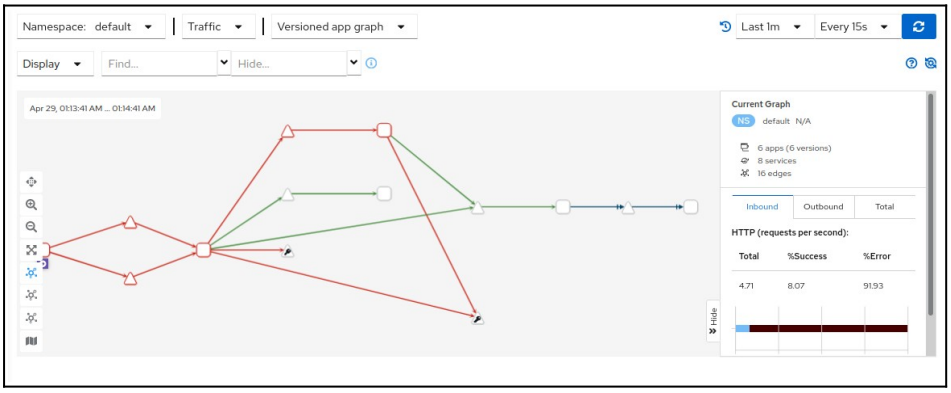

Finalmente, tenemos que ir al enlace de "*localhost:20001*" en el explorador para ver la interfaz de Kiali, como podemos observar, nuestro clúster necesita más recursos de CPU para poder correr todos los contenedores:






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]

**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 este de 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 cuál 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