



Centro Universitario de Ciencias Exactas e
Ingenierías.



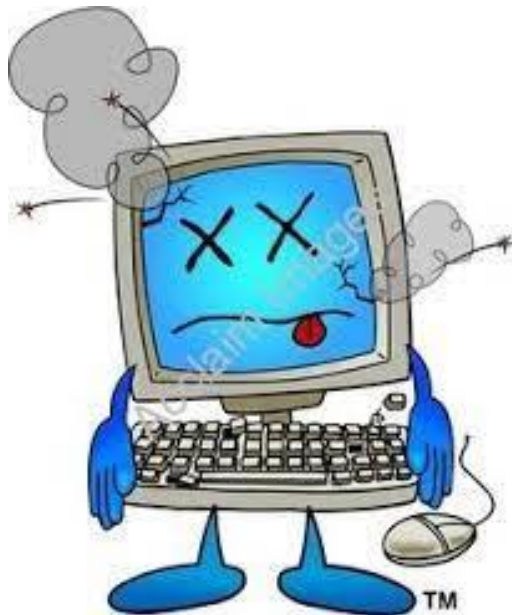
Ingeniería en computación.

Alumno: Vargas López David Guadalupe.

Computación tolerante a fallos.

Profesor: López Franco Michel Emanuel.

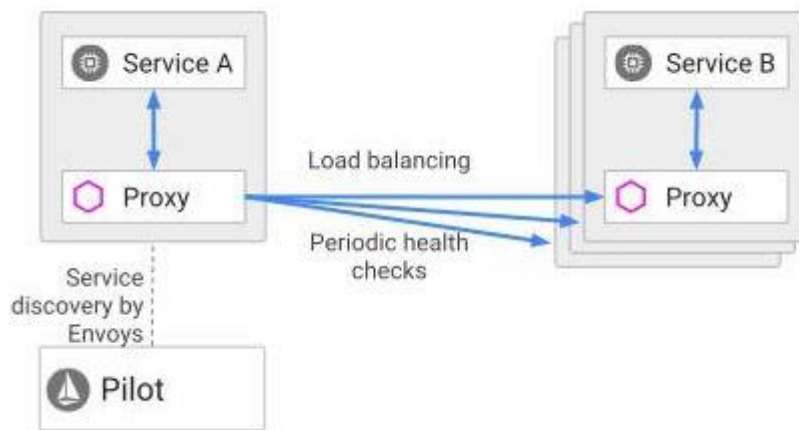
Sección: D06.



Guadalajara Jal. abril del 2022.

¿Qué es Istio?

Istio es una plataforma de malla de servicios con tecnología de open source que permite controlar el intercambio de datos entre los microservicios. Incluye API que le permiten integrarse a cualquier plataforma de registro, telemetría o sistema de políticas. El diseño de esta plataforma facilita su ejecución en distintos entornos: on-premise, alojados en la nube, en contenedores de Kubernetes y en servicios que se ejecutan en máquinas



virtuales, entre otros.

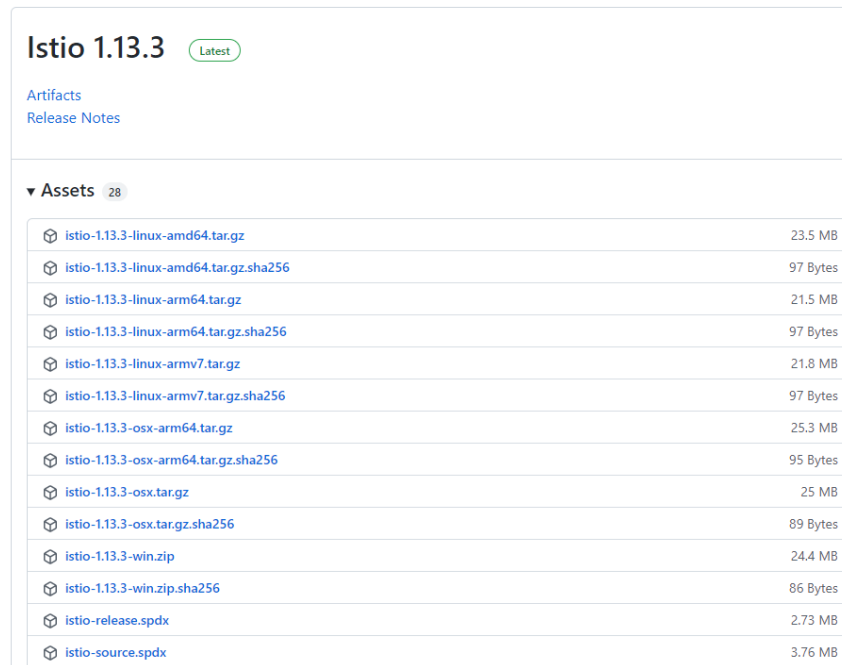
Istio Kubernetes es una red (o malla) de servicio que proporciona gestión de tráfico, aplicación de políticas de cumplimiento y recolección de

métricas. Una malla de servicios ("Service mesh") es una capa de infraestructura dedicada para gestionar la comunicación de servicio a servicio.

En Istio Kubernetes, esto se consigue configurando proxies basados en "Envoy", que es añadido a los pods como container "sidecar", e impone el flujo natural del tráfico al backend apropiado, mientras inhabilita a otros servicios que se comuniquen con este. Además, los servicios no se comunican directamente, sino que lo hacen a través de sus contenedores sidecar ("Envoy"). El responsable de este proceso es el "Pilot".

Aplicación usando istio:

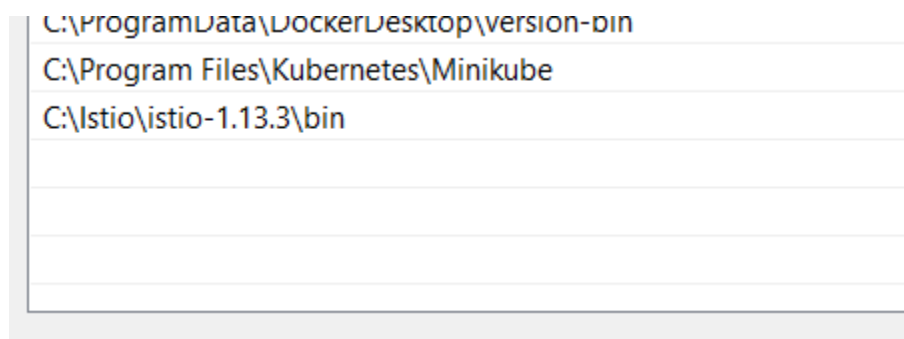
Ingresamos a la página oficial de Istio e instalamos la versión de acuerdo con el equipo que se tenga para su correcto funcionamiento, esta instalación se realiza en la página oficial de ISTIO dentro de un repositorio de GitHub.



The screenshot shows the GitHub release page for Istio 1.13.3. The page title is "Istio 1.13.3" with a "Latest" badge. Below the title are links for "Artifacts" and "Release Notes". The "Assets" section is expanded, showing a table of 28 assets. The assets are categorized by operating system and architecture, including Linux (amd64, arm64, armv7) and OSX (arm64). Each asset is a tar.gz or zip file, except for the release and source spdx files. The table lists the asset name and its size.

Asset Name	Size
istio-1.13.3-linux-amd64.tar.gz	23.5 MB
istio-1.13.3-linux-amd64.tar.gz.sha256	97 Bytes
istio-1.13.3-linux-arm64.tar.gz	21.5 MB
istio-1.13.3-linux-arm64.tar.gz.sha256	97 Bytes
istio-1.13.3-linux-armv7.tar.gz	21.8 MB
istio-1.13.3-linux-armv7.tar.gz.sha256	97 Bytes
istio-1.13.3-osx-arm64.tar.gz	25.3 MB
istio-1.13.3-osx-arm64.tar.gz.sha256	95 Bytes
istio-1.13.3-osx.tar.gz	25 MB
istio-1.13.3-osx.tar.gz.sha256	89 Bytes
istio-1.13.3-win.zip	24.4 MB
istio-1.13.3-win.zip.sha256	86 Bytes
istio-release.spdx	2.73 MB
istio-source.spdx	3.76 MB

Agregamos al path la instalación de Istio y posteriormente verificamos que funcione, de donde le pasamos la ruta creada después de hacer una carpeta en disco c de la computadora.



The screenshot shows a Windows command prompt with the following text entered:

```
C:\ProgramData\DockerDesktop\version-bin  
C:\Program Files\Kubernetes\Minikube  
C:\Istio\istio-1.13.3\bin
```

Con el comando `Istioctl` observamos que se encuentra totalmente funcional, para observar si su instalación se realizó de manera correcta.

```
Prueba la nueva tecnología PowerShell multiplataforma https://aka.ms/pscore6

PS C:\Users\david> istioctl
Istio configuration command line utility for service operators to
debug and diagnose their Istio mesh.

Usage:
  istioctl [command]

Available Commands:
  admin          Manage control plane (istiod) configuration
  analyze        Analyze Istio configuration and print validation messages
  authz          (authz is experimental. Use 'istioctl experimental authz')
  bug-report     Cluster information and log capture support tool.
  completion     Generate the autocompletion script for the specified shell
  create-remote-secret Create a secret with credentials to allow Istio to access remote Kubernetes apiservers
  dashboard      Access to Istio web UIs
  experimental   Experimental commands that may be modified or deprecated
  help           Help about any command
  install        Applies an Istio manifest, installing or reconfiguring Istio on a cluster.
  kube-inject    Inject Istio sidecar into Kubernetes pod resources
  manifest       Commands related to Istio manifests
  operator       Commands related to Istio operator controller.
  profile        Commands related to Istio configuration profiles
  proxy-config   Retrieve information about proxy configuration from Envoy [kube only]
  proxy-status   Retrieves the synchronization status of each Envoy in the mesh [kube only]
  remote-clusters Lists the remote clusters each istiod instance is connected to.
  tag            Command group used to interact with revision tags
  upgrade        Upgrade Istio control plane in-place
  validate       Validate Istio policy and rules files
  verify-install Verifies Istio Installation Status
  version        Prints out build version information

Flags:
  --context string      The name of the kubeconfig context to use
  -h, --help            help for istioctl
  -i, --istioNamespace string Istio system namespace (default "istio-system")
  -c, --kubeconfig string Kubernetes configuration file
  -n, --namespace string Config namespace
  --vlog level          number for the log level verbosity. Like -v flag. ex: --vlog=9

Additional help topics:
  istioctl options      Displays istioctl global options

Use 'istioctl [command] --help' for more information about a command.
PS C:\Users\david>
```

Levantamos el contenedor de Docker con 4gb de RAM, así como también utilizando 4 CPU para tener un buen rendimiento, aunque entre más recursos se le asignen será mucho mejor.

```
david@LAPTOP-OMJ7SMPI MINGW64 ~/OneDrive/Escritorio/6to semestre/tolerante a fal
los/istio codigo (main)
$ minikube start --cpus 4 --memory 4000
* minikube v1.25.2 en Microsoft Windows 10 Home 10.0.19044 Build 19044
* Using the docker driver based on existing profile
! You cannot change the memory size for an existing minikube cluster. Please fir
st delete the cluster.
! You cannot change the CPUs for an existing minikube cluster. Please first dele
te the cluster.
* Starting control plane node minikube in cluster minikube
* Pulling base image ...
* Restarting existing docker container for "minikube" ...
* Preparando Kubernetes v1.23.3 en Docker 20.10.12...
  - kubelet.housekeeping-interval=5m
* Verifying Kubernetes components...
  - Using image kubernetesui/dashboard:v2.3.1
  - Using image kubernetesui/metrics-scraper:v1.0.7
  - Using image gcr.io/k8s-minikube/storage-provisioner:v5
* Complementos habilitados: storage-provisioner, default-storageclass, dashboard
* Done! kubectl is now configured to use "minikube" cluster and "default" namesp
ace by default
```

Observamos si se encuentra instalado los recursos necesarios para utilizar Istio y como se observa no, por lo que se deberá instalarlo.

```
david@LAPTOP-OMJ7SMPI MINGW64 ~/OneDrive/Escritorio/6to semestre/tolerante a fallos/istio codigo (main)
$ kubectl get ns
NAME                STATUS   AGE
default             Active   20d
kube-node-lease     Active   20d
kube-public         Active   20d
kube-system         Active   20d
kubernetes-dashboard Active   20d
```

Posteriormente instalamos istioctl, así como istiod que es el proceso principal de istio.

```
david@LAPTOP-OMJ7SMPI MINGW64 ~/OneDrive/Escritorio/6to semestre/tolerante a fallos/istio codigo (main)
$ istioctl install
This will install the Istio 1.13.3 default profile with ["Istio core" "Istiod" "Ingress gateways"] components into the cluster. Proceed? (y/N) y
✓ 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 about your install/upgrade experience! https://forms.gle/pzWZpAvMVBecaQ9h9
```

Y corroboramos que efectivamente se instaló istio en el clúster.

```
david@LAPTOP-OMJ7SMPI MINGW64 ~/OneDrive/Escritorio/6to semestre/tolerante a fallos/istio codigo (main)
$ kubectl get ns
NAME                STATUS   AGE
default             Active   20d
istio-system        Active   4m13s
kube-node-lease     Active   20d
kube-public         Active   20d
kube-system         Active   20d
kubernetes-dashboard Active   20d
```

Y con el siguiente comando para observar los pods que se están corriendo actualmente.

```
PS C:\Users\david> kubectl get pod -n istio-system
NAME                                                    READY   STATUS    RESTARTS   AGE
istio-ingressgateway-6dc56fc9f9-49zk9                 1/1     Running   0           16m
istiod-8488b9bdc7-cbh4m                               1/1     Running   0           17m
PS C:\Users\david>
```

Se modifican los recursos definidos y se aplican los manifiestos para esto simplemente se realizó la aplicación con base a lo ya establecido.

```
PS C:\Users\david\OneDrive\Escritorio\istiocodigo\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
```

Se revisan los microservicios creados con el comando get pod

```
PS C:\Users\david\OneDrive\Escritorio\istiocodigo\microservices-demo\release> kubectl get pod
NAME                                READY   STATUS              RESTARTS   AGE
adservice-75656d5f44-2mv5s         0/1     ContainerCreating   0           7m36s
cartservice-8c64564d4-qsjhc        0/1     Running             5 (7s ago)  7m39s
checkoutservice-5d45565464-6m96f   1/1     Running             4 (2m44s ago)  7m41s
currencyservice-7dc56c8-b5xfn      0/1     Running             0           7m39s
emailservice-67b75bf988-rxb68      0/1     CrashLoopBackOff    6 (45s ago)  7m42s
flask-test-app-7846b7bb8f-7hfbn    1/1     Running             2 (11m ago)  21d
flask-test-app-7846b7bb8f-7vkwb    1/1     Running             2 (11m ago)  21d
flask-test-app-7846b7bb8f-nh9qp    1/1     Running             2 (11m ago)  21d
flask-test-app-7846b7bb8f-qztrl    1/1     Running             2 (11m ago)  21d
flask-test-app-7846b7bb8f-x4brq    1/1     Running             2 (11m ago)  21d
frontend-5db5d7b788-d4tl4         1/1     Running             0           7m40s
loadgenerator-77bc9cbc96-tpzn1     0/1     Init:0/1            0           7m39s
paymentservice-6f69f8b58d-5bx98    1/1     Running             2 (2m52s ago)  7m40s
productcatalogservice-67f5c88476-gd5lg 1/1     Running             5 (7s ago)  7m40s
recommendationservice-7ddd87dccc-2w8p7 1/1     Running             5 (2m57s ago)  7m41s
redis-cart-78746d49dc-8bvnn        0/1     ContainerCreating   0           7m37s
shippingservice-55bd6c45bb-kpjml    0/1     ContainerCreating   0           7m38s
PS C:\Users\david\OneDrive\Escritorio\istiocodigo\microservices-demo\release>
```

Ahora se pueden observar las etiquetas de la aplicación de servicios.

```
PS C:\Users\david\OneDrive\Escritorio\istiocodigo\microservices-demo\release> kubectl get ns default --show-labels
NAME     STATUS   AGE   LABELS
default Active   21d   kubernetes.io/metadata.name=default
PS C:\Users\david\OneDrive\Escritorio\istiocodigo\microservices-demo\release>
```

Ahora se realiza una etiqueta namespace en donde se maneja un valor por default.

```
PS C:\Users\david\OneDrive\Escritorio\istiocodigo\microservices-demo\release> kubectl label namespace default istio-injection=enabled
namespace/default labeled
PS C:\Users\david\OneDrive\Escritorio\istiocodigo\microservices-demo\release> |
```

Ahora se borran los microservicios especificados en kubernetes-manifest.yaml para observar el funcionamiento de los proxies.

```
PS C:\Users\david\OneDrive\Escritorio\istiocodigo\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
```

Ahora se hace apply para volver a activar los microservicios que se encuentran en el archivo .yaml.

```
PS C:\Users\david\OneDrive\Escritorio\istiocodigo\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
```

A continuación, se observan el estatus de los pods para observar las instancias que se están corriendo.

```
Unable to connect to the server: net/http: TLS handshake timeout
PS C:\Users\david\OneDrive\Escritorio\istiocodigo\microservices-demo\release> kubectl get pod
NAME                                READY   STATUS             RESTARTS   AGE
adservice-75656d5f44-t48gf          0/1     ContainerCreating   0           8m16s
cartservice-8c64564d4-9vxn timer 0/1     Running         1 (6m1s ago) 9m15s
checkoutservice-5d45565464-44x7z    0/1     ContainerCreating   0           9m37s
currencyservice-7dc56c8-ggfh5        0/1     ContainerCreating   0           8m18s
emailservice-67b75bf988-4zc2r        0/1     ContainerCreating   0           9m39s
frontend-5db5d7b788-4c9vc           1/1     Running             0           9m27s
loadgenerator-77bc9cbc96-2tbx9       0/1     Init:0/1            0           8m18s
paymentservice-6f69f8b58d-9sj85      0/1     Running             0           9m20s
productcatalogservice-67f5c88476-fmf5v 0/1     ContainerCreating   0           9m17s
recommendationservice-7ddd87dcd-hx5j6 0/1     ContainerCreating   0           9m36s
redis-cart-78746d49dc-ml2pm          0/1     ContainerCreating   0           8m16s
shippingservice-55bd6c45bb-r58k4      0/1     ContainerCreating   0           8m17s
PS C:\Users\david\OneDrive\Escritorio\istiocodigo\microservices-demo\release> █
```

Con el comando describe se accede por separado a cualquiera de los pods por medio de su nombre, para observar los detalles de cada uno de ellos.

```
PS C:\Users\david\OneDrive\Escritorio\istiocodigo\microservices-demo\release> kubectl describe pod cartservice-8c64564d4-9vxn timer
Name:          cartservice-8c64564d4-9vxn timer
Namespace:     default
Priority:       0
Node:          minikube/192.168.49.2
Start Time:    Mon, 25 Apr 2022 23:46:31 -0500
Labels:        app=cartservice
               pod-template-hash=8c64564d4
Annotations:   <none>
Status:        Running
IP:            172.17.0.10
IPs:           172.17.0.10
Controlled By: ReplicaSet/cartservice-8c64564d4
Containers:
  server:
    Container ID:  docker://4da0e279703afe6146297b4631502319b14bc5019c5ccb1d3d16b6e0b49b242d
    Image:         gcr.io/google-samples/microservices-demo/cartservice:v0.3.6
    Image ID:      docker-pullable://gcr.io/google-samples/microservices-demo/cartservice@sha256:eb0ac54c81a8f60ddba39921a4052dd6dfe88273805983c43d5283b446a584ee
    Port:         7070/TCP
    Host Port:     0/TCP
    State:         Running
```


Se aplican las integraciones para poder utilizarlas dentro de istio son archivos de configuración de los mismos kubernetes.

```
PS C:\Istio\istio-1.13.3\samples> kubectl apply -f addons
serviceaccount/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
clusterrole.rbac.authorization.k8s.io/kiali-viewer 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
clusterrolebinding.rbac.authorization.k8s.io/prometheus created
```

Posteriormente observamos los pods, contando con los de istio y los manifest.yaml las cuales son integraciones necesarias para utilizar los microservicios.

```
PS C:\Istio\istio-1.13.3\samples> kubectl get pod -n istio-system
NAME                                READY   STATUS    RESTARTS   AGE
grafana-67f5ccd9d7-rwzct            1/1     Running   1 (7m50s ago)   54m
istio-ingressgateway-6dc56fc9f9-rl46b 1/1     Running   0             108m
istiod-8488b9bdc7-d6xkg             1/1     Running   0             109m
jaeger-78cb4f7d4b-tj9tk             1/1     Running   1             53m
kiali-c946fb5bc-1thqt               1/1     Running   1 (15m ago)     52m
prometheus-7cc96d969f-jlkn6         2/2     Running   0             52m
```

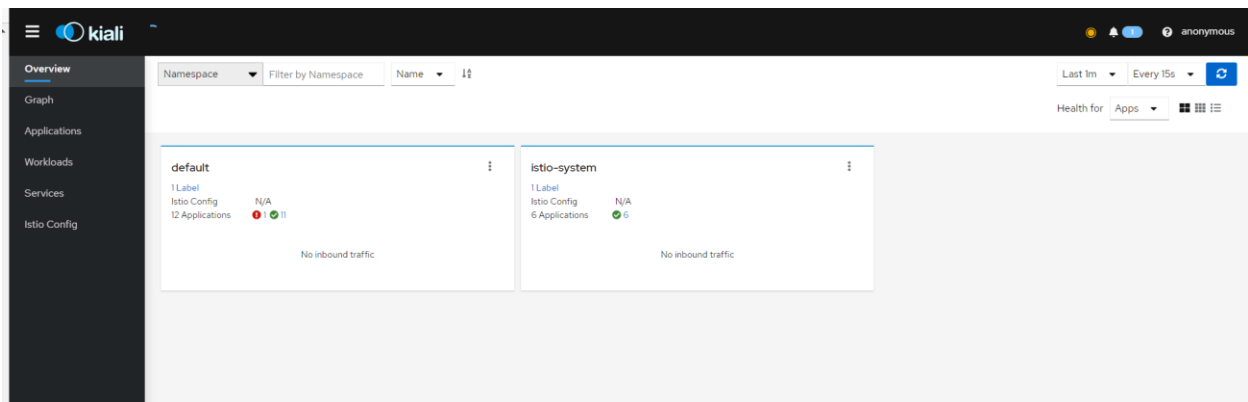
Y con el siguiente comando se accede a estos pods, para poder observar el funcionamiento continuo de los pods y su estatus actual.

```
PS C:\Istio\istio-1.13.3\samples> kubectl get svc -n istio-system
NAME                                TYPE                CLUSTER-IP      EXTERNAL-IP      PORT(S)                                     AGE
grafana                             ClusterIP           10.99.50.251    <none>            3000/TCP                                   12m
istio-ingressgateway                LoadBalancer       10.110.14.5     <pending>        15021:32608/TCP,80:30907/TCP,443:31442/TCP 66m
istiod                               ClusterIP           10.107.135.179  <none>            15010/TCP,15012/TCP,443/TCP,15014/TCP      68m
jaeger-collector                    ClusterIP           10.103.147.5    <none>            14268/TCP,14250/TCP,9411/TCP              11m
kiali                                ClusterIP           10.110.191.199  <none>            20001/TCP,9090/TCP                        10m
prometheus                           ClusterIP           10.96.142.47    <none>            9090/TCP                                   10m
tracing                             ClusterIP           10.96.187.229   <none>            80/TCP,16685/TCP                         12m
zipkin                               ClusterIP           10.103.126.96   <none>            9411/TCP                                   11m
PS C:\Istio\istio-1.13.3\samples>
```

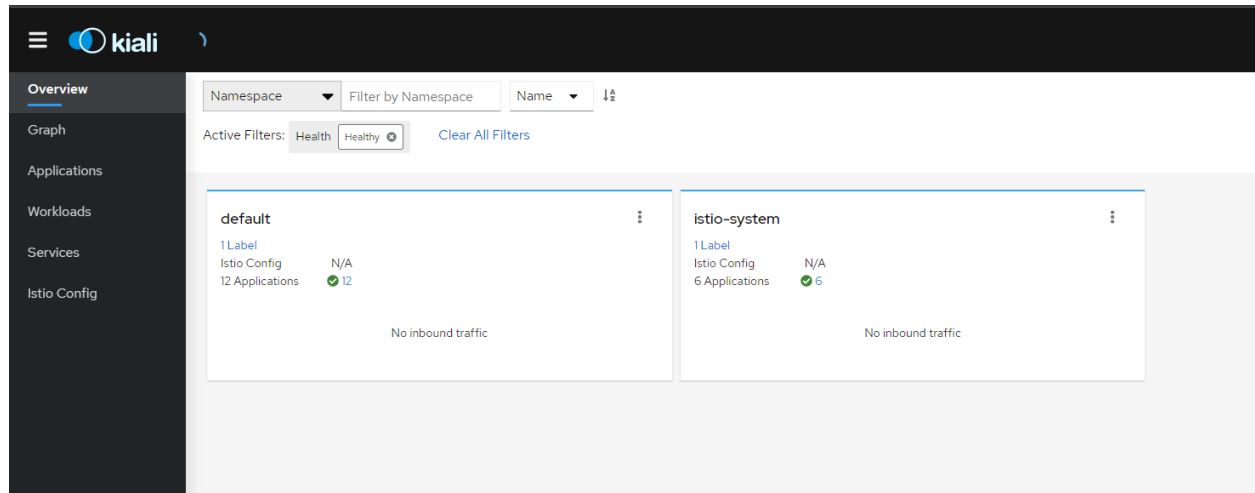
A continuación se habilita el kiali en el localhost para poder visualizar sus componentes de forma grafica.

```
PS C:\Istio\istio-1.13.3\samples> kubectl port-forward svc/kiali -n istio-system 20001
Forwarding from 127.0.0.1:20001 -> 20001
Forwarding from [::1]:20001 -> 20001
Handling connection for 20001
Handling connection for 20001
Handling connection for 20001
Handling connection for 20001
Handling connection for 20001
Handling connection for 20001
Handling connection for 20001
Handling connection for 20001
Handling connection for 20001
Handling connection for 20001
Handling connection for 20001
Handling connection for 20001
Handling connection for 20001
Handling connection for 20001
Handling connection for 20001
Handling connection for 20001
Handling connection for 20001
```

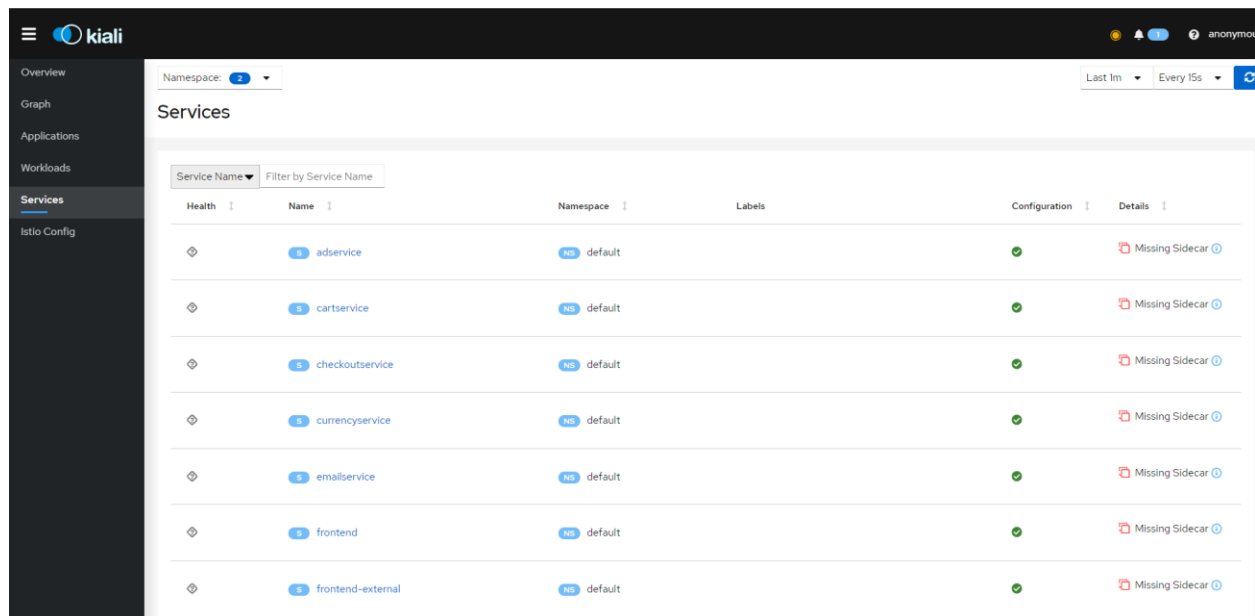
Entramos a kiali en el navegador utilizando la liga localhost/20001, en donde te direccionara de forma automático.



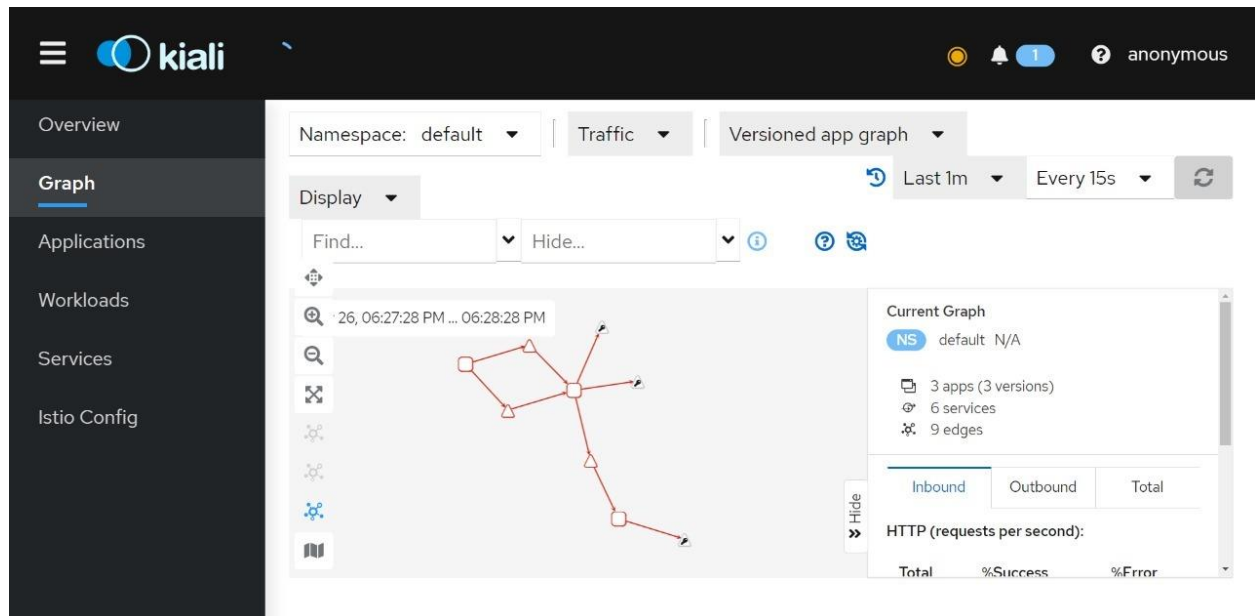
A continuación, se presentan el dashboard de kiali que se direcciona en automático.



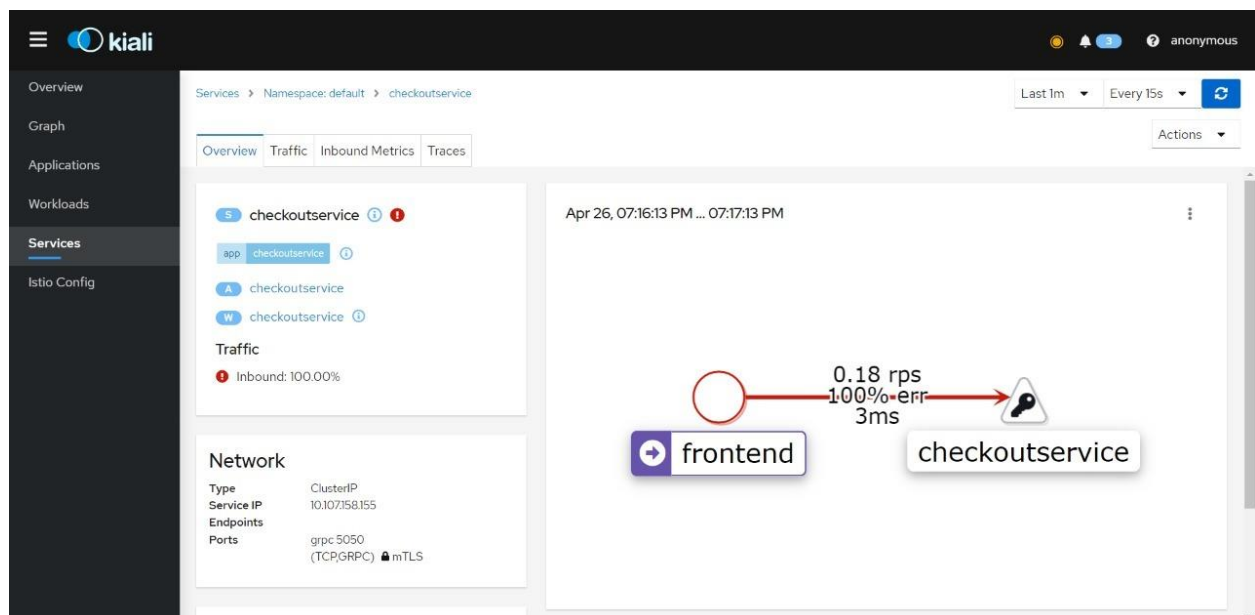
Después esta la parte de aplicaciones, en la cual se muestran detalles de los microservicios que se presentan.



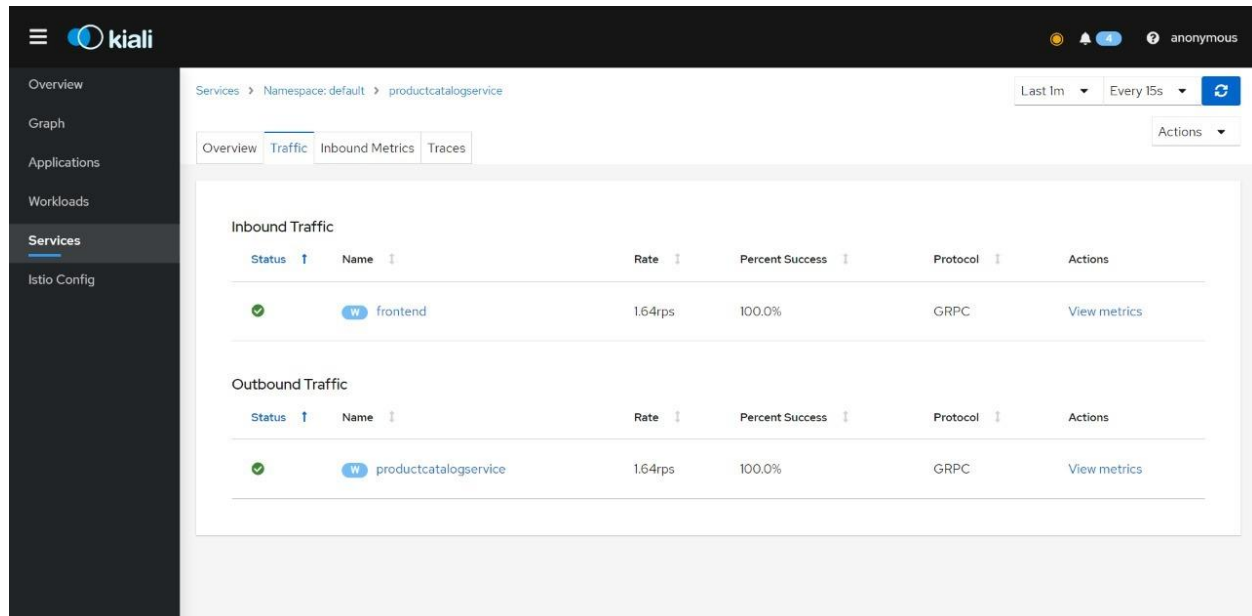
Y en la parte de graph se observa el grafo tipo árbol con los nodos correspondientes a los microservicios que se presentan.



Aunque también podemos abrir por separado cada uno de los microservicios y observar su grafo correspondiente además de sus características.



A continuación, se muestran las características de uno de los microservicios, así como su tráfico de entrada y tráfico de salida.



Link al repositorio:

<https://github.com/David-1212/istio>

Conclusiones:

Esta práctica me pareció bastante interesante, debido a que de esta manera se pueden realizar microservicios para una aplicación y de esta manera observar el funcionamiento mas a fondo de cada uno de ellos y en caso de que alguno de estos este fallando, poder evitar que la pagina caiga por completo, además de que en lo personal si lo utilizaría ya que en este caso se podría implementar en proyectos grandes o que estén a la venta.

En lo personal me pareció bastante engorroso debido a que como istio es nativo de Linux, al momento de estarlo realizando en Windows tenia bastantes problemas y bugs además de problemas con los proxys, los cuales se arreglaron cuando desactive los proxys y se dejo que la computadora cargara más tiempo para que no se sobrecargara de instrucciones.