

# SERVICE MESH EXAMPLE

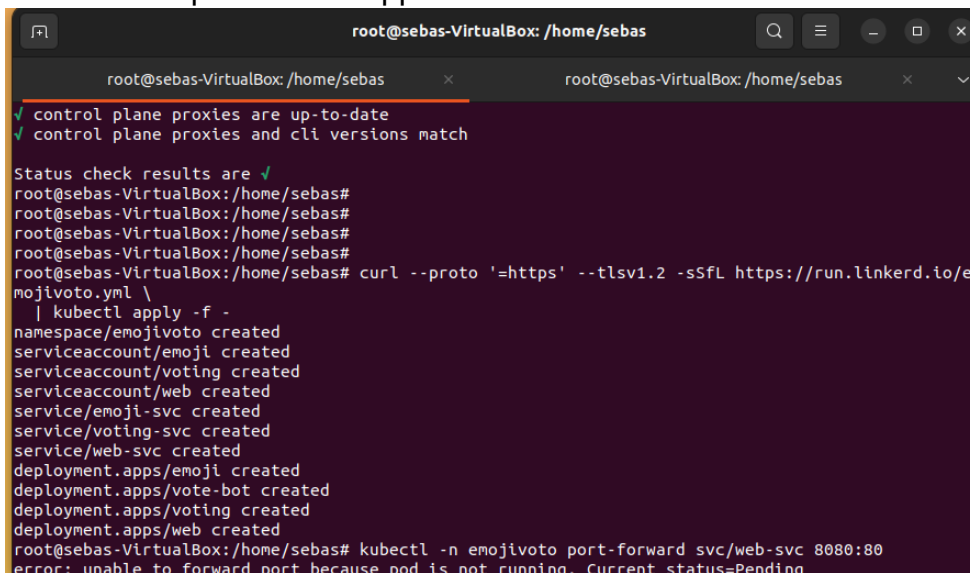
## LINKERD

### Experiencia

Mi experiencia corriendo el getting started fue un poco caótica pues no hay comandos para Windows, por lo que tuve que hacerlo en una máquina virtual de Linux Ubuntu y encima de eso no pude hacerlo en mi equipo portátil, tuve que utilizar mi computadora de escritorio pues la laptop se rehusó a instalar Docker, creo que es por la licencia de Windows que la versión home no cuenta con módulos del hipervisor necesarios para virtualización anidada, habiendo tenido la pc de escritorio con Windows 10 pro en la virtual de Linux instale el Docker Engine para usarlo como driver para mi cluster de kubernetes, en mi caso utilicé Minikube luego con minikube start forcé driver Docker levante el cluster y ya pude realizar la guía. Se realiza una comprobación del cluster que está corriendo correctamente, luego se instala linkerd en el cluster el segundo comando `linkerd install | kubectl apply -f -` me dio un error de proxy con la recomendación que me dio se solucionó poniéndolo después del install y el resto levantar la demo y el dashboard no tuvo inconvenientes.

### Aprendizaje

La función que más aprecio en el service mesh es la observabilidad ya que el dashboard nos da muchas métricas de la aplicación también como se enruta la aplicación. También observe un poco sobre la resiliencia al entrar a los emojis que retornan errores y como se muestran en el log y como esta conectado el trafico entre los endpoints de la app.

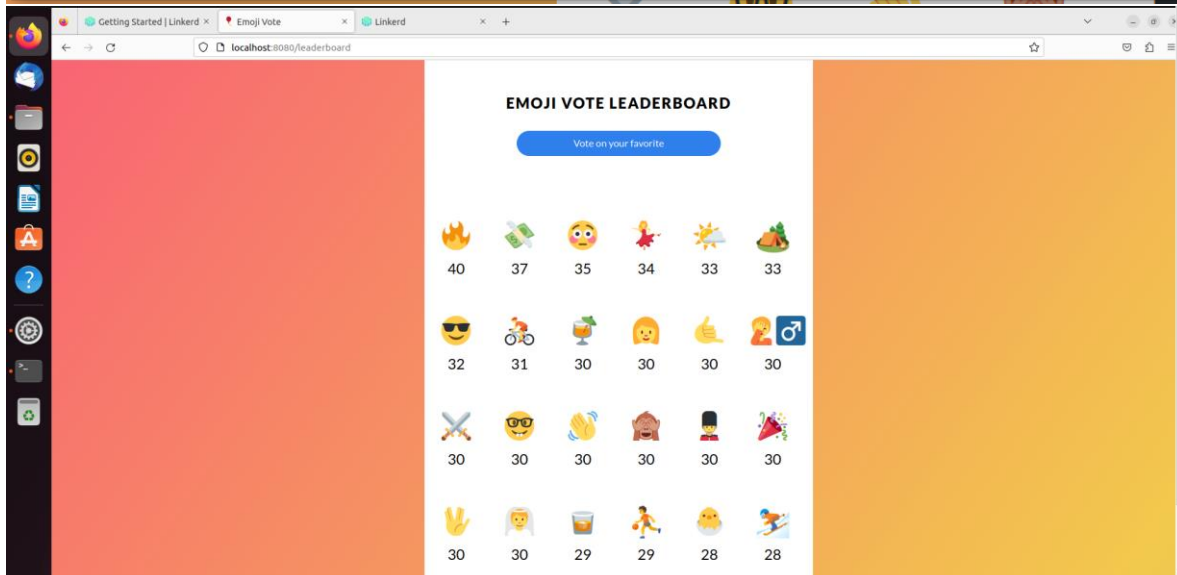


```
root@sebas-VirtualBox: /home/sebas
✓ control plane proxies are up-to-date
✓ control plane proxies and cli versions match

Status check results are ✓
root@sebas-VirtualBox:/home/sebas#
root@sebas-VirtualBox:/home/sebas#
root@sebas-VirtualBox:/home/sebas#
root@sebas-VirtualBox:/home/sebas#
root@sebas-VirtualBox:/home/sebas# curl --proto 'https' --tlsv1.2 -sSfL https://run.linkerd.io/emojivoto.yml \
| kubectl apply -f -
namespace/emojivoto created
serviceaccount/emoji created
serviceaccount/voting created
serviceaccount/web created
service/emoji-svc created
service/voting-svc created
service/web-svc created
deployment.apps/emoji created
deployment.apps/vote-bot created
deployment.apps/voting created
deployment.apps/web created
root@sebas-VirtualBox:/home/sebas# kubectl -n emojivoto port-forward svc/web-svc 8080:80
error: unable to forward port because pod is not running. Current status=Pending
```

```
linkerd-data-plane
-----
✓ data plane namespace exists
✓ data plane proxies are ready
✓ data plane is up-to-date
✓ data plane and cli versions match
✓ data plane pod labels are configured correctly
✓ data plane service labels are configured correctly
✓ data plane service annotations are configured correctly
✓ opaque ports are properly annotated

Status check results are ✓
root@sebas-VirtualBox:/home/sebas# kubectl -n emojioto port-forward svc/web-svc 8080:80
Forwarding from 127.0.0.1:8080 -> 8080
Forwarding from [::1]:8080 -> 8080
Handling connection for 8080
E0314 16:25:44.458213 81020 portforward.go:379] error copying from remote stream to local connection: readfrom tcp4 127.0.0.1:8080->127.0.0.1:48404: write tcp4 127.0.0.1:8080->127.0.0.1:48404: write: broken pipe
Handling connection for 8080
Handling connection for 8080
Handling connection for 8080
```



Getting Started | Linkerd xEmoji Vote xLinkerd x

localhost:50750/namespaces/emojivoto

LINKERD

Namespace > emojivoto

CLUSTER

- Namespaces
- Plano de Control

EMOJIVOTO

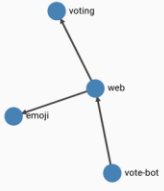
CARGAS DE TRABAJO

- Cron Jobs
- Daemon Sets
- Deployments
- menuItemsServices
- Jobs
- Pods
- Replica Sets
- Replication Controllers
- Stateful Sets

CONFIGURACIÓN

- Traffic Splits

MEJORAMIENTO



```
graph TD; voting --- web; emoji --- web; web --- vote-bot
```

Deployments

Deployment	En la malla de servicios	Tasa de éxito	PPS	Latencia P50	Latencia P95	Latencia P99
emoji	1/1	100.00%	2.27	1 ms	5 ms	9 ms
vote-bot	1/1	100.00%	0.3	1 ms	4 ms	4 ms
voting	1/1	89.47%	1.27	1 ms	2 ms	3 ms
web	1/1	94.20%	2.3	4 ms	28 ms	38 ms

Pods