



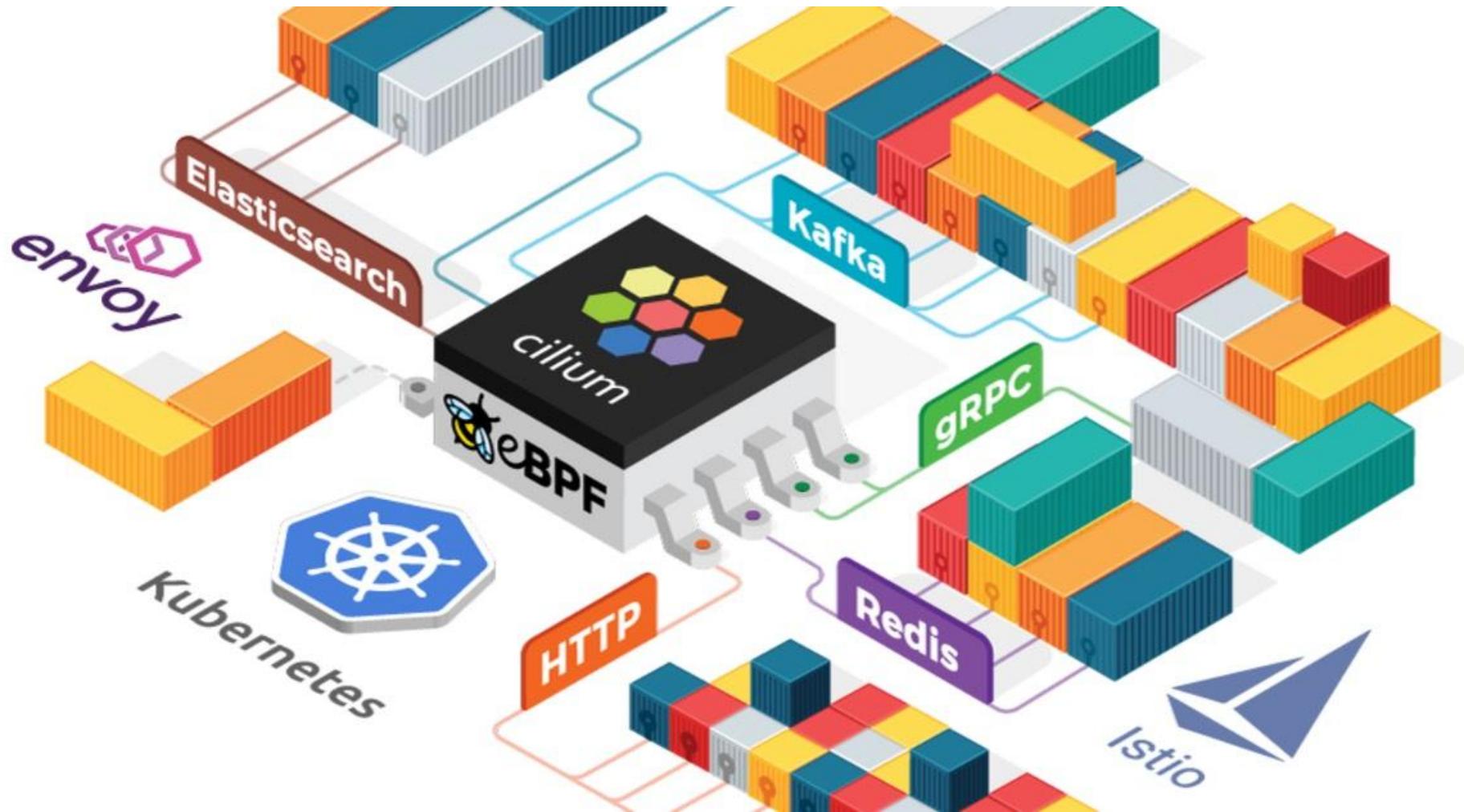
Introduction to Cilium Network with eBPF Technology

(Prparn Lueangpoonlap)

Agenda

- What is Cilium ?
- Why this matter ?
- Demo session
- What next ?

What is Cilium?

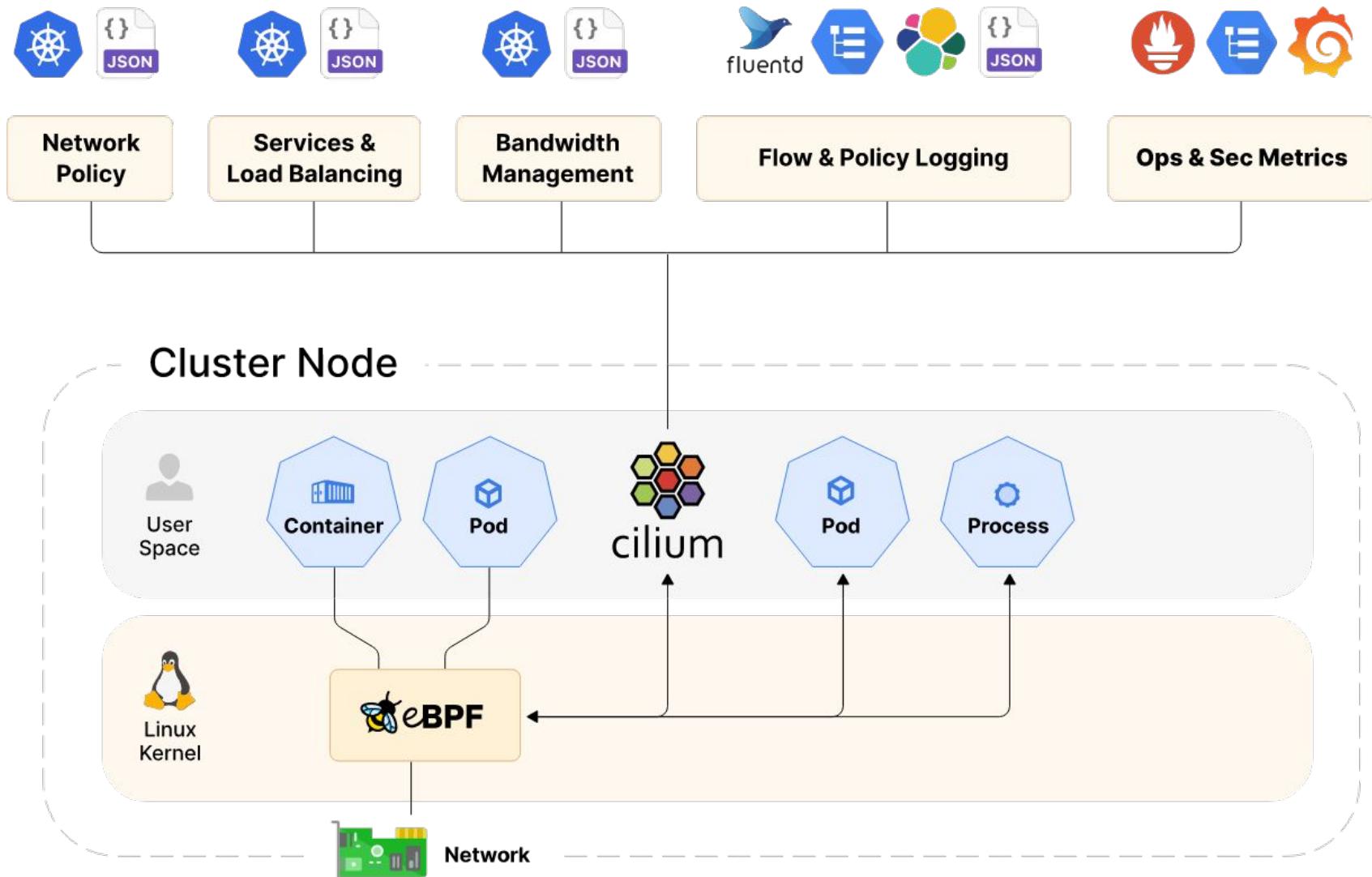


Introduction to Cilium Network

What is Cilium?

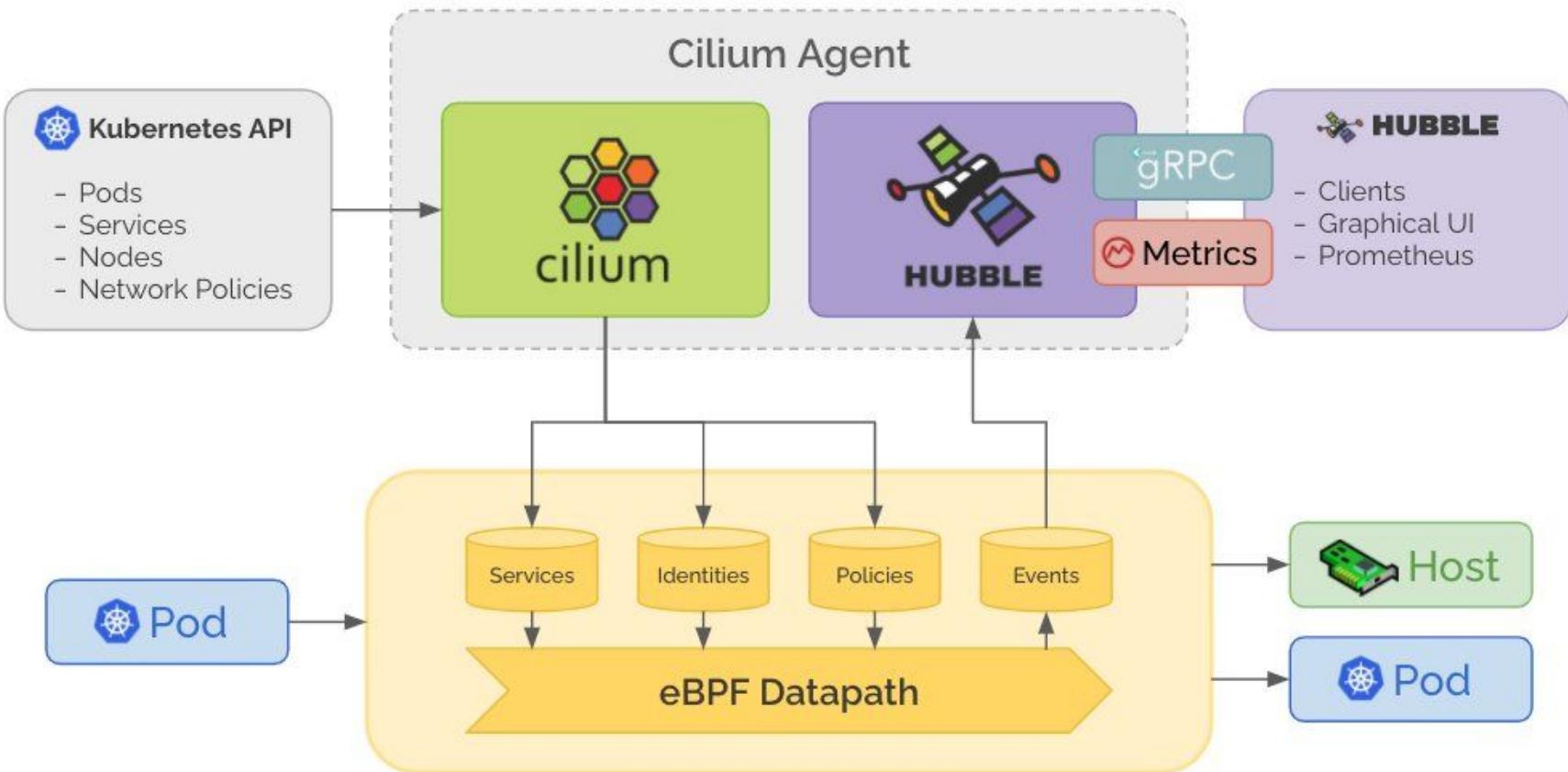
- **Cilium** is an open source, **cloud native solution** for providing, securing, and observing network connectivity between workloads, fueled by the revolutionary **Kernel technology eBPF**.
- As CNCF member (Graduation). Cilium is now open source project for provide **networking, security, observability** in cloud native environment (**Kubernetes as CNI standard**)
- With incredible **performance in Cilium**. Many cloud provider had been added cilium to their Kubernetes platform already since year 2021

What is Cilium?



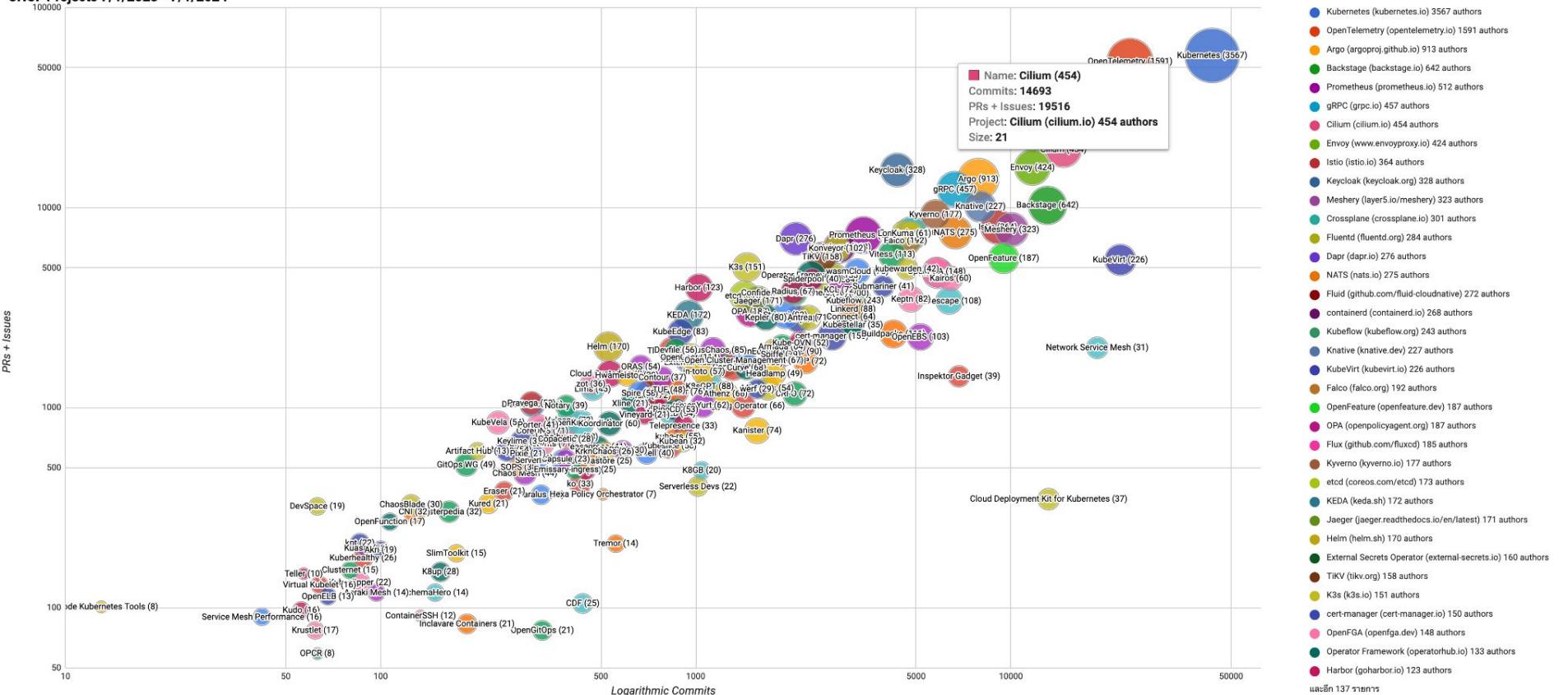
Introduction to Cilium Network

What is Cilium?



What is Cilium?

CNCF Projects 7/1/2023 - 7/1/2024



Ref: https://docs.google.com/spreadsheets/d/1FyVjdO9kMnz4hmEkyYLpI101cBDq_ZGHMKR5j_ce5JU/edit?gid=976519966#gid=976519966

Introduction to Cilium Network

What is Cilium?

- Networking



High Performance
Networking (CNI) →



Layer 4 Load Balancer →



Cluster Mesh →



Bandwidth and Latency
Optimization →



Kube-proxy Replacement →



BGP →



Egress Gateway →



Service Mesh →

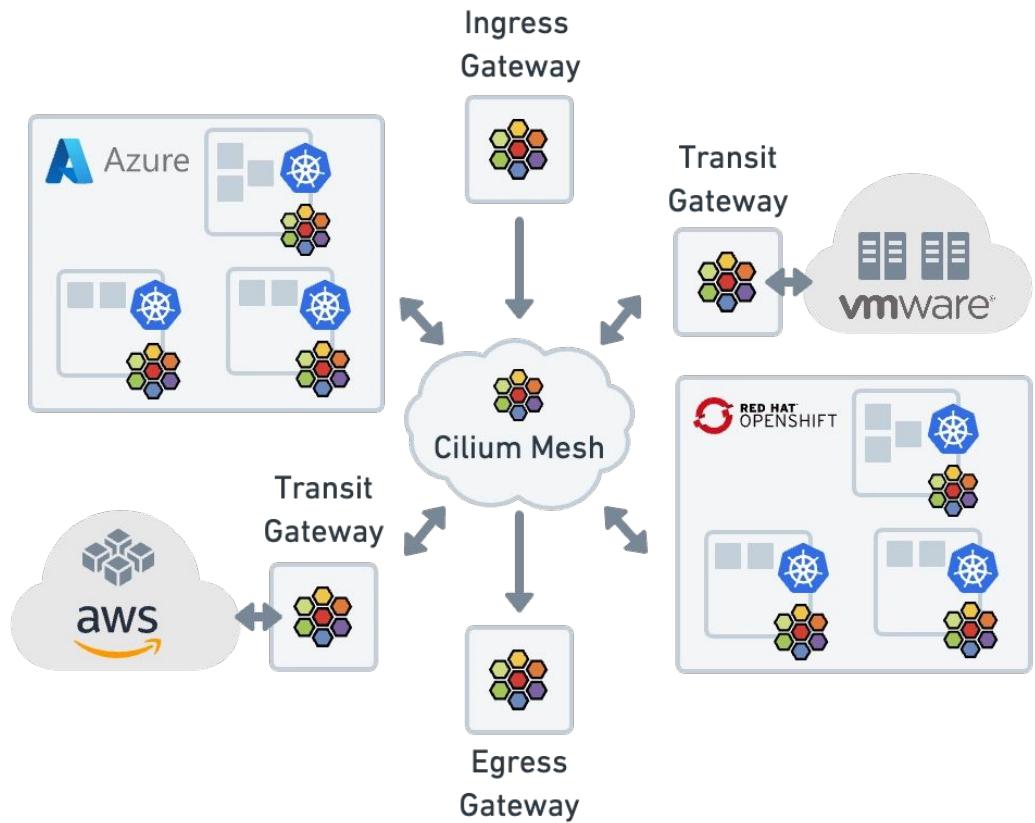


Gateway API →



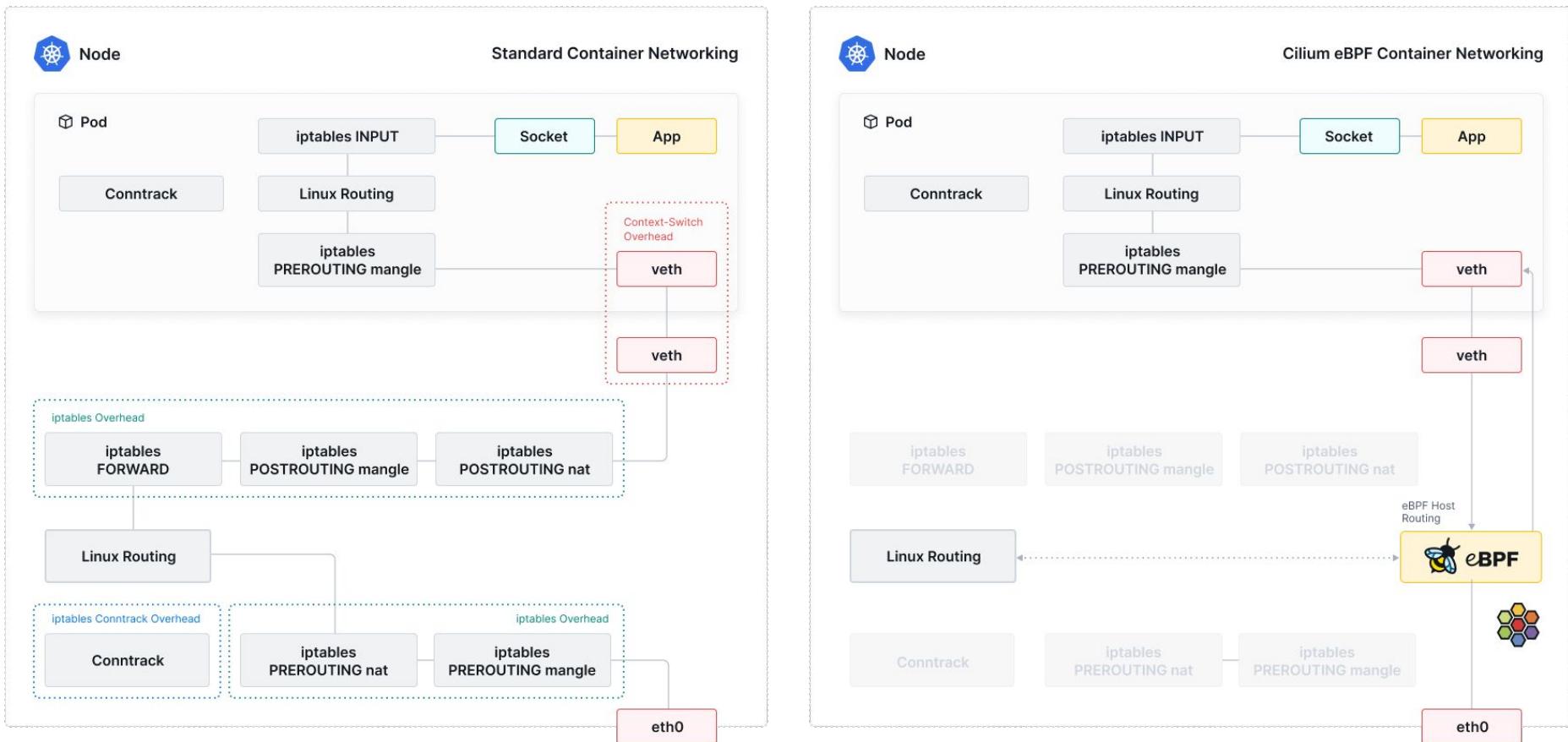
Multicast →

What is Cilium ?



Introduction to Cilium Network

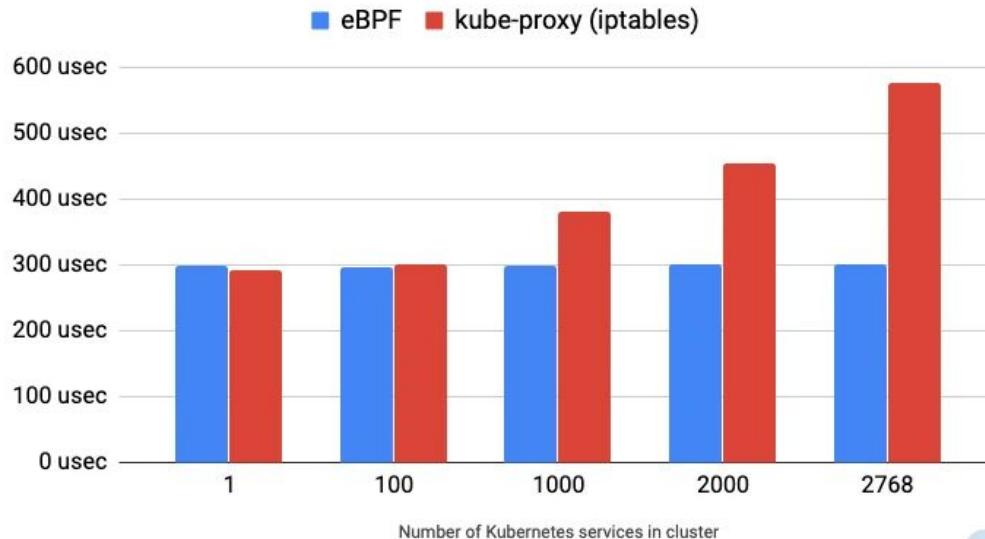
What is Cilium?



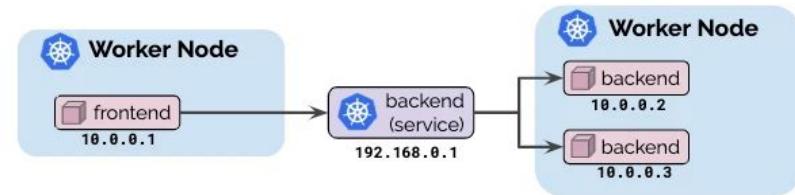
Introduction to Cilium Network

What is Cilium?

HTTP request latency via k8s service (usec)



Number of Kubernetes services in cluster



Network-based load-balancing

→ connect("192.168.0.1")

Local Socket

#1 10.0.0.1→192.168.0.1
#2 10.0.0.1←192.168.0.1
#3 10.0.0.1→192.168.0.1
#n [...]

Network DNAT

10.0.0.1→10.0.0.2
10.0.0.1←10.0.0.2
10.0.0.1→10.0.0.2
[...]

Remote Socket

10.0.0.1→10.0.0.2
10.0.0.1←10.0.0.2
10.0.0.1→10.0.0.2
[...]

DNAT

Socket-based load-balancing

→ connect("192.168.0.1")

DNAT

Local Socket

#1 10.0.0.1→10.0.0.2
#2 10.0.0.1←10.0.0.2
#3 10.0.0.1→10.0.0.2
#n [...]

Remote Socket

10.0.0.1→10.0.0.2
10.0.0.1←10.0.0.2
10.0.0.1→10.0.0.2
[...]

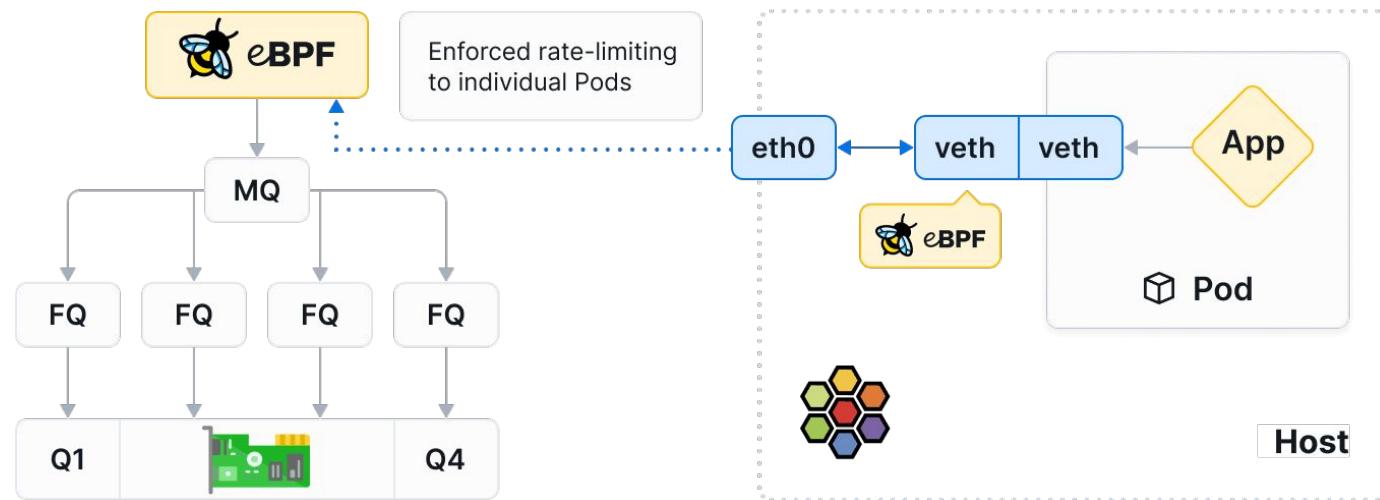
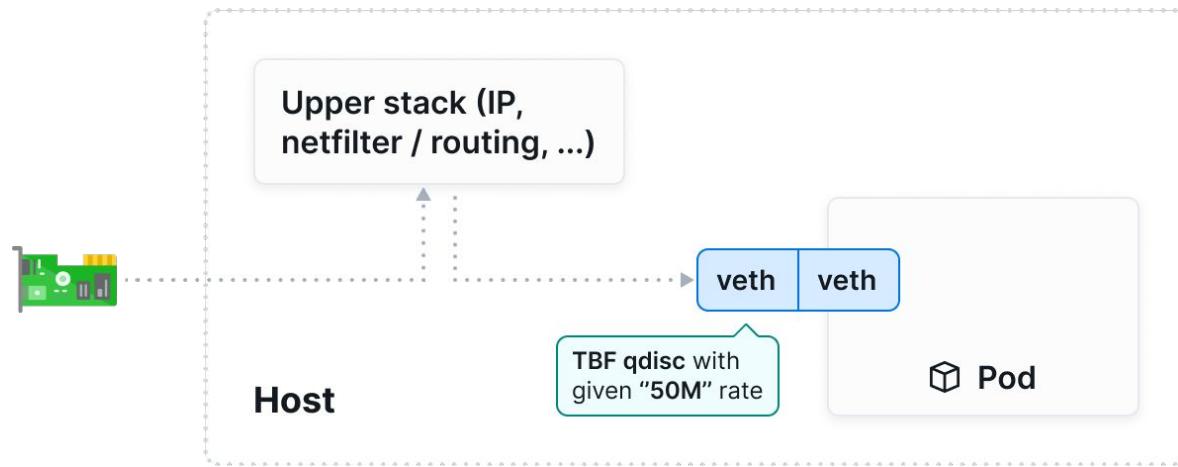
Ref:https://cilium.io/blog/2019/08/20/cilium-1_

Introduction to Cilium Network

What is Cilium?



kubernetes.io/ingress-bandwidth: "50M"



What is Cilium?

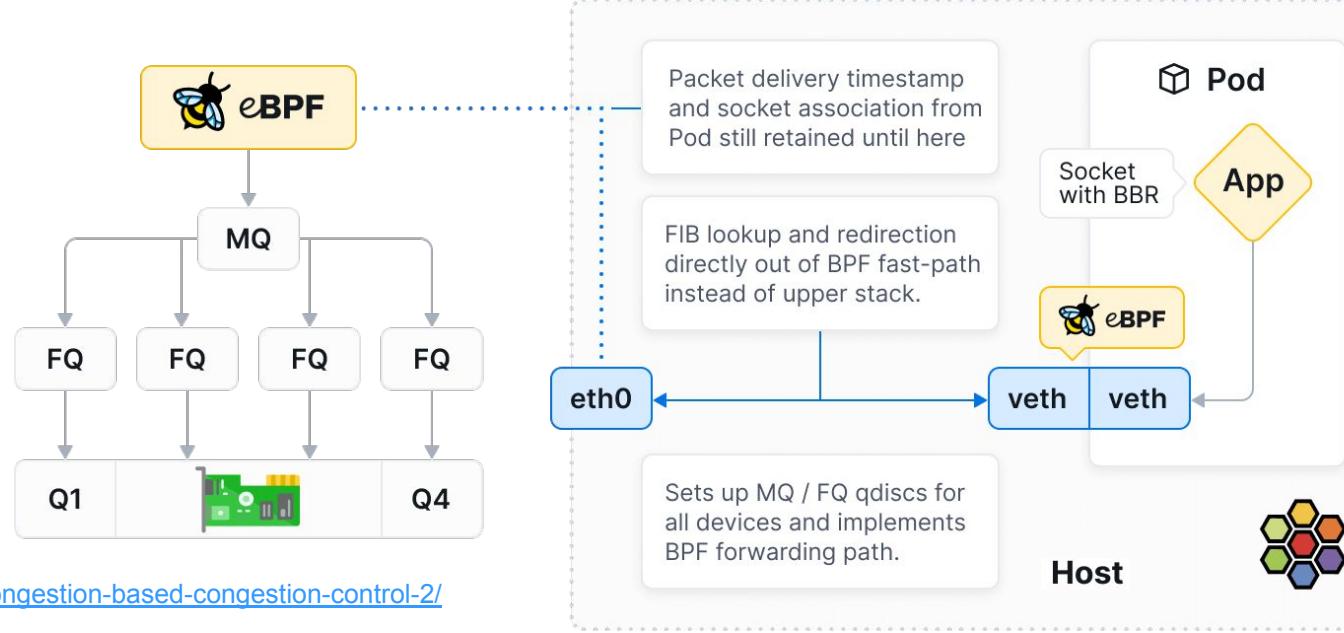
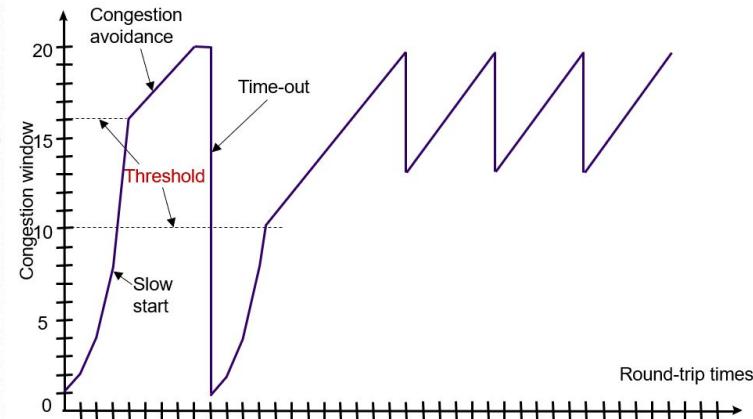
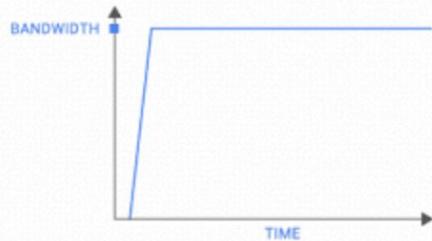
TCP before BBR

Today's Internet is not moving data as well as it should. TCP sends data at lower bandwidth because the 1980s-era algorithm assumes that packet loss means network congestion.



TCP BBR

BBR models the network to send as fast as the available bandwidth and is 2700x faster than previous TCPs on a 10Gb, 100ms link with 1% loss. BBR powers google.com, youtube.com, and apps using Google Cloud Platform services.



Ref:<https://research.google/pubs/bbr-congestion-based-congestion-control-2/>

Introduction to Cilium Network

What is Cilium?

- Observability



Service Map →



Metrics & Tracing Export →



Identity-aware L3/L4/DNS
Network Flow Logs →



Advanced Network Protocol
Visibility →

- Security



Transparent Encryption →

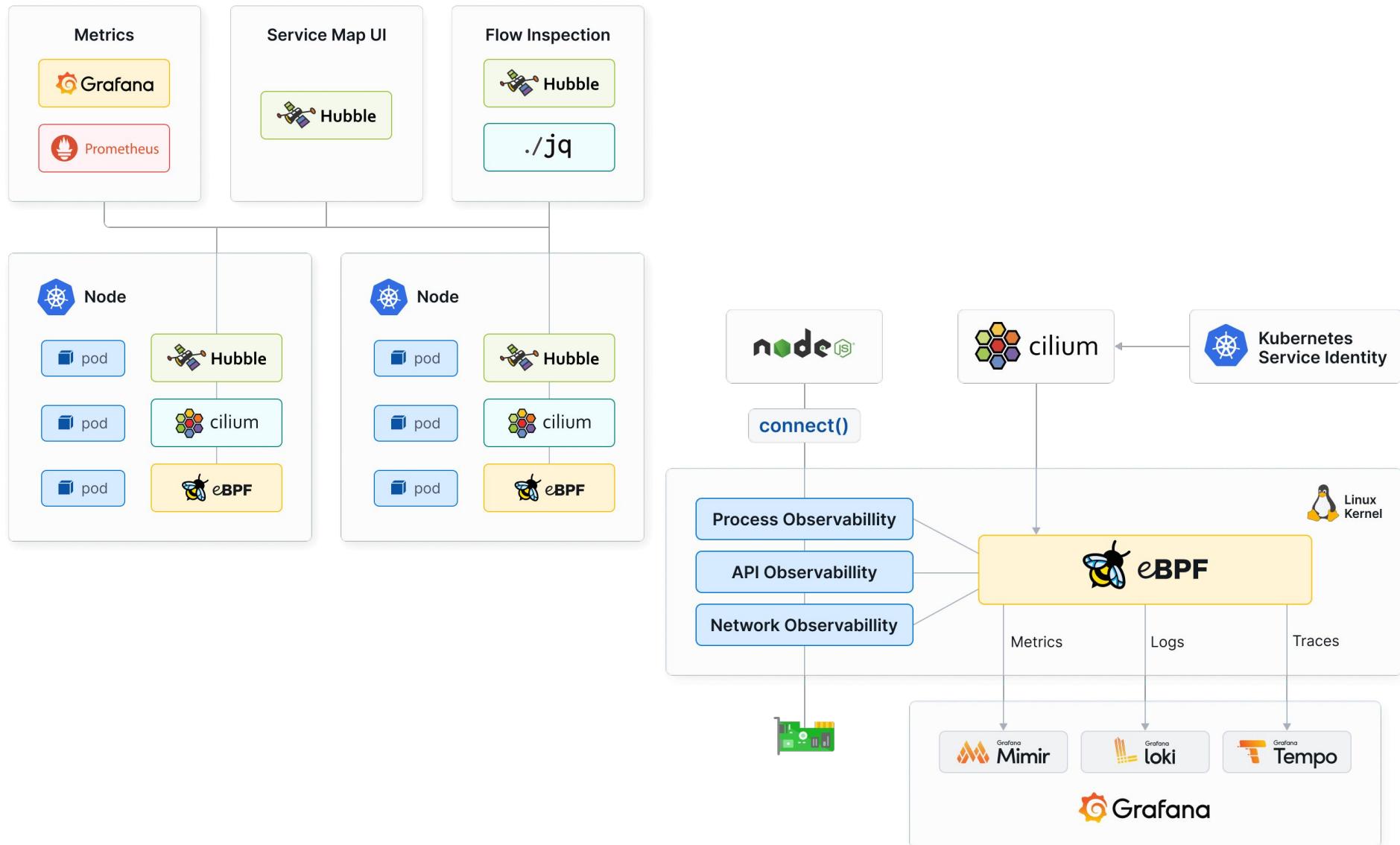


Network Policy →



Runtime Enforcement →

What is Cilium?



Introduction to Cilium Network

Platform and Application Teams

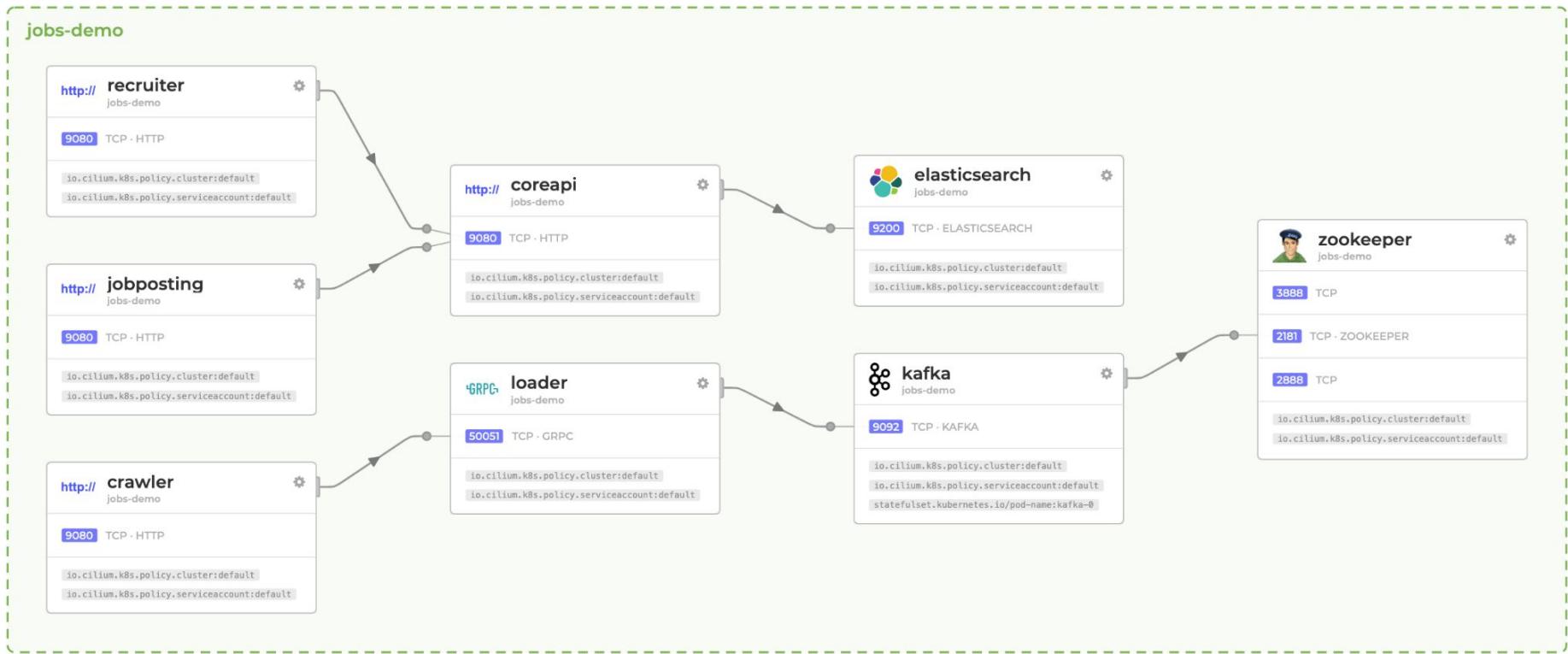
What is Cilium?

```
[ubuntu@ip-10-21-1-233:~$ hubble observe
Feb 12 01:36:06.300: 10.0.4.83:14888 -> 10.0.3.10:4240 to-endpoint FORWARDED (TCP Flags: ACK)
Feb 12 01:36:06.556: 10.0.3.10:4240 <-> 10.0.5.110:52348 to-overlay FORWARDED (TCP Flags: ACK)
Feb 12 01:36:06.556: 10.0.5.110:52348 -> 10.0.3.10:4240 to-endpoint FORWARDED (TCP Flags: ACK)
Feb 12 01:36:06.864: 10.0.3.142:15558 <-> 10.0.0.25:4240 to-overlay FORWARDED (TCP Flags: ACK)
Feb 12 01:36:06.870: 10.0.3.142:15558 -> 10.0.0.25:4240 to-endpoint FORWARDED (TCP Flags: ACK)
Feb 12 01:36:06.927: 10.0.3.142:58984 <-> 10.0.4.148:4240 to-overlay FORWARDED (TCP Flags: ACK)
Feb 12 01:36:06.935: 10.0.1.57:52076 <-> 10.0.4.148:4240 to-overlay FORWARDED (TCP Flags: ACK)
Feb 12 01:36:06.946: ingress-nginx/ingress-nginx-controller-8cf5559f8-h2sr7:443 <-> 10.0.1.57:46590 to-overlay FORWARDED (TCP Flags: SYN, ACK)
Feb 12 01:36:06.946: 10.0.1.57:46590 -> ingress-nginx/ingress-nginx-controller-8cf5559f8-h2sr7:443 to-endpoint FORWARDED (TCP Flags: ACK, RST)
Feb 12 01:36:06.946: 10.0.1.57:47310 -> ingress-nginx/ingress-nginx-controller-8cf5559f8-h2sr7:80 to-endpoint FORWARDED (TCP Flags: ACK, RST)
Feb 12 01:36:06.959: 10.0.1.57:47310 <-> ingress-nginx/ingress-nginx-controller-8cf5559f8-h2sr7:80 to-overlay FORWARDED (TCP Flags: SYN)
Feb 12 01:36:06.959: 10.0.1.57:46590 <-> ingress-nginx/ingress-nginx-controller-8cf5559f8-h2sr7:443 to-overlay FORWARDED (TCP Flags: SYN)
Feb 12 01:36:06.960: 10.0.1.57:46590 <-> ingress-nginx/ingress-nginx-controller-8cf5559f8-h2sr7:443 to-overlay FORWARDED (TCP Flags: ACK, RST)
Feb 12 01:36:06.960: 10.0.1.57:47310 <-> ingress-nginx/ingress-nginx-controller-8cf5559f8-h2sr7:80 to-overlay FORWARDED (TCP Flags: ACK, RST)
Feb 12 01:36:07.019: 10.0.3.142:16908 <-> 10.0.5.215:4240 to-overlay FORWARDED (TCP Flags: ACK)
Feb 12 01:36:07.026: 10.0.1.57:48222 <-> 10.0.5.215:4240 to-overlay FORWARDED (TCP Flags: ACK)
Feb 12 01:36:07.036: 10.0.3.142:24910 <-> 10.0.1.248:4240 to-overlay FORWARDED (TCP Flags: ACK)
Feb 12 01:36:07.045: 10.0.1.248:4240 <-> 10.0.3.142:24910 to-overlay FORWARDED (TCP Flags: ACK)
Feb 12 01:36:07.045: 10.0.1.57:14258 -> 10.0.1.248:4240 to-stack FORWARDED (TCP Flags: ACK)
Feb 12 01:36:07.045: 10.0.1.57:14258 -> 10.0.1.248:4240 to-endpoint FORWARDED (TCP Flags: ACK)
Feb 12 01:36:07.045: 10.0.3.142:24910 -> 10.0.1.248:4240 to-endpoint FORWARDED (TCP Flags: ACK)
Feb 12 01:36:07.068: 10.0.3.10:4240 <-> 10.0.1.57:39218 to-overlay FORWARDED (TCP Flags: ACK)
Feb 12 01:36:07.068: 10.0.3.142:55422 -> 10.0.3.10:4240 to-stack FORWARDED (TCP Flags: ACK)
Feb 12 01:36:07.068: 10.0.3.142:55422 -> 10.0.3.10:4240 to-endpoint FORWARDED (TCP Flags: ACK)
Feb 12 01:36:07.068: 10.0.1.57:39218 -> 10.0.3.10:4240 to-endpoint FORWARDED (TCP Flags: ACK)
Feb 12 01:36:07.077: 10.0.1.57:39218 <-> 10.0.3.10:4240 to-overlay FORWARDED (TCP Flags: ACK)
Feb 12 01:36:07.083: 10.0.2.154:4240 <-> 10.0.3.142:30542 to-overlay FORWARDED (TCP Flags: ACK)
Feb 12 01:36:07.083: 10.0.1.57:36562 -> 10.0.2.154:4240 to-endpoint FORWARDED (TCP Flags: ACK)
Feb 12 01:36:07.083: 10.0.3.142:30542 -> 10.0.2.154:4240 to-endpoint FORWARDED (TCP Flags: ACK)
Feb 12 01:36:07.084: 10.0.3.142:30542 <-> 10.0.2.154:4240 to-overlay FORWARDED (TCP Flags: ACK)
Feb 12 01:36:07.092: 10.0.1.57:36562 <-> 10.0.2.154:4240 to-overlay FORWARDED (TCP Flags: ACK)
```

Ref:<https://github.com/cilium/hubble>

Introduction to Cilium Network

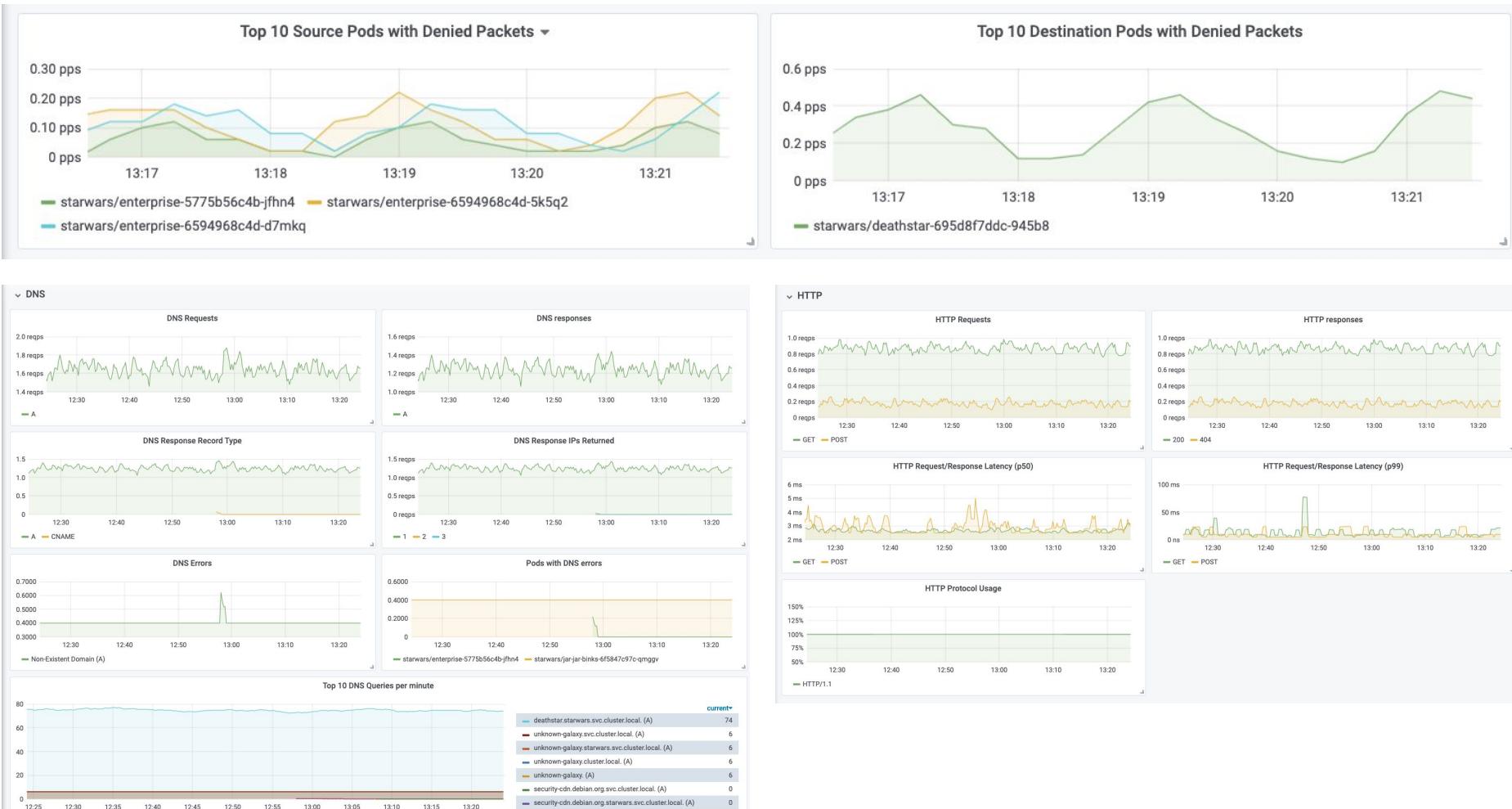
What is Cilium?



Ref:<https://github.com/cilium/hubble>

Introduction to Cilium Network

What is Cilium?



Ref:<https://github.com/cilium/hubble>

Introduction to Cilium Network

What is Cilium?

- Security



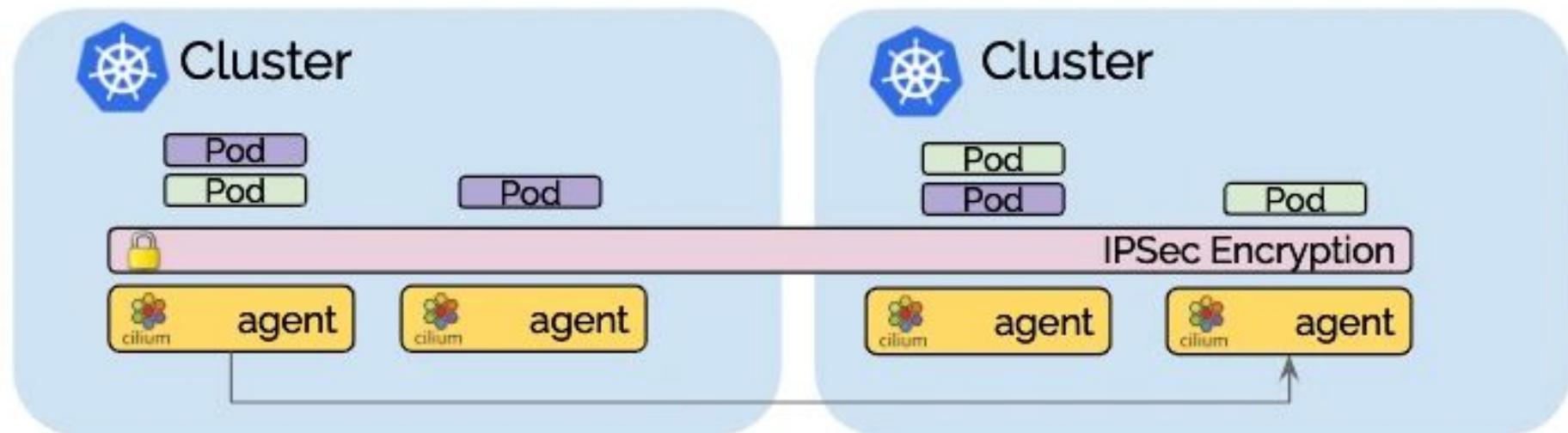
Transparent Encryption →



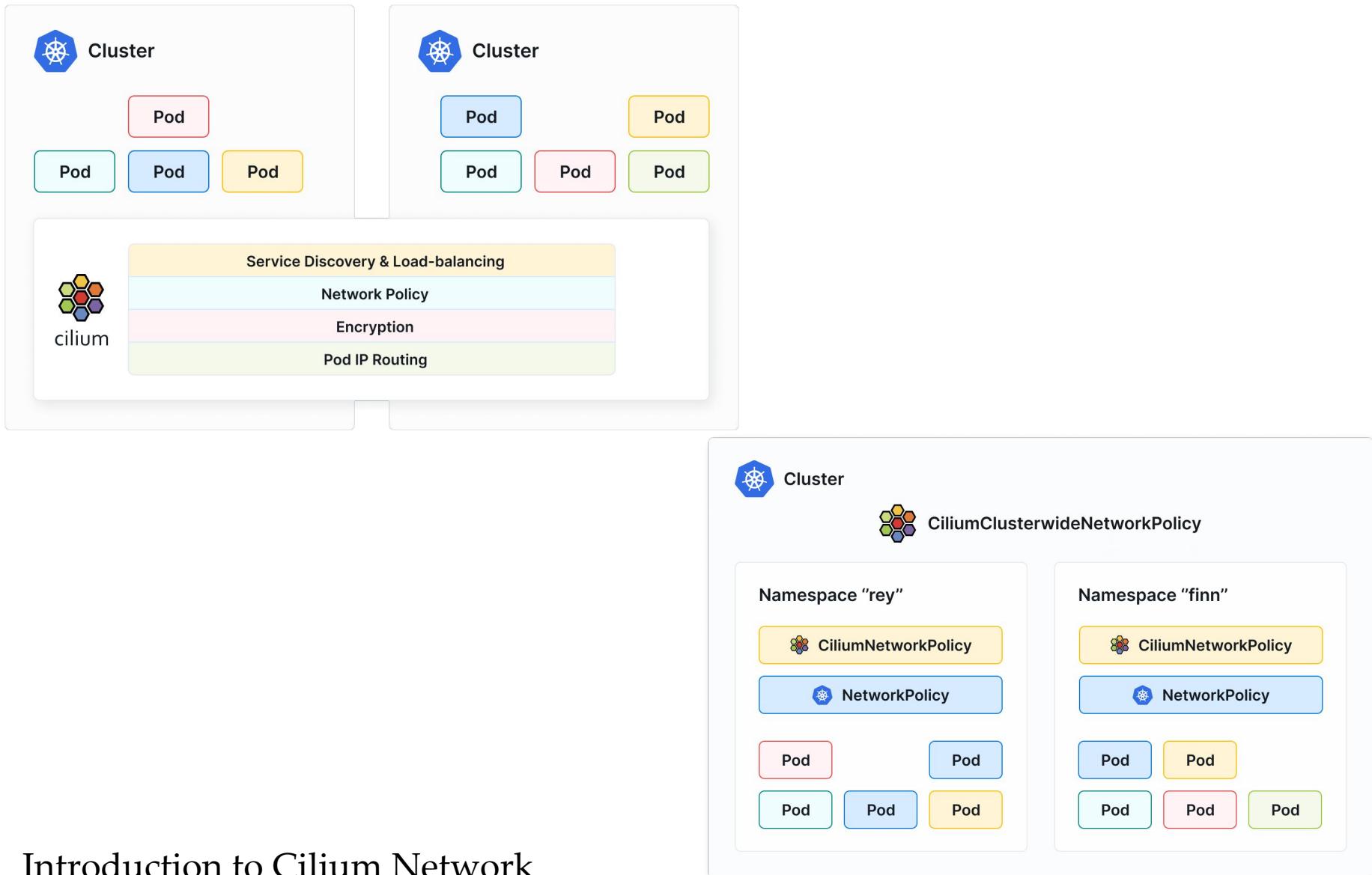
Network Policy →



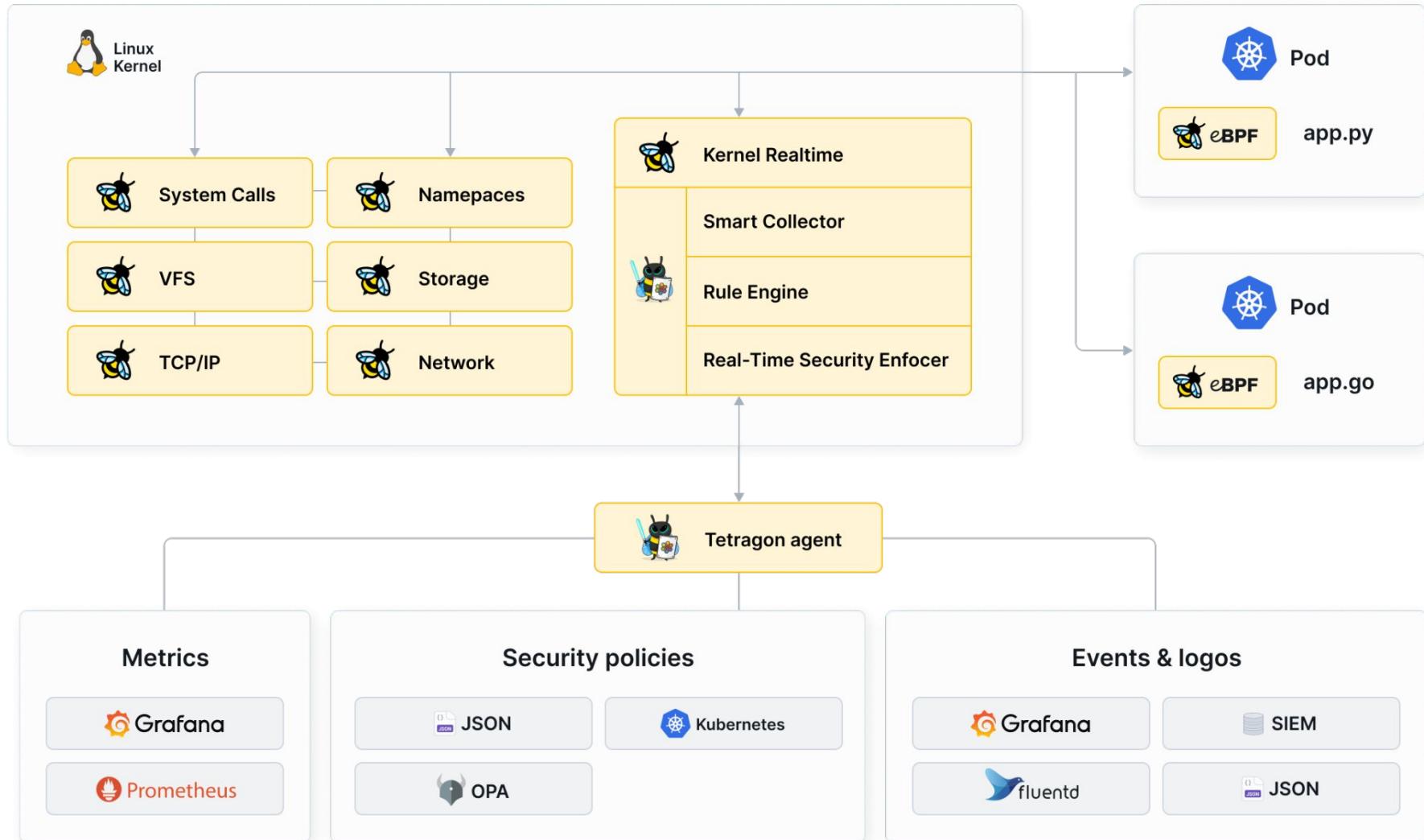
Runtime Enforcement →



What is Cilium?

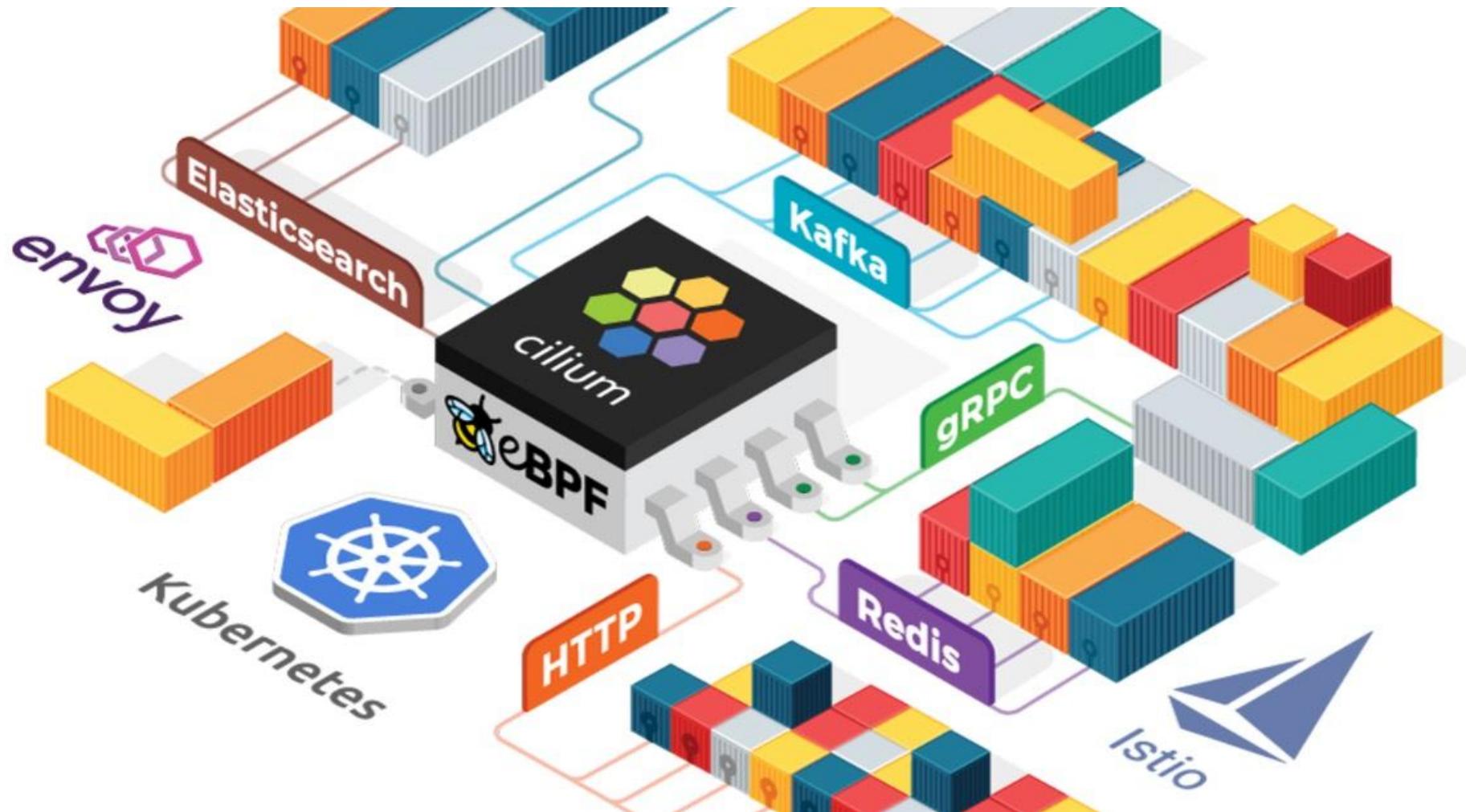


What is Cilium?



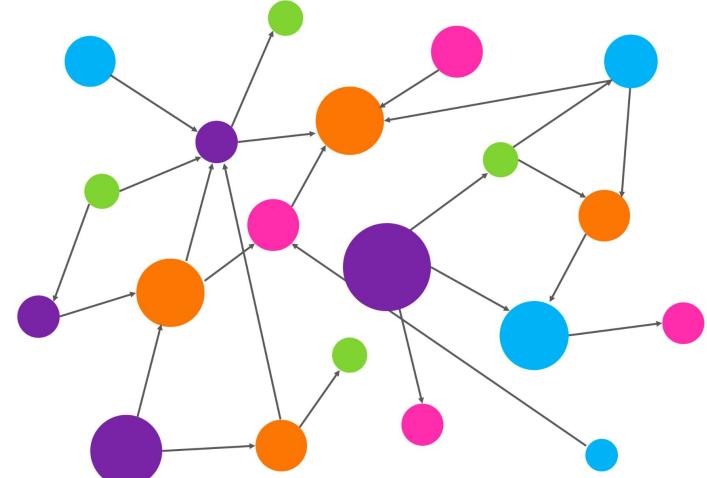
Introduction to Cilium Network

Why this matter ?

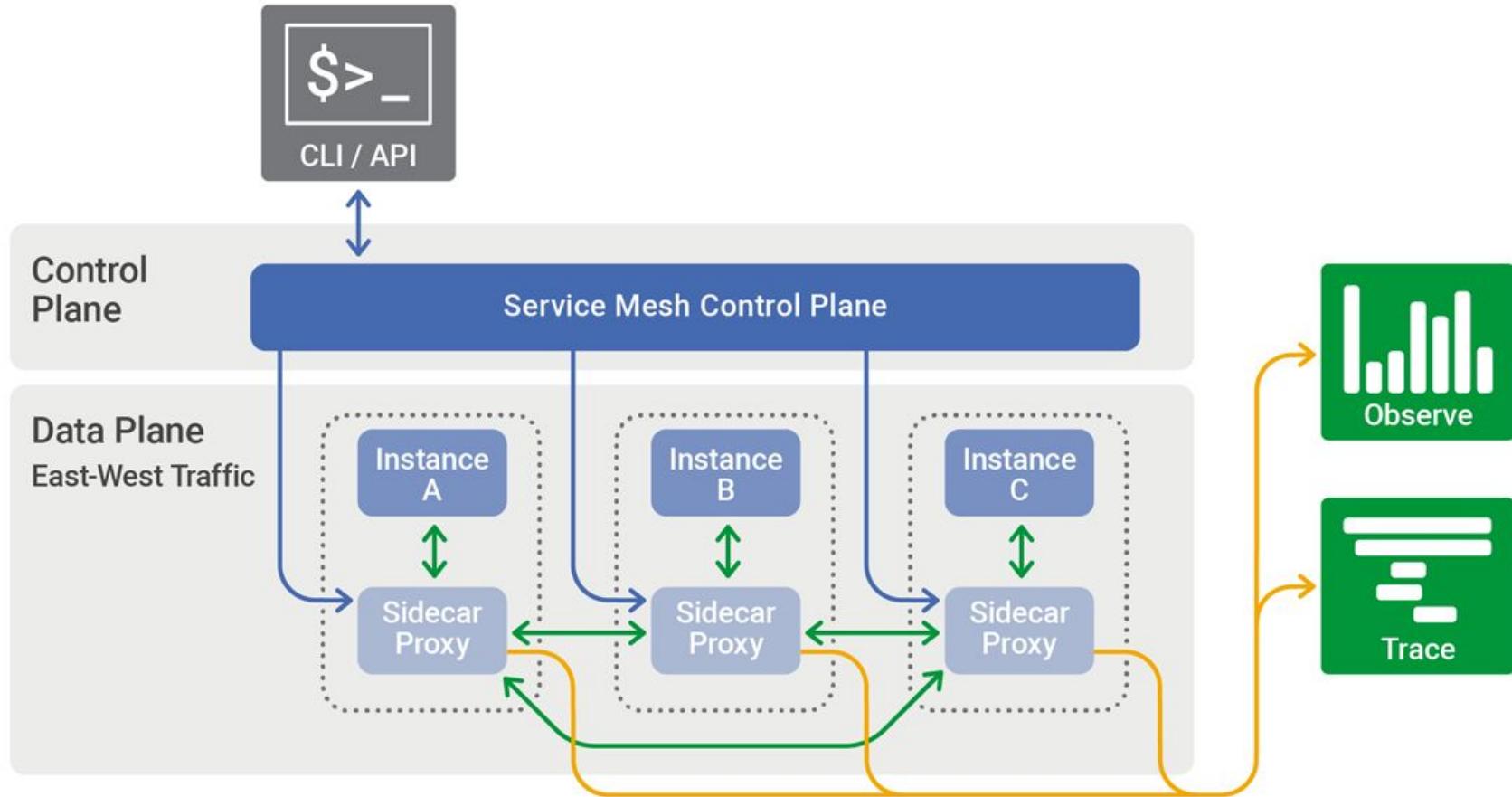


Why this matter ?

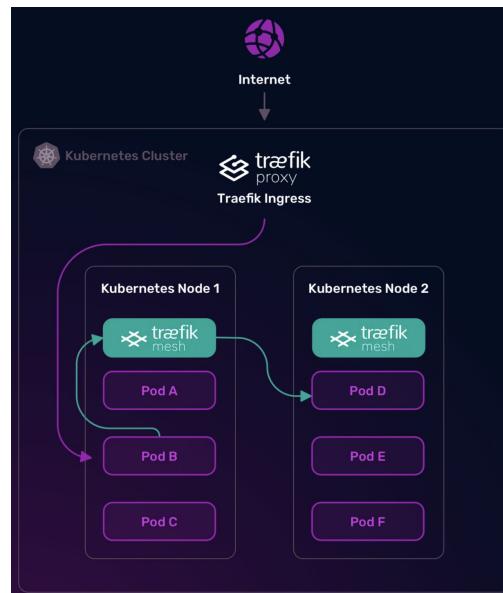
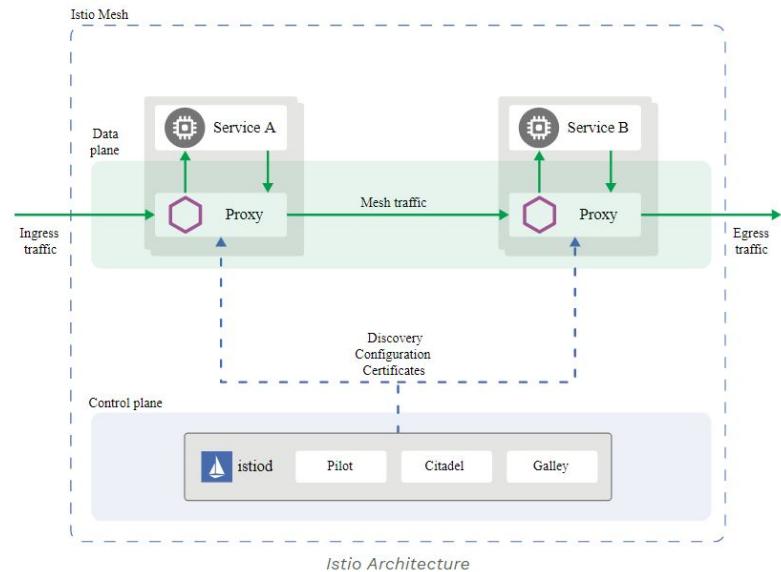
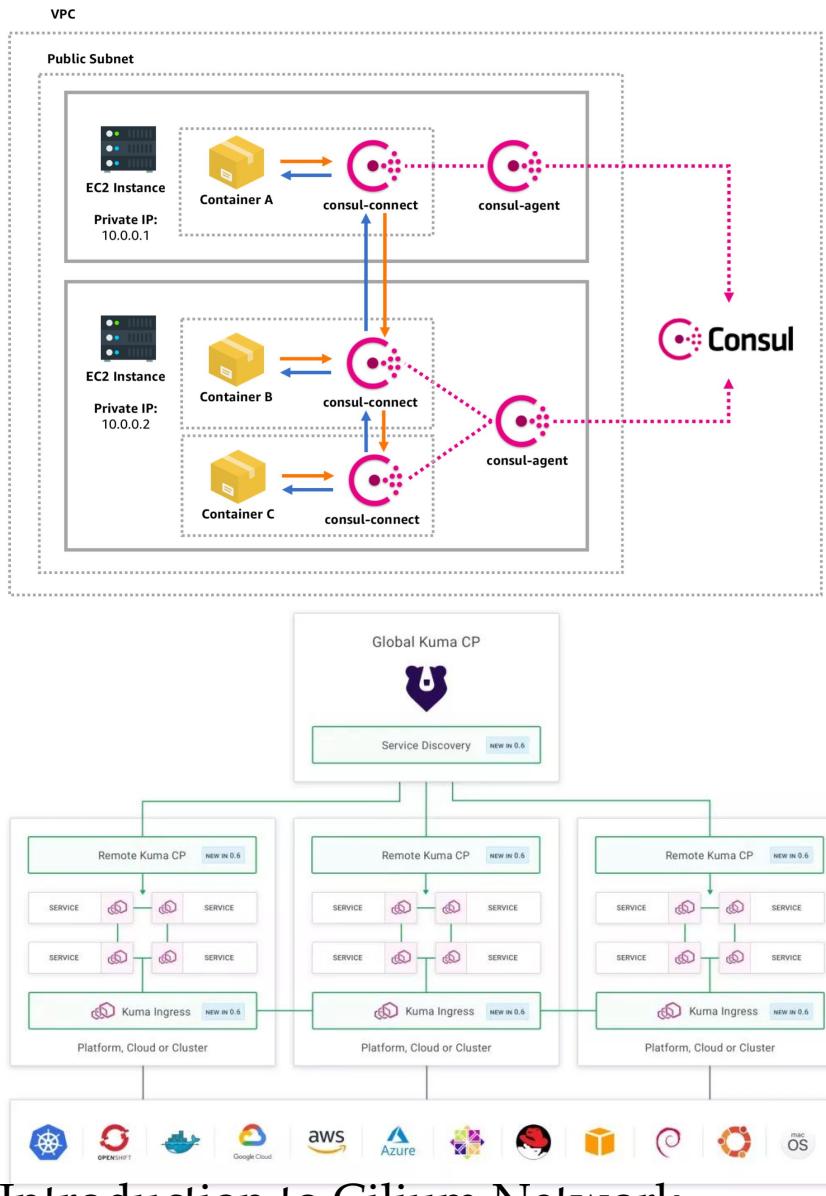
- **Cilium** given some special feature on network / observability / security than other CNI network on Kubernetes
- But why this matter ?
 - **Case Study:** Service Mesh & Observability
 - Many solution in option. How about Kuma/Istio/Dynatrace etc.



Service Mesh & Observability



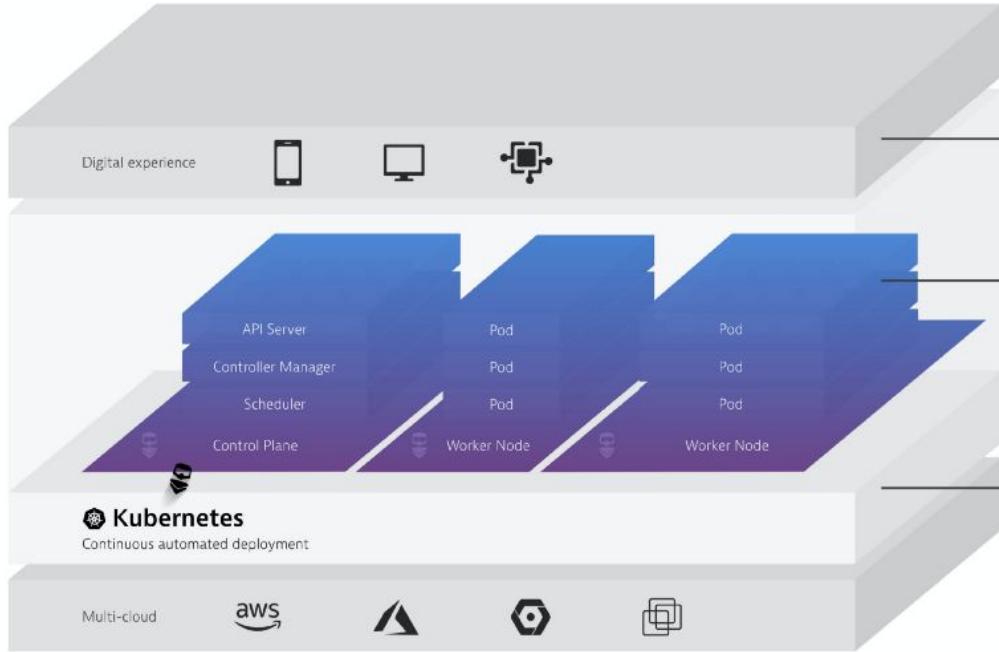
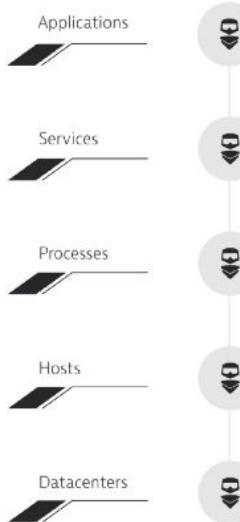
Service Mesh & Observability



Introduction to Cilium Network

Service Mesh & Observability

OneAgent deploys automatically via Operator to all layers and technologies in your environment



Monitor, analyze and optimize every digital interaction

Real-time auto discovery through OneAgent injection into containers without code or image changes

Automatic and continuous deployment of Dynatrace OneAgent to all cluster nodes

Full integration with all major cloud platforms

Service Mesh & Observability

The collage consists of four screenshots:

- Kiali (Top Left):** Shows a service graph for the bookinfo namespace. It includes nodes like istio-ingressgateway, productpage, reviews, ratings, and mongoDB, connected by various routes and annotations such as 'details' and 'reviews'.
- Linkerd (Top Right):** A browser-based interface showing the deployment/product-gateway-api-deploy. It displays metrics for three services: deploy/price-api-deploy, deploy/product-gateway-api-deploy, and deploy/product-api-deploy. For each, it shows Success Rate (SR), Requests Per Second (RPS), and P99 latency.
- Linkerd Metrics (Bottom Right):** A detailed view of the deployment/product-gateway-api-deploy service. It shows live calls and route metrics for requests from 10.244.0.1 to the service. Metrics include count, average response time, and success rate.
- Linkerd PurePath (Bottom Left):** A service flow diagram for nginxForMicroservices. It traces a request through multiple services: nginxForMicroservices, EasyTravel(BackendWebser...), JourneyService, and easyTravel-Business. Each step shows its contribution to the total response time and the number of requests handled.

Introduction to Cilium Network

Why this matter ?

Dave Streb
@dave_strebel

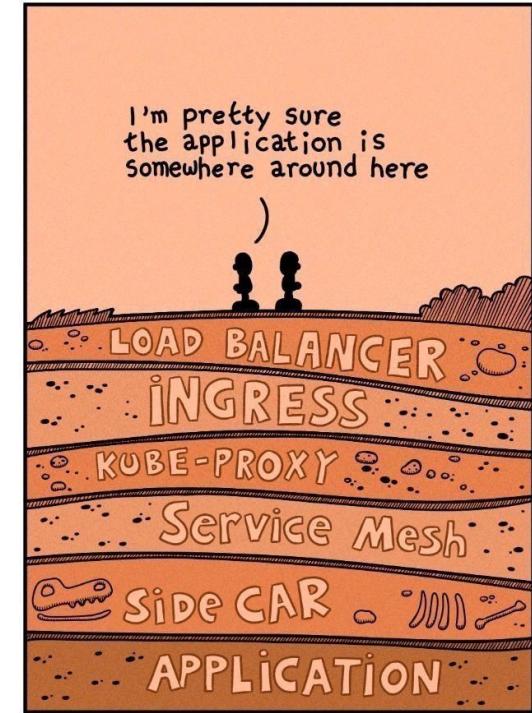
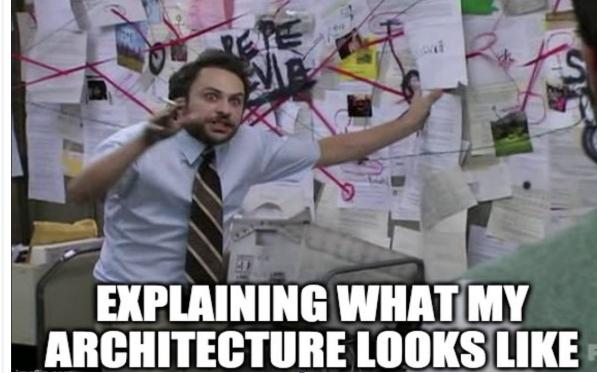
"Finally got Istio into production"



6:21 PM · Sep 17, 2019 from Apple Valley, MN · Tweetbot for iOS

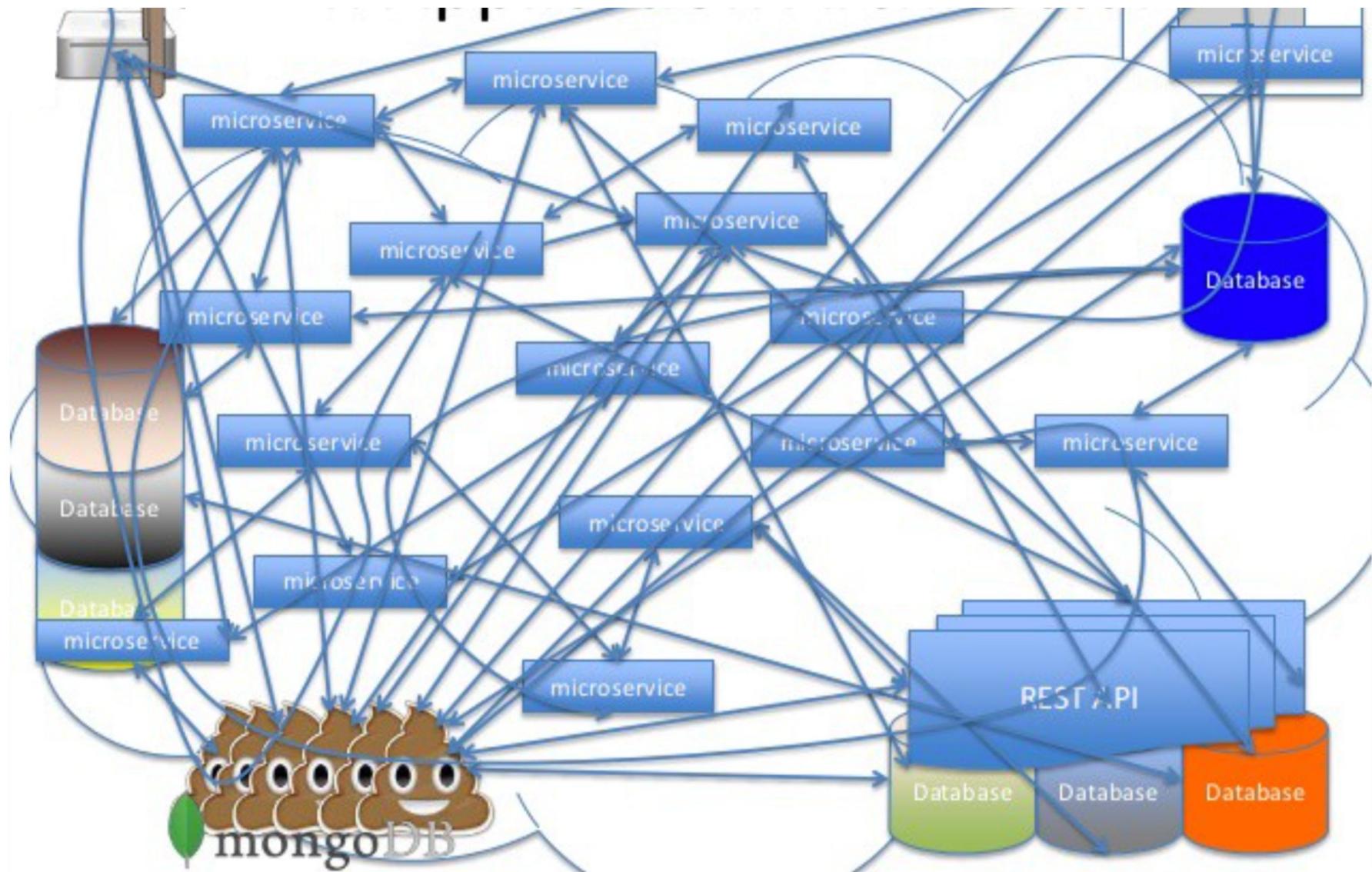


SPRING-BOOT RUNNING UNDER ISTIO
SIDECAR PROXY THROW HTTP 403 ERROR

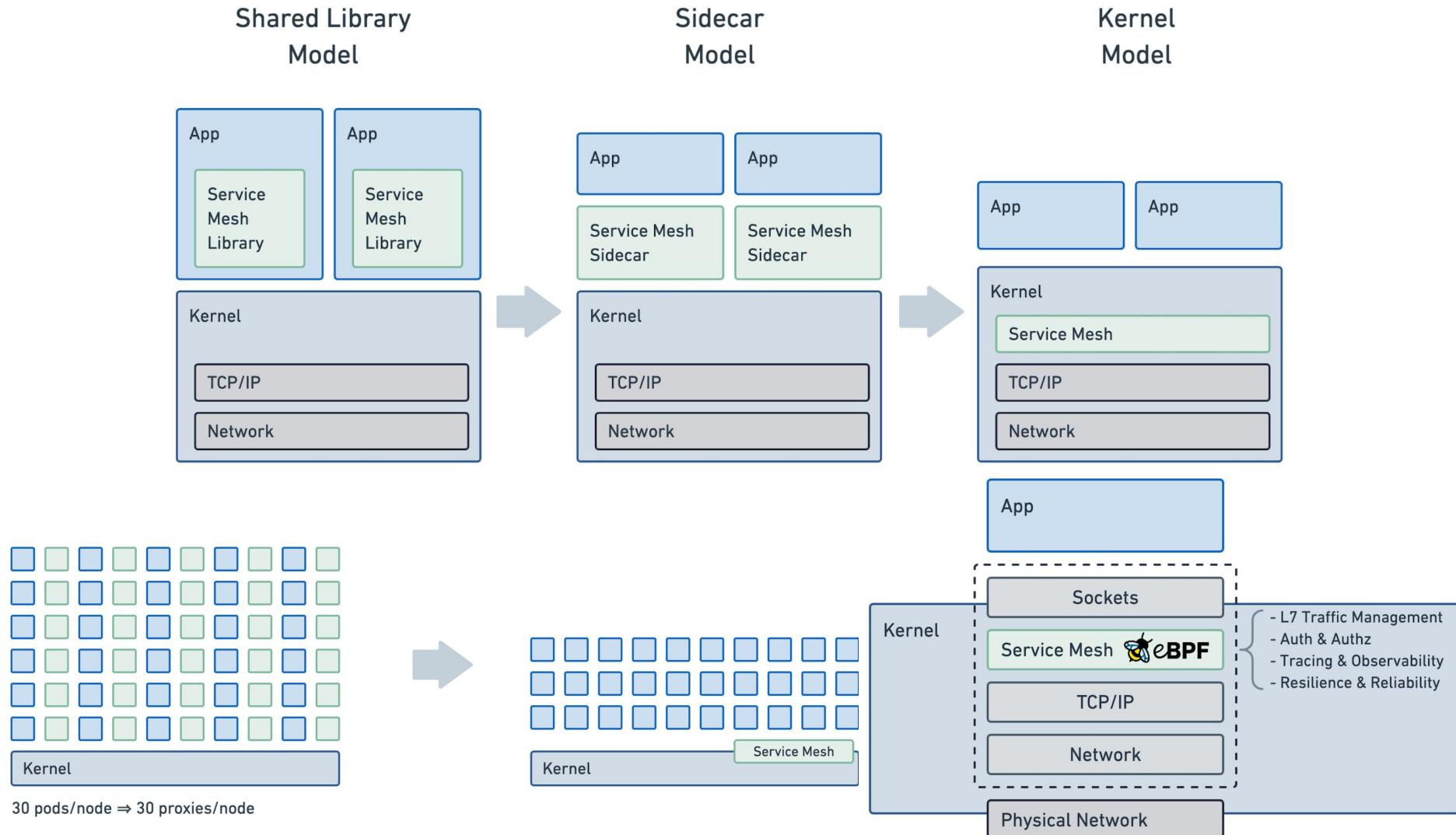


Introduction to Cilium Network

Why this matter ?



Why this matter ?

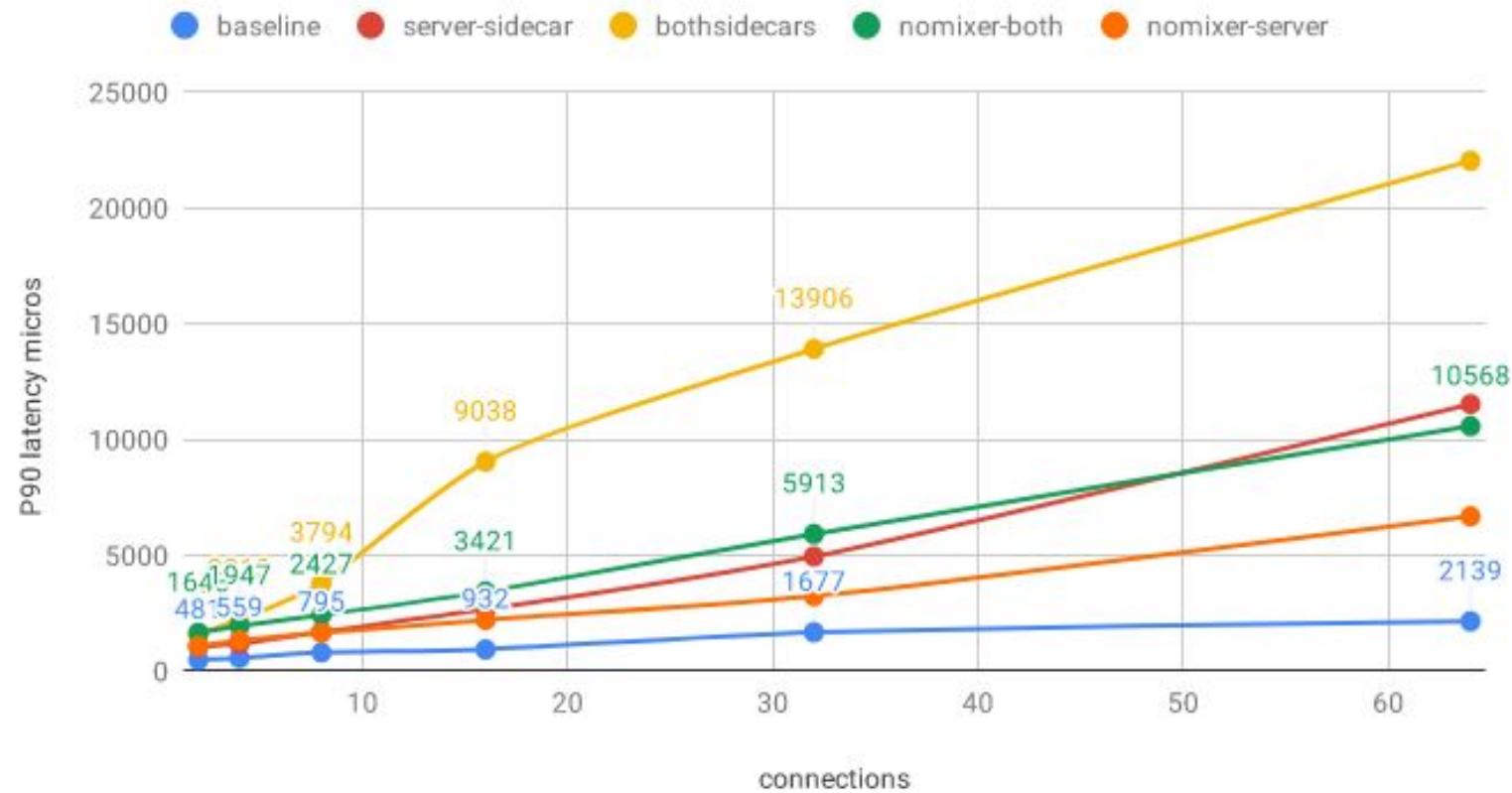


Ref: <https://isovalent.com/blog/post/2021-12-08-ebpf-servicemesh>

Introduction to Cilium Network

Why this matter ?

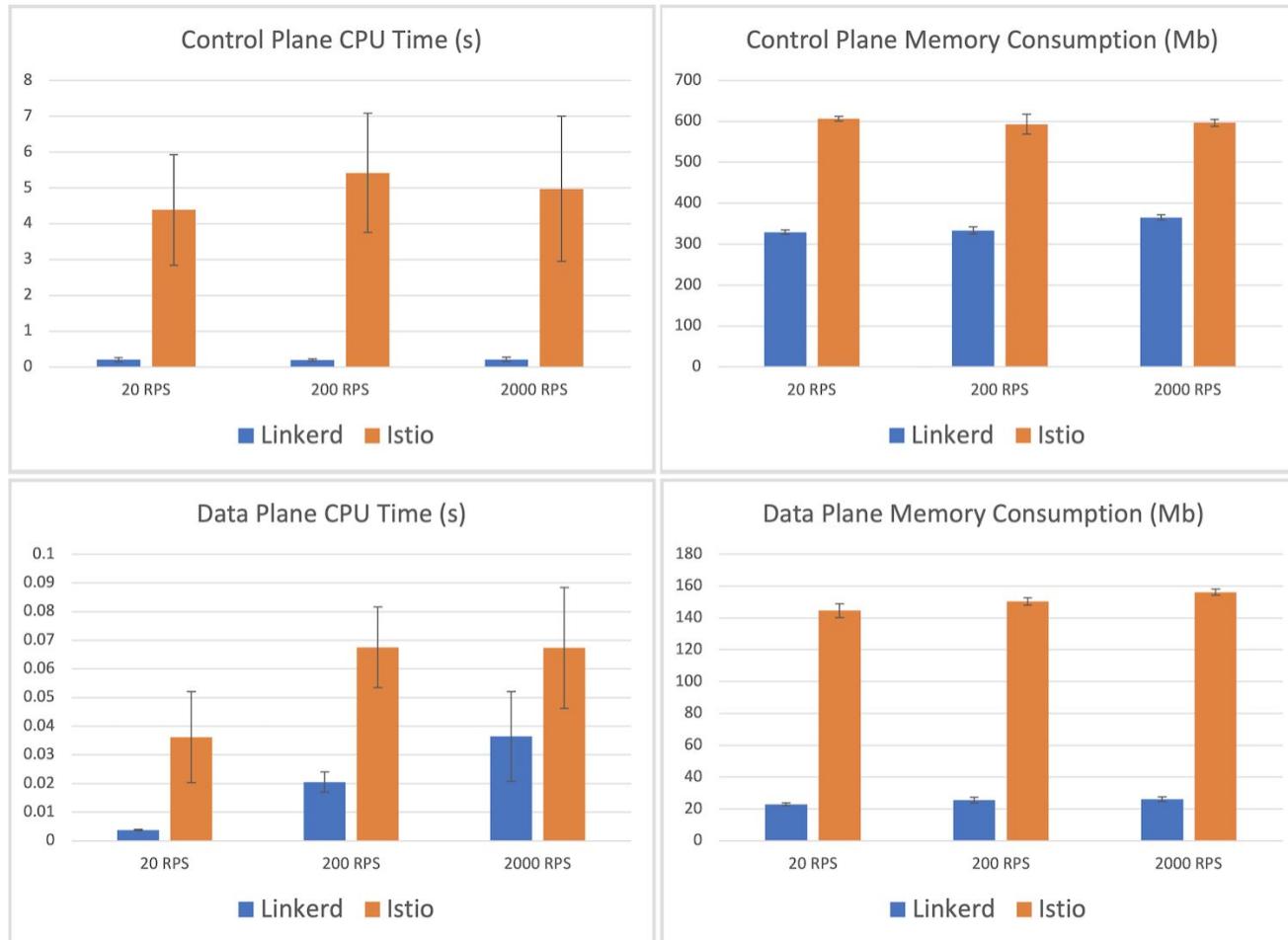
Latency at 1000 rps



Ref:<https://istio.io/v1.2/docs/concepts/performance-and-scalability/>

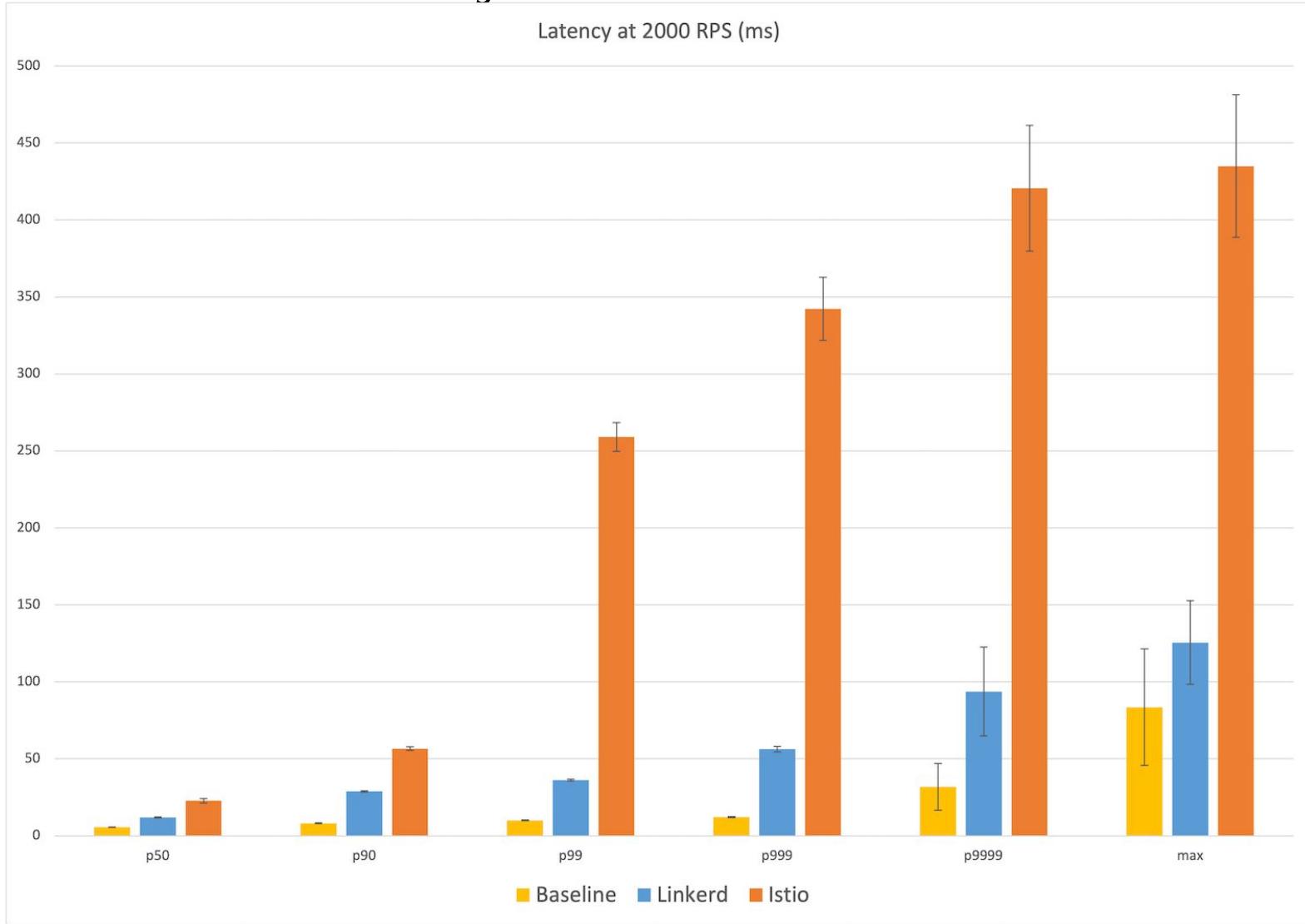
Introduction to Cilium Network

Why this matter ?



Ref:<https://linkerd.io/2021/11/29/linkerd-vs-istio-benchmarks-2021/>

Why this matter ?



<https://linkerd.io/2021/11/29/linkerd-vs-istio-benchmarks-2021/>

Introduction to Cilium Network

Why this matter ?

[About](#)[Blog](#)[News](#)[Get involved](#)[Documentation](#)[Try Istio](#)

Using eBPF for traffic redirection in Istio ambient mode

An alternative approach to redirecting application pod traffic to the per-node ztunnel.

Mar 29, 2023 | By Iris Ding - Intel, Chun Li - Intel

In Istio's new [ambient mode](#), the [istio-cni](#) component running on each Kubernetes worker node is responsible for redirecting application traffic to the zero-trust tunnel (ztunnel) on that node. By default it relies on iptables and [Generic Network Virtualization Encapsulation \(Geneve\)](#) overlay tunnels to achieve this redirection. We have now added support for an eBPF-based method of traffic redirection.

Why eBPF

Although performance considerations are essential in the implementation of Istio ambient mode redirection, it's also important to consider ease of programmability, to enable the implementation of versatile and customized requirements. With eBPF, you can leverage additional context in the kernel to bypass complex routing and simply send packets to their final destination.

Furthermore, eBPF enables deeper visibility and additional context for packets in the kernel, allowing for more efficient and flexible management of data flow compared with iptables.

How it works

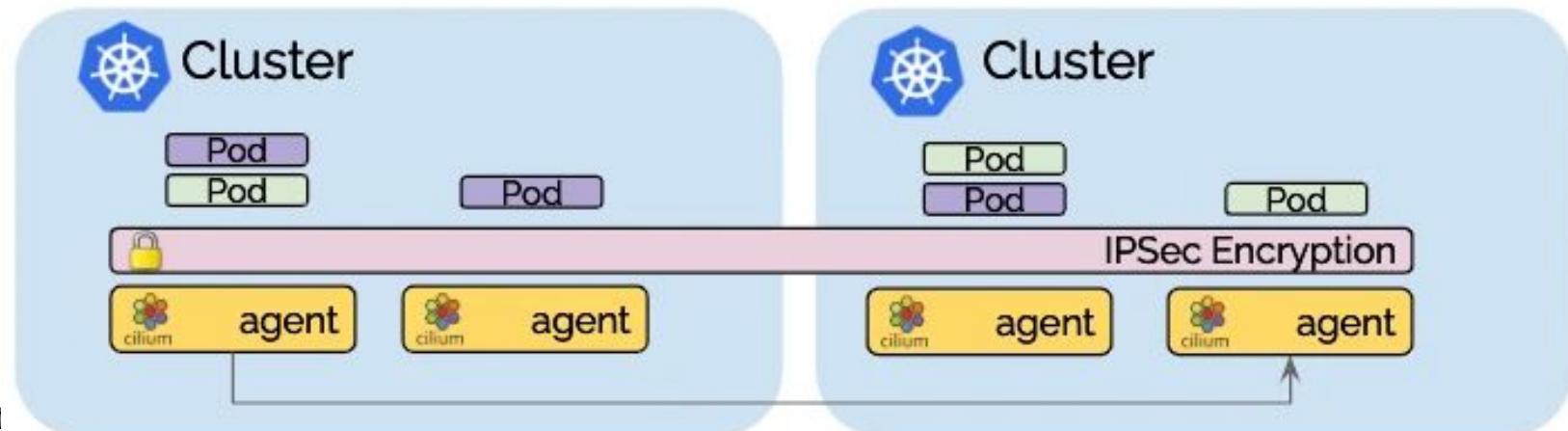
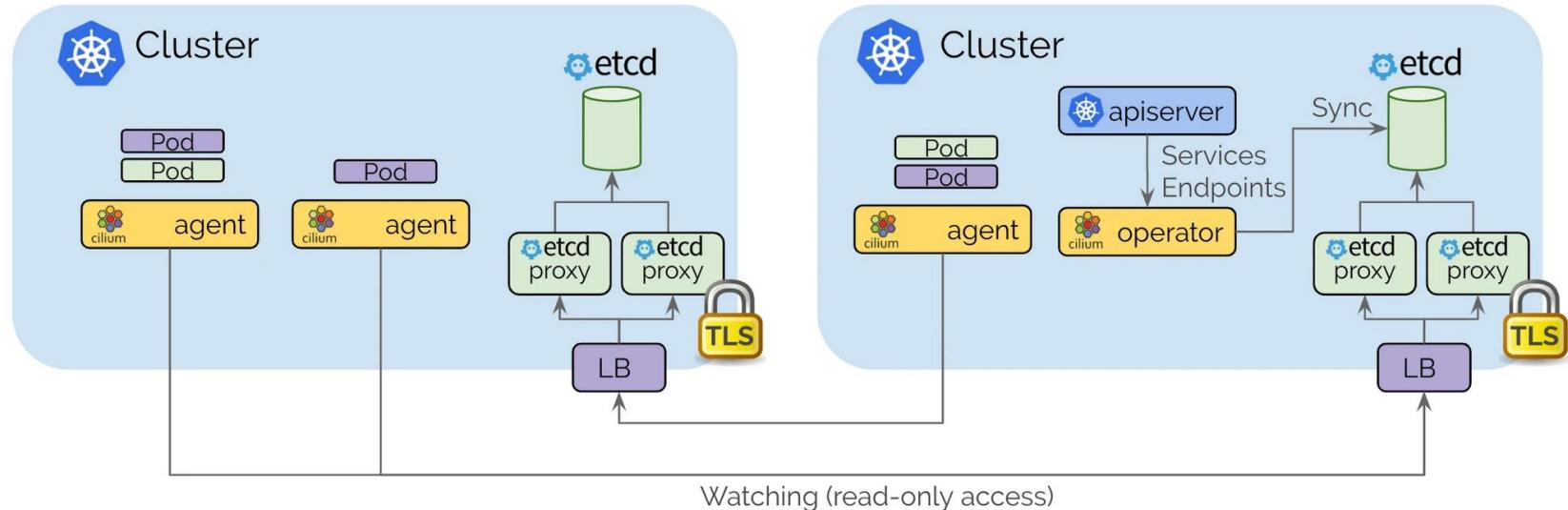
An eBPF program, attached to the [traffic control](#) ingress and egress hook, has been compiled into the Istio CNI component. [istio-cni](#) will watch pod events and attach/detach the eBPF program to other related network interfaces when the pod is moved into or out of ambient mode.

Ref: <https://istio.io/latest/blog/2023/ambient-ebpf-redirection/>

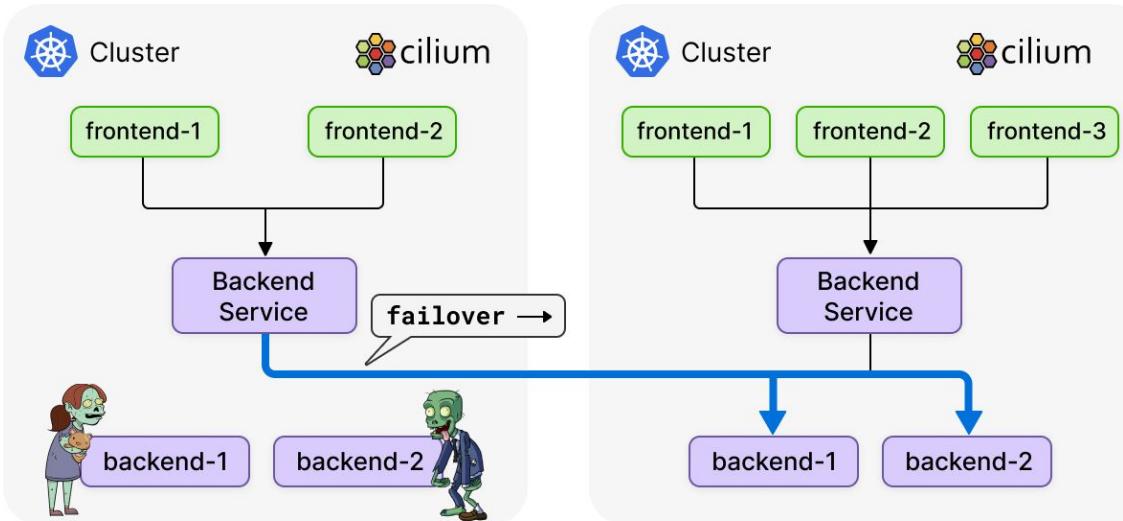
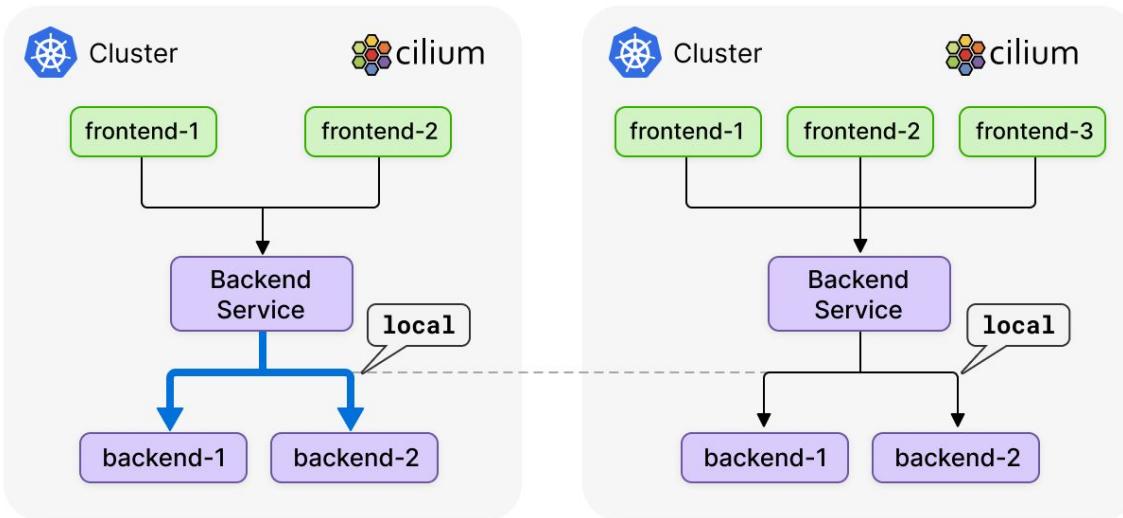
Introduction to Cilium Network

Why this matter ?

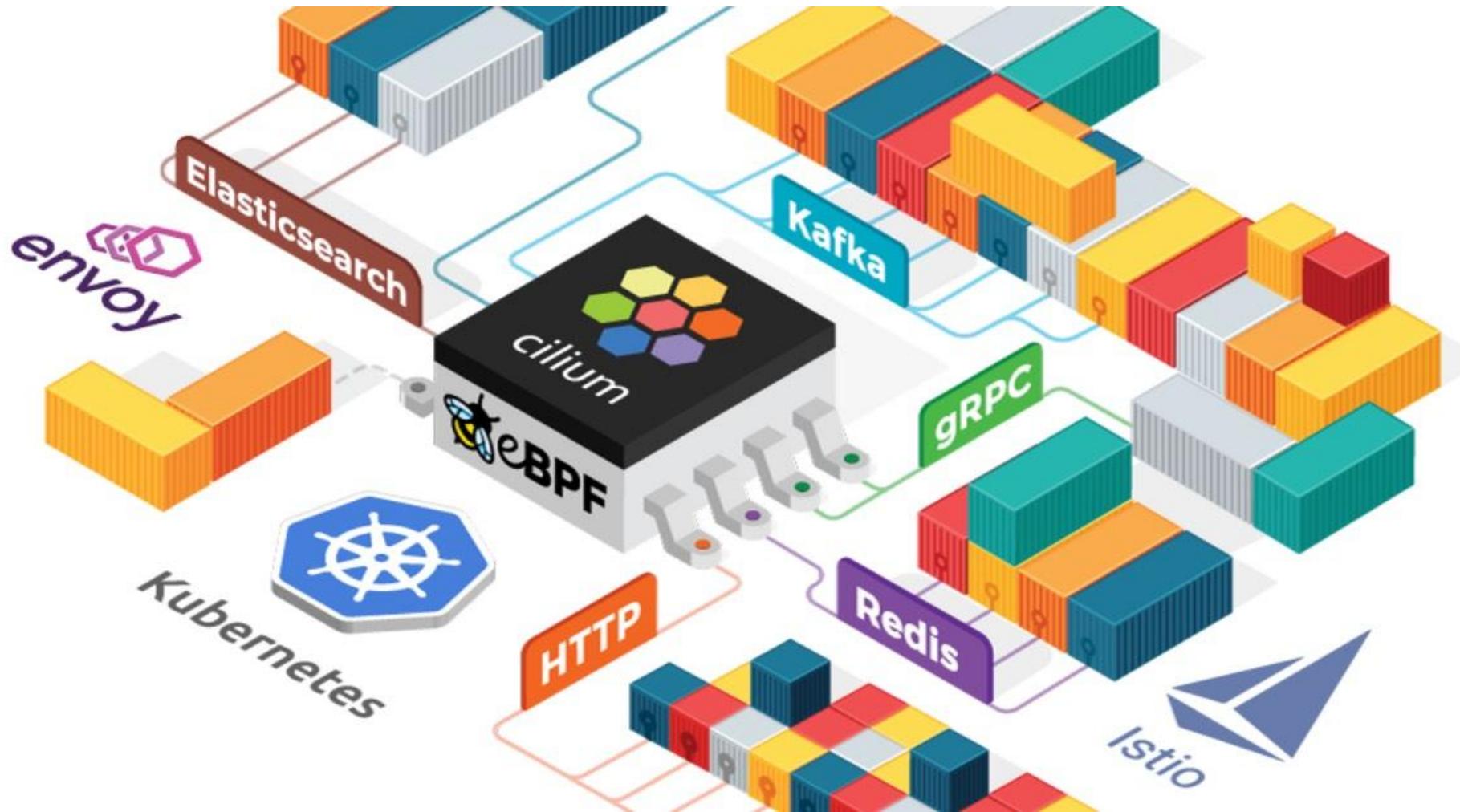
- **Case Study:** Cluster Mesh connection between Kubernetes cluster.



Why this matter ?



Demo Session



Introduction to Cilium Network

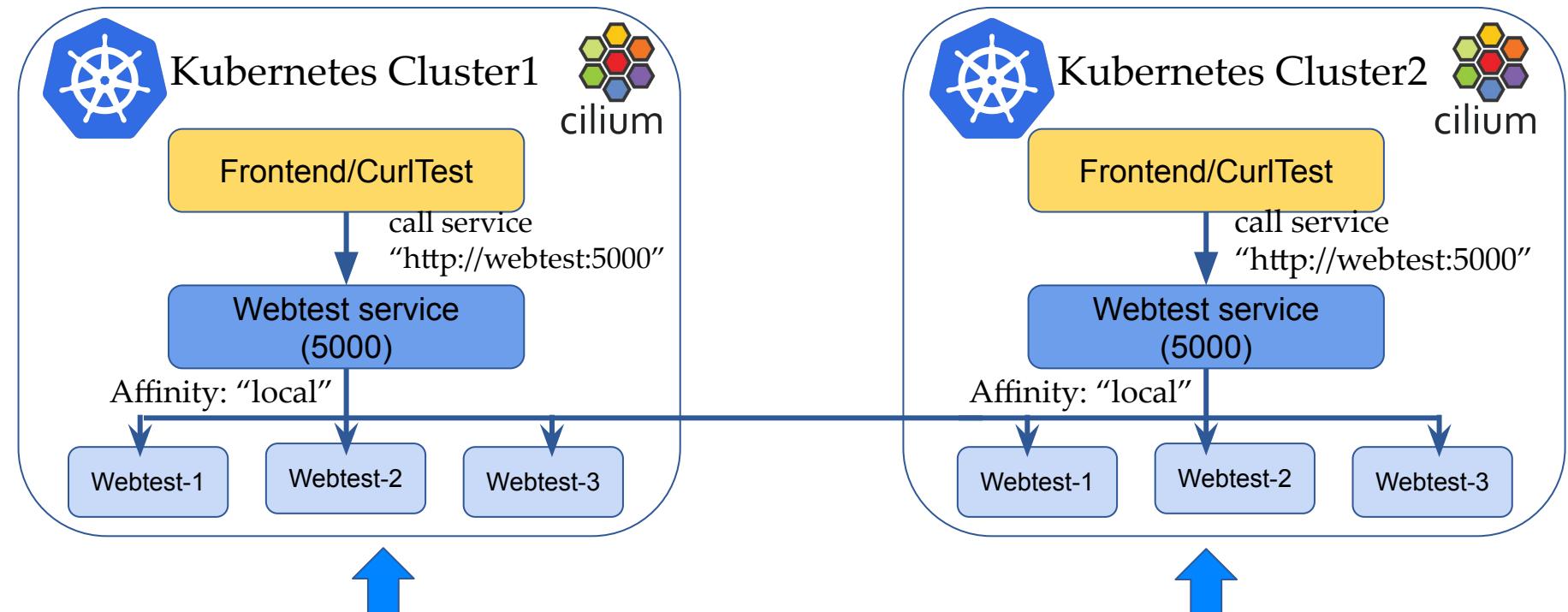
Demo Session

- In Demo session. We will provide 2 cluster (cluster1/cluster2) of Kubernetes with “Cluster Mesh” enable
- Each cluster will deploy application and service:
 - Cluster1: Deploy application **version 1.0**
 - Cluster2: Deploy application **version 1.51RC1**
- For make application extreme durability. We will enable “Cluster Mesh” with service affinity
- Normal condition:
 - Service will route traffic in local cluster for less latency
- Fail condition:
 - Service will kick traffic across cluster for HA and durability

Demo Session

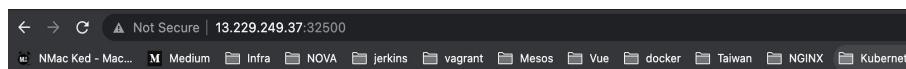
- Failover period:
 - Cilium take 30 second for failover traffic across cluster
- Fail back period:
 - Cilium will switch back when local resource available within 1 - 2 seconds

Demo Session



Welcome Page from Container Python Lab Web Version 1.00

Checkpoint Date/Time: Sat Aug 20 08:27:02 2022



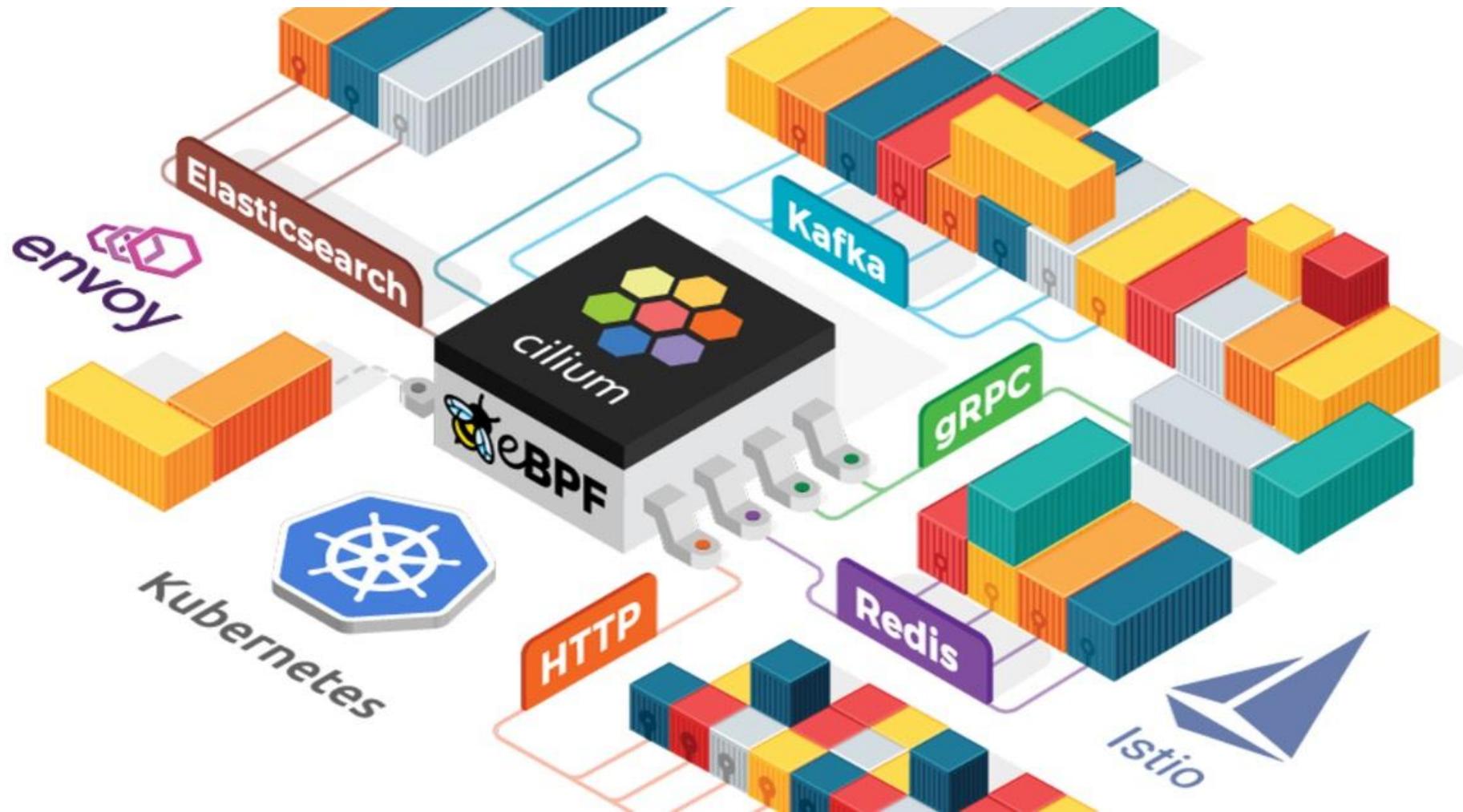
Welcome Page from Container Python Lab Web Version 1.51 RC

Checkpoint Date/Time: Sat Aug 20 08:27:00 2022

Demo 1: <https://youtu.be/LSSNA-W2GWA?si=xKbq3Jr-BiJmXvuW>

Demo 2: <https://youtu.be/f84rzH8ERCs?si=RgJs5mlp5b8PCKY8>

What Next ?



What Next ?

- eBPF around the World ?
- User space vs Kernel space to make eBPF difference ?
- Step-by-Step Go programming with eBPF !!!
- “Maps” user application and kernel program ?
- Leverage ultimate capability of eBPF for our job ?

