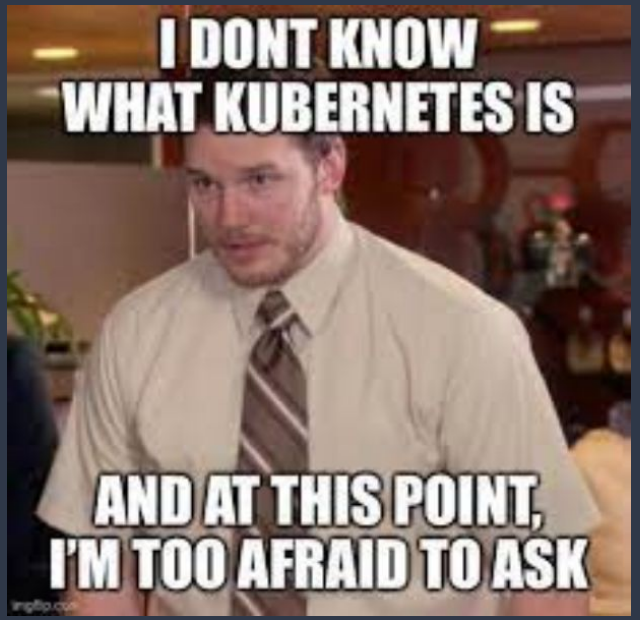
Cilium: Le Bon, la Brute et le Truand

Intro

- whoami
- Scope of this talk



CNI

- What is CNI
- Which are the commonly used ones

Calico, Flannel, Cilium, …

Flannel

- Approachable
- Easy to use
- Multiple compatible backends (VXLAN, host-gateway)
- Overlay network model overhead
- No network policy support

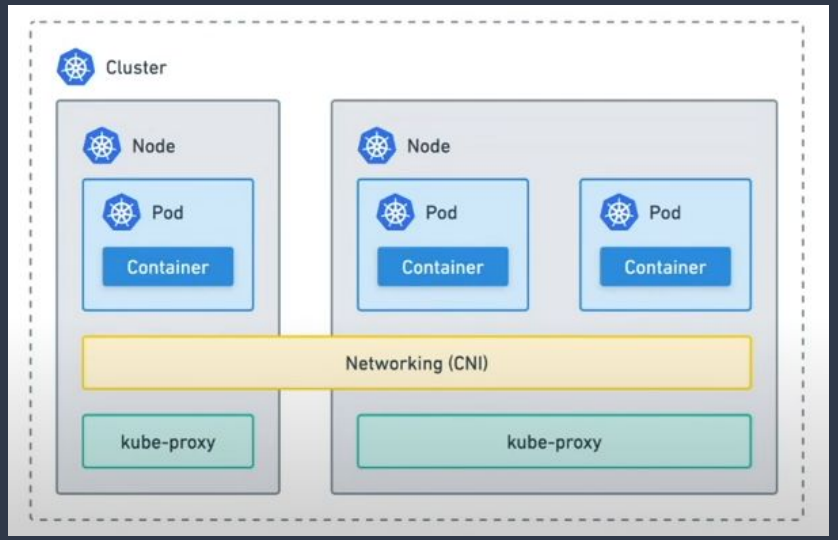
Calico

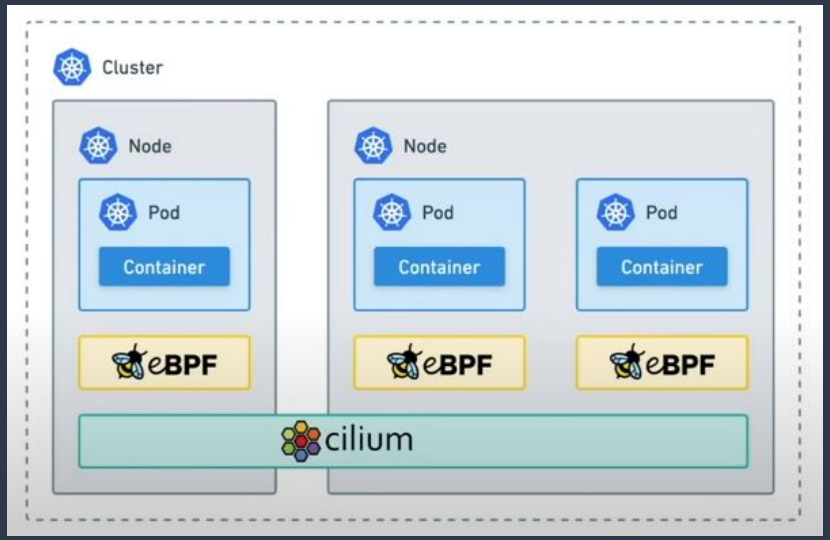
- Iptables, eBPF recently (different receive path)
- Privately owned
- Well established and battle tested
- Purely IP solution
- Rules based on IP address, subnets, ports
- Low latency
- Reliable, high performance
- Network policies

Cilium

- Based on eBPF right away
- Owned by CNCF
- Community resourced
- Wide and fast adoption by big companies
- Extensive feature set
- Fast, low latency, high performance
- L3/L4, L7 (HTTP, gRPC, Kafka)
- Identity based security
- Network policies

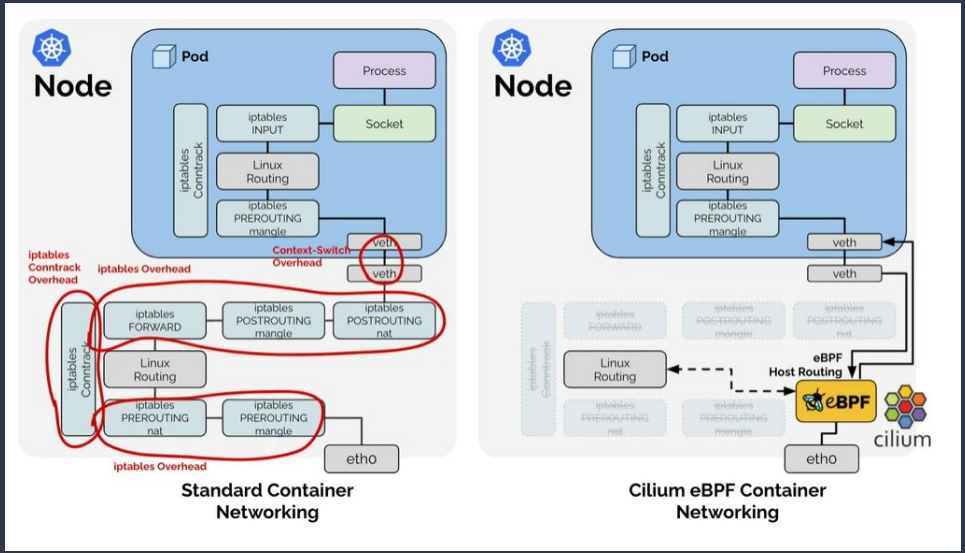
How it looks like





The network overhead

- Bypass all of the iptables and upper stack overhead in the host namespace
- Context-switching overhead when traversing through the veth pairs

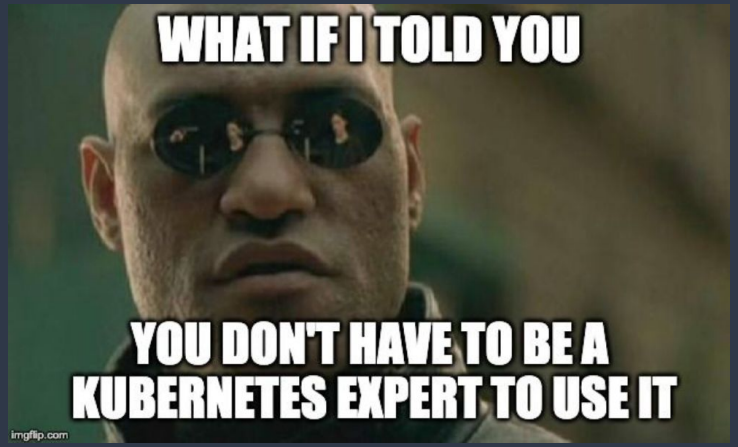


Cilium killer features

- Understand and filter network traffic (HTTP, gRPC, Kafka, …)
- Network policies that understand application-level concepts
- Visibility into the network, app protocol metadata and security (eBPF)
- eBPF supported by pretty much all linux distros
- Developed hand-in-hand with eBPF (sometimes by the exact same people)
- In-house expertise with eBPF, built Cilium on top
- Kubernetes identity awareness

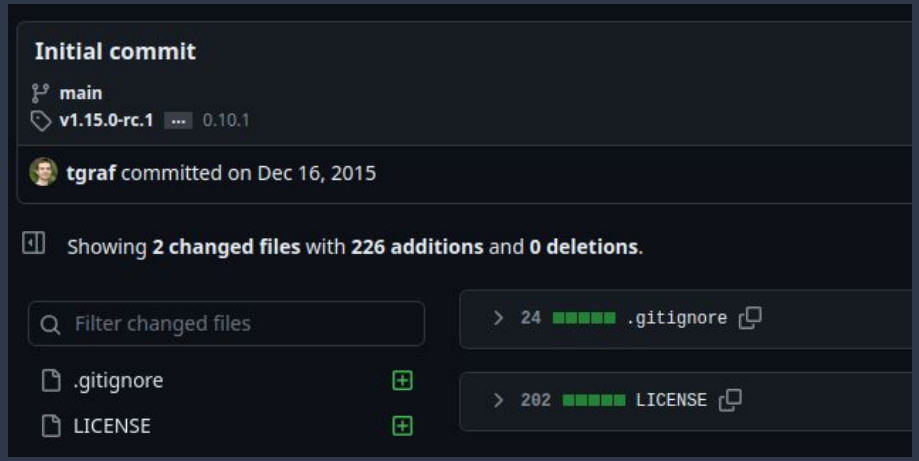
Step back, Cilium in a nutshell

- OpenSource && CloudNative software
- Provides network connectivity
- Secures network connectivity
- Observes network connectivity
- CNCF most advanced CNI



How it all started?

- First commit 12/2015, Thomas Graf
- Started as a networking solution with strong policy capabilities
- Now picked by major vendors (AWS, Google, Azure)
- Now the only graduated network project in CNCF landscape



The Cilium ecosystem and foundations

- Cilium
- Hubble
- Tetragon
- eBPF

Cilium (CNI)

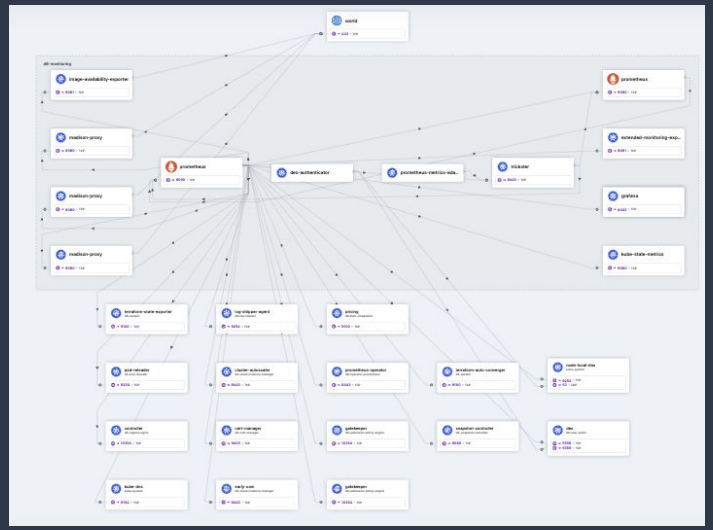
- Network connectivity
- Load balancing
- Operates at L3/L4 and L7
- Supports modern app protocols (HTTP, gRPC, Kafka, …)
- Hundreds of [big] adopters (Google, Yahoo, NY Times, …)
- eBPF as its core strength
- CNCF most advanced CNI

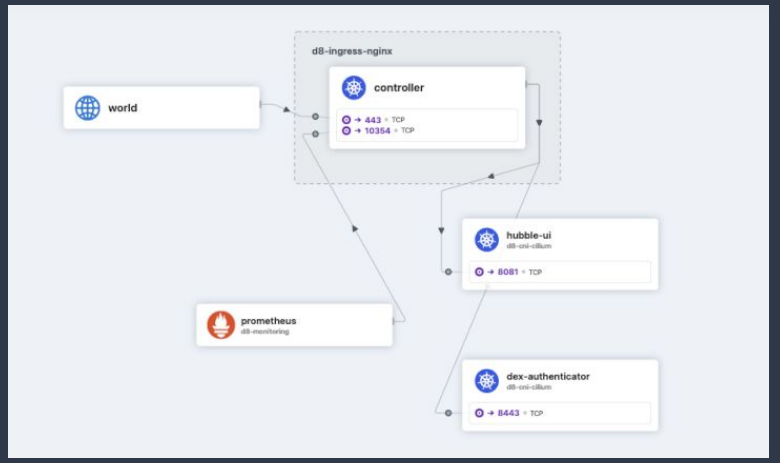
Hubble

- Networking and security observability
- Flow logs, service dependencies, communication maps, metrics
- GUI and CLI tool
- Troubleshooting capabilities
- Network policy editor
- Ops alerting/monitoring, app monitoring, sec observability

Hubble

```
cilium main 5s
master node ✗ hubble observe --pod deathstar --protocol http
Jan 29 12:22:37.872: default/tiefighter:58162 (ID:16408) -> default/deathstar-7848d6c4d5-wldf9:80 (ID:42455) http-request FORWARDED (HTTP/1.1 POST http://deathstar.default.svc.cluster.local/v1/request-landing)
Jan 29 12:22:37.872: default/tiefighter:58162 (ID:16408) <- default/deathstar-7848d6c4d5-wldf9:80 (ID:42455) http-response FORWARDED (HTTP/1.1 200 0ms (POST http://deathstar.default.svc.cluster.local/v1/request-landing))
Jan 29 12:22:54.785: default/tiefighter:50356 (ID:16408) -> default/deathstar-7848d6c4d5-wldf9:80 (ID:42455) http-request DROPPED (HTTP/1.1 PUT http://deathstar.default.svc.cluster.local/v1/exhaust-port)
Jan 29 12:22:54.785: default/tiefighter:50356 (ID:16408) <- default/deathstar-7848d6c4d5-wldf9:80 (ID:42455) http-response FORWARDED (HTTP/1.1 403 0ms (PUT http://deathstar.default.svc.cluster.local/v1/exhaust-port))
cilium main
master node ✗ hubble observe --pod deathstar --verdict DROPPED
Jan 29 12:22:54.785: default/tiefighter:50356 (ID:16408) -> default/deathstar-7848d6c4d5-wldf9:80 (ID:42455) http-request DROPPED (HTTP/1.1 PUT http://deathstar.default.svc.cluster.local/v1/exhaust-port)
Jan 29 12:23:17.066: default/xwing:52020 (ID:4425) <> default/deathstar-7848d6c4d5-grz7m:80 (ID:42455) policy-verdict:none INGRESS DENIED (TCP Flags: SYN)
Jan 29 12:23:17.066: default/xwing:52020 (ID:4425) <> default/deathstar-7848d6c4d5-grz7m:80 (ID:42455) Policy denied DROPPED (TCP Flags: SYN)
Jan 29 12:23:18.080: default/xwing:52020 (ID:4425) <> default/deathstar-7848d6c4d5-grz7m:80 (ID:42455) policy-verdict:none INGRESS DENIED (TCP Flags: SYN)
Jan 29 12:23:18.080: default/xwing:52020 (ID:4425) <> default/deathstar-7848d6c4d5-grz7m:80 (ID:42455) Policy denied DROPPED (TCP Flags: SYN)
Jan 29 12:23:19.093: default/xwing:52020 (ID:4425) <> default/deathstar-7848d6c4d5-grz7m:80 (ID:42455) policy-verdict:none INGRESS DENIED (TCP Flags: SYN)
Jan 29 12:23:19.093: default/xwing:52020 (ID:4425) <> default/deathstar-7848d6c4d5-grz7m:80 (ID:42455) Policy denied DROPPED (TCP Flags: SYN)
Jan 29 12:23:20.106: default/xwing:52020 (ID:4425) <> default/deathstar-7848d6c4d5-grz7m:80 (ID:42455) policy-verdict:none INGRESS DENIED (TCP Flags: SYN)
Jan 29 12:23:20.106: default/xwing:52020 (ID:4425) <> default/deathstar-7848d6c4d5-grz7m:80 (ID:42455) Policy denied DROPPED (TCP Flags: SYN)
Jan 29 12:23:21.119: default/xwing:52020 (ID:4425) <> default/deathstar-7848d6c4d5-grz7m:80 (ID:42455) policy-verdict:none INGRESS DENIED (TCP Flags: SYN)
Jan 29 12:23:21.120: default/xwing:52020 (ID:4425) <> default/deathstar-7848d6c4d5-grz7m:80 (ID:42455) Policy denied DROPPED (TCP Flags: SYN)
Jan 29 12:23:23.306: default/xwing:52020 (ID:4425) <> default/deathstar-7848d6c4d5-grz7m:80 (ID:42455) policy-verdict:none INGRESS DENIED (TCP Flags: SYN)
Jan 29 12:23:23.306: default/xwing:52020 (ID:4425) <> default/deathstar-7848d6c4d5-grz7m:80 (ID:42455) Policy denied DROPPED (TCP Flags: SYN)
Jan 29 12:23:27.359: default/xwing:52020 (ID:4425) <> default/deathstar-7848d6c4d5-grz7m:80 (ID:42455) policy-verdict:none INGRESS DENIED (TCP Flags: SYN)
Jan 29 12:23:27.359: default/xwing:52020 (ID:4425) <> default/deathstar-7848d6c4d5-grz7m:80 (ID:42455) Policy denied DROPPED (TCP Flags: SYN)
Jan 29 12:23:35.467: default/xwing:52020 (ID:4425) <> default/deathstar-7848d6c4d5-grz7m:80 (ID:42455) policy-verdict:none INGRESS DENIED (TCP Flags: SYN)
Jan 29 12:23:35.467: default/xwing:52020 (ID:4425) <> default/deathstar-7848d6c4d5-grz7m:80 (ID:42455) Policy denied DROPPED (TCP Flags: SYN)
Jan 29 12:23:52.533: default/xwing:52020 (ID:4425) <> default/deathstar-7848d6c4d5-grz7m:80 (ID:42455) policy-verdict:none INGRESS DENIED (TCP Flags: SYN)
Jan 29 12:23:52.533: default/xwing:52020 (ID:4425) <> default/deathstar-7848d6c4d5-grz7m:80 (ID:42455) Policy denied DROPPED (TCP Flags: SYN)
Jan 29 12:24:24.960: default/xwing:52020 (ID:4425) <> default/deathstar-7848d6c4d5-grz7m:80 (ID:42455) policy-verdict:none INGRESS DENIED (TCP Flags: SYN)
Jan 29 12:24:24.960: default/xwing:52020 (ID:4425) <> default/deathstar-7848d6c4d5-grz7m:80 (ID:42455) Policy denied DROPPED (TCP Flags: SYN)
```





Tetragon

- Realtime security observability and runtime enforcement
- Detects and reacts upon security significant events
  - Process execution
  - System call activity
  - Network and file access
  - I/O activity
- Youngest of the family

eBPF

- THE force
- Acronym for extended Berkeley Packet Filter
- Runs inside the Linux kernel, logic can be applied and updated on the fly
- Brings programmability for the operating system
- No application change
- No container configuration

Want more?

- Kube-proxy replacement (bpfilter)
- Mutual auth
- mTLS ready with 2 lines of yaml
- Load balancer
- Transparent encryption (IPsec, WireGuard)
- Service mesh, Cluster mesh
- IPv6
- SCTP
- Simultaneously handle multiple streams of data
- Advanced BGP features (timers, graceful restart, eBGP multihop)

…

Why do we need another CNI 1/2

- Because kubernetes is not complicated, so you know exactly what is happening

Why do we need another CNI 2/2

- Because kubernetes is secure out of box

Common challenges we face

- How do I tell which svc is communicating with other svc
- How do I enforce rules so svc A can talk to svc B
  … and NOT talk to other svc
- How do I put security in this place
- How do I observe network and all the communication between my apps

Answer?


SERVICE MESH
LIGHTYEAR
SERVICE MESH EVERYWHERE
imgflip.com

Wait, what?

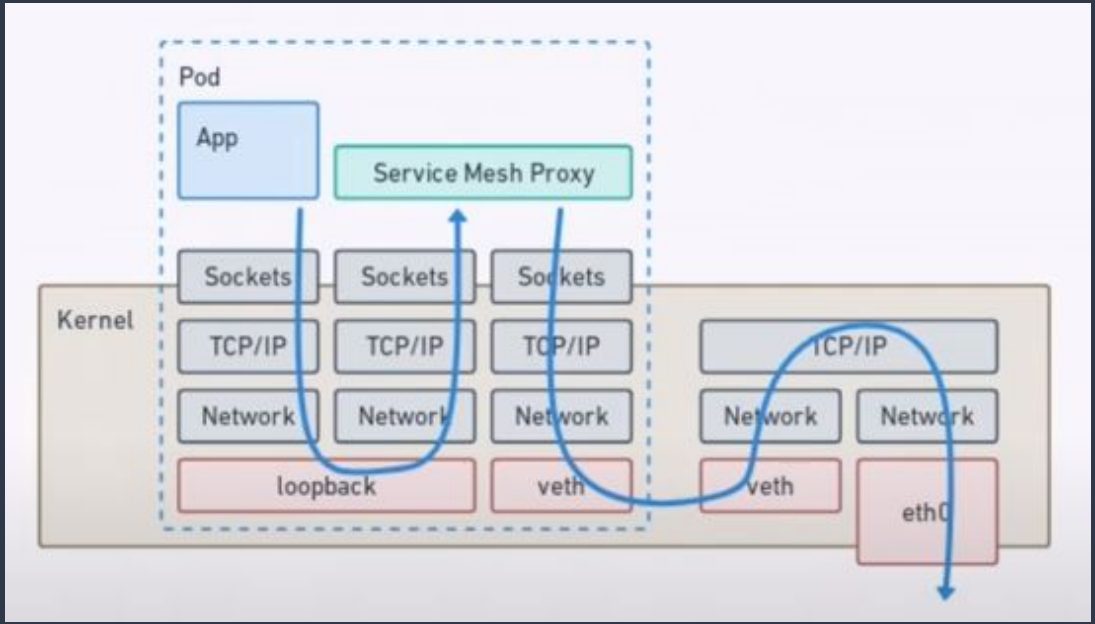
- We already have service mesh with Istio, Linkerd, …
- But with all the service meshes, there is [at least] one catch …

Sidecars

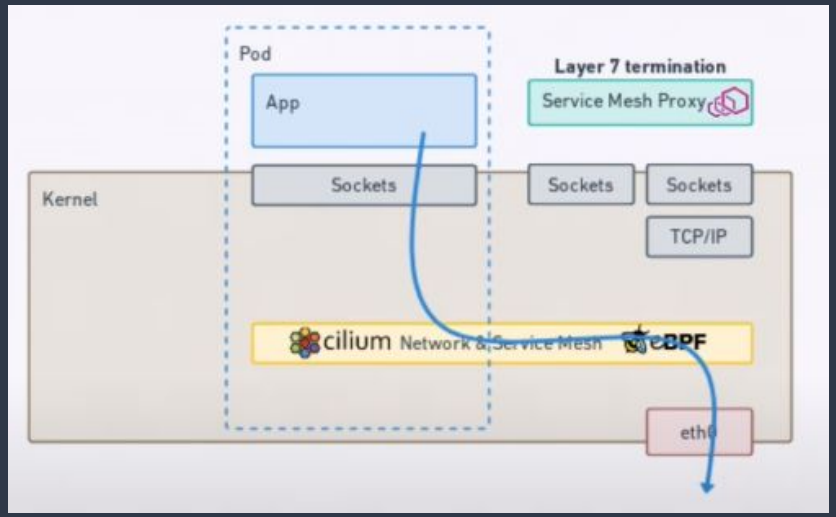
Problems with sidecars

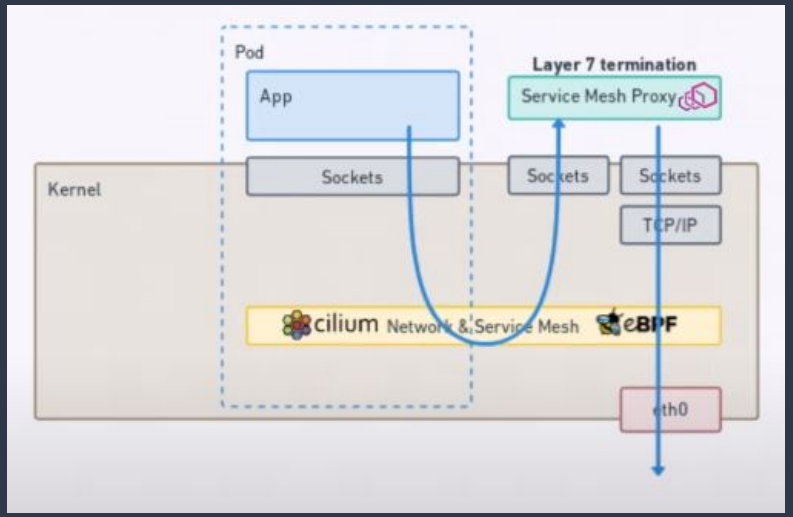
- Extra latency
- Complexity
- Resources (cost)
- Start-up/shut-down race conditions
- Pod startup time (wait for proxy to be ready)

Network path with sidecar



Network path without sidecar (L3/L4, L7)





The sidecar model controversy

- Envoy for L7 policies, observability and traffic management rules
  - CEC (cilium envoy config to configure envoy listeners)
- Only one proxy per node
  - 2 services communicating on the same node do not need to have 2 proxies
- Per pod sidecar vs per node proxy security
  - Isolate network stacks on kernel level already (namespaces, cgroups)
  - Envoy listeners in user namespace
  - Noisy neighbours
  - Isolation for debugging
  - One proxy per node is a spof
  - Troubleshoot user space vs kernel based networking layer

A bit of service mesh history (aka the WHY)

- Application needs to communicate to its fellas
- Multiple programming languages in one app
- Canary rollouts
- Circuit breaking
- Rate limiting
- Load balancing
- Service discovery
- Retries

… all can be solved by a sidecar proxy container … [for EVERY single pod]

This is all nice and cool only until you receive the bill from your provider.

There must be a better way

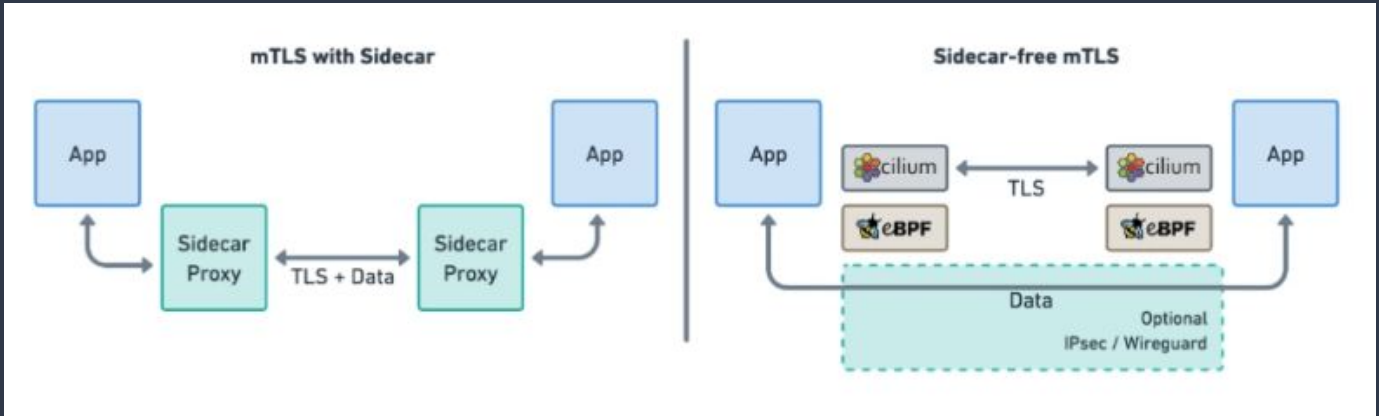
- eBPF

Cilium service mesh

- Observability
- Ingress
- Load balancing
- Protocol parsing
- Path based routing
- Service load balancing
- Rules (canary, retries, …)
- Identity based security
- mTLS at L7 (beta, only with Spiffe identity)

The mTLS challenge

- Separate authentication and encryption path (proxy based designed to work for TCP, but when separated, UDP, SCTP or multicast can be used)
- Cilium already with strong network layer encryption (IPsec, WireGuard)
- What about L7?
  - How to get certificate injected into the kernel and authenticate?



Why is it more than a service mesh?

- Understands traffic that flows through
- Enforce policies based on it
- Policy for Cassandra or Kafka traffic
- Work in kernel, much faster than anything you implement with Istio/Linkerd
- Explore traffic between containers down to TCP packet level
  - Similar to Linkerd Tap but much lower level
- Combine Cilium with existing service meshes

So eBPF everywhere and for everything?

- Not really… [yet?]
- Great for
  - Xdp (examine raw packet buffers as they enter the system, decide what to do with them)
  - Connect-time load balancing (balance at the source using program loaded in kernel instead of virtual IP, no DNAT need on packet processing path so NAT overhead from service connections is removed)
- Not that great for (expensive)
  - Packet-by-packet processing (CPU intensive)
  - Decryption and re-encryption for encrypted flows (need to build a structure and do a lookup for every packet)

Recap

+

- Available out of box with managed k8s solutions
- eBPF
- Strong policy engine
- Kubernetes identities aware
- Great integration with Prometheus and Grafana stack
- Active and fast-paced project

-

- Setup can be finicky
- Hubble ui looks dated
- Might require nev cluster setup
- Some features not “there” yet (mTLS on L7)

Study materials

- rtfm
- Labs @ isovalent.com/resource-library/labs/

Where to seek for help

- Documentation is very, very good. Read it
- Slack
- Weekly eCHO livestream on YT
- Supportive community

May the source be with You!

Tomáš Jamrich
Prusa Research a.s.
tomas.jamrich@prusa3d.cz

Q&A