



ISOVALENT

Cilium : Le meilleur de Linux pour vos réseaux Kubernetes



Paul Chaignon | @pchaigno

Staff Software Engineer, Isovalent at Cisco



cilium

Created by ISOVALENT



eBPF-based:

- Networking
- Security
- Observability
- Service Mesh & Ingress



Microsoft chooses Cilium for Azure networking



AWS picks Cilium for Networking & Security on EKS Anywhere



Google chooses Cilium for Google Kubernetes Engine (GKE) networking



Building High-Performance Cloud Native Pod Networks



Scaleway uses Cilium as the default CNI for Kubernetes Kapsule



Managed Kubernetes: 1.5 Years of Cilium Usage at DigitalOcean



What Makes a Good Multi-tenant Kubernetes Solution



Kubernetes Network Policies in Action with Cilium



Bell uses Cilium and eBPF for telco networking



TSI uses Cilium for its Open Sovereign Cloud product



Scaling a Multi-Tenant Kubernetes Clusters in a Telco



F5 uses Cilium VXLAN tunnel integration with BIG-IP



Radio France uses Cilium in their self hosted clusters on AWS



Deezer uses Cilium as their CNI for all on-prem Kubernetes clusters for its performance and security



Rabobank uses Cilium to implement zero trust networking in their API platform



Bloomberg leverages Cilium to construct data sandboxes

Foundation



Technology



Who Am I?



Paul Chaignon

Staff Software Engineer @ Isovalent
Equipe datapath/eBPF pour Cilium



ISOVALENT



cilium



Tetragon

- Company behind Cilium
- Remote-first startup
- Just acquired by Cisco!



Cilium

- Programmer le noyau : eBPF
- Connecter les conteneurs
- Répartir la charge entre pods
- Segmenter son réseau
- Connecter l'existant
- Conclusion

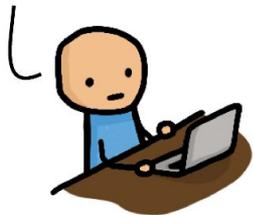


Cilium

- ◆ Programmer le noyau : eBPF
- ◆ Connecter les conteneurs
- ◆ Répartir la charge entre pods
- ◆ Segmenter son réseau
- ◆ Connecter l'existant
- ◆ Conclusion

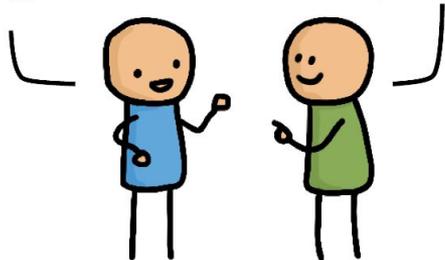
Application Developer:

I want this new feature to observe my app



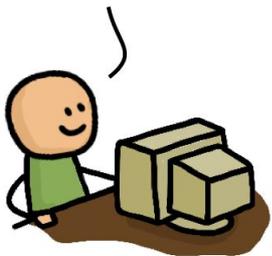
Hey kernel developer! Please add this new feature to the Linux kernel

OK! Just give me a year to convince the entire community that this is good for everyone.

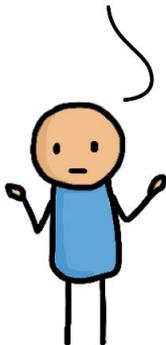


1 year later...

I'm done. The upstream kernel now supports this.



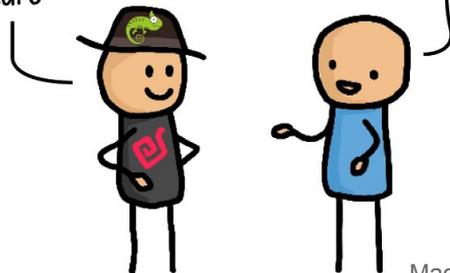
But I need this in my Linux distro



5 year later...

Good news. Our Linux distribution now ships a kernel with your required feature

OK but my requirements have changed since...



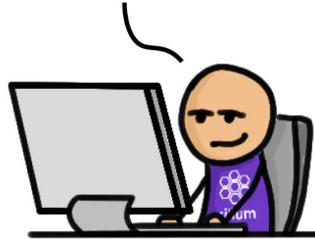
Application Developer:

i want this new feature
to observe my app



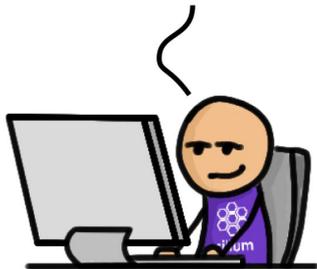
eBPF Developer:

OK! The kernel can't do this so let
me quickly solve this with eBPF.

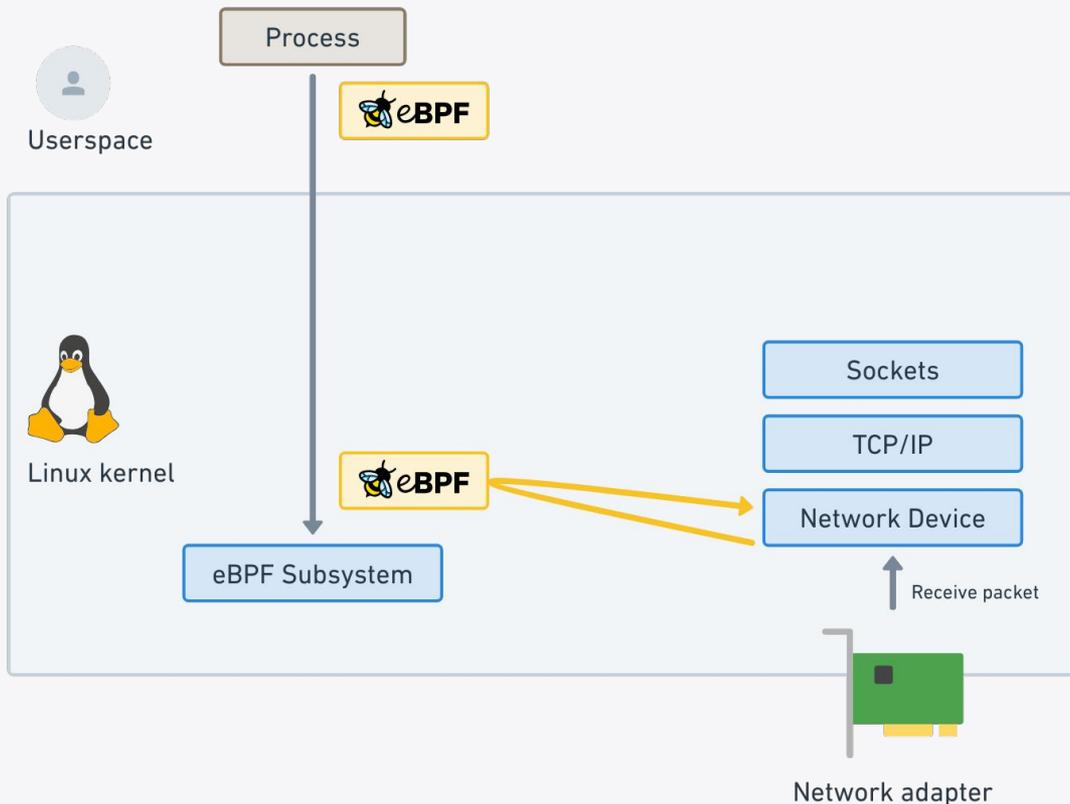


A couple of days later...

Here is a release of our eBPF project that has this feature
now. BTW, you don't have to reboot your machine.



Programmer le noyau



- Programme chargé dans le noyau, vérifié statiquement
- Attaché à des évènements
 - Réception de paquets
 - Appel de fonctions kernel
 - ...
- Exécuté pour chaque évènement



Cilium

- ◆ Programmer le noyau : eBPF
- ◆ **Connecter les conteneurs**
- ◆ Répartir la charge entre pods
- ◆ Segmenter son réseau
- ◆ Connecter l'existant
- ◆ Conclusion

Connecter les conteneurs

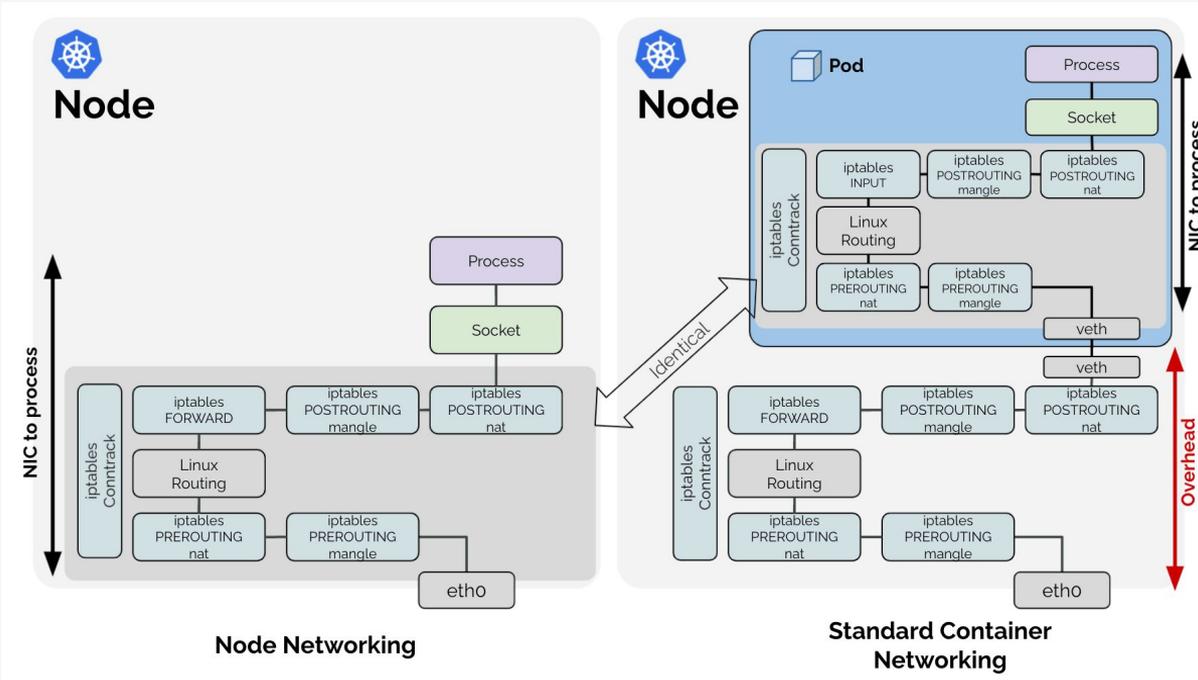
Modèle Kubernetes :

- Les conteneurs d'un Pod partagent le même namespace réseau
- Chaque Pod a son adresse IP
- Tous les pods du cluster peuvent communiquer entre eux (sans NAT!)

Avec Cilium :

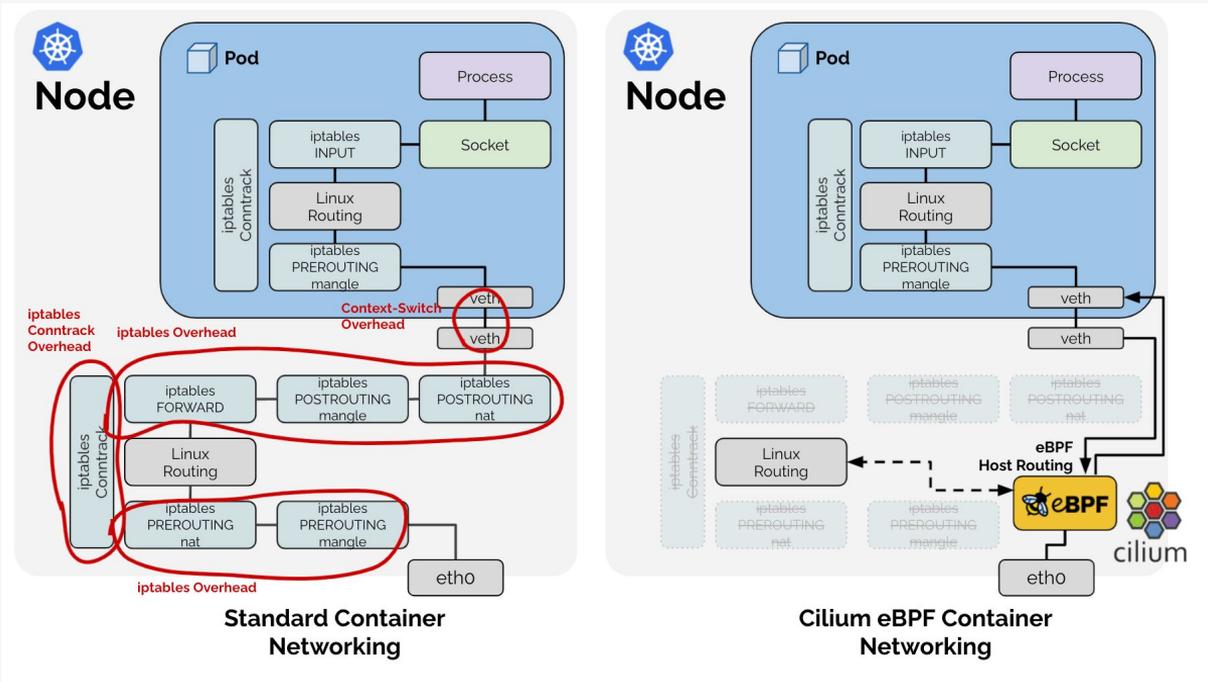
- Pod connecté au namespace hôte via des interfaces veth
- Pod connectés entre eux via un overlay (ex. VXLAN) ou directement si le réseau le permet

Connecter les conteneurs... avec eBPF



- Stack réseau Linux est assez générale mais lourde
- Stack traversée 2+ fois dans le cas de conteneurs

Connecter les conteneurs... avec eBPF

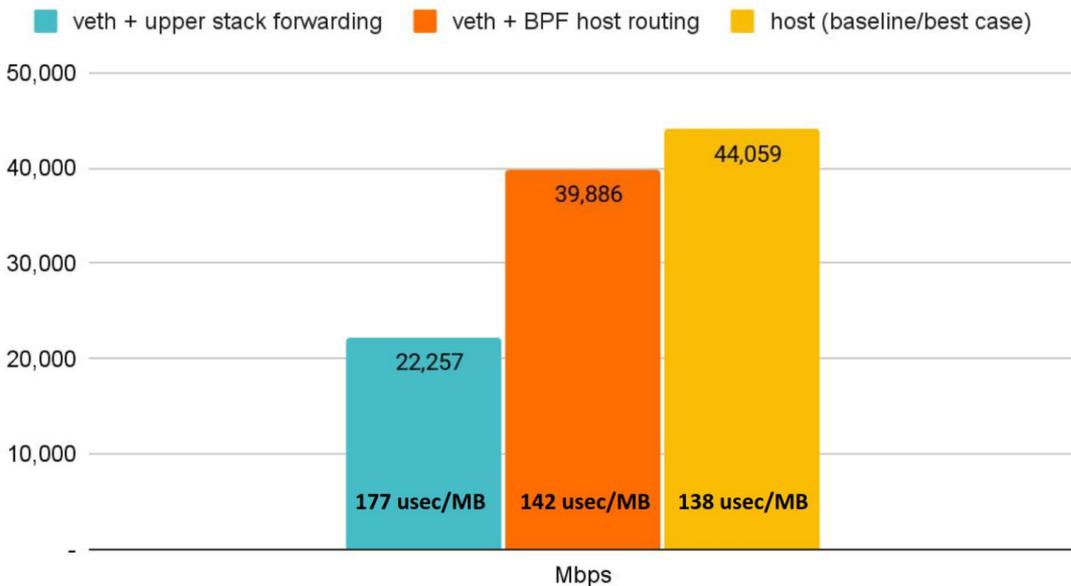


Avec eBPF :

- Spécialisation de la stack réseau
- Bypass tout ce qui n'est pas requis
- Beaucoup à bypasser pour les réseaux de conteneurs

Connecter les conteneurs... avec eBPF

TCP stream single flow Pod to Pod over wire (higher is better)



Back to back: AMD Ryzen 9 3950X @ 3.5 GHz, 128G RAM @ 3.2 GHz, PCIe 4.0, ConnectX-6 Dx, mlx5 driver, striding mode, LRO off, 1.5k MTU
Receiver: taskset -a -c <core> `tcp_mmap` -s (non-zero-copy mode), Sender: taskset -a -c <core> `tcp_mmap` -H <dst host>

Cilium

- ◆ Programmer le noyau : eBPF
- ◆ Connecter les conteneurs
- ◆ Répartir la charge entre pods
- ◆ Segmenter son réseau
- ◆ Connecter l'existant
- ◆ Conclusion

Répartir la charge entre pods

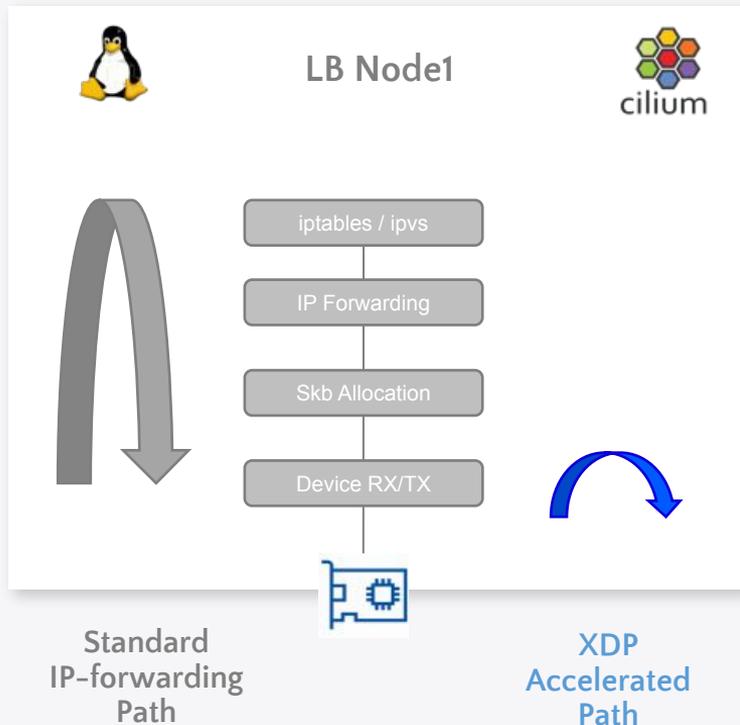
Modèle Kubernetes :

- Les fonctions de load balancing sont incluses
- Tous les noeuds sont des load-balancers

Avec Cilium :

- Support des standards (ClusterIP, NodePort, etc.)
- Possibilité de déployer un load-balancer “standalone”

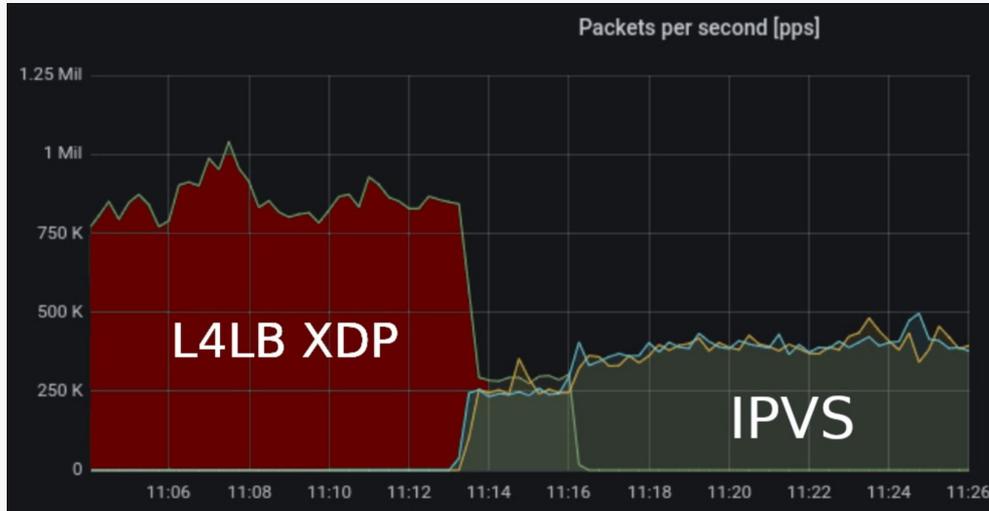
Répartir la charge entre pods... avec eBPF



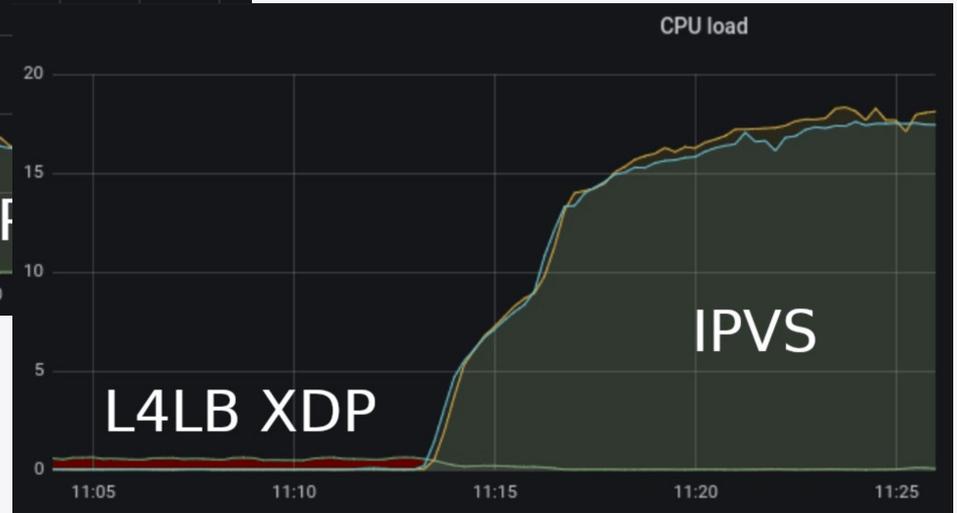
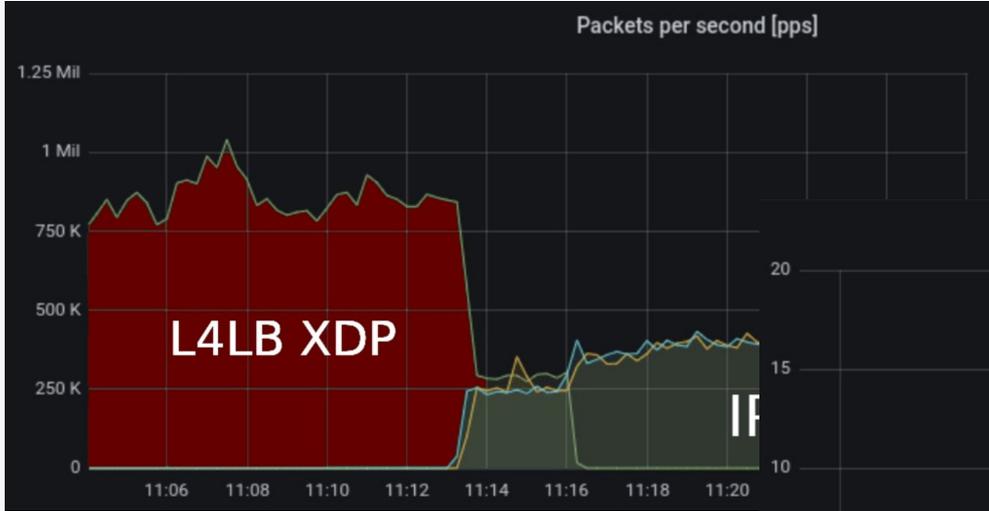
Redirection du load-balancer au plus tôt

- *North-south* : Au niveau du driver (XDP) pour les paquets entrants le cluster
- *East-west* : Au niveau du socket pour le load-balancing entre pods

Répartir la charge entre pods



Répartir la charge entre pods





Cilium

- ◆ Programmer le noyau : eBPF
- ◆ Connecter les conteneurs
- ◆ Répartir la charge entre pods
- ◆ **Segmenter son réseau**
- ◆ Connecter l'existant
- ◆ Conclusion

Segmenter son réseau

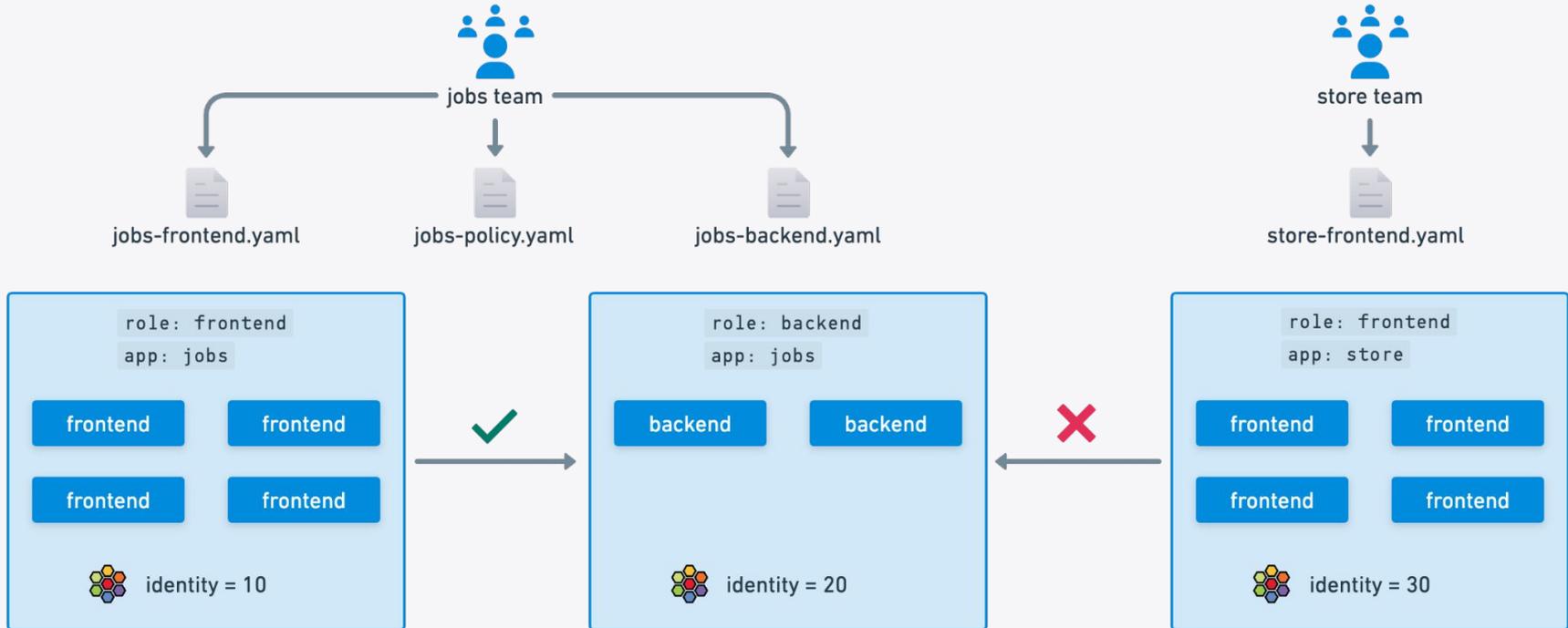
Modèle réseau Kubernetes :

- Isolation réseau des pods par application de politiques de sécurité L3 et L4

Avec Cilium :

- Support du standard NetworkPolicy
- + Support de politiques L7, FQDN-based, ICMP, etc.

Segmenter son réseau



Segmenter son réseau... avec eBPF

eBPF based

Iptables based

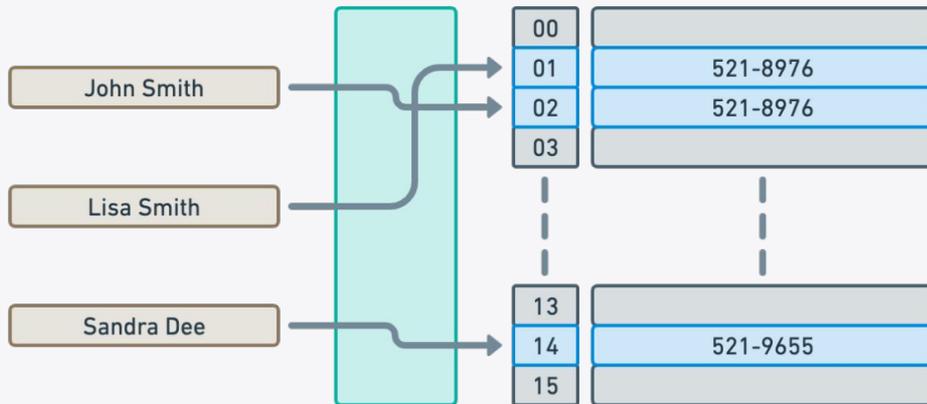
- Liste :
 - Lookup en $O(n)$
 - Update en $O(n)$



Segmenter son réseau... avec eBPF

eBPF based

- Packet classifier :
 - Lookup en $O(\log n)$
 - Update en $O(1)$



Iptables based

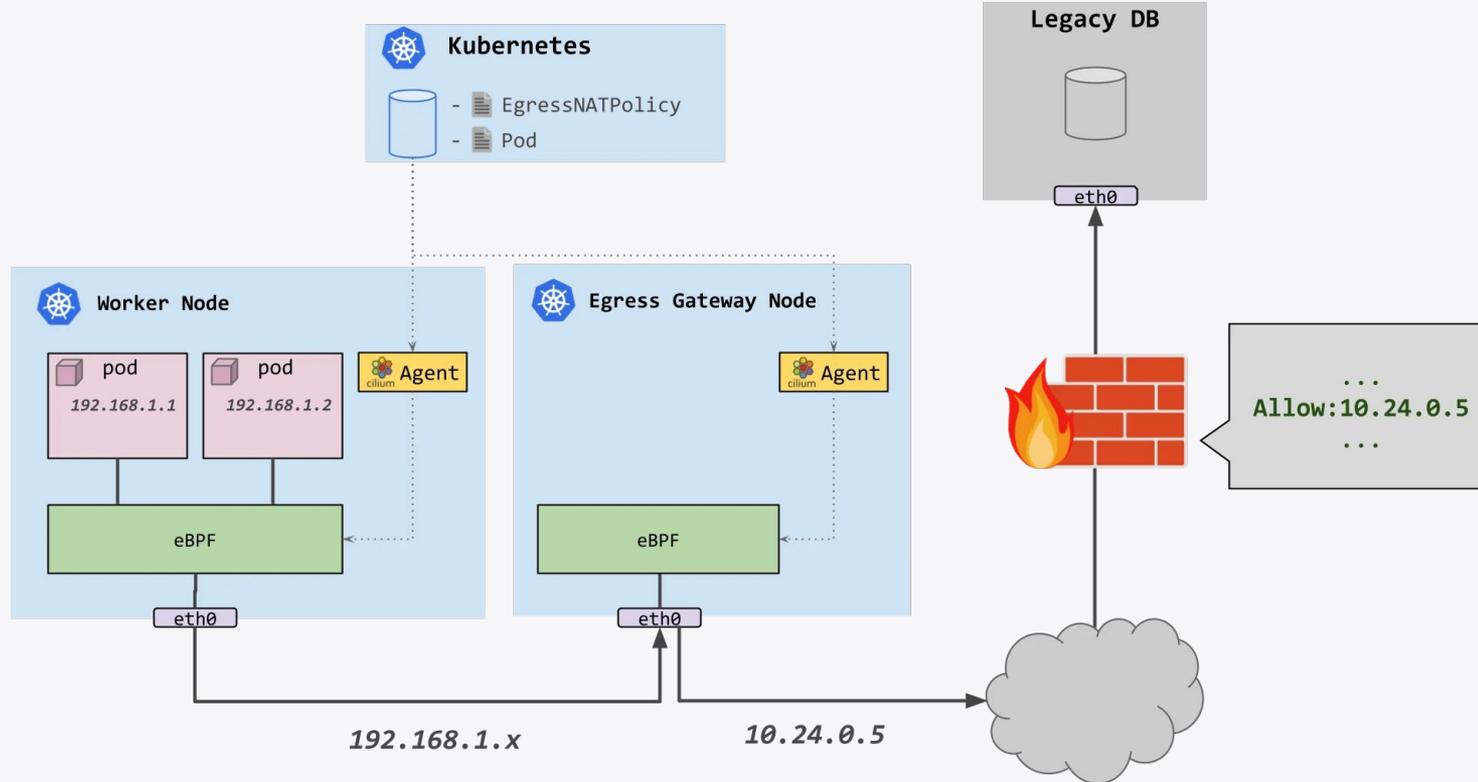
- Liste :
 - Lookup en $O(n)$
 - Update en $O(n)$



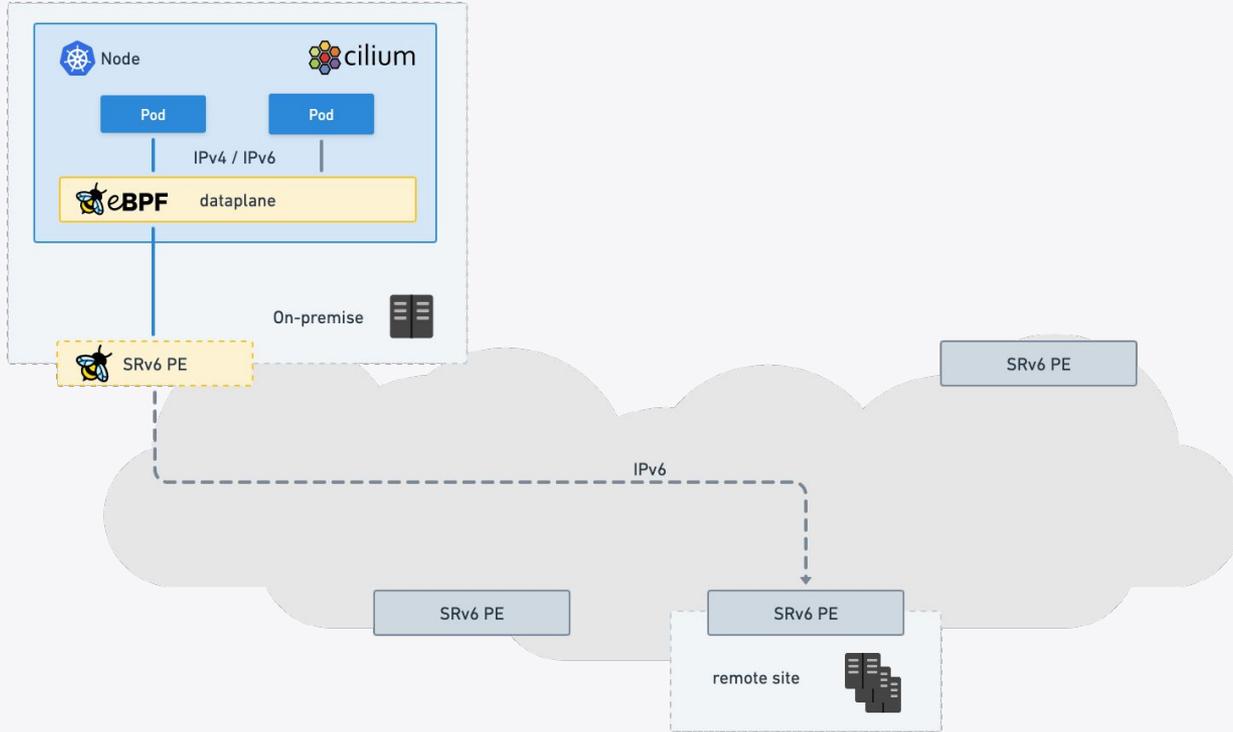
Cilium

- ◆ Programmer le noyau : eBPF
- ◆ Connecter les conteneurs
- ◆ Répartir la charge entre pods
- ◆ Segmenter son réseau
- ◆ **Connecter l'existant**
- ◆ Conclusion

Cilium Egress Gateway

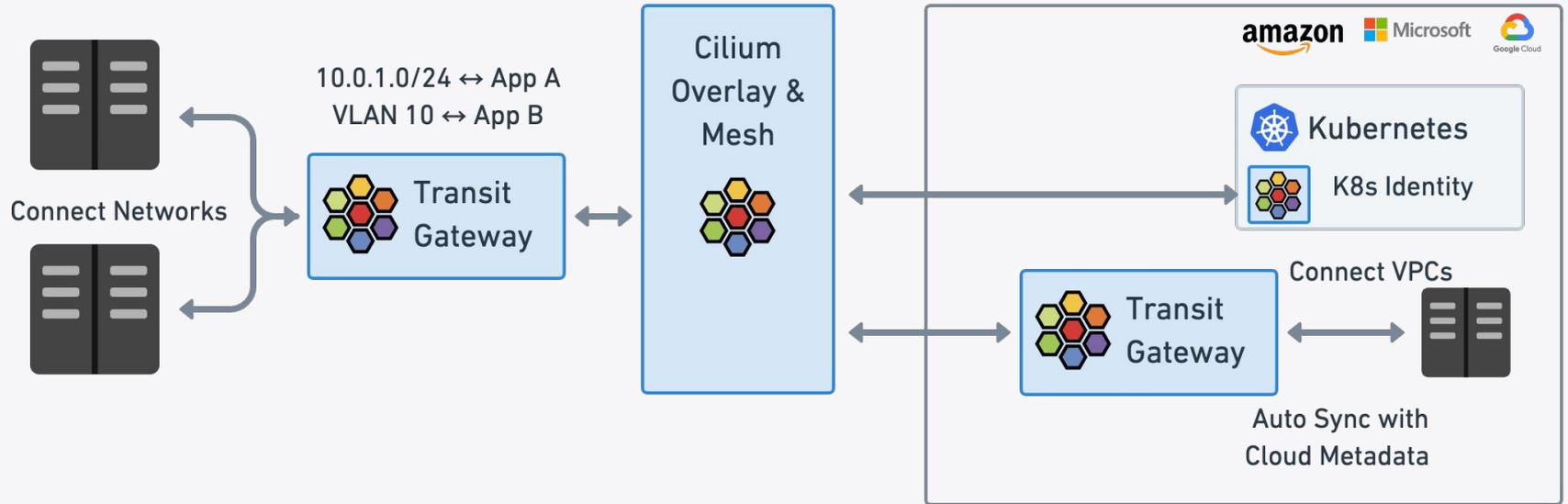


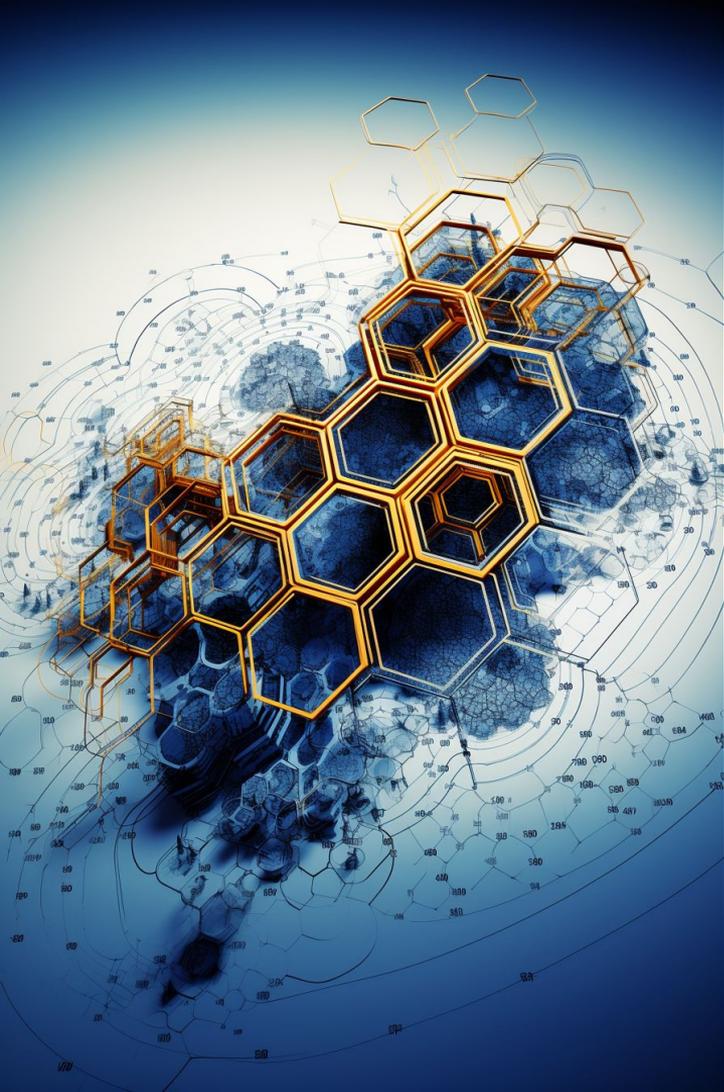
Cilium SRv6 L3VPN



- H.Encaps, H.Encaps.Red, End.DT4
- SID allocator
- BGP integration

Cilium Mesh





Cilium

- ◆ Programmer le noyau : eBPF
- ◆ Connecter les conteneurs
- ◆ Répartir la charge entre pods
- ◆ Segmenter son réseau
- ◆ Connecter l'existant
- ◆ Conclusion

Conclusion

- Cilium, le plugin réseau pour Kubernetes
- S'appuie sur les dernières innovations du noyau (pas que BPF!)
 - Principalement des gains en efficacité
- Fonctionnalités pour connecter l'existant au "cloud-native"

ISOVALENT

Merci !

Merci à Raymond de Jong, Raphaël Pinson et Vadim Shchekoldin pour beaucoup des slides et images !





ISOVALENT



cilium Service Mesh

Ingress Authentication Traffic Management

spiffe Gateway API

cilium **hubble** Observability

Metrics Tracing Service Map Logs

SIEM fluentd Grafana OpenTelemetry

cilium Networking

CNI

Network Policy: DNS, L3/L4, L7

Encryption: IPsec, Wireguard

Load-Balancing: K8s, Maglev, DSR

Multi-Cluster Networking NAT46

IPv4 IPv6 Cloud SDN BGP Overlay SRv6 Egress Gateway

Runtime Security

Tetragon

SIEM fluentd Grafana

Observability

Enforcement

Kubernetes Container VM Metal



