How Cilium uses BPF to Supercharge Kubernetes Networking & Security

Mark Darnell, Senior PM Networking, SUSE
Roger Klorese, Senior PM Kubernetes, SUSE
Dan Wendlandt, Co-founder, Isovalent

CLOUD NATIVE COMPUTING FOUNDATION

Agenda

- Why is SUSE Presenting Cilium? – A Little Context
- Why Cilium?
  - A Bit of Cilium Background
  - Use Cases – Some Fundamental Networking Needs
  - Luxury/Future Use Cases – What we see on the Horizon
- A Deeper Dive into Cilium Internals
  - What is BPF and How Does it Work?
  - How Does Cilium Use BPF?

Agenda

- Why is SUSE Presenting Cilium? – A Little Context
- Why Cilium?
  - A Bit of Cilium Background
  - Use Cases – Some Fundamental Networking Needs
  - Luxury/Future Use Cases – What we see on the Horizon
- A Deeper Dive into Cilium Internals
  - What is BPF and How Does it Work?
  - How Does Cilium Use BPF?

What is Cilium? (100 level Course)

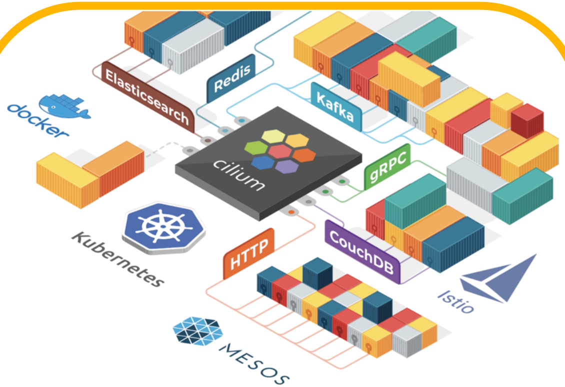
Cilium is open source software for transparently securing the network connectivity between application services deployed using Linux container management platforms

What is Cilium? (400 level course)

Service Identity
Visibility + filtering based on Kubernetes service labels, DNS-names, etc, not IP addresses.

API-Aware Security
Goes beyond TCP/UDP ports, natively understanding HTTP, gRPC, Kafka, DNS, & more.

Performance & Scale
BPF datapath and control plane optimized for highly dynamic, large scale envs with high throughput.,



Service and API-Aware Linux Networking & Security

Open Source

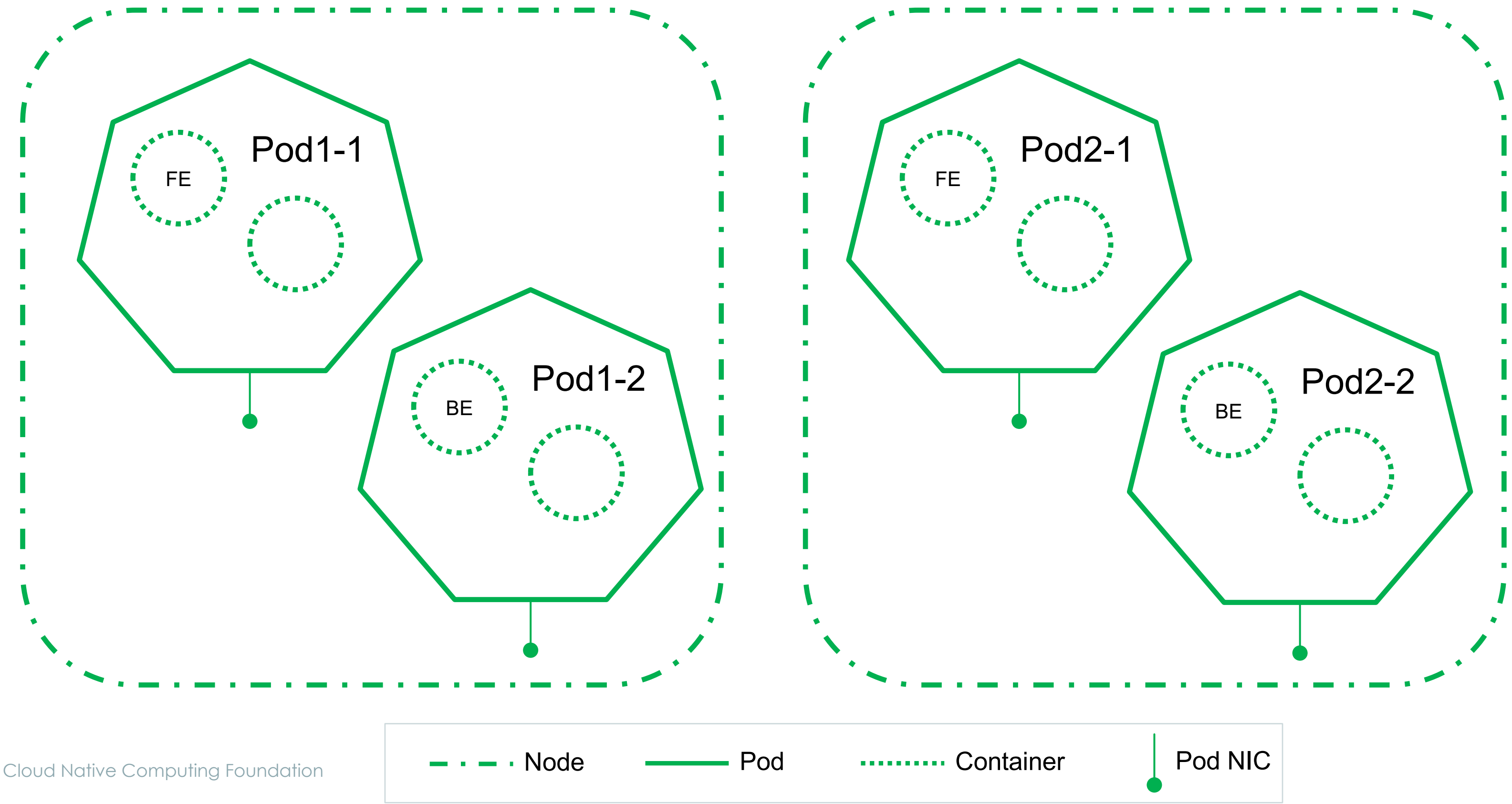
Powered by BPF

Multi-Cluster Routing
Provides simple, efficient, and secure connectivity between multiple Kubernetes clusters

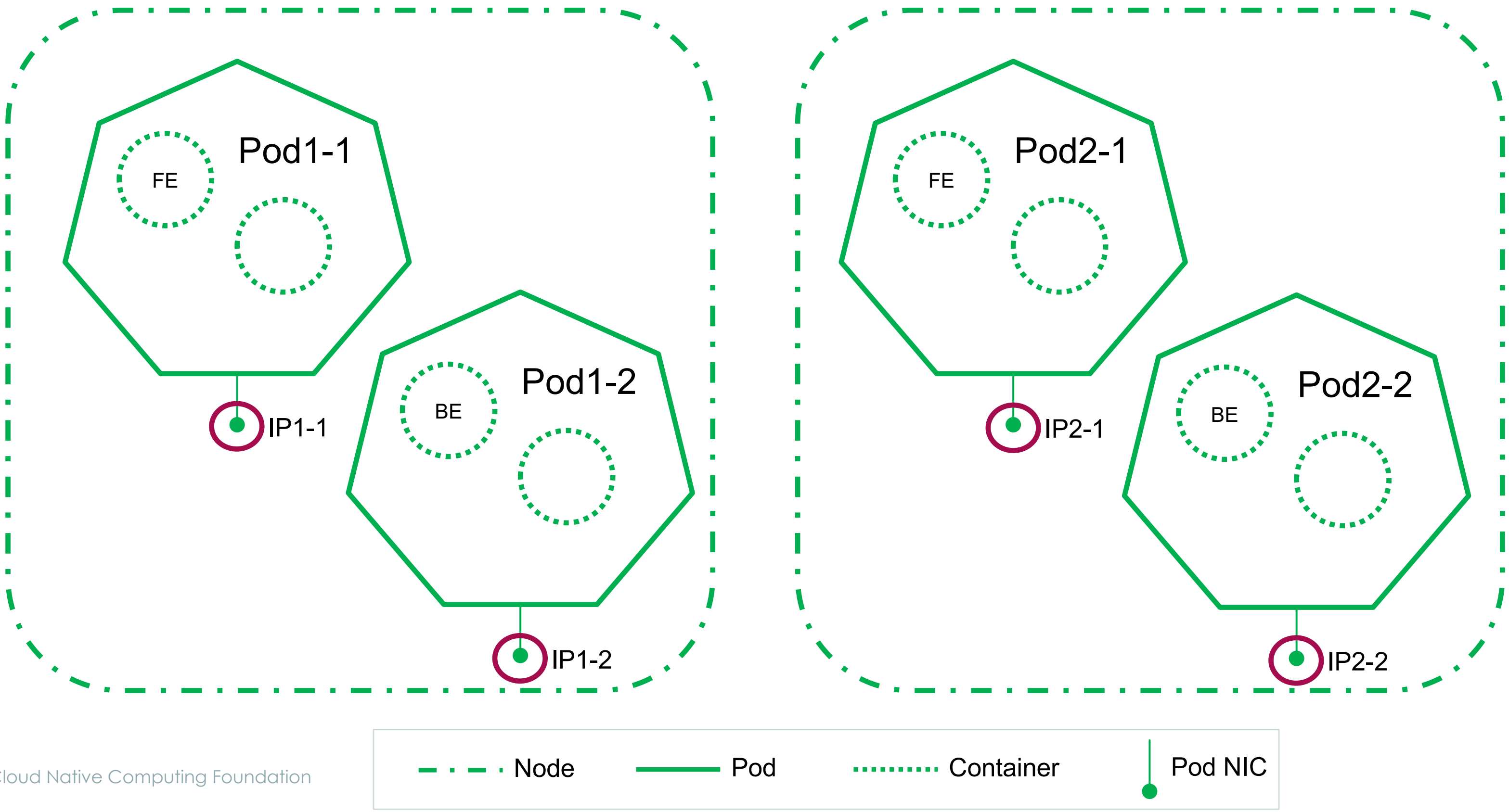
Universal Encryption
Adds encryption to all traffic between Cilium endpoints with no application/pod changes.

Transparent to Apps
By running in the kernel, BPF + Cilium require no changes by or coordinate with app teams.

A (Tiny) Cloud Native Web Farm – Standard Kubernetes



Security Requires Pod IPs – Standard Kubernetes



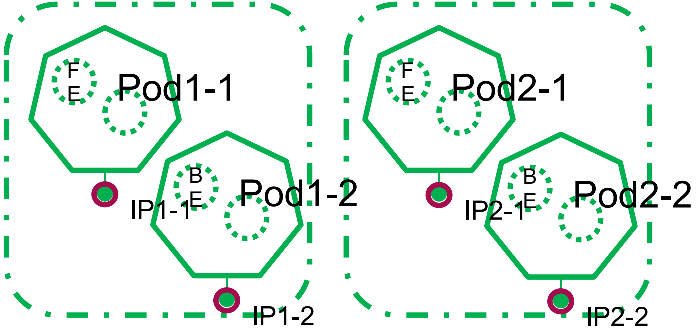
iptables Rules to Enforce App/DB Security

```
# Allow http protocol access to Apache running in pods [1,2]-1
iptables –i eth0 –p tcp –dport 80 –s ${clientIPRange} –d ${IP1-1}
iptables –i eth0 –p tcp –dport 80 –s ${clientIPRange} –d ${IP2-1}

# Allow Apache to access Vitess/Mysql in pods [1,2]-2
iptables –i eth0 –p tcp –dport 3306 –s ${IP1-1} –d ${IP1-2}
iptables –i eth0 –p tcp –dport 3306 –s ${IP2-1} –d ${IP1-2}
iptables –i eth0 –p tcp –dport 3306 –s ${IP1-1} –d ${IP2-2}
iptables –i eth0 –p tcp –dport 3306 –s ${IP2-1} –d ${IP2-2}

# Allow related packets on established or related connections
iptables -m state --state ESTABLISHED,RELATED -j ACCEPT

# Drop all other packets
iptables –i eth0 –j DROP # or –j REJECT
iptables –o eth0 –j DROP # or –j REJECT
```



Better! Cilium Label-Aware Security & Visibility

Pod

role=frontend

HTTP GET / →

allow

Pod

role=backend

← HTTP GET /

deny

Pod

role=other

Label-based Security Policy:

```
endpointSelector:
    matchLabels:
      role = "backend"
ingress:
    matchLabels:
      role = "frontend"
```

Label-based Security Visibility Logs:

```
23:15:01:  allow: role=frontend →role=backend
23:16:34:  deny: role=other →role=backend
…
```

Cilium Label-Aware Scalability

Create This ONE Time and Scale Out

```
endpointSelector:
    matchLabels:
      role = "backend"
ingress:
    matchLabels:
      role = "frontend"
```

OR...modify and grow this each time a pod is added

```
# Allow Apache to access Vitess/Mysql in pods [1,2]-2
iptables –i eth0 –p tcp –dport 3306 –s ${IP1-1} –d ${IP1-2}
iptables –i eth0 –p tcp –dport 3306 –s ${IP2-1} –d ${IP1-2}
iptables –i eth0 –p tcp –dport 3306 –s ${IP1-1} –d ${IP2-2}
iptables –i eth0 –p tcp –dport 3306 –s ${IP2-1} –d ${IP2-2}
…
```

iptables Scalability/Performance Challenges

- iptables in Kubernetes
  - Used for L3/L4 load-balancing (kube-proxy), security filtering (some CNI plugins)
  - Each pod create/delete => add/delete of iptables config, <u>across all hosts</u>
- Control Plane
  - Highly ephemeral Kubernetes pods
  - iptables rules can’t be add/removed incrementally (CPU, latency to update rules)
- Data Plane
  - kube-proxy relies on per-packet linear traversal of rules for load-balancing (CPU, packet latency)

kubernetes / kubernetes

Watch 3,016 Star 57

Code | Issues 2,201 | Pull requests 1,054 | Actions | Projects 9 | Security | Insights

KubeProxy is sometimes consumes 0.7 CPU on 100 Node clusters when doing nothing #44613

https://github.com/kubernetes/kubernetes/issues/44613

Latency to Add IPTables Rules

- Time spent to add one rule when there are 5k services (40k rules): 11 minutes
- 20k services (160k rules): 5 hours

IPTables Service Routing Performance

Where is latency generated?

- Long list of rules in a chain
- Enumerate through the list to find a service and pod

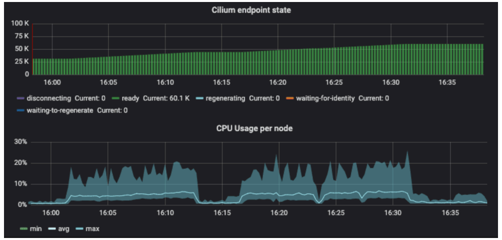
In this test, there is one entry per service in KUBE-SERVICES chain.

	1 Service (µs)	1000 Services (µs)	10000 Services (µs)	50000 Services (µs)
First Service	575	614	1023	1821
Middle Service	575	602	1048	4174
Last Service	575	631	1050	7077

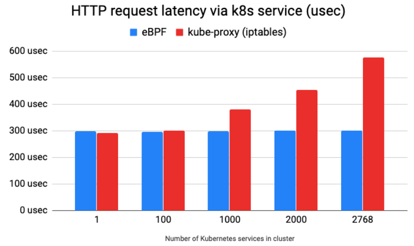
https://www.slideshare.net/LCChina/scale-kubernetes-to-support-50000-services

Cilium Scalability / Performance

- Cilium + Kubernetes
  - Implements L3/L4 LB, security filtering as highly-optimized BPF programs
- Control Plane
  - Incremental BPF map updates + BPF templating make pod addition lightweight
  - Scales to: 5K nodes, 100K pods, 20K svcs
- Data Plane Scalability/Performance
  - Highly optimized BPF programs
  - Efficient hash-lookups, rather than linear traversals via kube-proxy

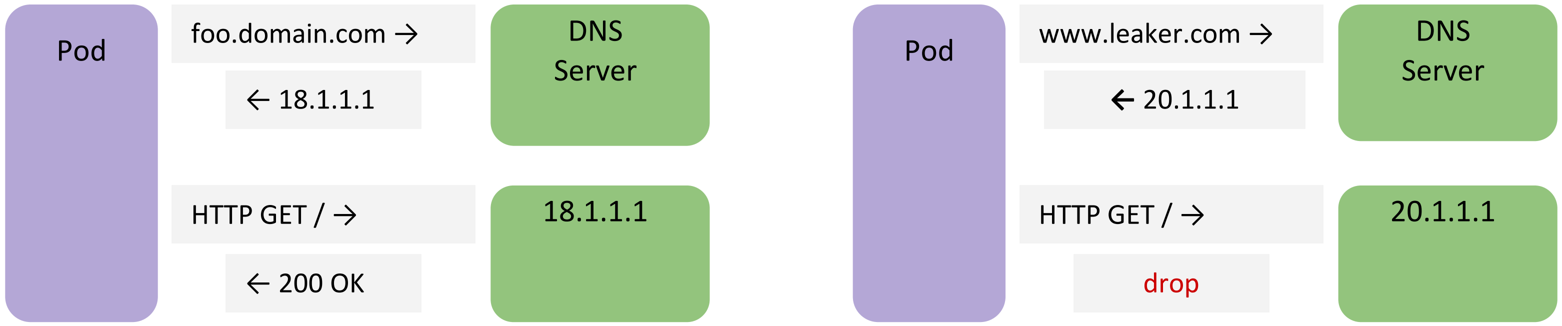


Scaling K8s env from 30K ➔ 60K pods has minimal and temporary CPU consumption.
https://cilium.io/blog/2019/04/24/cilium-15/



Latency with BPF-based NodePort vs. kube-proxy(iptables).
https://cilium.io/blog/2019/08/20/cilium-16/

Another Cilium Option - DNS-aware **Security**

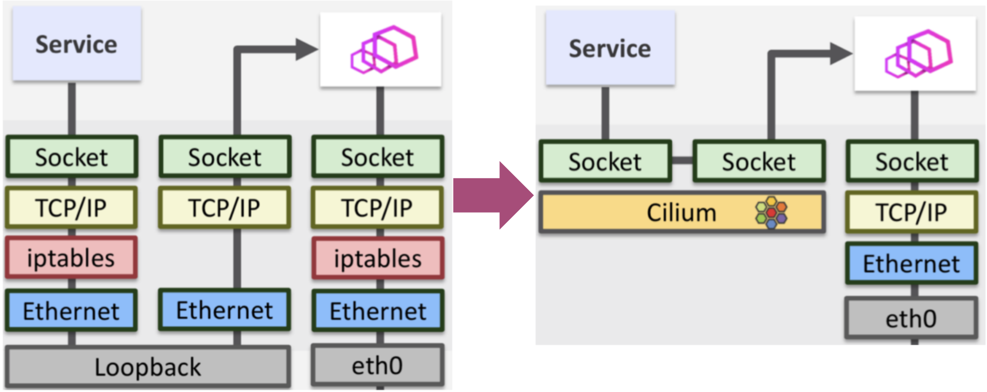


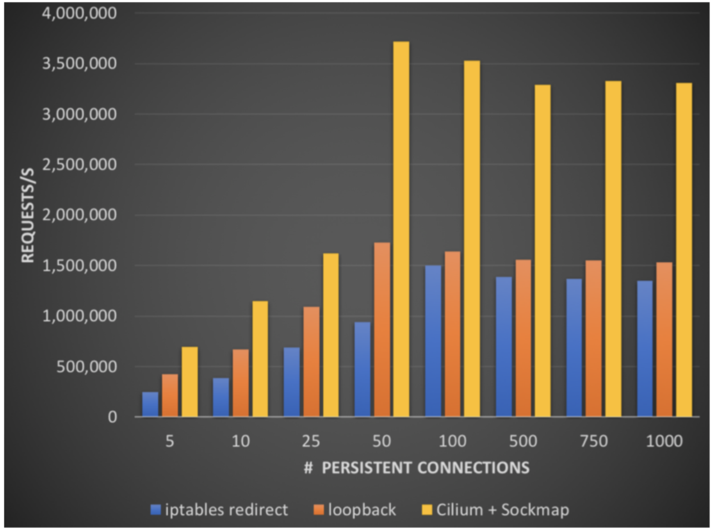

```yaml
- toFQDNs:
  - matchPattern: "*.domain.com"
  toPorts:
  - ports:
    - port: '443'
      protocol: TCP
```

Why Did SUSE Choose Cilium for CaaSP V4?

- Identity Aware Security (labels or DNS) **Reduces Op-Ex** via simple policy declarations that require no manual intervention as pods/nodes scale
- Underlying tool (BPF/eBPF) is architecturally superior and more efficient for highly dynamic workloads and their corresponding networking requirements => **Reduces Cap-Ex** via better hardware utilization
- **Reducing Op-Ex and Cap-Ex with one feature and its underlying BPF is more than sufficient reason to take a strong look at Cilium**
- Advanced functionality and additional performance optimizations occurring rapidly within Cilium…and let’s take a look at some of those

Cilium Envoy Acceleration (3X gain)



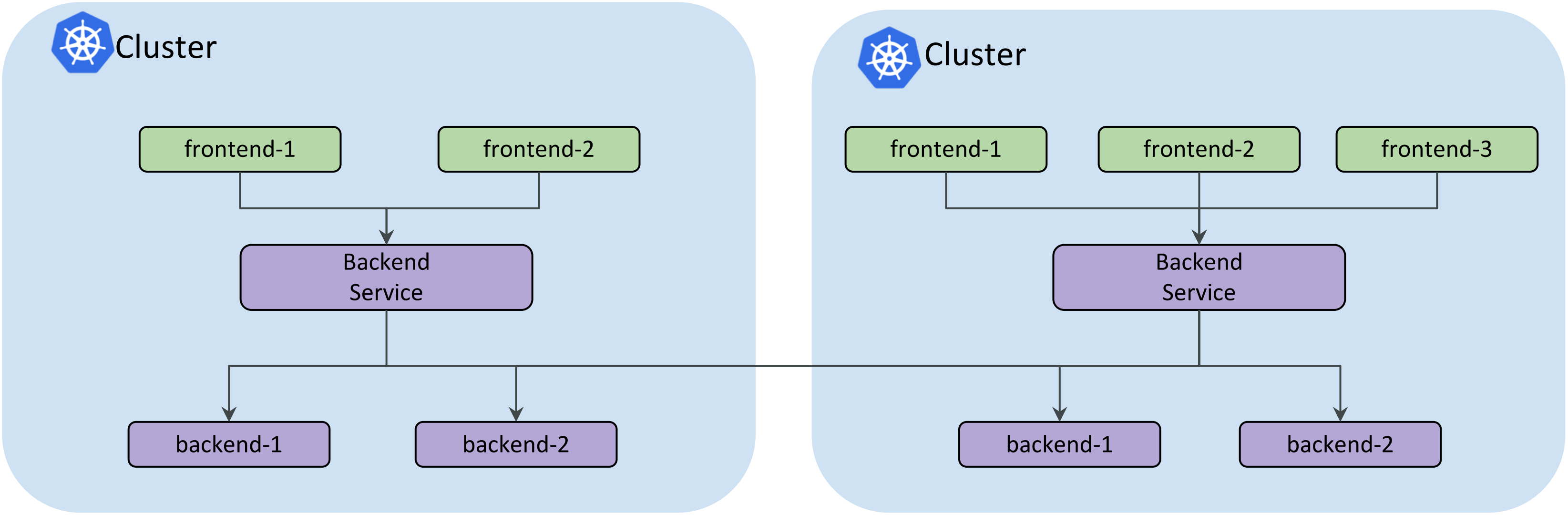


More info in KubeCon EU 2018 slides:

Accelerating Envoy and Istio with Cilium and the Linux Kernel

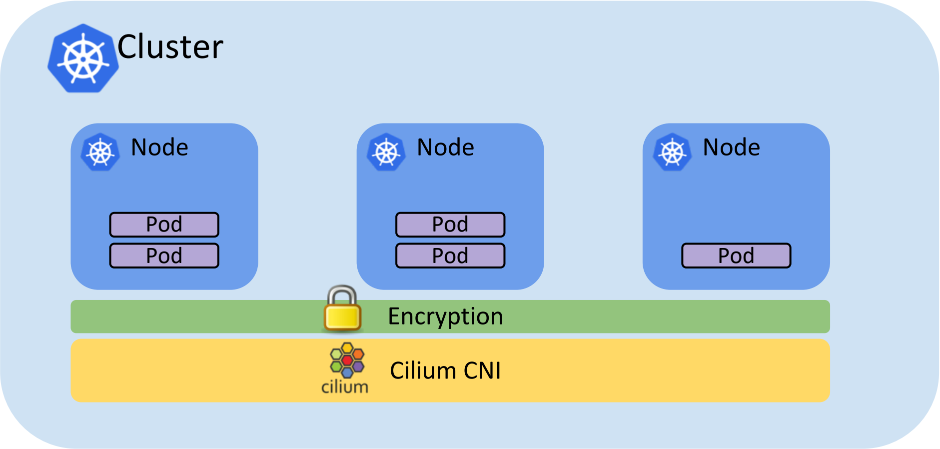
https://bit.ly/2G7DfIY

Multi-Cluster Service Routing




```
metadata:
  annotations:
    io.cilium/global-service: "true"
```

Transparent Encryption



API Firewall

```yaml
apiVersion: "cilium.io/v2"
kind: CiliumNetworkPolicy
description: "Allow HTTP GET /public from env=prod to app=service"
metadata:
  name: "rule1"
spec:
  endpointSelector:
    matchLabels:
      app: service
  ingress:
  - fromEndpoints:
    - matchLabels:
        env: prod
    toPorts:
    - ports:
      - port: "80"
        protocol: TCP
      rules:
        http:
        - method: "GET"
          path: "/public"
```

Data Store Authorization

```yaml
apiVersion: "cilium.io/v2"
kind: CiliumNetworkPolicy
[…]
specs:
- endpointSelector:
   matchLabels:
     app: cassandra
ingress:
- toPorts:
  - ports:
    - port: "9042"
      protocol: TCP
      l7proto: cassandra
      l7:
      - query_action: "select"
        query_table: "myTable"
```

Agenda

- Why is SUSE Presenting Cilium? – A Little Context
- Why Cilium?
  - A Bit of Cilium Background
  - Use Cases – Some Fundamental Networking Needs
  - Luxury/Future Use Cases – What we see on the Horizon
- A Deeper Dive into Cilium Internals
  - What is BPF and How Does it Work?
  - How Does Cilium Use BPF?

What is BPF / eBPF?

Berkeley Packet Filter

Flexible: Executes custom logic in the Linux kernel.

Safe: BPF code is verified to not crash/hang kernel.

Fast: JIT-compiled to run at native speed.

Humble origins:

```
tcpdump -n dst host 192.168.1.1
```

Learn More: http://docs.cilium.io/en/latest/bpf

BPF Tech Adoption

- L3-L4 Load balancing
- Network security
- Traffic optimization
- Profiling

https://code.fb.com/open-source/linux/

- Replacing iptables with BPF
- NFV & Load balancing (XDP)
- Profiling & Tracing

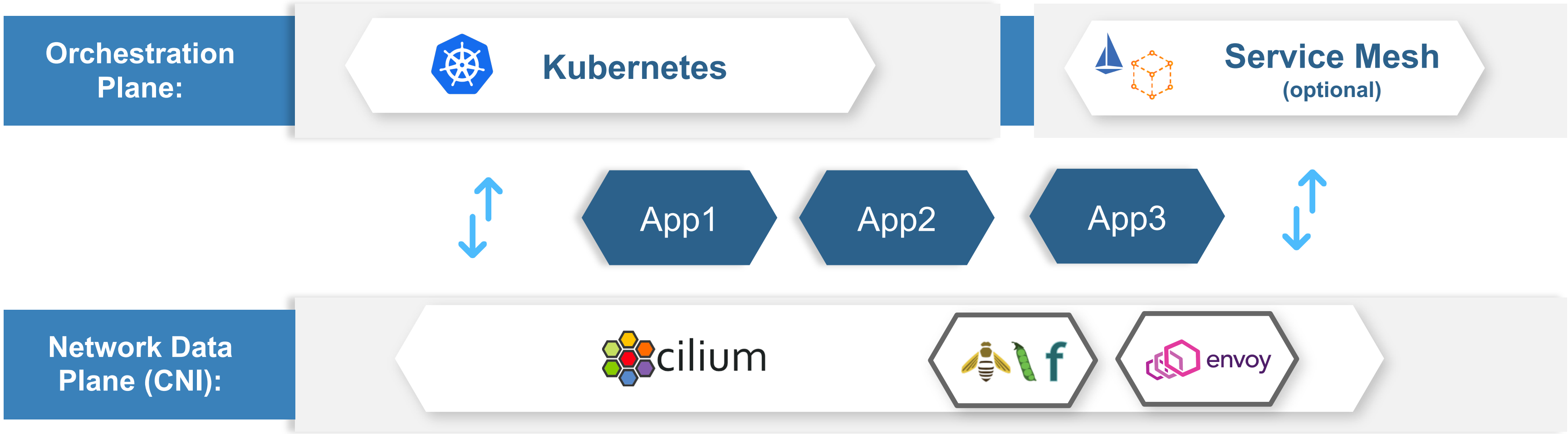
https://goo.gl/6JYYJW

- QoS & Traffic optimization
- Network Security
- Profiling
- http://vger.kernel.org/lpc-bpf2018.html#session-1

- Performance Troubleshooting
- Tracing & Systems Monitoring
- Networking

http://www.brendangregg.com/blog/2016-03-05/linux-bpf-superpowers.html

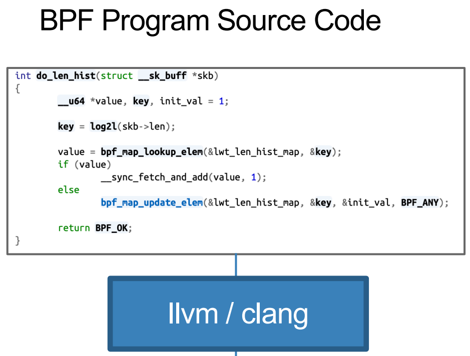
Cilium: Bringing the Power of BPF to Kubernetes & Service Mesh

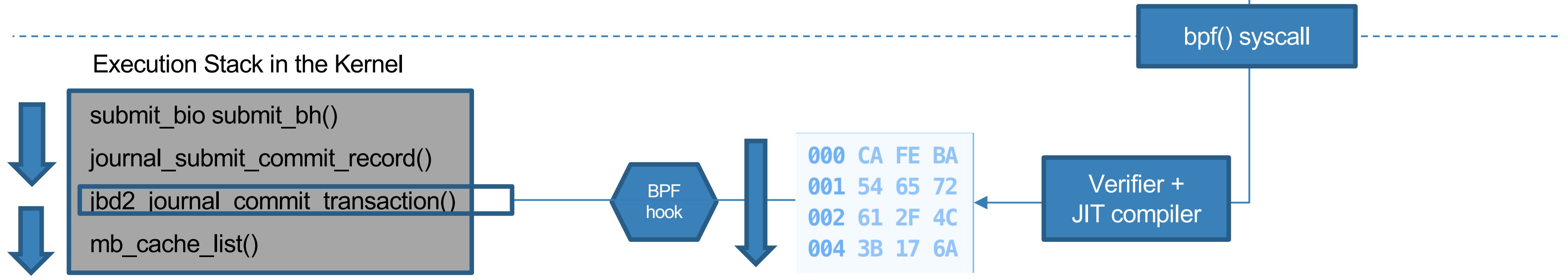


BPF Concepts #1: Programs and Hook Points

“Function-as-a-Service” for kernel events

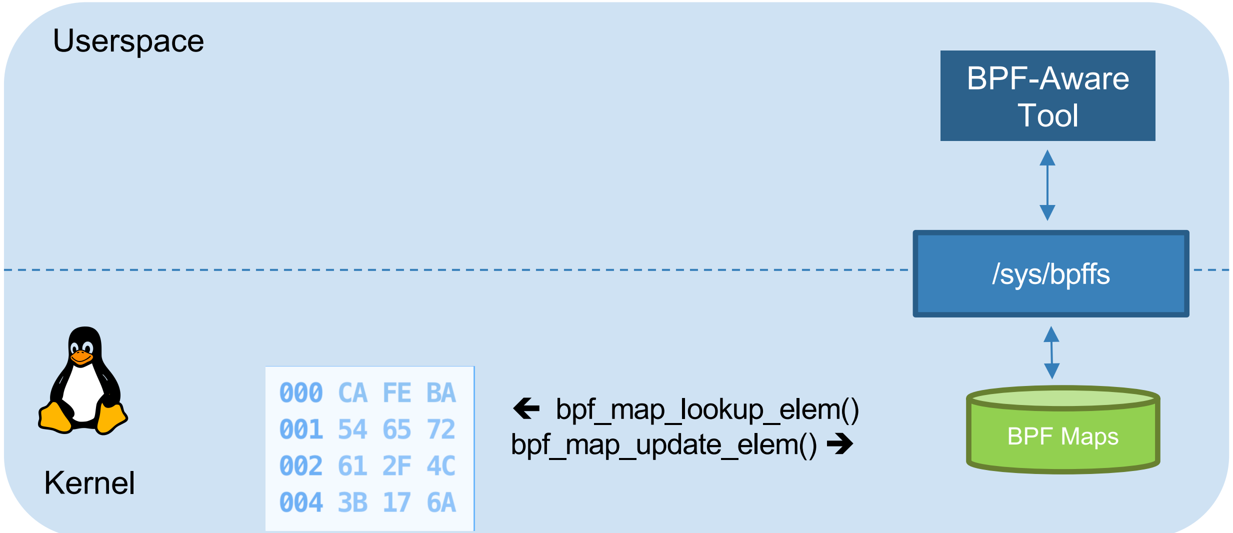
with strong safety guarantees and native kernel performance





BPF Concepts #2: Maps

Efficient data structures that persist across function invocation.

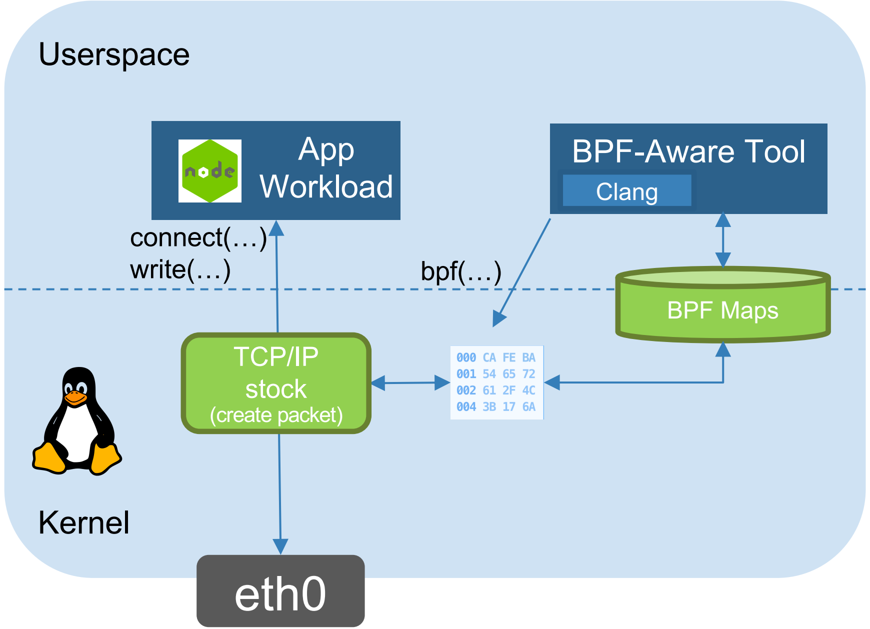


Highly Efficient:

- Fine-grained update of BPF program config data (e.g., policy/load-blancing rules)
- Accumulation of visibility data in-kernel, with only summaries exported to userspace.

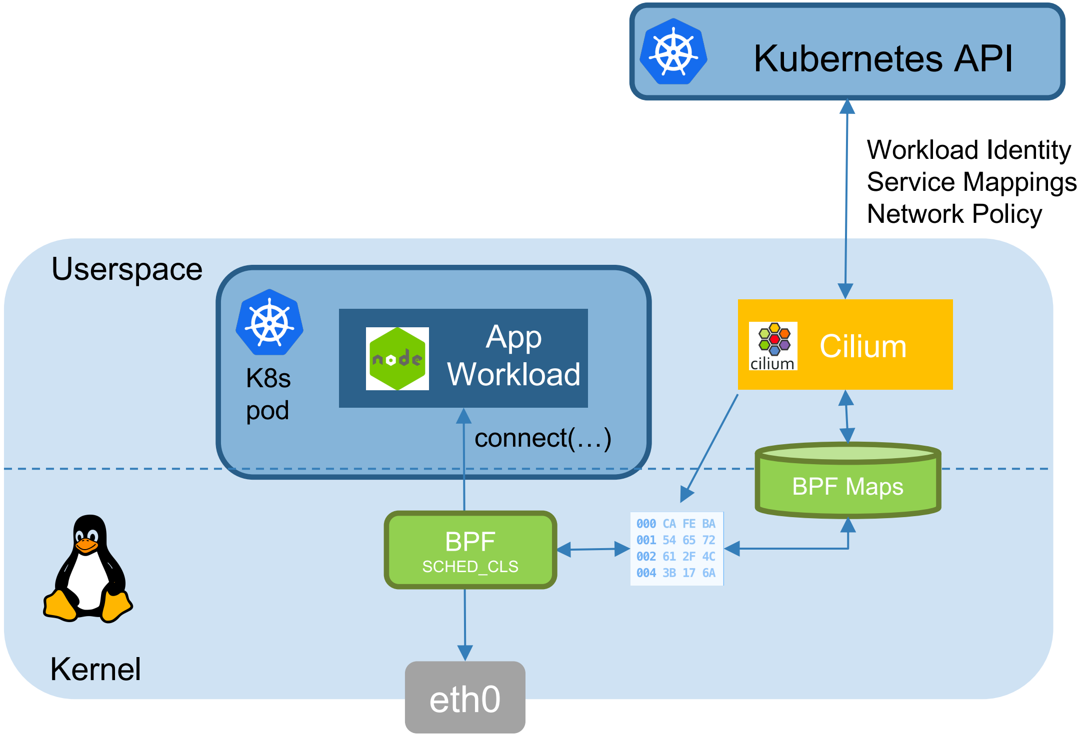
https://lwn.net/Articles/664688/

Putting it Together: BPF Networking Filtering Example



1. Tool generates BPF program to filter packets based on contents of a BPF Map.
2. Tool compiles to BPF program to byte code
3. Tool uses bpf() syscall to load byte code into kernel at hook point that sees each IP packet.
4. Kernel verifies safety of code, JIT-compiles for native perf.
5. Userspace tool inserts IPs to block to as entry in a BPF map.
6. Application calls connect() and writes data to socket. BPF program is run for each packet.

How Cilium Uses BPF



Cilium-generated BPF programs control:

- Pod-to-Pod Network Connectivity.
- Service-based Load-balancing.
- Network Visibility and Security Enforcement

Questions

CLOUD NATIVE
COMPUTING FOUNDATION

Cilium Community

https://github.com/cilium/cilium

cilium Star 4,334

slack https://cilium.io/slack

10,000+ Commits

from

107 Contributors

ISOVALENT · SUSE · Palantir · Adobe · DATADOG · Digital Ocean · Ctrip · Capital One · overstock.com

KubeCon | CloudNativeCon

North America 2019

Nov. 18 - 21, 2019
San Diego, CA
kubecon.io

Feel Free to Reach Out...

Dan Wendlandt, dan@isovalent.com
Mark Darnell, mark.darnell@suse.com
Roger Klorese, roger.klorese@suse.com

Thank You

From SUSE and Isovalent – be sure to visit us at Kubecon San Diego!

CLOUD NATIVE
COMPUTING FOUNDATION