Service MESH without the MESS

FOSDEM 2023

Speaker: Raymond de Jong
@dejongraymond
Agenda

- eBPF & Cilium Introduction
- Service Mesh Evolution
- Cilium Service Mesh
- Features
- Demo
eBPF & Cilium Introduction
eBPF

Makes the Linux kernel programmable in a secure and efficient way.

“What JavaScript is to the browser, eBPF is to the Linux Kernel”
Run eBPF programs on events

Attachment points
- Kernel functions (kprobes)
- Userspace functions (uprobe)
- System calls
- Tracepoints
- Sockets (data level)
- Network devices (packet level)
- Network device (DMA level) [XDP]
- ...

Diagram showing attachment points at various levels:
- File Descriptor
- VFS
- Block Device
- TCP retrans
- Sockets
- TCP/IP
- Network Device
- Disk
- NIC
What is Cilium?

- **Networking & Load-Balancing**
  - CNI, Kubernetes Services, Multi-cluster, VM Gateway
- **Network Security**
  - Network Policy, Identity-based, Encryption
- **Observability**
  - Metrics, Flow Visibility, Service Dependency

At the foundation of Cilium is the new Linux kernel technology eBPF, which enables the dynamic insertion of powerful security, visibility, and networking control logic within Linux itself. Besides providing traditional network level security, the flexibility of BPF enables security on API and process level to secure communication within a container or pod.

[Read More](#)
cilium
Created by ISOVALENT

eBPF-based:
- Networking
- Security
- Observability
- Service Mesh & Ingress

Foundation
- CLOUD NATIVE COMPUTING FOUNDATION
- eBPF
- envoy

Technology
- eBPF
- envoy

- Adobe
  What Makes a Good Multi-tenant Kubernetes Solution
- Capital One
  Building a Secure and Maintainable PaaS
- Alibaba Cloud
  Building High-Performance Cloud-Native Pod Networks
- MÁSMÓVIL
  Scaling a Multi-Tenant k8s Cluster in a Telco
- Trip.com
  First step towards cloud native networking
- CENGN
  Cloud Native Networking with eBPF
- DigitalOcean
  Managed Kubernetes: 1.5 Years of Cilium Usage at DigitalOcean
- Google
  Google chooses Cilium for Google Kubernetes Engine (GKE) networking
- Bell
  Why eBPF is changing the Telco networking space?
- GitLab
  Kubernetes Network Policies in Action with Cilium
- AWS
  AWS picks Cilium for Networking & Security on EKS Anywhere
- Scaleway
  Scaleway uses Cilium as the default CNI for Kubernetes Kapsule
- sportradar
  Sportradar is using Cilium as their main CNI plugin in AWS (using kops)
- Utmost
  Utmost is using Cilium in all tiers of its Kubernetes ecosystem to implement zero trust
- Yahoo!
  Yahoo is using Cilium for L4 North-South Load Balancing for Kubernetes Services

ISOVALENT
Service Mesh Introduction
Service Mesh

- App
- Observability
- Security
- App
- Traffic Management
- Resilience
Each application requires a service mesh library written in the language framework of the application.
Service Mesh with Sidecars

Sidecar Service Mesh Model

Python App

Sidecar Proxy
Mesh Library

Sidecar Proxy
Mesh Library

Go App

Service mesh is embedded in a proxy running outside of the application.
Service Mesh Evolution

Shared Library Model
- App
  - Service Mesh Library
- Kernel
  - TCP/IP
  - Network

Sidecar Model
- App
  - Service Mesh Sidecar
- App
  - Service Mesh Sidecar
- Kernel
  - TCP/IP
  - Network

Kernel Model
- App
- App
- Kernel
  - Service Mesh
  - envoy
  - CBPF
  - TCP/IP
  - Network
Layer 7 is the only part which is not yet there

- **L7**: HTTP, gRPC...
- **L4**: TCP, UDP, ICMP...
- **L3**: IP
- **L2**: Ethernet...
Yet, Cilium already has L7 network policies and visibility.
Cilium Service Mesh
Cilium agent per node

- Dynamic eBPF programs
- Envoy for L7 policies & observability
Cilium for sidecarless service mesh

- Dynamic eBPF programs
- Envoy for L7 policies & observability and traffic management rules etc
What is different with Cilium Service Mesh?

- Reduced operational complexity
- Reduced resource usage
- Better performance
- Avoid sidecar startup/shutdown race conditions
Reduce resource usage - sidecar vs proxy per node

Total number of proxies required

30 pods/node → 30 proxies/node
Cost of sidecar injection
eBPF powered network path for L3/L4 traffic
Envoy for Layer 7 termination when needed
Latency performance

Latency at different target request per second (rps)
Lower is better

All data & Scripts: https://isovalent.com/blog/post/2022-05-03-servicemesh-security
Throughput performance

Max Throughput (rps)

Higher is better

All data & Scripts: https://isovalent.com/blog/post/2022-05-03-servicemesh-security
Pod ready performance

Time it takes for naked pods to become Ready
Lower is better

<table>
<thead>
<tr>
<th></th>
<th>Pod Ready (sec)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Cilium - 200 pods/25qps - No workload (99th%ile)</td>
<td>49.1458</td>
</tr>
<tr>
<td>Istio - 200 pods/25qps - No workload (99th%ile)</td>
<td>154.4898</td>
</tr>
</tbody>
</table>

Time it takes for Job & Deployment pods to become Ready
Lower is better

<table>
<thead>
<tr>
<th></th>
<th>Pod Ready (sec)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Cilium - 100 pods/5qps - Workload (99th%ile)</td>
<td>8.1064</td>
</tr>
<tr>
<td>Istio - 100 pods/5qps - Workload (99th%ile)</td>
<td>123.4904</td>
</tr>
</tbody>
</table>

All data & Scripts: [https://isovalent.com/blog/post/2022-05-03-servicemesh-security](https://isovalent.com/blog/post/2022-05-03-servicemesh-security)
Cilium 1.12 Release

- Production Ready Cilium Service Mesh
- Conformant Ingress Controller
- Using Kubernetes as Service Mesh Control Plane
  - Simple to use sidecar-free Service Mesh configured using Kubernetes Services and Ingress
- Prometheus metrics and OpenTelemetry
- CiliumEnvoyConfig and CiliumClusterEnvoyConfig CRD
- Extended Grafana dashboards for L7 visibility
Roadmap 1.13

- Gateway API
  - HTTP Routing
  - TLS Termination
  - HTTP Traffic Splitting / Weighting
- Multiple Ingress per Load Balancer
- More L7 metrics collection through Isovalent Tetragon Enterprise
## Layer 7 Traffic Management Options

<table>
<thead>
<tr>
<th>Ingress</th>
<th>Services</th>
<th>Gateway API</th>
<th>EnvoyConfig</th>
</tr>
</thead>
<tbody>
<tr>
<td>Original L7 load-balancing standard in K8s</td>
<td>Use of K8s services with annotations</td>
<td>Originally labelled Ingress v2. Richer in features.</td>
<td>Raw Envoy Config via CustomResource</td>
</tr>
<tr>
<td>Simple</td>
<td>Simple</td>
<td>Simple</td>
<td>Advanced Users &amp; Integrations</td>
</tr>
<tr>
<td>Supported since Cilium 1.12</td>
<td>Support coming in Cilium 1.13</td>
<td>Support for v0.5.1 coming in Cilium 1.13</td>
<td>Supported since Cilium 1.12</td>
</tr>
</tbody>
</table>
**Ingress**

- Ingress can be used for path-based routing and TLS termination
- Cilium manages Ingress resources without external Ingress Controller
- Cilium Service Mesh Ingress Controller requires ability to create Service of Type LoadBalancer using either Cloud Provider integration or e.g. MetalLB
- Ingress CRD with `ingressClassName: cilium`
Ingress HTTP Example
Ingress gRPC Example

gRPC requests to: `grpcurl hipstershop.ProductCatalogService/ListProducts` and `grpcurl hipstershop.CurrencyService/GetSupportedCurrencies`
TLS Termination
Gateway API

Use of Gateway and HTTPRoute objects for path-based routing

```yaml
apiVersion: gateway.networking.k8s.io/v1beta1
kind: Gateway
metadata:
  name: my-gateway
spec:
  gatewayClassName: cilium
  listeners:
  - protocol: HTTP
    port: 80
    name: web-gw
  allowedRoutes:
    namespaces:
      from: Same
```

```yaml
apiVersion: gateway.networking.k8s.io/v1alpha2
kind: HTTPRoute
metadata:
  name: http-app-1
spec:
  parentRefs:
  - name: my-gateway
    namespace: default
  rules:
  - matches:
      - path:
        type: PathPrefix
        value: /details
  backendRefs:
  - name: details
    port: 9080
```
Gateway API

Use of Gateway and HTTPRoute for TLS Termination

```yaml
apiVersion: gateway.networking.k8s.io/v1beta1
kind: Gateway
metadata:
  name: tls-gateway
spec:
gatewayClassName: cilium
listeners:
  - name: https
    protocol: HTTPS
    port: 443
    hostname: "bookinfo.cilium.rocks"
    tls:
      certificateRefs:
      - kind: Secret
        name: demo-cert
```

```yaml
apiVersion: gateway.networking.k8s.io/v1beta1
kind: HTTPRoute
metadata:
  name: https-app-route
spec:
  parentRefs:
    - name: tls-gateway
  hostnames:
    - "bookinfo.cilium.rocks"
  rules:
    - matches:
        - path:
            type: PathPrefix
            value: /details
      backendRefs:
        - name: details
          port: 9080
```
Gateway API
Traffic Splitting with Weighted Routes

```yaml
apiVersion: gateway.networking.k8s.io/v1alpha2
class: HTTPRoute
metadata:
  name: example-weighted-route
spec:
  parentRefs:
  - name: my-gateway
  rules:
  - matches:
    - path:
      type: PathPrefix
      value: /echo
  backendRefs:
  - kind: Service
    name: echo-1
    port: 8080
    weight: 75
  - kind: Service
    name: echo-2
    port: 8090
    weight: 25
```
Service + Annotations

Simple way to enable gRPC weighted-least-request load balancing

```yaml
apiVersion: v1
kind: Service
metadata:
  name: backend
annotations:
  io.cilium/lb-protocol: "grpc"
  io.cilium/lb-mode: "weighted-least-request"
spec:
  type: ClusterIP
  ports:
    - port: 80
  selector:
    name: backend
```
Service + Annotations + Multi-Cluster

Compatible with multi-cluster load balancing

```yaml
apiVersion: v1
kind: Service
metadata:
  name: backend
annotations:
  io.cilium/global-service: "true"
  io.cilium/lb-protocol: "grpc"
  io.cilium/lb-mode: "weighted-least-request"
spec:
  type: ClusterIP
  ports:
  - port: 80
  selector:
    name: backend
```
Learn more!

For the Enterprise
Hardened, enterprise-grade eBPF-powered networking, observability, and security.

isovalent.com/product
isovalent.com/labs

OSS Community
eBPF-based Networking, Observability, Security

cilium.io
cilium.slack.com
Regular news

Base technology
The revolution in the Linux kernel, safely and efficiently extending the capabilities of the kernel.

ebpf.io
What is eBPF? - ebook
Thank you!