ISOVALENT

cilium

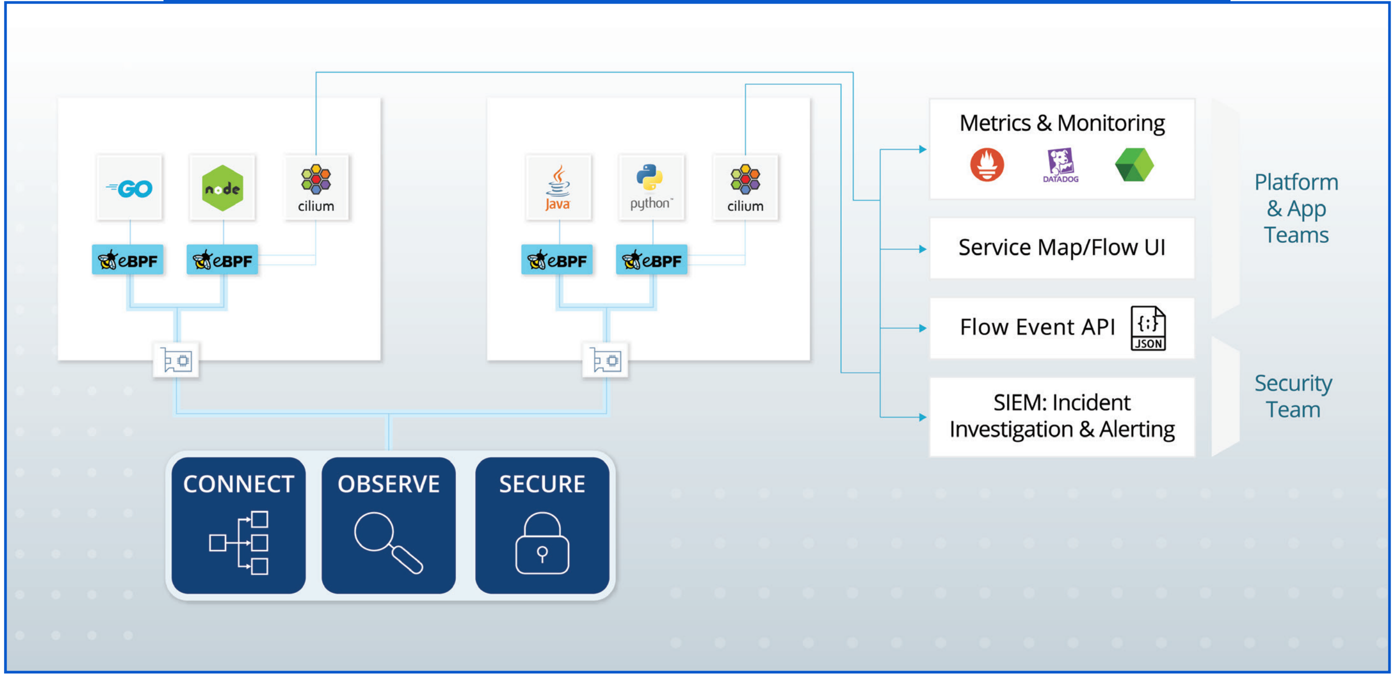
Accelerating the Journey to Cloud Native Microservices

Overcoming the Networking, Observability and Security challenges that slow down adoption of production Kubernetes platforms

Executive Summary

Enterprises are rapidly adopting Cloud Native technologies to efficiently support the rollout of business applications. Unfortunately, they quickly realize that legacy approaches that rely on IPs and ports as their main notion of identity are useless in the highly dynamic world of Kubernetes. eBPF, a technology that allows one to safely run programs in the Linux Kernel, promises to fundamentally change the way cloud-native networking is done. Cilium leverages a new and powerful Linux kernel technology, eBPF, to provide high-performance, cloud native-aware networking, observability, and security. Cilium's unique capabilities enable Kubernetes architects to solve a wide array of challenges commonly faced as Kubernetes platforms mature.

Cilium: eBPF-Powered Networking, Observability + Security



The 3 top challenges faced by Kubernetes architects as their platforms become more mission critical include:

1. **Addressing Security & Compliance Requirements**
2. **Providing Advanced Connectivity**
3. **Ensuring Identity-Aware Observability for Platform and Application teams**

Let's look at each of these challenges and the approaches to resolving them:

Addressing Zero Trust Networking, Security Visibility and Compliance Requirements

Description

Out of the box, Kubernetes allows workload to workload / service to service communication and infiltration of any workload would hand the keys of your whole kingdom to an attacker. Therefore a key requirement to migrating business critical workloads onto the Kubernetes environment is adopting a Zero Trust security model. Compliance requirements for these workloads dictate network isolation between tenant workloads within a cluster and restricted access to external workloads. Implementing a default deny model helps prevent breaches, insider attacks, and resulting lateral movement by the attacker. Security visibility becomes critical for SecOps teams to sign-off and allow critical workloads to run in a Kubernetes environment, as they require the tools to perform efficient incident investigations and monitor all key compliance requirements.

Challenges with legacy approaches

Traditional IP-based firewalls cannot implement the required network isolation requirements for compliance because of the dynamic nature by which pods are created and assigned IP addresses. By default Kubernetes environments allow any pod to connect to any other pod, even across namespaces. As a result, an attacker can leverage an intrusion to perform lateral attacks, dramatically increasing the potential harm. Implementing Kubernetes Network Policies is a great first step toward Zero Trust but teams often run into a range of challenges as they try to lock down their environment. Not only can it be painful to get the YAML syntax and formatting just right, but there are many subtleties in the behavior of the network policy specification (e.g. default allow/deny, namespacing, wildcarding, rules combination, etc.) which can lead to broken apps or unforeseen vulnerable configurations.

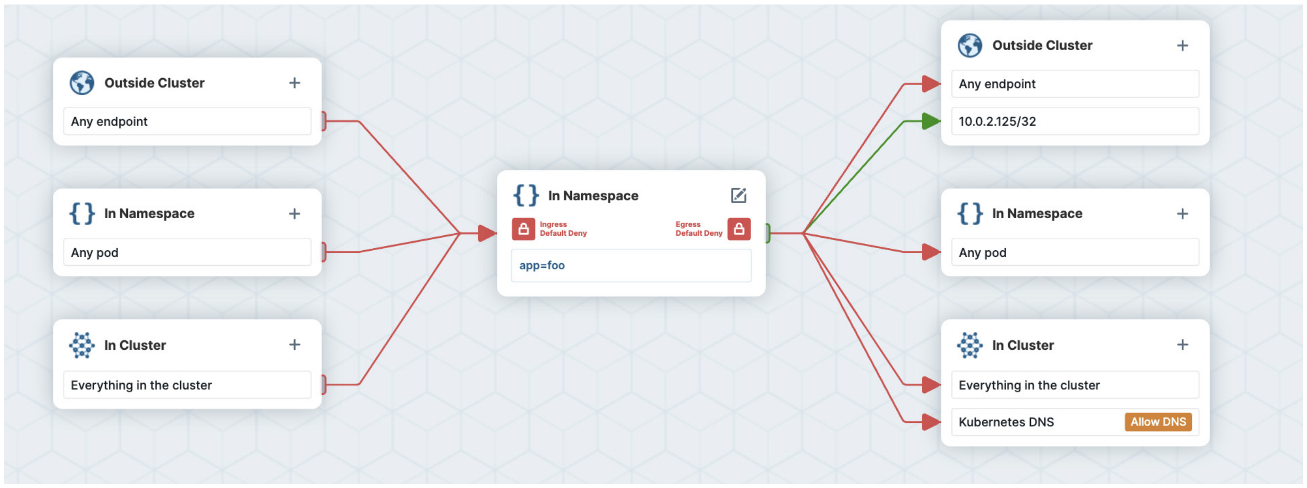
Legacy approaches to network security visibility provide little help when performing threat detection, compliance monitoring, or incident investigations for Kubernetes workloads. As many tools operate only at the network perimeter and/or lack application context, they miss the vast majority of service-to-service communications and are limited in their visibility by relying on IP and port-based flow logs. Many legacy approaches map IPs back to a host, erroneously assuming only one application for each host, while in the Kubernetes environment there can in fact be dozens.

Compliance requirements also often dictate that sensitive traffic is encrypted in flight. Implementing encryption at the app layer adds work and complication, and ensuring all traffic that needs to be encrypted is protected by the appropriate TLS version and ciphers is often manual and incomplete.

How Cilium Enterprise solves these challenges

Zero Trust Security

Going well beyond what is possible with traditional Linux networking like iptables, Cilium's eBPF-powered datapath natively understands an application's identity and purpose, implementing not only basic Kubernetes Network Policy (e.g. Label + CIDR matching) but also supporting DNS-aware network policies (e.g. allow to *.google.com), which dramatically simplifies defining zero-trust policies for accessing services outside of the Kubernetes cluster.

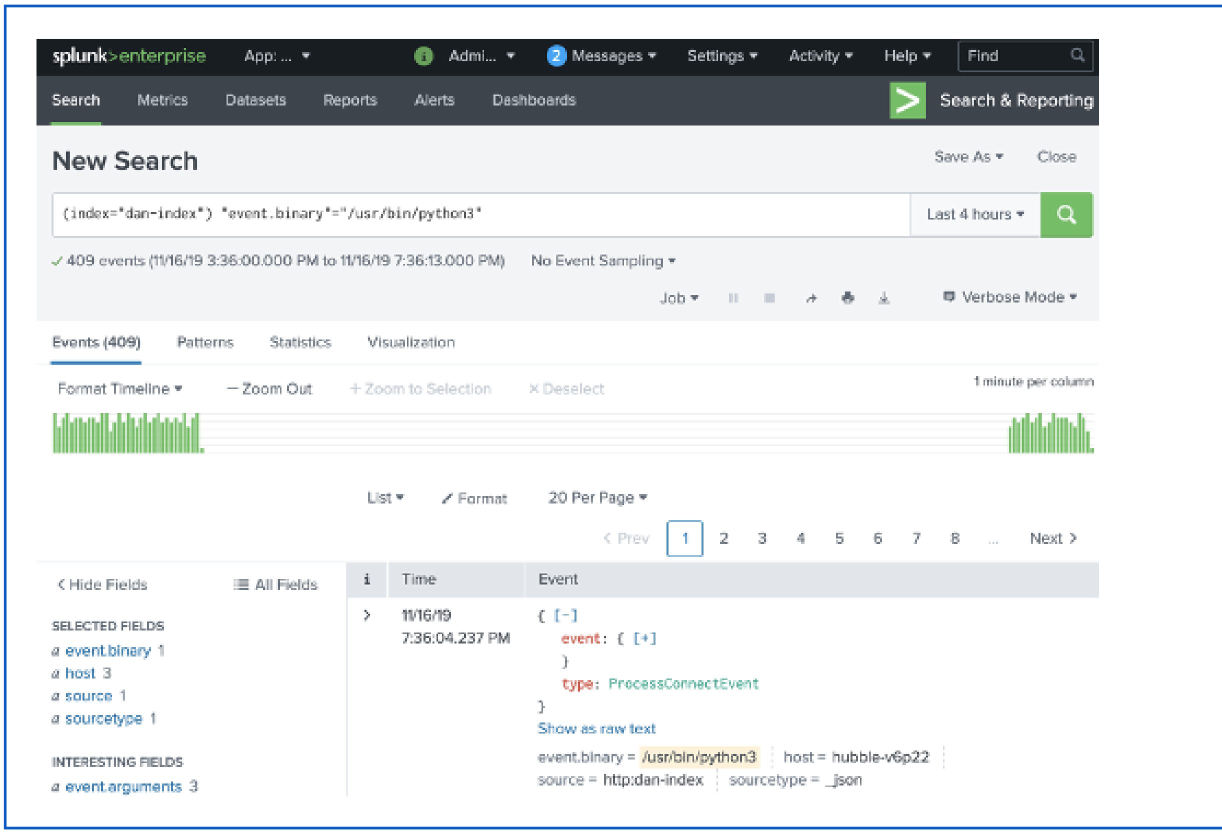


Additionally, Cilium supports L7 policies (e.g. allow HTTP GET /foo) for fine-grained access control to shared API services running common cloud native protocols like HTTP, gRPC, Kafka, etc. Cilium also supports deny-based network policies, cluster-wide network policy, and host-layer firewalling.

Security visibility - Incident investigation, alerting and flow audit

Cilium efficiently monitors the precise Linux process and command, container, and Kubernetes pod identity for each connection. Cilium exports this data to a SecOps team's existing SIEM (e.g. Splunk, ELK etc.), providing all the visibility needed to identify potential breaches, investigate attacks and lateral movement, and audit the environment for security compliance.

The identity-aware flow logs from Cilium's Hubble can also be stored to enable long-term forensics of network connectivity and attack investigation. Cilium's API-awareness, optionally combined with TLS-termination enables security visibility even at the HTTP-layer.



```
(index="dan-index") "event.binary"="/usr/bin/python3"
```

✓ 409 events (11/16/19 3:36:00.000 PM to 11/16/19 7:36:13.000 PM)

SELECTED FIELDS
a event.binary 1
a host 3
a source 1
a sourcetype 1

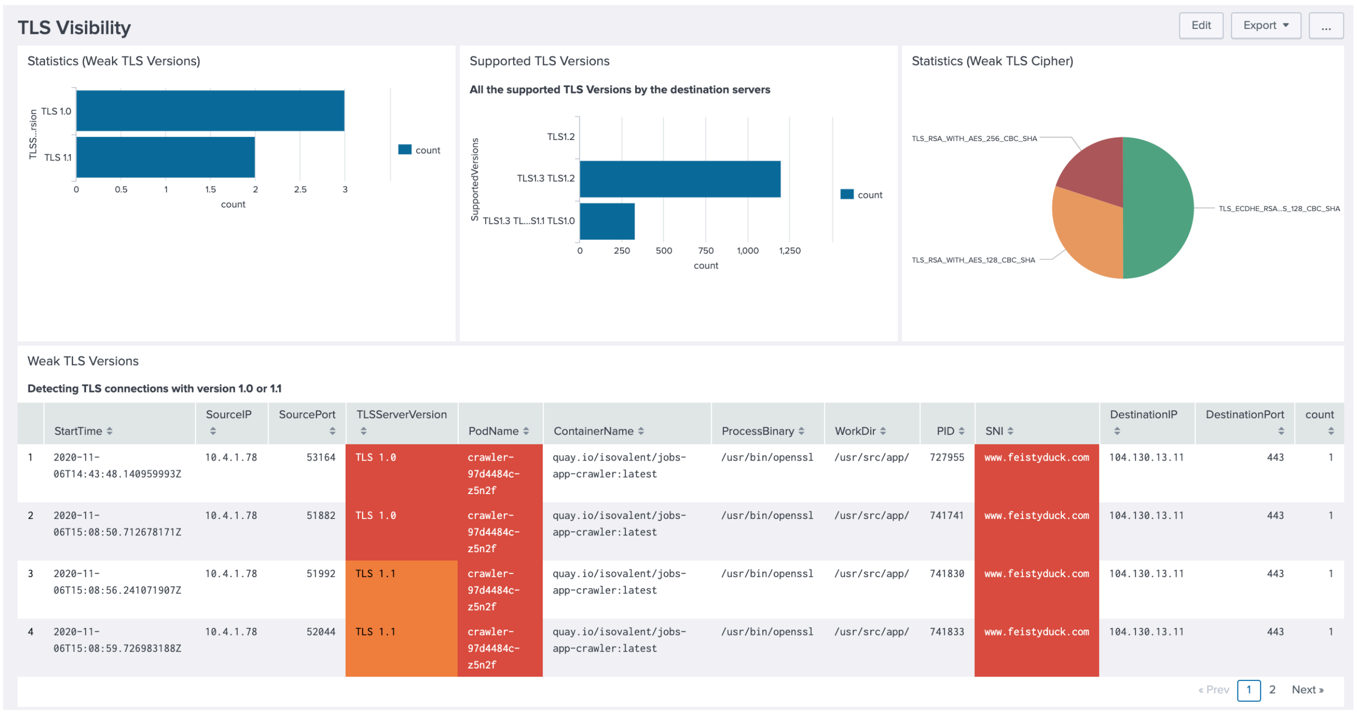
INTERESTING FIELDS
a event.arguments 3

i	Time	Event
›	11/16/19 7:36:04.237 PM	{ [-] event: { [+] } type: ProcessConnectEvent }

```
event.binary = /usr/bin/python3 | host = hubble-v6p22
source = http:dan-index | sourcetype = _json
```

Encryption & Compliance Monitoring

Cilium's transparent encryption capabilities use the highly efficient IPsec capabilities built into the Linux kernel to automatically encrypt communications between all workloads within, or between, Kubernetes clusters to ensure workload compliance. This mechanism is simple: it requires only a single configuration setting in Cilium and no application changes. Cilium's compliance monitoring capabilities ensure all traffic that needs to be encrypted is protected by the appropriate TLS version and ciphers, that the SNI matches the original destination DNS name, and that the certificate received is signed by a trusted certificate authority.


TLS Visibility
Edit
Export
...
Statistics (Weak TLS Versions)
TLSS...rsion
TLS 1.0
TLS 1.1
0
0.5
1
1.5
2
2.5
3
count
count
Supported TLS Versions
All the supported TLS Versions by the destination servers
SupportedVersions
TLS1.2
TLS1.3 TLS1.2
TLS1.3 TL...S1.1 TLS1.0
0
250
500
750
1,000
1,250
count
count
Statistics (Weak TLS Cipher)
TLS_RSA_WITH_AES_256_CBC_SHA
TLS_ECDHE_RSA...S_128_CBC_SHA
TLS_RSA_WITH_AES_128_CBC_SHA

Weak TLS Versions

Detecting TLS connections with version 1.0 or 1.1

	StartTime	SourceIP	SourcePort	TLSServerVersion	PodName	ContainerName	ProcessBinary	WorkDir	PID	SNI	DestinationIP	DestinationPort	count
1	2020-11-06T14:43:48.140959993Z	10.4.1.78	53164	TLS 1.0	crawler-97d4484c-z5n2f	quay.io/isovalent/jobs-app-crawler:latest	/usr/bin/openssl	/usr/src/app/	727955	www.feistyduck.com	104.130.13.11	443	1
2	2020-11-06T15:08:50.712678171Z	10.4.1.78	51882	TLS 1.0	crawler-97d4484c-z5n2f	quay.io/isovalent/jobs-app-crawler:latest	/usr/bin/openssl	/usr/src/app/	741741	www.feistyduck.com	104.130.13.11	443	1
3	2020-11-06T15:08:56.241071907Z	10.4.1.78	51992	TLS 1.1	crawler-97d4484c-z5n2f	quay.io/isovalent/jobs-app-crawler:latest	/usr/bin/openssl	/usr/src/app/	741830	www.feistyduck.com	104.130.13.11	443	1
4	2020-11-06T15:08:59.726983188Z	10.4.1.78	52044	TLS 1.1	crawler-97d4484c-z5n2f	quay.io/isovalent/jobs-app-crawler:latest	/usr/bin/openssl	/usr/src/app/	741833	www.feistyduck.com	104.130.13.11	443	1

« Prev 1 2 Next »

Providing Advanced Connectivity

Description

As the enterprise continues to deploy complex, distributed applications on Kubernetes, it is crucial to have a network infrastructure that will scale to support the demands of hundreds if not thousands of containerized application components being created and replaced continuously. Networking would be easy if all communications stayed within small Kubernetes clusters. Unfortunately, architects have to manage connectivity to legacy workloads, across multiple clusters at unprecedented scale as clusters get ever larger. Forwarding, load balancing and monitoring mechanisms that work fine in a small environment can suffer dramatic performance degradation as the environment and number of connections grows.

Challenges with legacy approaches

Networking:

Linux networking historically has relied on the fundamental construct of iptables as the primary tool to implement Linux firewalls and packet filters. Iptables were designed for an earlier time, when rules were created manually and environments were more static and at far lower scale. When network speeds were slow iptables' use of sequential processing of rules did not introduce significant latency or performance issues. The network has evolved as have demands placed on it. Network speeds increased, performance suffered, and the need to process rule sets grew from tens to hundreds and even thousands.

Using IP/port based mechanisms also has many other obvious disadvantages. As containers are deployed and torn down frequently individual IP addresses often have a very short lifetime. An IP address that was used by a container for just seconds is then reused by another container a couple of seconds later. This puts stress on systems which rely on using IP addresses for security filtering purposes as all nodes in a cluster must be constantly aware of the latest IP to container mapping. Maintaining these mappings is very challenging in an individual cluster and exponentially more difficult across multiple clusters.

It is also important to realize that no Kubernetes cluster is an island. If all communications happened within a single cluster things would be a lot simpler. Alas, many applications rely on services from different clusters, or even ones that remain physical (e.g. Databases). This introduces greater complexity. One approach commonly seen with early Kubernetes deployments is to rely on the applications themselves to manage their own networking and security requirements. Yet this can create a complex spaghetti soup of different implementations that can be a nightmare to manage and troubleshoot.

Multi cluster connectivity:

With standard Kubernetes networking, each cluster is independent, requiring proxies to connect workloads that run in different clusters for the purposes of migration, disaster-recovery, or geographic locality. However, not only do proxies add complexity (as another point in the infrastructure that may fail or come under heavy load), but they also present a security challenge since internal services must be "exposed" to the entire remote cluster due to the lack of shared identity between clusters.

Load balancing:

As the size of a Kubernetes cluster grows and the number of endpoints needing to be connected increases, many environments find that performance slows down to unacceptable levels. Service-based load-balancing is a core network function in Kubernetes, but using kube-proxy for load-balancing can lead to severe performance challenges as scale increases due to its reliance on iptables.

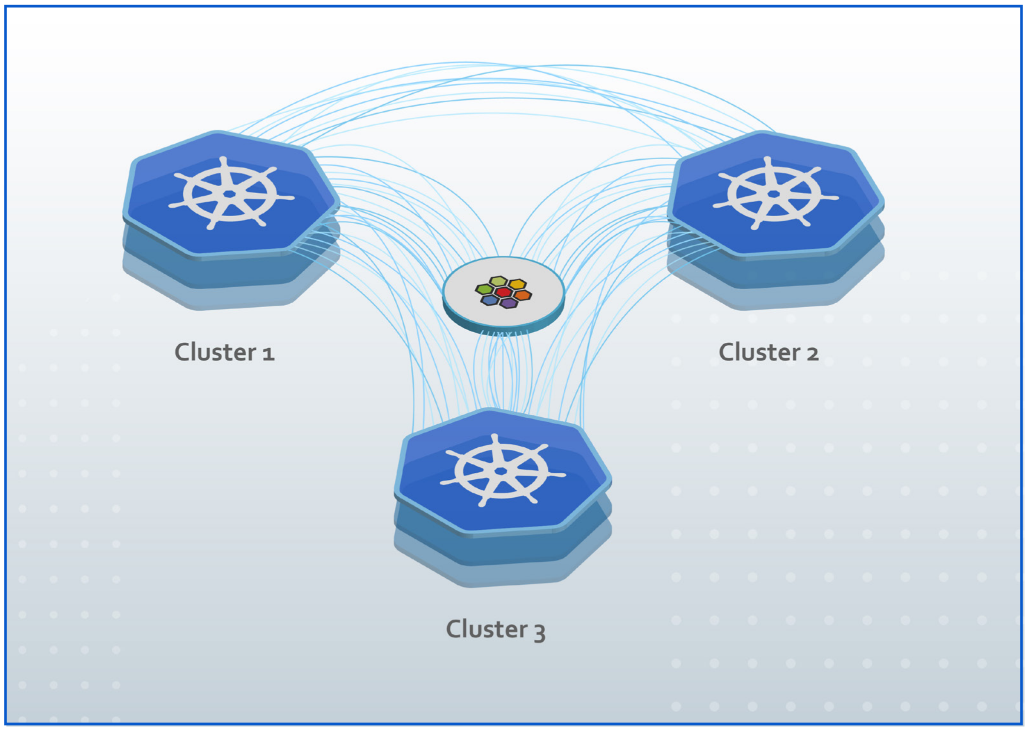
Why Cilium Enterprise

Networking:

Cilium is a **cloud native centric implementation of IPv4 and IPv6 Kubernetes networking & security**, and implements the Container Network Interface (CNI) standard to easily integrate with any existing Kubernetes cluster. Cilium's control and data plane has been **built from the ground up for large-scale and highly dynamic cloud native environments** where 100s and even 1000s of containers are created and destroyed within seconds. Cilium's control plane is highly optimized, running in Kubernetes clusters of up to 5K nodes and 100K pods.
Cilium uses bpfilter to address the performance, latency and processing issues of legacy iptables. It replaces the in-kernel implementation of iptables with high-performance network filtering, and guarantees a non-disruptive transition for Linux users. Cilium also supports both **overlay and direct routing models,** and has native integration with cloud provider networking for AWS, GCP, and Azure.

Multi Cluster Connectivity:

Cilium Cluster Mesh creates a single mesh of connectivity for load-balancing, observability and security between nodes across multiple clusters, enabling simple, high-performance cross-cluster connectivity.



Cluster Mesh also preserves **workload identity** for cross-cluster traffic, meaning that network visibility tooling, such as Hubble and network security policies, continue to work just as they do within a single cluster.

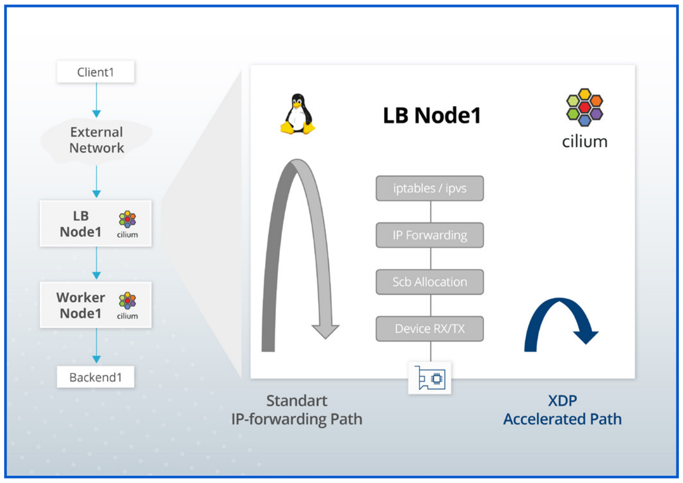
Finally, Cluster Mesh supports the important requirement to have workloads running in “edge” k8s clusters that are in a particular geographic location for latency or data sovereignty concerns. In both cases, such requirements are highly correlated to business critical workloads.

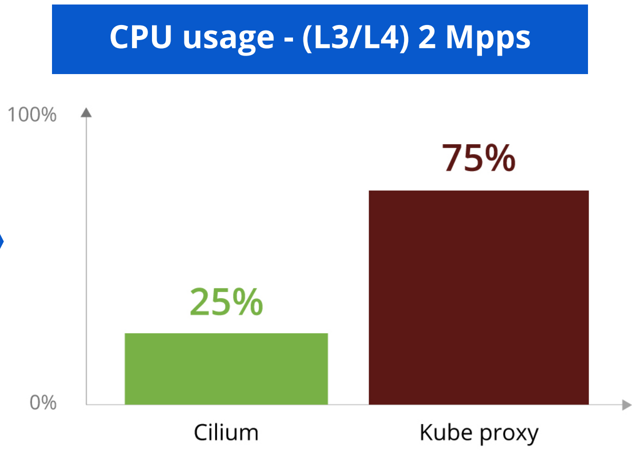
Load balancing:

To directly address the limitations of kube-proxy, Cilium acts as a 100% kube-proxy replacement to provide all service load-balancing in a Kubernetes cluster. If possible, Cilium will perform the load balancing on the system call level and translate the address directly in the connect() system call instead of relying on network address translation throughout the entire duration of a network connection.

Cilium’s data plane uses eBPF for high performance L3/4 load-balancing and incremental updates, avoiding the pitfalls of iptables-based CNI plugins. Specifically for pod-to-pod service load-balancing, Cilium performs this operation at the socket-layer, besting packet-based load-balancing solutions like iptables or IPVS.

For load-balancing of connections inbound to the cluster (e.g. type LoadBalancer/NodePort services and ExternalIPs), Cilium leverages XDP for NIC-hardware accelerated forwarding and supports Direct-Server-Return (DSR), both of which provide significant latency improvements and reduced server load. Cilium leverages the advanced Maglev algorithm for high-performance consistent hashing, at high scale.





Use of XDP and DSR significantly reduce latency and server load.

Source: https://cilium.io/blog/2020/06/22/cilium-18

Ensuring Identity Aware Observability for Platform and Application Teams

Description

As application teams architect and run highly distributed API-driven services in Kubernetes, visibility into network connectivity behavior is critical to running production-grade services. Quickly being able to root-cause common failures, especially whether they lie in the application layer or due to an issue in the underlying infrastructure, is critical to ensuring uptime and availability to customers. Without the right data, failures often cause immediate finger pointing exercises as teams have no easy way to determine if the cause of an issue lies in the network (DNS lookup failure, network policy drop, TCP layer connection failures, resets) vs. isolated in the applications layer. Unfortunately, Kubernetes, provides little visibility into the network behavior of the workloads it runs as pods.

Challenges with Legacy Solutions

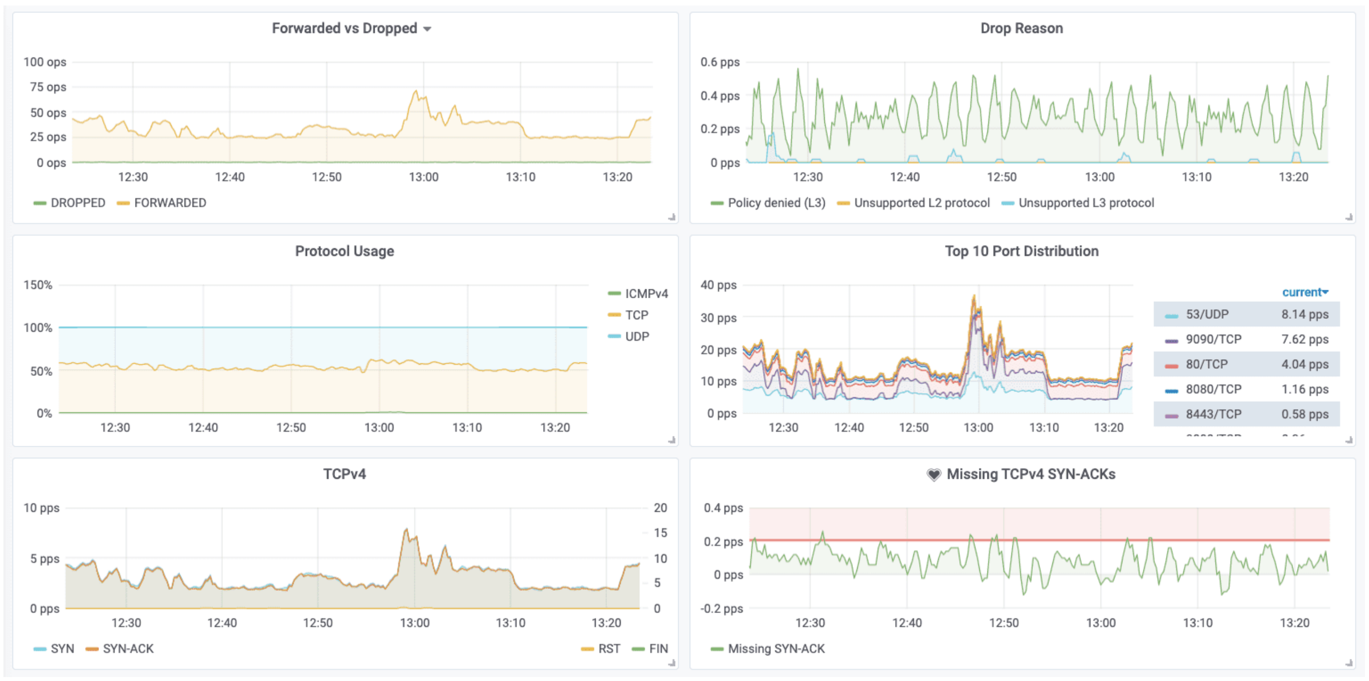
Traditional IP-based network monitoring tools are ineffective as ephemeral pod IPs do not identify the services that are impacted, and such traditional monitoring tools have no ability to restrict an application team's view to only the data relevant to their application. As a result, Kubernetes platform teams are often pulled in to troubleshoot problems.

Why Cilium Enterprise

Cilium dramatically speeds up the investigation of application and platform-layer issues. Cilium leverages eBPF to efficiently monitor the precise Linux process / command, container, and Kubernetes pod identity for each connection and makes this data available to all teams that need it to diagnose performance and availability challenges.

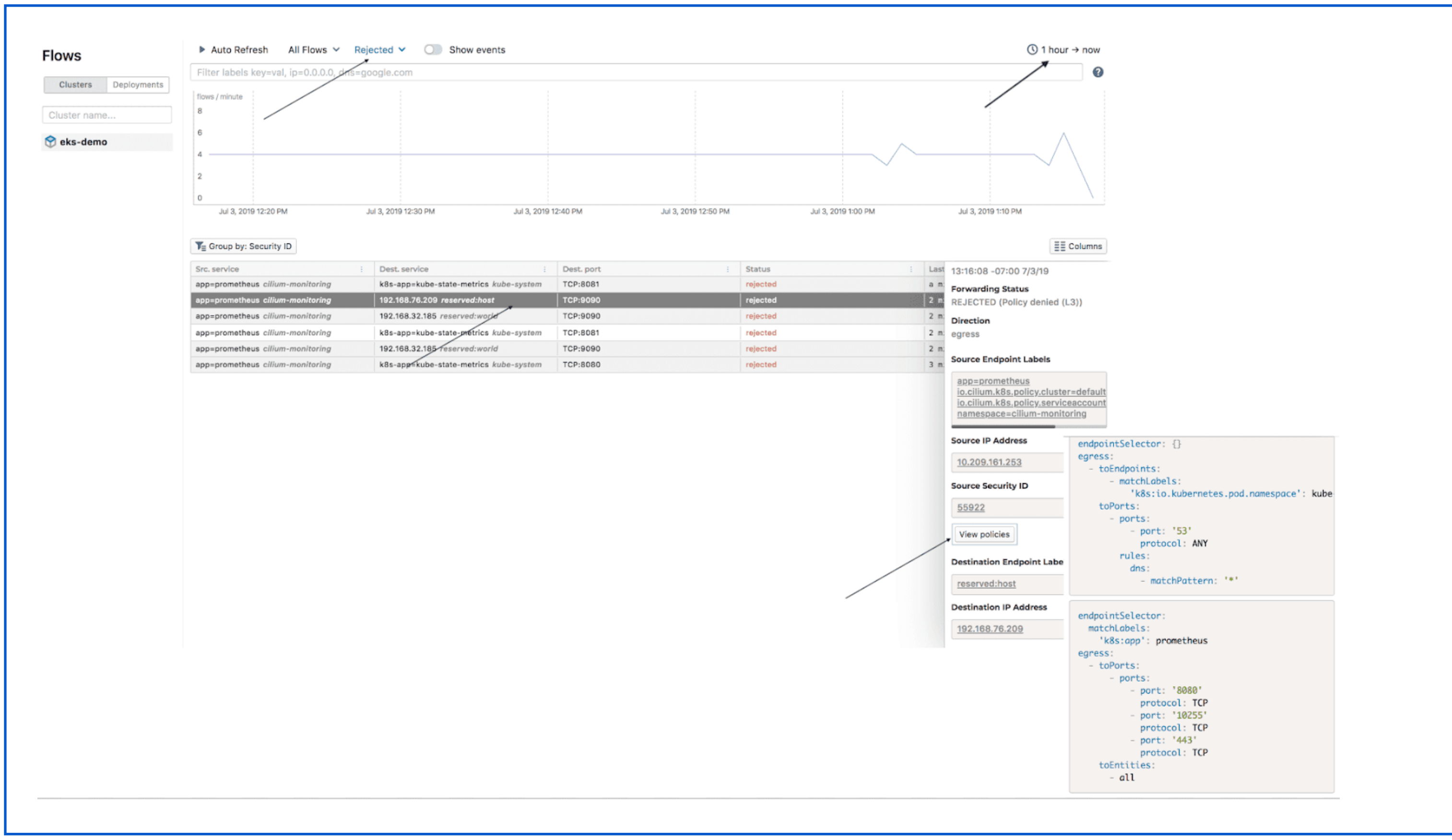
Multi-tenant Connectivity Data & Metrics

Cilium provides each application team self-service access to tools that provide rich streams of data about the health of connectivity between their services. This data is critical in solving the classic “finger-pointing” between application and infrastructure operations teams. Cilium also securely gives application tenants access only to the connectivity data associated with their Kubernetes namespaces.



Historical Data Views + Analytics

When troubleshooting a connectivity issue, having as much historical context as possible is critical to quickly resolving the incident. For example, comparing connectivity behavior to a baseline from before the incident, or identifying the exact timing of intermittent errors or faults that happened several hours ago. Cilium stores flow data and enables later querying and analytics on this data. Cilium also annotates flow data with additional metadata, such as the details about policies that were applied when a flow was allowed or denied, that further simplifies troubleshooting.



Src. service	Dest. service	Dest. port	Status	Last
app=prometheus *cilium-monitoring*	k8s-app=kube-state-metrics *kube-system*	TCP:8081	rejected	a m
app=prometheus *cilium-monitoring*	192.168.76.209 *reserved:host*	TCP:9090	rejected	2 m
app=prometheus *cilium-monitoring*	192.168.32.185 *reserved:world*	TCP:9090	rejected	2 m
app=prometheus *cilium-monitoring*	k8s-app=kube-state-metrics *kube-system*	TCP:8081	rejected	2 m
app=prometheus *cilium-monitoring*	192.168.32.185 *reserved:world*	TCP:9090	rejected	2 m
app=prometheus *cilium-monitoring*	k8s-app=kube-state-metrics *kube-system*	TCP:8080	rejected	3 m

13:16:08 -07:00 7/3/19

Forwarding Status
REJECTED (Policy denied (L3))

Direction
egress

Source Endpoint Labels
app=prometheus
io.cilium.k8s.policy.cluster=default
io.cilium.k8s.policy.serviceaccount
namespace=cilium-monitoring

Source IP Address
10.209.161.253

Source Security ID
55922

Destination Endpoint Labe
reserved:host

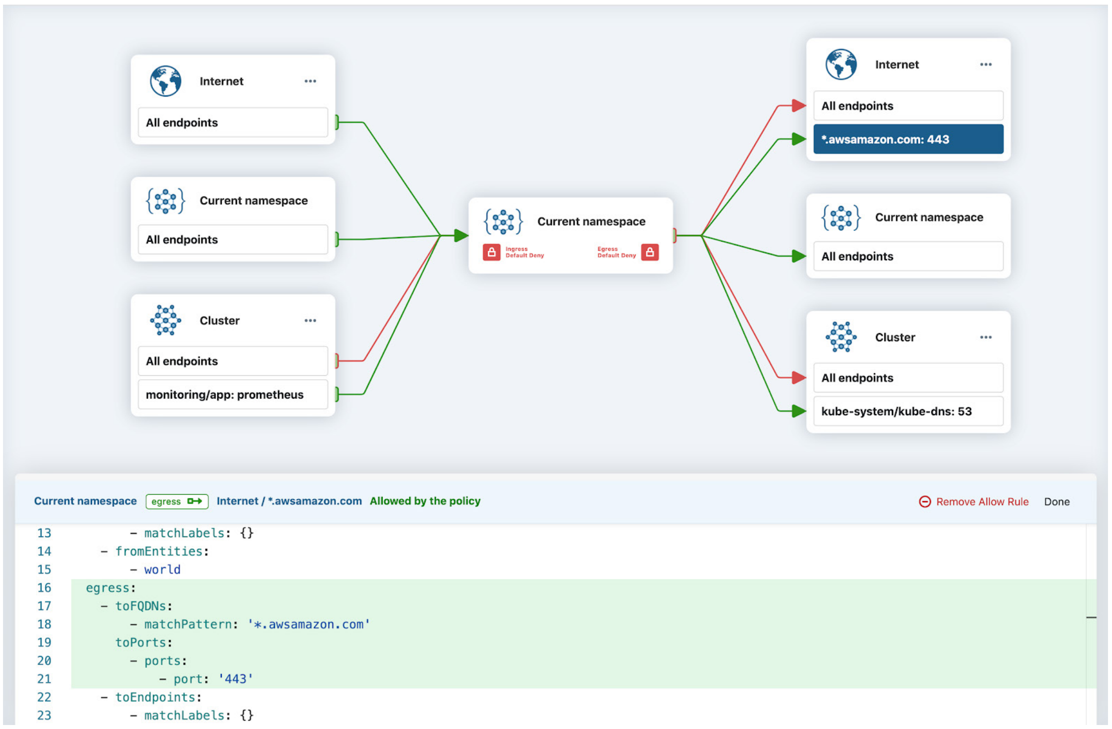
Destination IP Address
192.168.76.209

```
endpointSelector: {}
egress:
  - toEndpoints:
      - matchLabels:
          'k8s:io.kubernetes.pod.namespace': kube
    toPorts:
      - ports:
          - port: '53'
            protocol: ANY
        rules:
          dns:
            - matchPattern: '*'
```

```
endpointSelector:
  matchLabels:
    'k8s:app': prometheus
egress:
  - toPorts:
      - ports:
          - port: '8080'
            protocol: TCP
          - port: '10255'
            protocol: TCP
          - port: '443'
            protocol: TCP
    toEntities:
      - all
```

Simplified Network Policy Creation

One major cause of failures lies with the implementation of network policies. With a “default deny” model, a missed dependency leads to a broken application. Cilium provides tooling to simplify and automate the creation of Network Policy based on labels and DNS-aware data. APIs enable integration into CI/CD workflows while visualizations help teams understand the expected behavior of a given policy. Collectively, these capabilities dramatically reduce the barrier to entry to creating Network Policies and the ongoing overhead of maintaining them as applications evolve.



```
        - matchLabels: {}
    - fromEntities:
        - world
  egress:
    - toFQDNs:
        - matchPattern: '*.awsamazon.com'
      toPorts:
        - ports:
            - port: '443'
    - toEndpoints:
        - matchLabels: {}
```

About Isovalent

Isovalent builds software to connect, observe and secure cloud native workloads via it’s Cilium open source project and Cilium Enterprise product.

Cilium Open Source

Cilium Open Source provides eBPF-based networking, observability, and security with optimal scale and performance for platform teams operating Kubernetes environments across cloud and on-prem infrastructure.

Cilium Enterprise

Cilium Enterprise addresses the complex workflows related to security automation, forensics, compliance, role-based access control, and integration with legacy infrastructure that arise as platform teams engage with application and security teams within an enterprise organization.

US HEADQUARTERS

444 Castro St. STE 730
Mountain View, CA 94041 USA

SWISS HEADQUARTERS

Industriestrasse 25 8604
Volketswil Zurich, Switzerland

ISOVALENT

cilium