ISOVALENT

Zero Trust Networking in Kubernetes

Using Cilium, Hubble & Grafana

Raphaël Pinson | @raphink | @raphink@mastodon.social

Solutions Architect, Isovalent

Key Questions when Enterprises Adopting Network Policy

- Where to start?
- How to troubleshoot Network Policies?
- How to stay up-to-date with Network Policies while applications evolve?
- How to prevent application teams from simple allowing everything?
- How can security teams prove that default deny policies are enforced?
- How can security teams be alerted on denied connections?

There is no “Easy Road” for Adopting Network Policy

- **Default Deny**
  - Each service-to-service communication explicitly allowed
  - But high chance of misconfiguration ⇒ application **unavailability** and **high adoption friction**

- Better Approach: Focus on **Risk Reduction**
  - Define **metrics** for Risk Exposure
  - Focus on the **most security sensitive** namespaces first
  - Leverage **network observability** to identify which network policy patterns easily reduce risk with minimal friction.

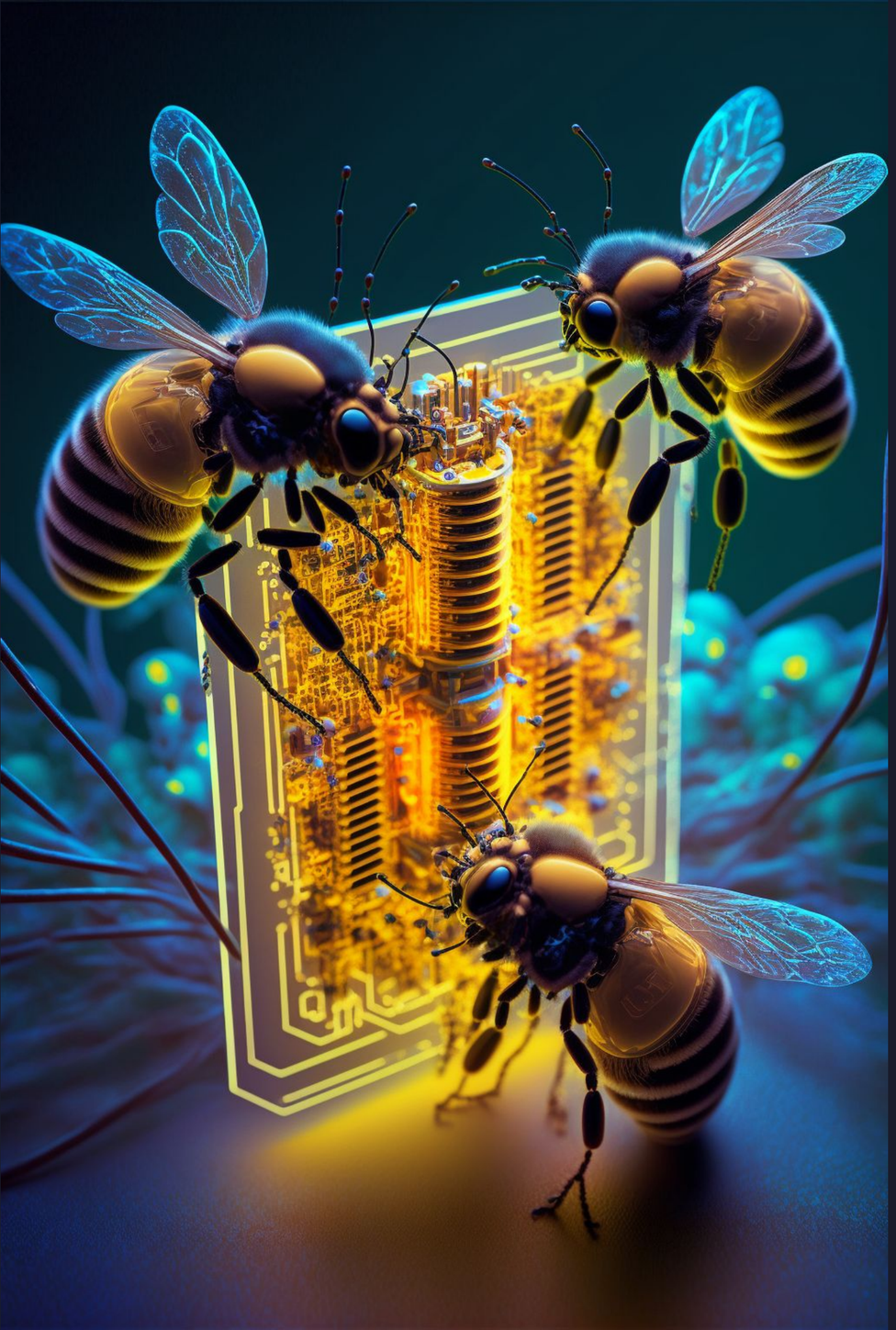
ISOVALENT

Who am I

Raphaël Pinson

Solutions Architect @ Isovalent

CNCF Ambassador



Zero Trust Networking

Using Cilium, Hubble & Grafana

- Cilium & eBPF
- Network Policies Strategies
- Network Policies Observability

What this talk is NOT about: Network Policy 101

ISOVALENT

Labs

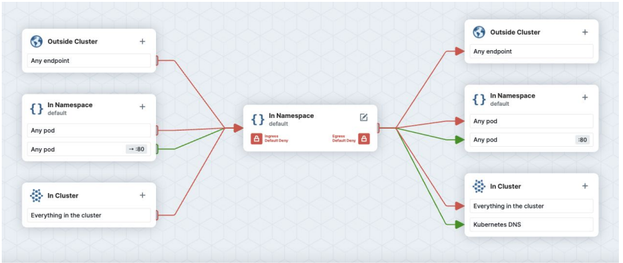
isovalent.com/labs

cilium

Tutorials

Getting started guides and examples.

cilium.io
networkpolicy.io



Tools

Learn how to create
Network Policies for Kubernetes

editor.cilium.io

This session IS about strategies to overcome the operational & organizational challenges of adopting Network Policy in complex an enterprise environments.

Zero Trust Networking

Using Cilium, Hubble & Grafana

- **Cilium & eBPF**
- Network Policies Strategies
- Network Policies Observability

- Open Source Projects

ISOVALENT

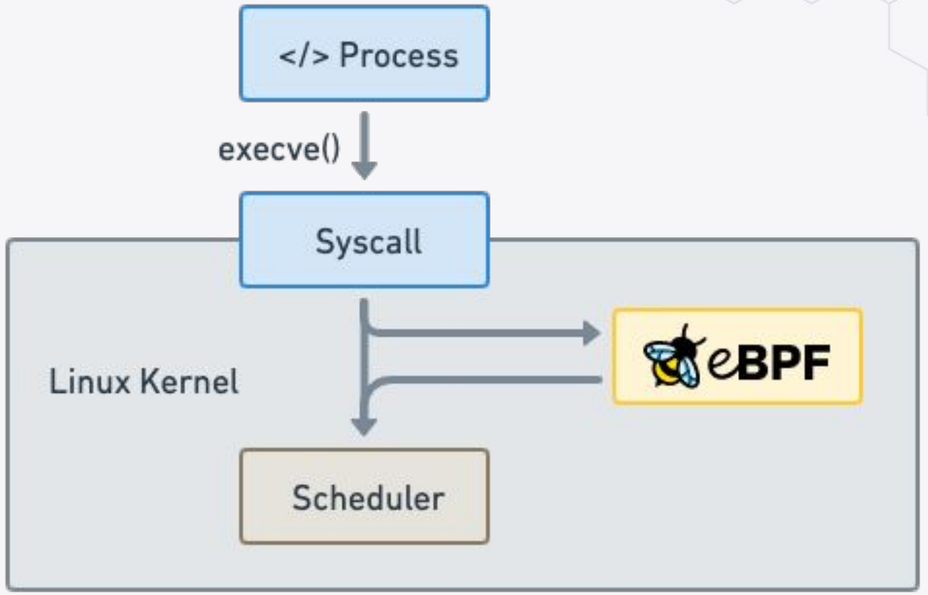
- Company behind Cilium
- Provides Cilium Enterprise


eBPF

eBPF

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

“What JavaScript is to the browser, eBPF is to the Linux Kernel”

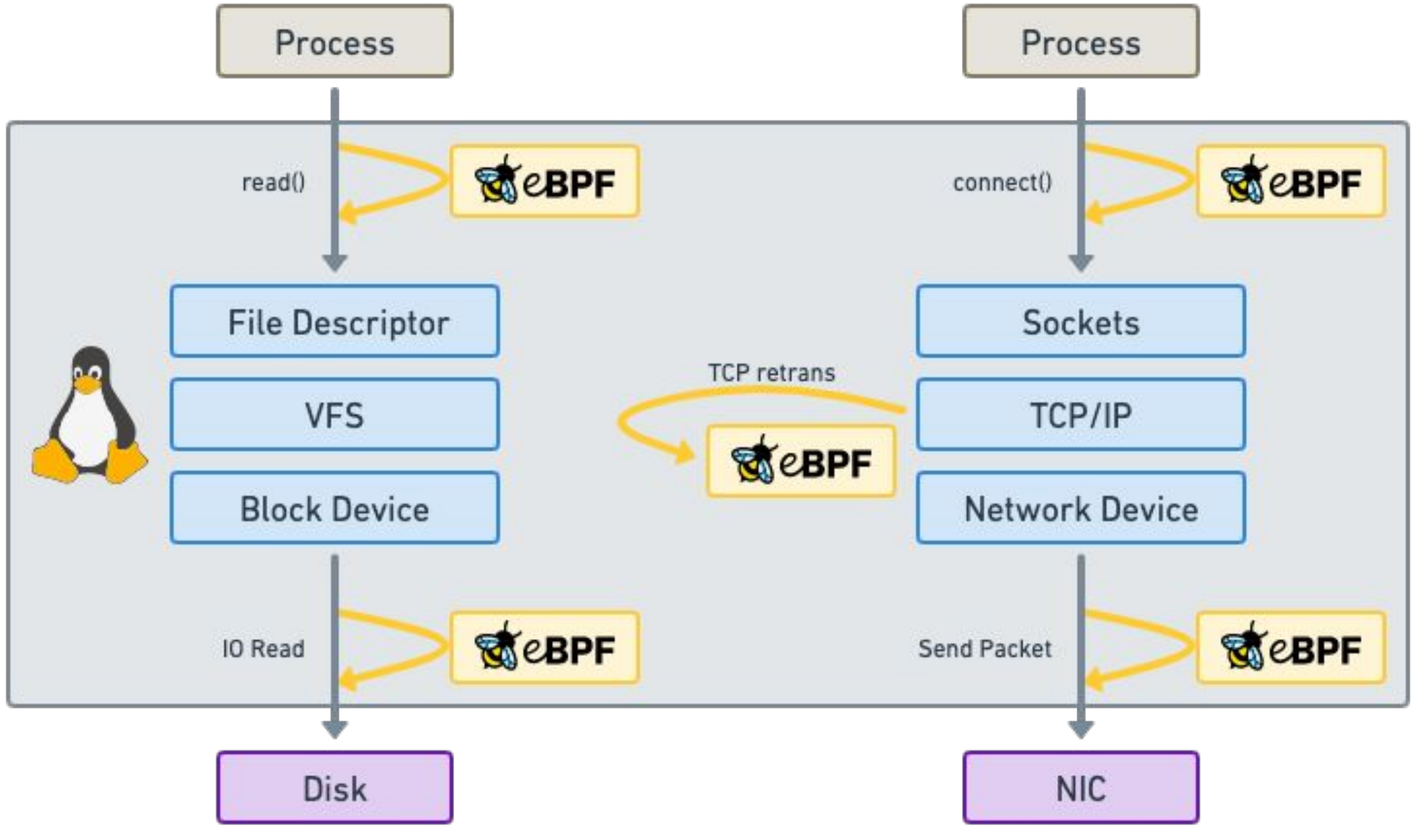



```c
int syscall__ret_execve(struct pt_regs *ctx)
{
        struct comm_event event = {
                .pid = bpf_get_current_pid_tgid() >> 32,
                .type = TYPE_RETURN,
        };

        bpf_get_current_comm(&event.comm, sizeof(event.comm));
        comm_events.perf_submit(ctx, &event, sizeof(event));

        return 0;
}
```

Run eBPF programs on events



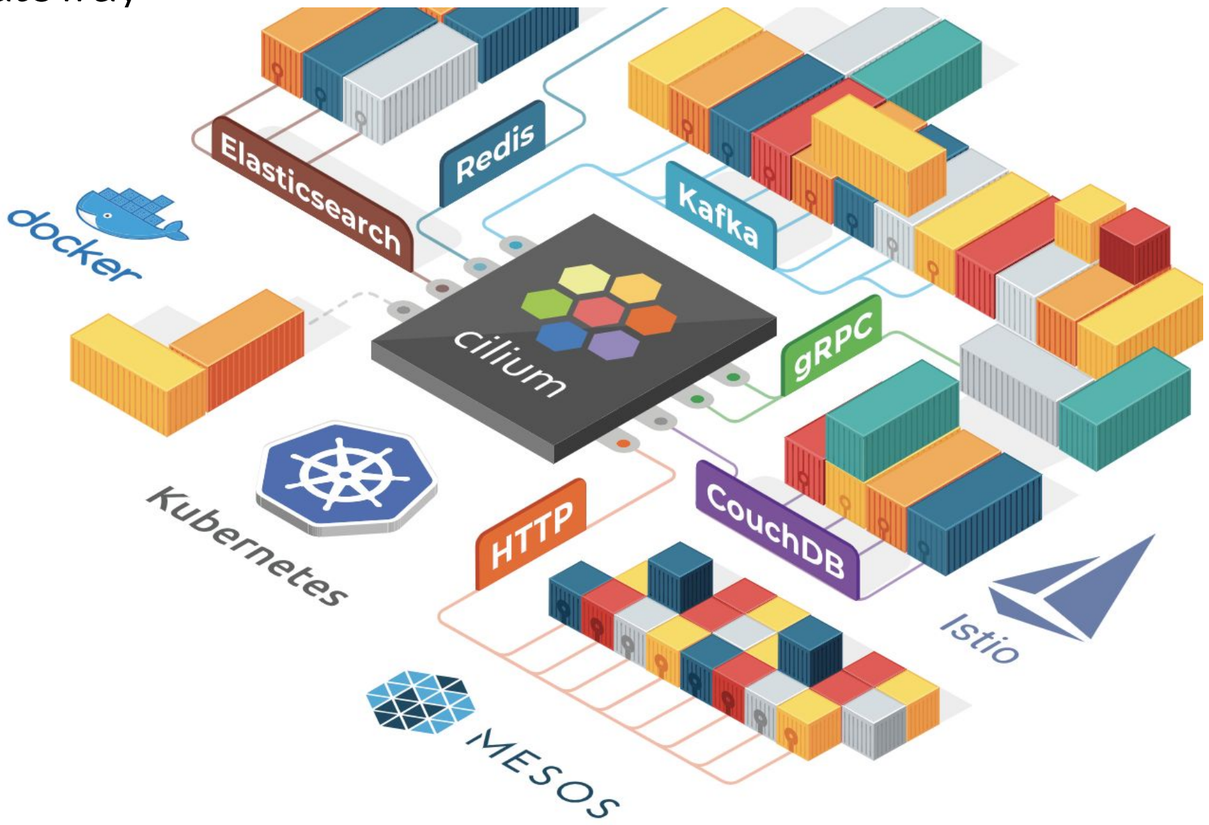
Attachment points

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

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 Service Mesh

Service Mesh

Ingress | Authentication | Traffic Management | spiffe | Gateway API

cilium hubble

Observability

Metrics | Tracing | Service Map | Logs | SIEM | fluentd | Grafana | OpenTelemetry

cilium CNI

Networking

Network Policy: DNS | L3/L4 | L7

Encryption: IPsec | Wireguard

Load-Balancing: K8s | Maglev | DSR

Networking: Multi-Cluster | NAT46 | IPv4 | IPv6 | Cloud SDN | BGP | Overlay | SRv6 | Egress Gateway

Runtime Security

Tetragon

SIEM | fluentd | Grafana

Observability

Enforcement

Kubernetes | Container | VM | Metal

aws | Google Cloud | Azure | Alibaba Cloud | RED HAT OPENSHIFT | vmware

cilium

Created by ISOVALENT

eBPF-based:

- Networking
- Security
- Observability
- Service Mesh & Ingress

Foundation

CLOUD NATIVE COMPUTING FOUNDATION

Technology

eBPF envoy

Building a Global Multi Cluster Gaming Infrastructure with Cilium

---

What Makes a Good Multi-tenant Kubernetes Solution

Building a Secure and Maintainable PaaS

Building High-Performance Cloud-Native Pod Networks

Scaling a Multi-Tenant k8s Cluster in a Telco

First step towards cloud native networking

Cloud Native Networking with eBPF

Managed Kubernetes: 1.5 Years of Cilium Usage at DigitalOcean

---

Google chooses Cilium for Google Kubernetes Engine (GKE) networking

---

Why eBPF is changing the Telco networking space?

Kubernetes Network Policies in Action with Cilium

AWS picks Cilium for Networking & Security on EKS Anywhere

Scaleway uses Cilium as the default CNI for Kubernetes Kapsule

---

Sportradar is using Cilium as their main CNI plugin in AWS (using kops)

Utmost is using Cilium in all tiers of its Kubernetes ecosystem to implement zero trust

---

Yahoo is using Cilium for L4 North-South Load Balancing for Kubernetes Services

ISOVALENT

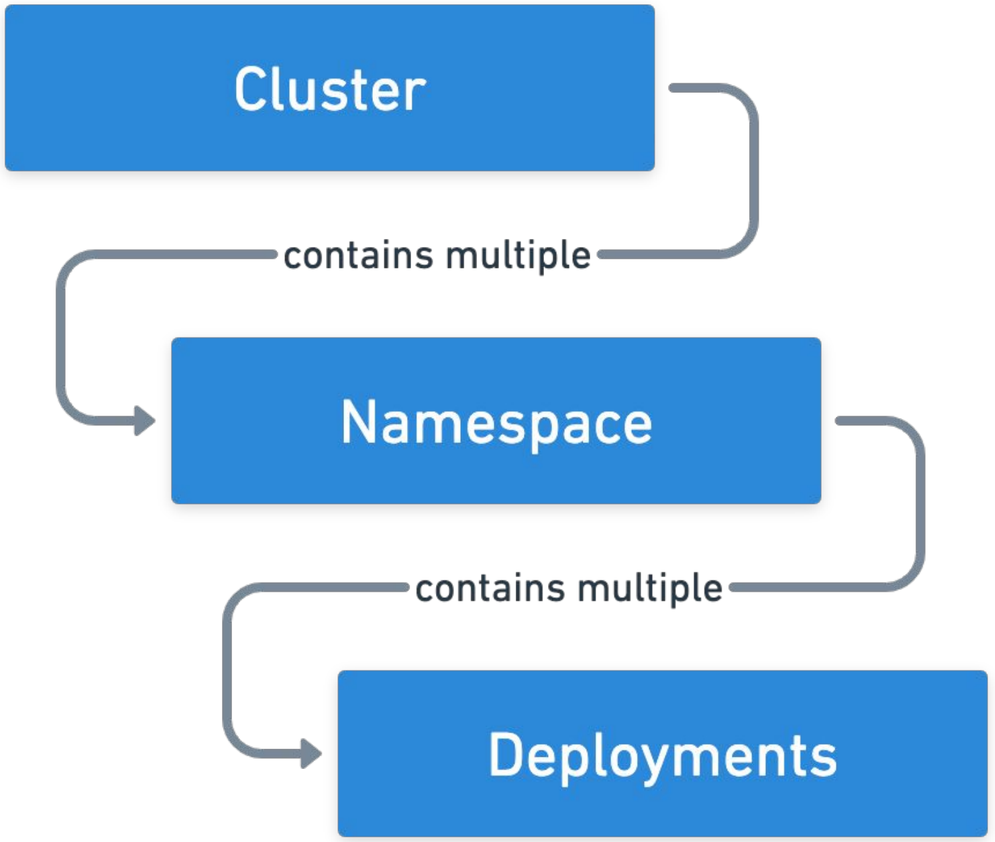
Zero Trust Networking

Using Cilium, Hubble & Grafana

- Cilium & eBPF
- **Network Policies Strategies**
- Network Policies Observability

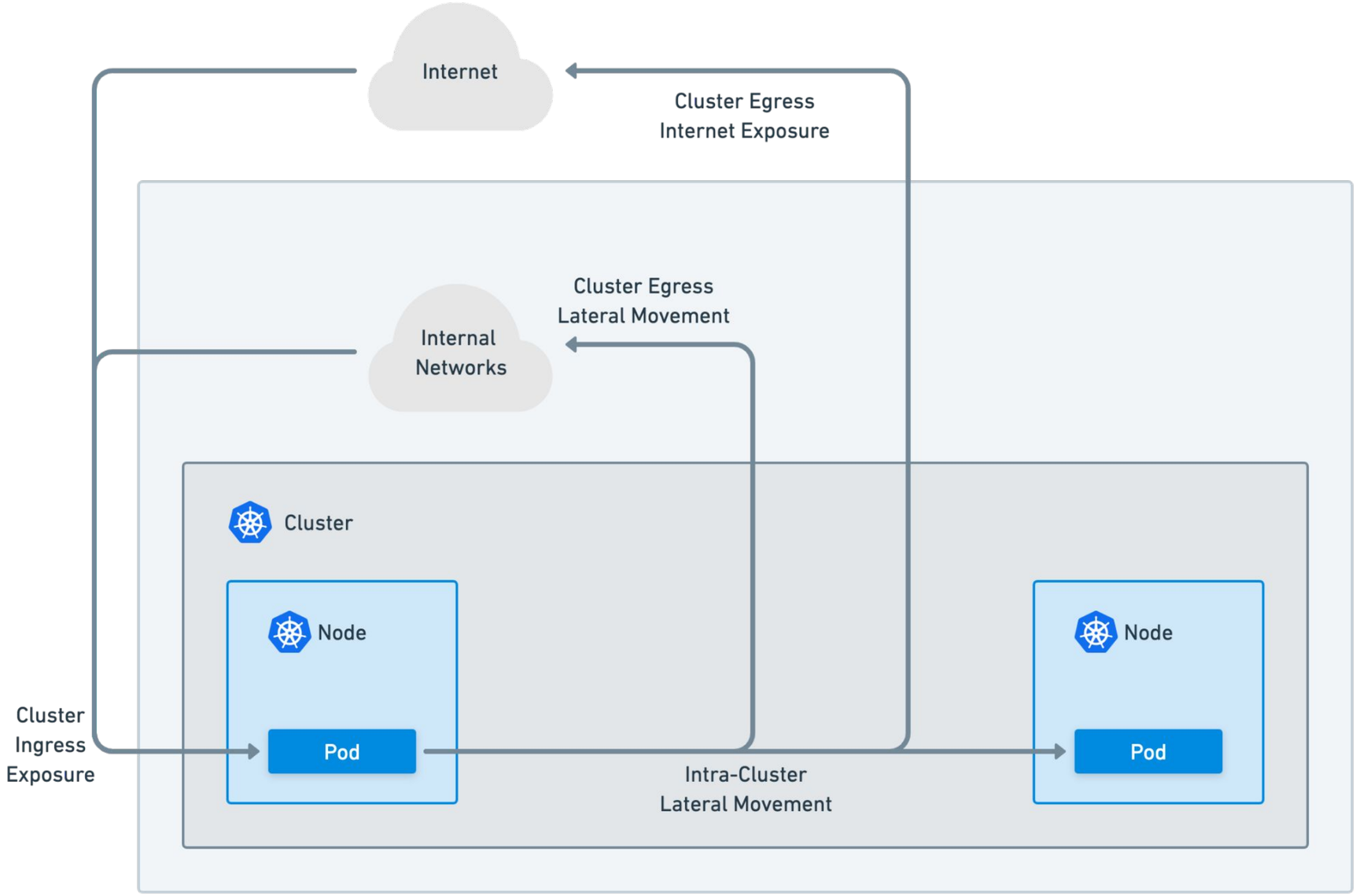
Application Teams & Multi-tenancy

- A Kubernetes cluster is often used by multiple teams, each managing one or more services.
- Each team has one or more namespaces to run their applications.
- Applying network policy at namespace level is a common first step in enabling multi-tenancy.

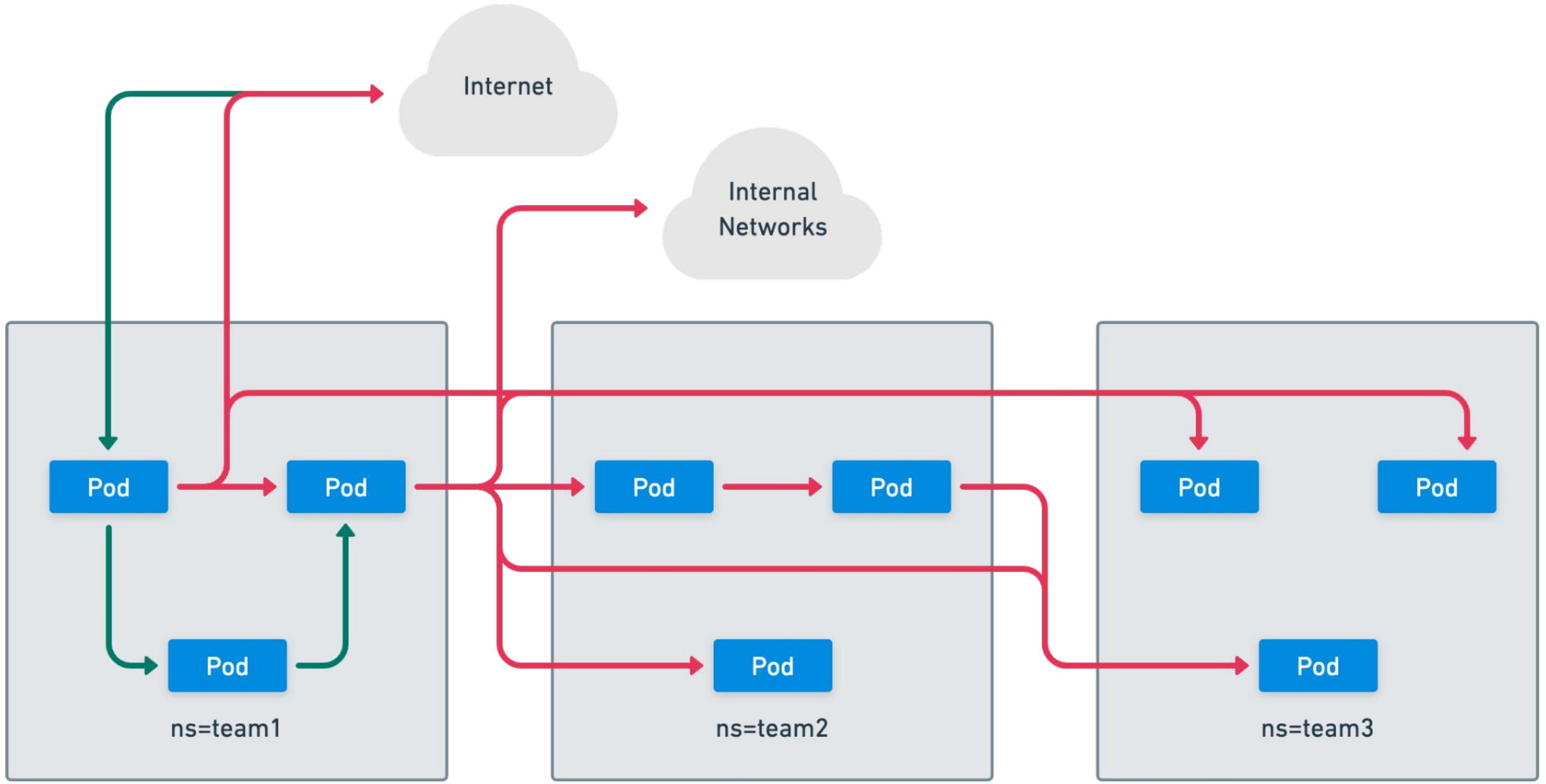


Note: Not always 1-to-1 team-to-namespace. Team may be a namespace-label, or pod label within a larger namespace, but same concept applies.

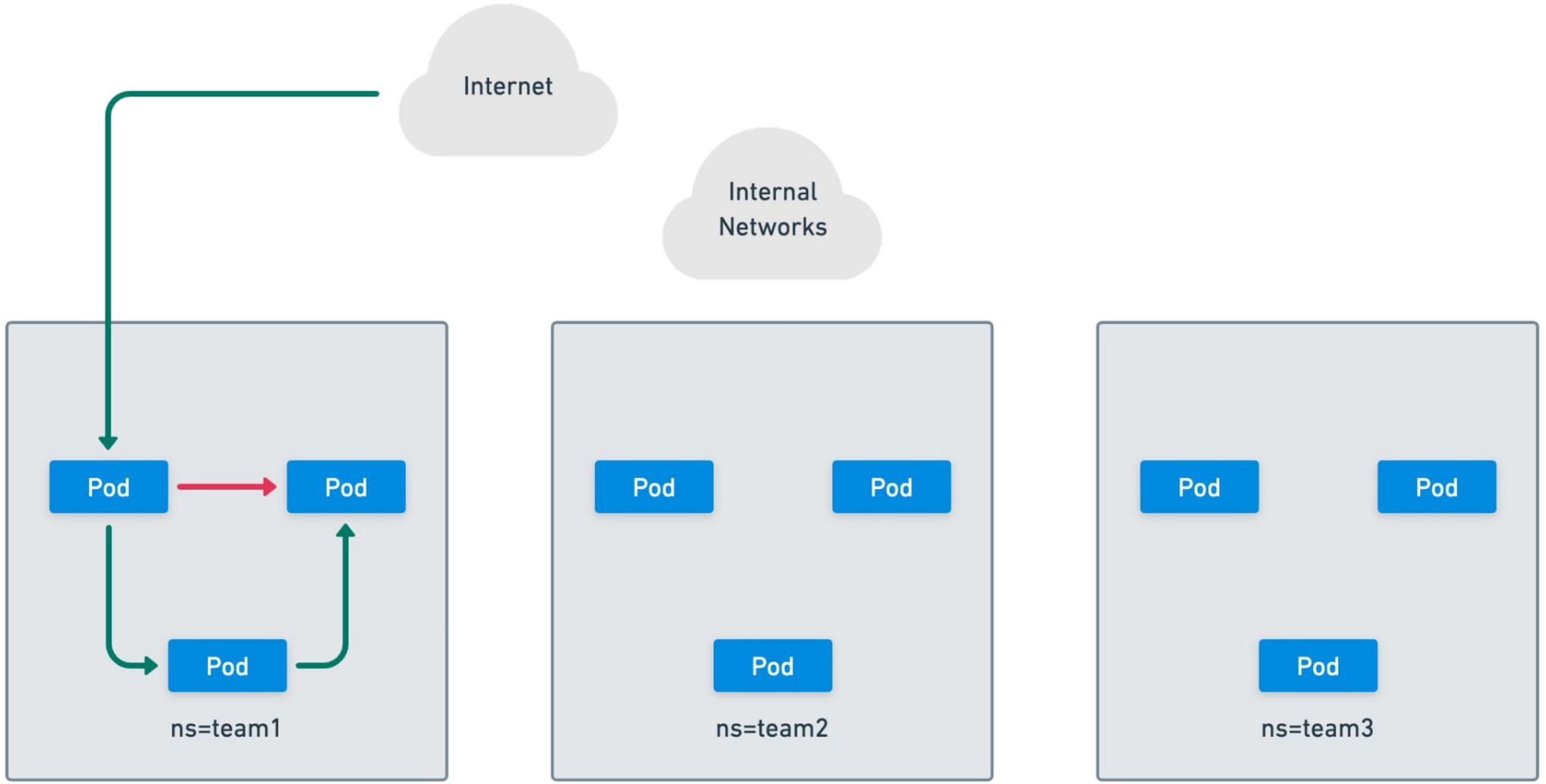
Key Types of Kubernetes Network Risk Exposure



Risk Reduction Example - Wide blast radius



Risk Reduction Example - Limited blast radius



Measuring Risk: Prioritization

Concrete Measurements to guide prioritization

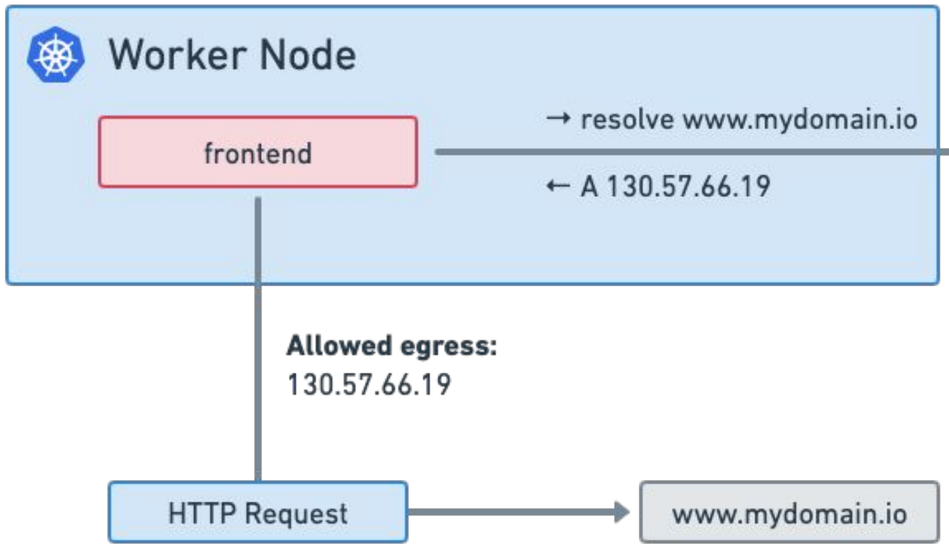
- # of services reachable via Cluster Ingress with no matching traffic.
- # of services reachable from other namespaces with no matching traffic.
- # of services with access to External Networks or Internet with no matching connections.

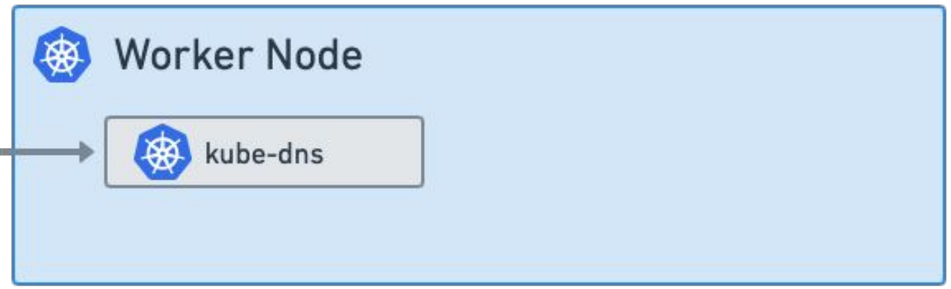
The Cost of “Overfitting”

- Strong network observability tools enables you to translate each observed flow into a rule.
- However, simplified approaches to translating traffic flow to rule tend to be cumbersome and brittle.
- Why?:
  - IPs for services outside the cluster can change or are opaque.
  - App dependencies or shared services: kube-dns, external logging, vaults servers are required by all apps. Creating pair-wise rules leads to friction and bloat.
  - Communication within a single app/team. Complex and frequent change. Leads to increased number of rules which change frequently.

Best Practices - DNS Rules

DNS-aware Cilium Network Policies for External Services which IPs frequently change.






```yaml
apiVersion: "cilium.io/v2"
kind: CiliumNetworkPolicy
[...]
specs:
- endpointSelector:
  matchLabels:
    app: frontend
  egress:
  - toFQDNs:
    - matchName: "*.mydomain.io"
    toPorts:
    - ports:
      - port: "443"
        protocol: TCP
```

Initial “Coarse-Grained” Per-Namespace Strategy

Pattern to avoid “overfitting”

1. **Allow all** ingress/egress communication within a namespace

2. Use **Hubble Observability** data to identify and permit:
    a. Which (service + port) within the namespace are **“public services”**:
        - Allow from cluster-only.
        - Allow from “world” (type LB/NP) or ingress namespace.
    b. **Egress** Access to:
        - Other services in the cluster (optional, limit ports).
        - Other services in the external private network (optional, limit ports).
        - Other services on the Internet (optional, limit ports).

3. **Coarse-grained** nature of policies ⇒ policies need only change **rarely**

Key Pattern: Baseline vs. Per-Namespace Policies

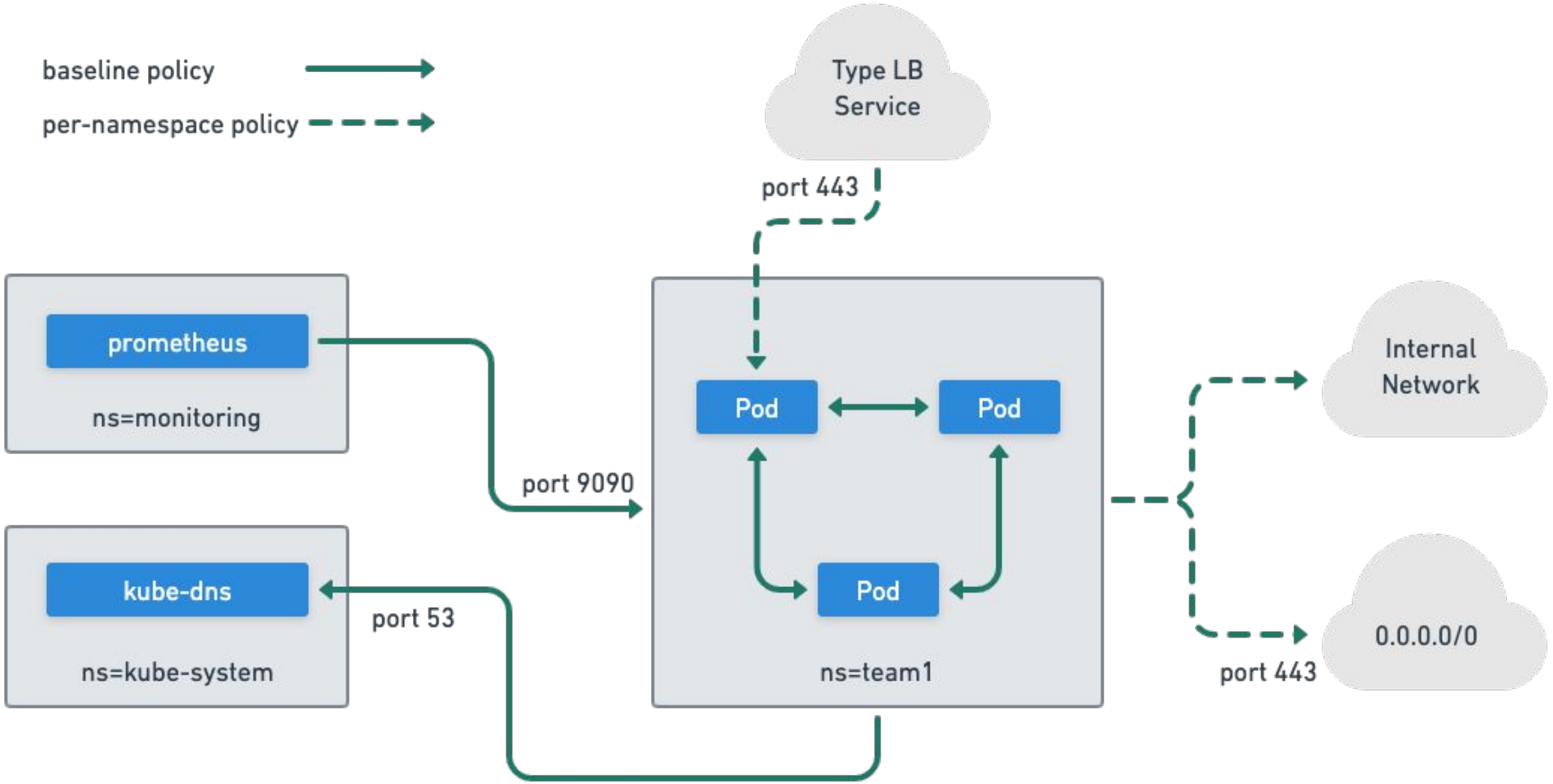
Global Baseline Policies

- Default deny ingress/egress.
- Allow all ingress/egress within namespace.
- Egress to “public services”
  - cluster-wide shared services (e.g. kube-dns, prometheus).
  - external shared CIDR/DNS (logging, monitoring, vault, etc.)
- Often implemented by **CiliumClusterWideNetworkPolicies**

Per-Namespace Policies

- Per-service ingress (limited by port):
  - Exposed within cluster via ClusterIP service.
  - Exposed externally via ingress or Load-Balancer/NodePort.
- Namespace-specific egress:
  - No access.
  - Egress to specific CIDR/DNS + port
  - Unrestricted access on specific ports (fallback).

Example Baseline + Coarse-grained Per Namespace Policies



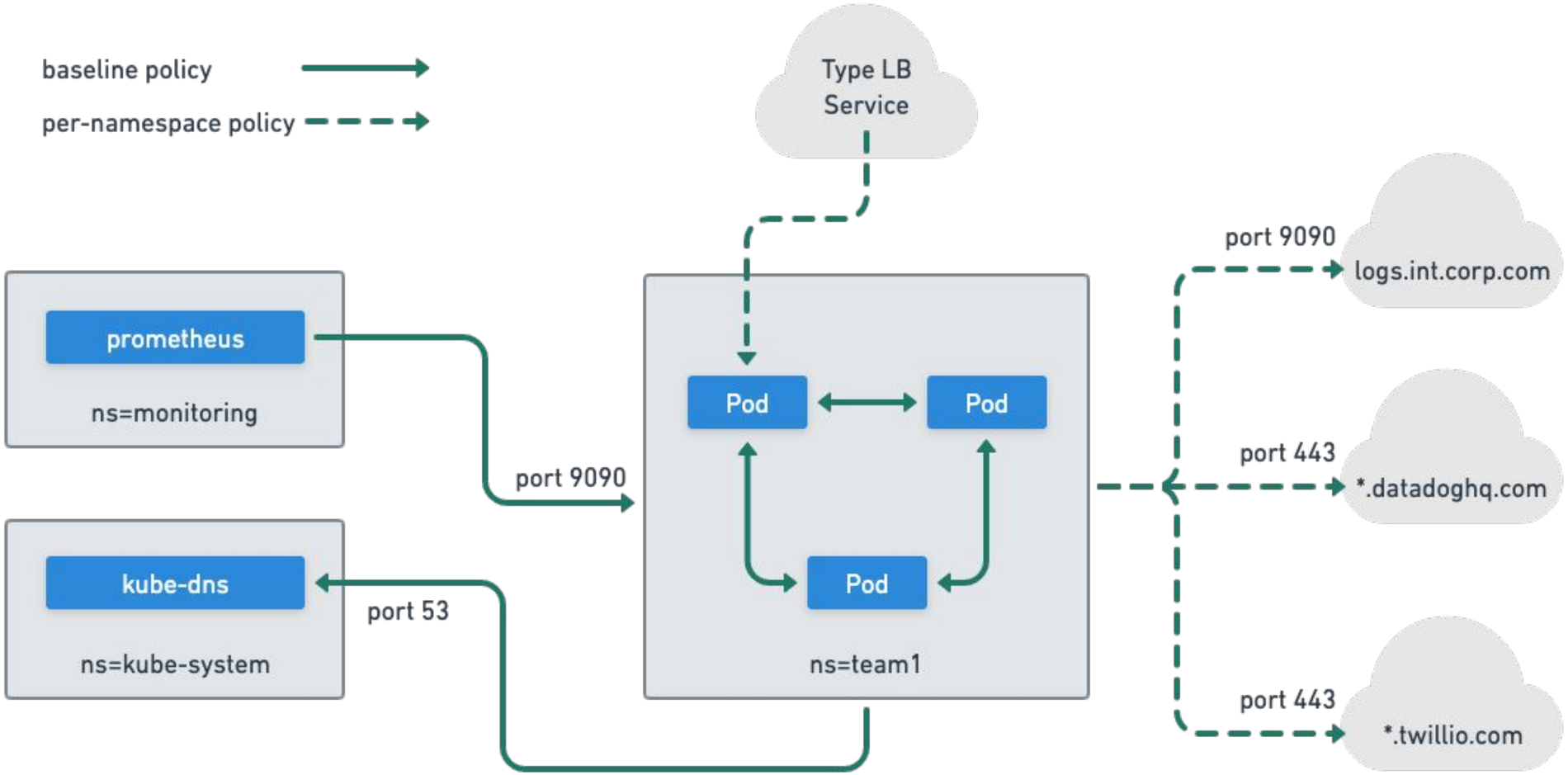
Transitioning from “Coarse” to “Fine-Grained” Policies

- **Prioritize** namespaces based on:
  - Most **security-sensitive** applications
  - Measurement of exposure vs. used connectivity

- **Transition for egress** outside the cluster:
  - Access to all private network → Access to specific FQDNs or smaller CIDRs (with port)
  - Access to Internet → Access to specific FQDNs or small CIDRs (with port)

- **Transitions within** the cluster:
  - Shift rules that allow all to/from cluster to allowing to/from specific namespaces or services.

Example Baseline + Fine-grained Per Namespace Policies

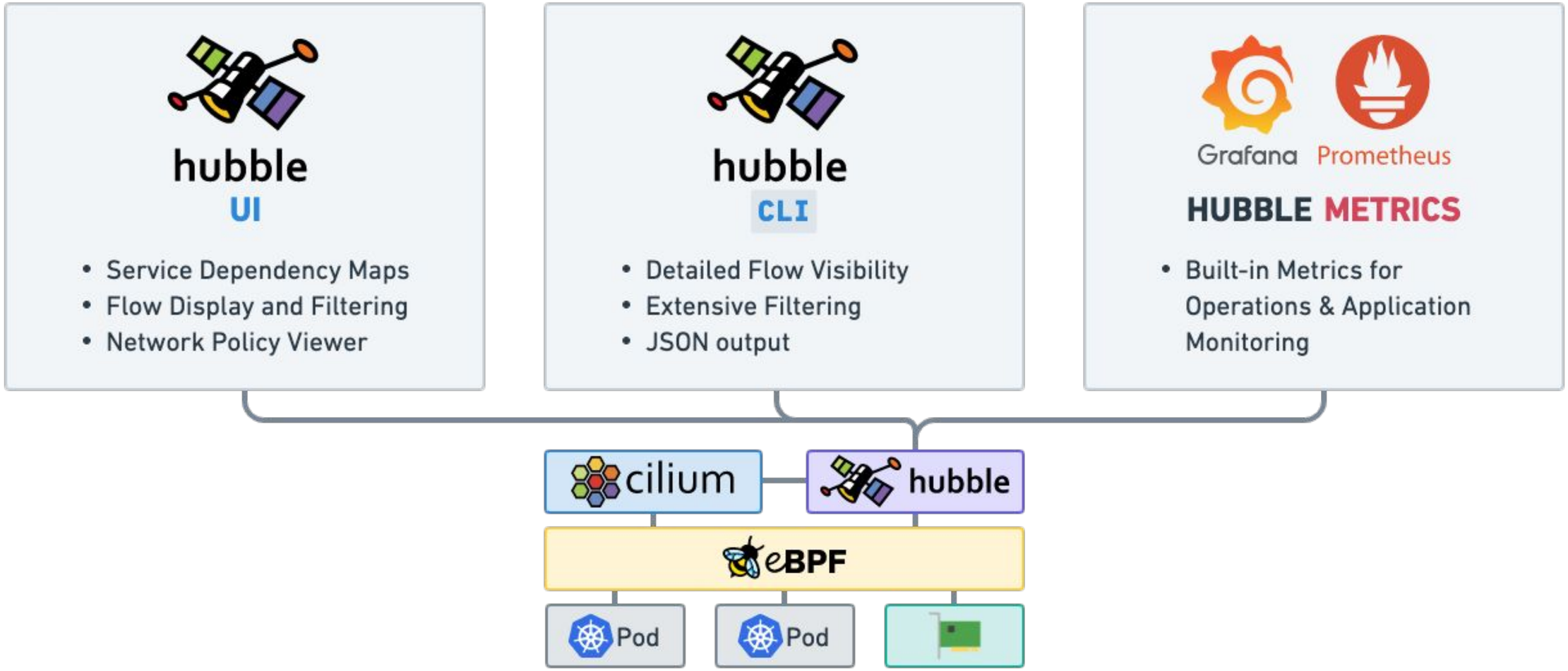


Zero Trust Networking

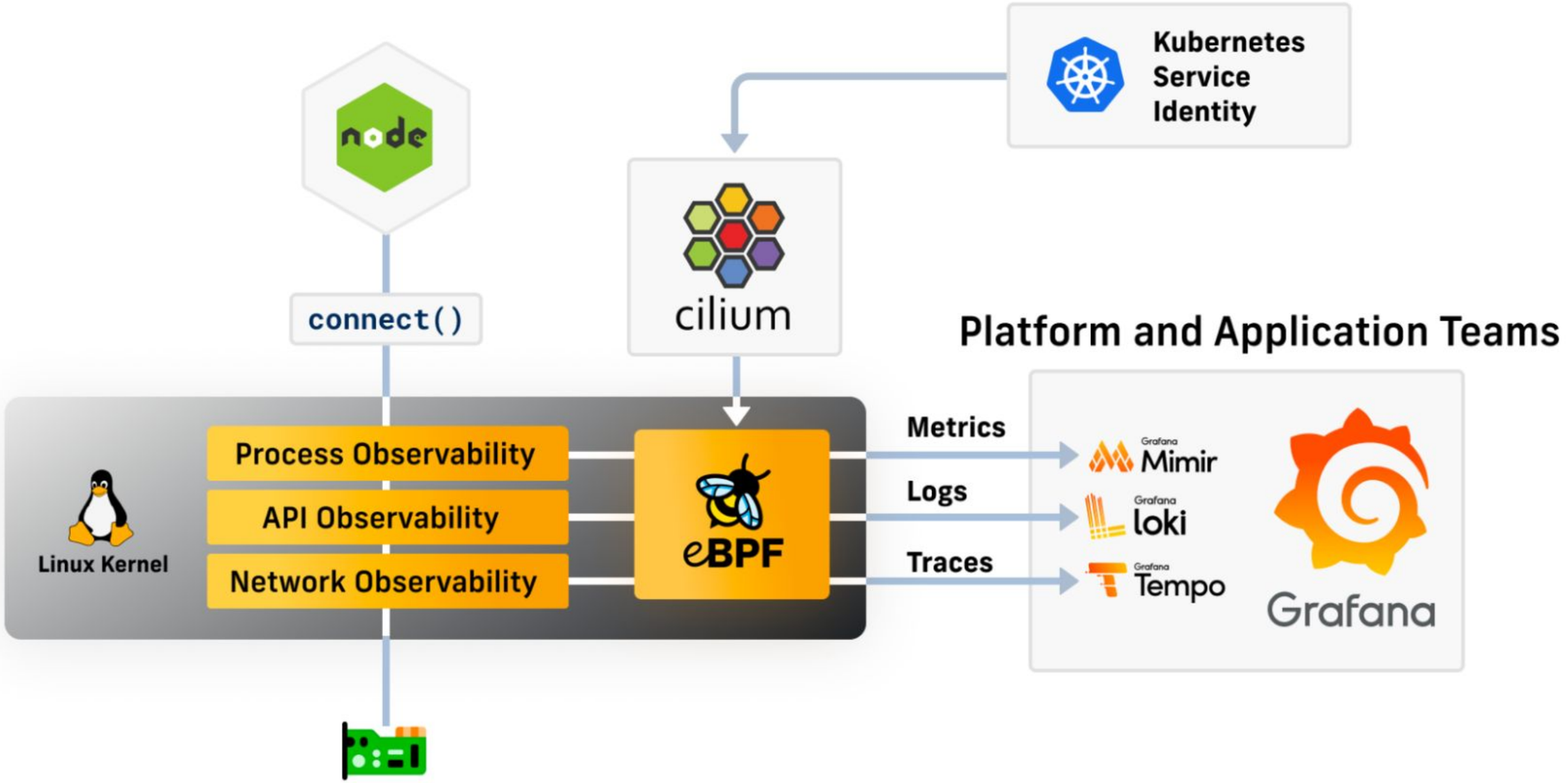
Using Cilium, Hubble & Grafana

- Cilium & eBPF
- Network Policies Strategies
- **Network Policies Observability**

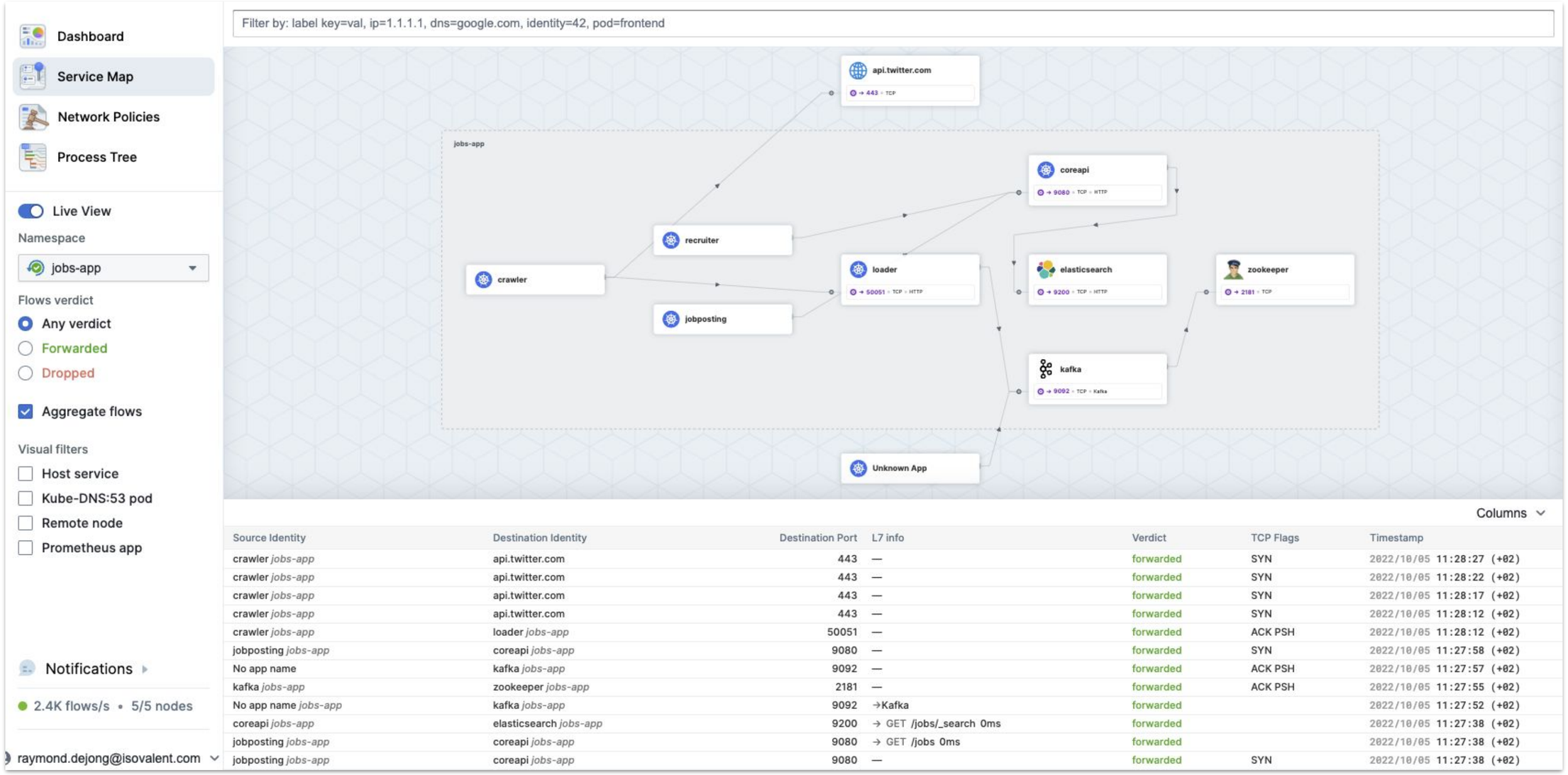
Hubble Overview



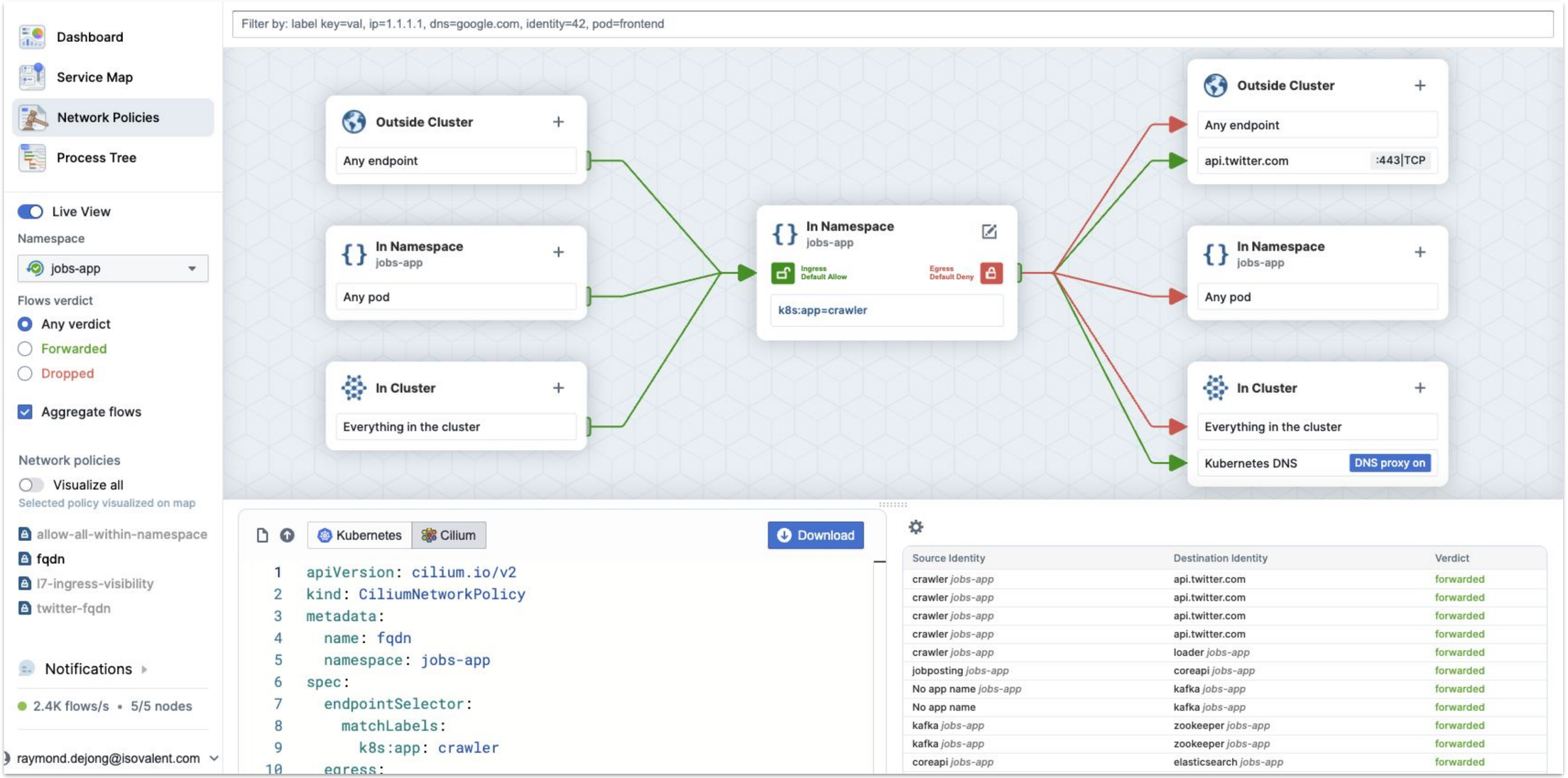
Cilium + Hubble ❤ Grafana



Using Hubble Observability Data to Build Policies

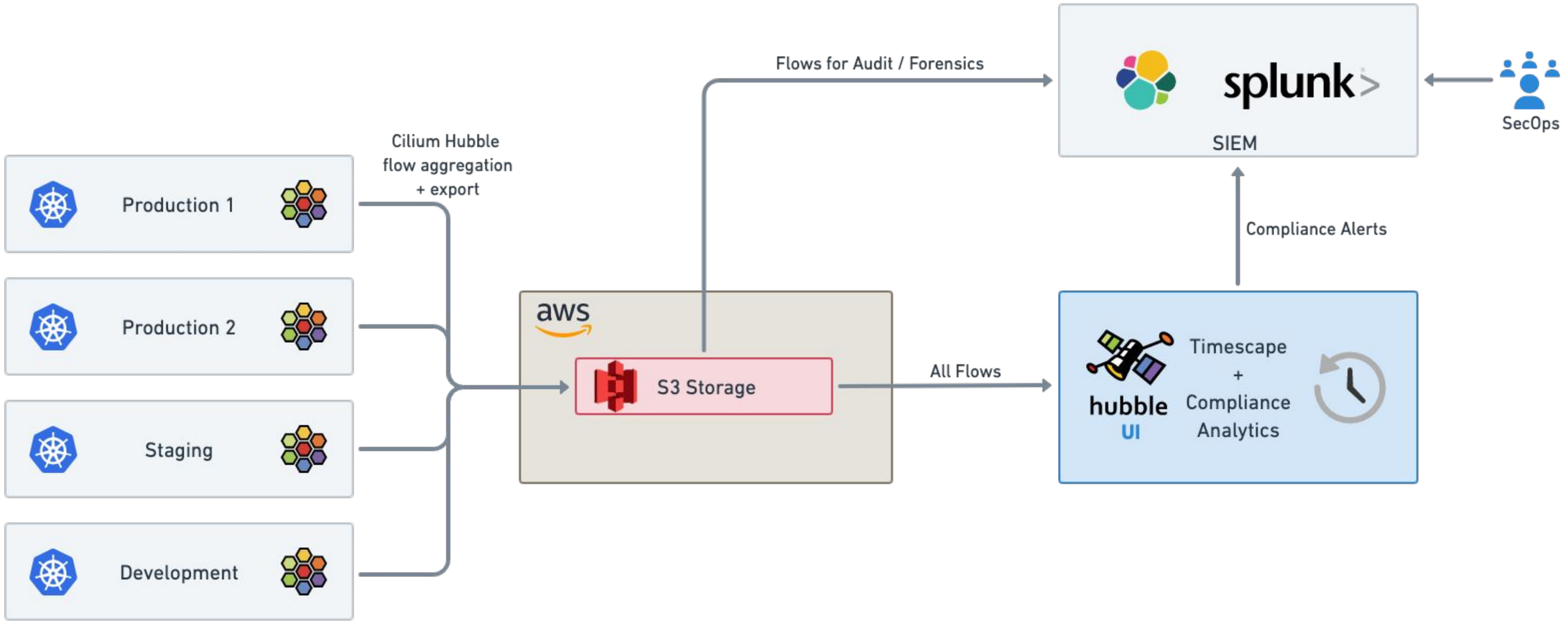


Hubble Network Policy Editor



Cilium Policy Verdicts in Grafana

Using Hubble Observability Data for Security Compliance



Observability-Driven Zero Trust Approach

Day 1:

- identify commonly shared services to include in baseline policy.
- identify “low-hanging fruit” namespaces easily locked down at ingress/egress (no per-namespace exceptions required).
- identify the set of “exceptions” required for a given per-namespace policy

Day 2:

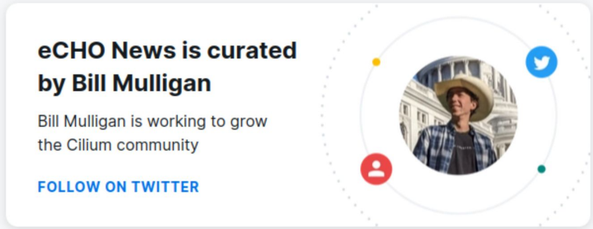
- Flow data can be sourced from dev/staging clusters as well as production to identify the network policy changes required to ship a new app version.
- Historical flow data can be analyzed to predict impact of new more stringent baseline or per-namespace policies.
- Historical flow data combined with enforced policies can be used to detect overly broad rules. For example, “egress allow 0.0.0.0/0 port 443” when a rule to a few specific DNS names would suffice.

ISOVALENT

eBPF resources

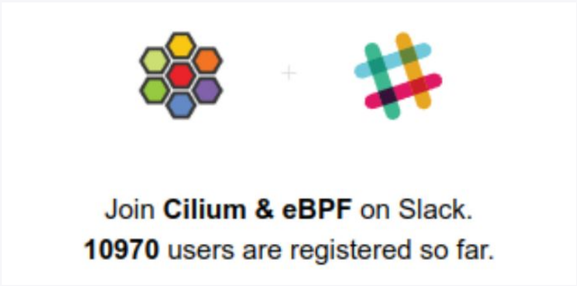
eCHO

eBPF YouTube podcast:
https://www.youtube.com/channel/UCJFUxkVQTBJh3LD1wYBWvuQ



eCHO News

Bi-weekly eBPF newsletter:
https://cilium.io/newsletter/



eBPF & Cilium Slack

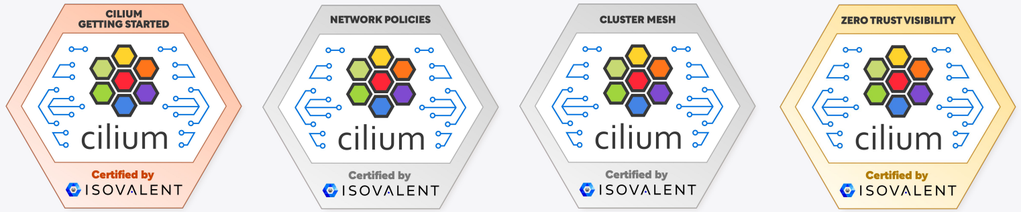
http://slack.cilium.io/

ISOVALENT

Practical Labs

... to become a Cilium & eBPF Jedi

isovalent.com/labs



Get **badges**

ISOVALENT

Zero Trust Visibility

Using Cilium, Hubble & Grafana for namespace security

https://isogo.to/zero-trust-lab

ISOVALENT

Workshops

Zurich — 1st June

Basel — 7th June

Geneva — 14th September

isovalent.com/workshop-tour

ISOVALENT

Thank you!