# Kubernetes Security with Cilium and Tetragon

Conscia
Secure progress

# Who are we?

Fadi Dasus

Senior Cloud Security Architect

fad@conscia.com

Anders Pedersen

Architect

anp@conscia.com

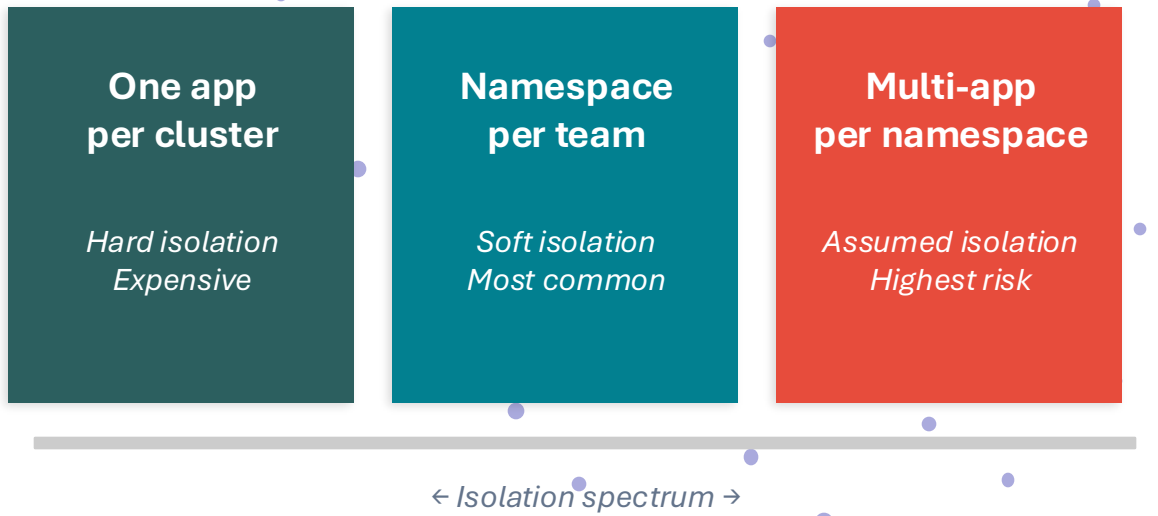
Conscia
Secure progress

# The black box problem.

Kubernetes moves fast — and security teams are often left blind to what's actually running.

**Like cloud, but worse:** the attack surface lives inside your own cluster.

And most clusters aren't single-tenant.




**The further right you go, the harder security becomes.**

# Allow-all by default.

- Fully open by default — no firewall, no segmentation
- Namespaces create soft boundaries, holes needed for shared services
- Network policies exist, but are complex and rarely adopted
- Pods are ephemeral — IPs change, policies become stale
- No visibility → can't write policies without breaking things

THE DILEMMA

**Too strict**

Block the dev teams

**Too loose**

Accept the risk

**Without visibility, you can't win.**

# Service mesh arrived.

*An elegant answer — with real trade-offs.*

*Some customers adopt service mesh purely for the observability.*

**But is there a better way?**

| WHAT IT SOLVES | THE COST |
| --- | --- |
| ✓ **Security**<br>mTLS + identity-based access control, east-west | ✕ **Complexity**<br>Another control plane to manage and upgrade |
| ✓ **Observability**<br>Full service-to-service visibility, zero code changes | ✕ **Overhead**<br>Extra sidecar per pod CPU, memory, latency |

# Cilium & Hubble.

**2015** Kubernetes v1.0

**2018** Service mesh goes GA (Istio 1.0)

**2021** **Cilium hits mainstream (GKE default, CNCF incubating)**

## 01 Observability

Hubble gives real-time, identity-aware view of all network flows — what's talking, what's dropped.

## 02 Security

eBPF hooks in at the kernel level — deep policy enforcement with full pod identity context.

## 03 No sidecars

All the benefits of a service mesh. None of the sidecar complexity.

**Built on eBPF · One solution · No sidecar complexity**

# From breach to zero trust

Visibility, isolation, and runtime defense inside Kubernetes.





eBPF · Cilium · Hubble · Tetragon

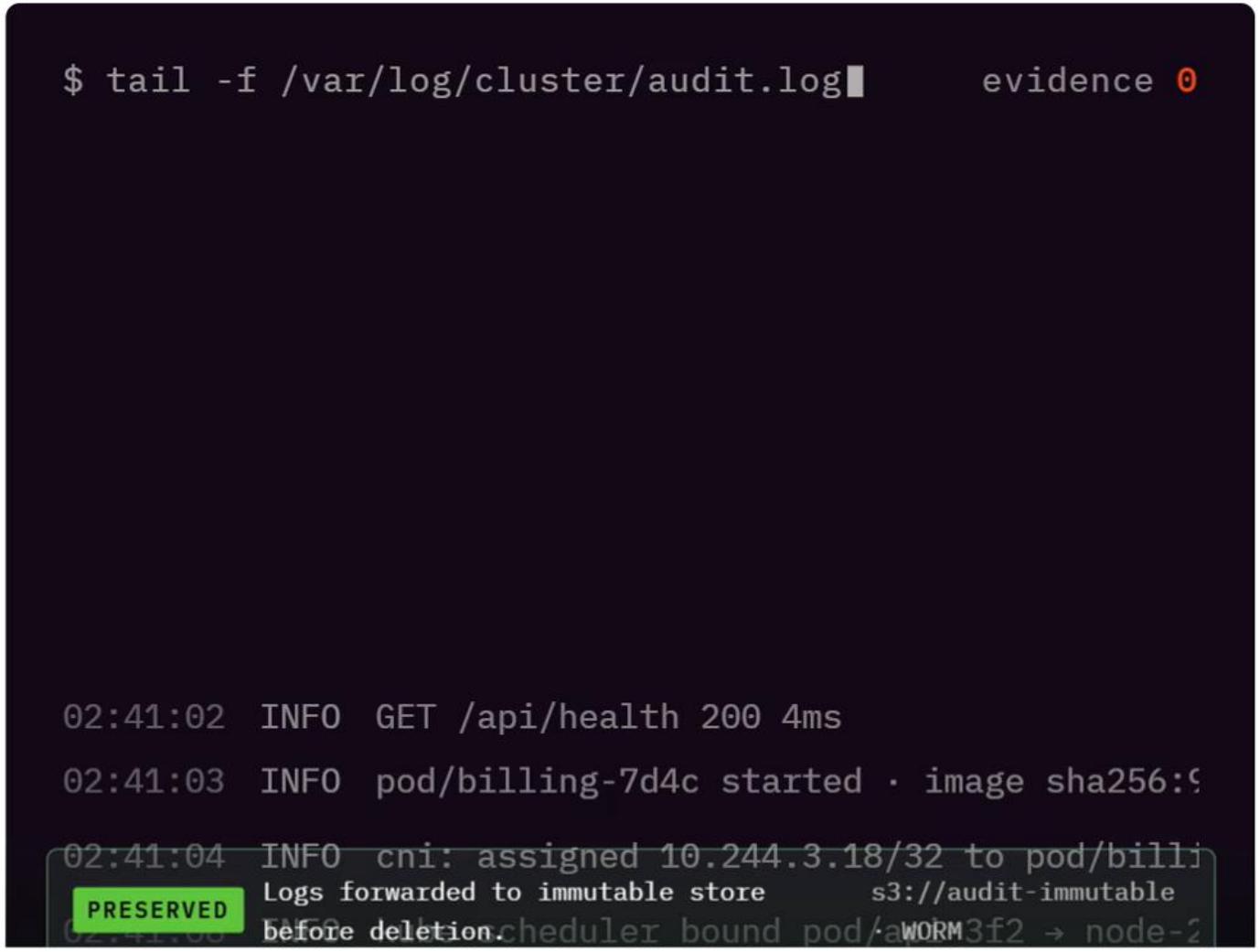

# A real customer, a real Tuesday and a compromise measured in minutes.

| | |
|---|---|
| T+00:00 | Initial foothold via workload vuln |
| T+01:14 | Lateral movement across pods |
| T+02:31 | Privilege escalation to cluster root |
| T+03:00 | Log storm begins. The useful ones, deleted. |

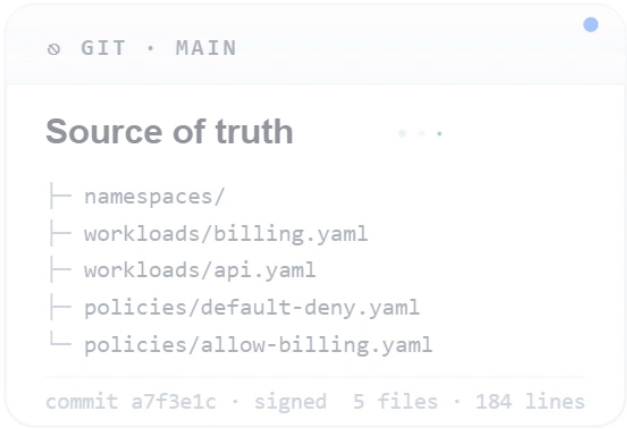
# Logs betray you first.

In a real incident, the useful logs are drowned in attacker-generated noise or deleted outright. You are blind in the one moment you most need to see.

NIS2 and CIS Benchmarks **require** centralized log management with backup and integrity controls. Take care of your logs, or they won't be there when you need them.



# A source of truth

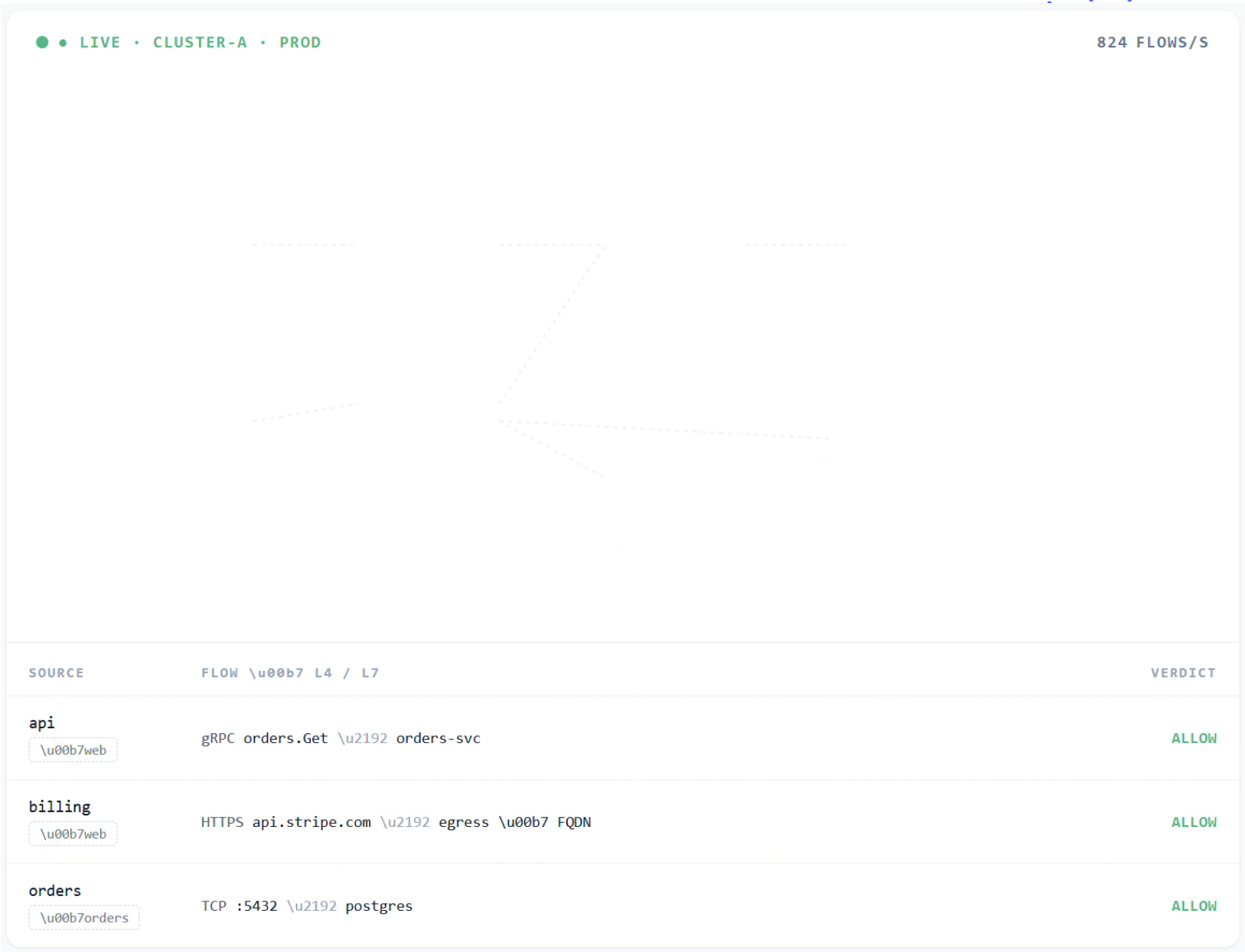



GitOps gives the cluster a **declared healthy state** : the business logic that has to be running, the services that must exist, the policies that must apply.

If you don't know what "healthy" looks like, you can't tell when you're compromised. A Git repo, reconciled continuously, is the baseline everything else is measured against.

# Visibility

Identity-aware flow visibility into every connection in the cluster who is talking to whom, on which protocol, with which result. No app instrumentation. Kernel-speed, eBPF-cheap.



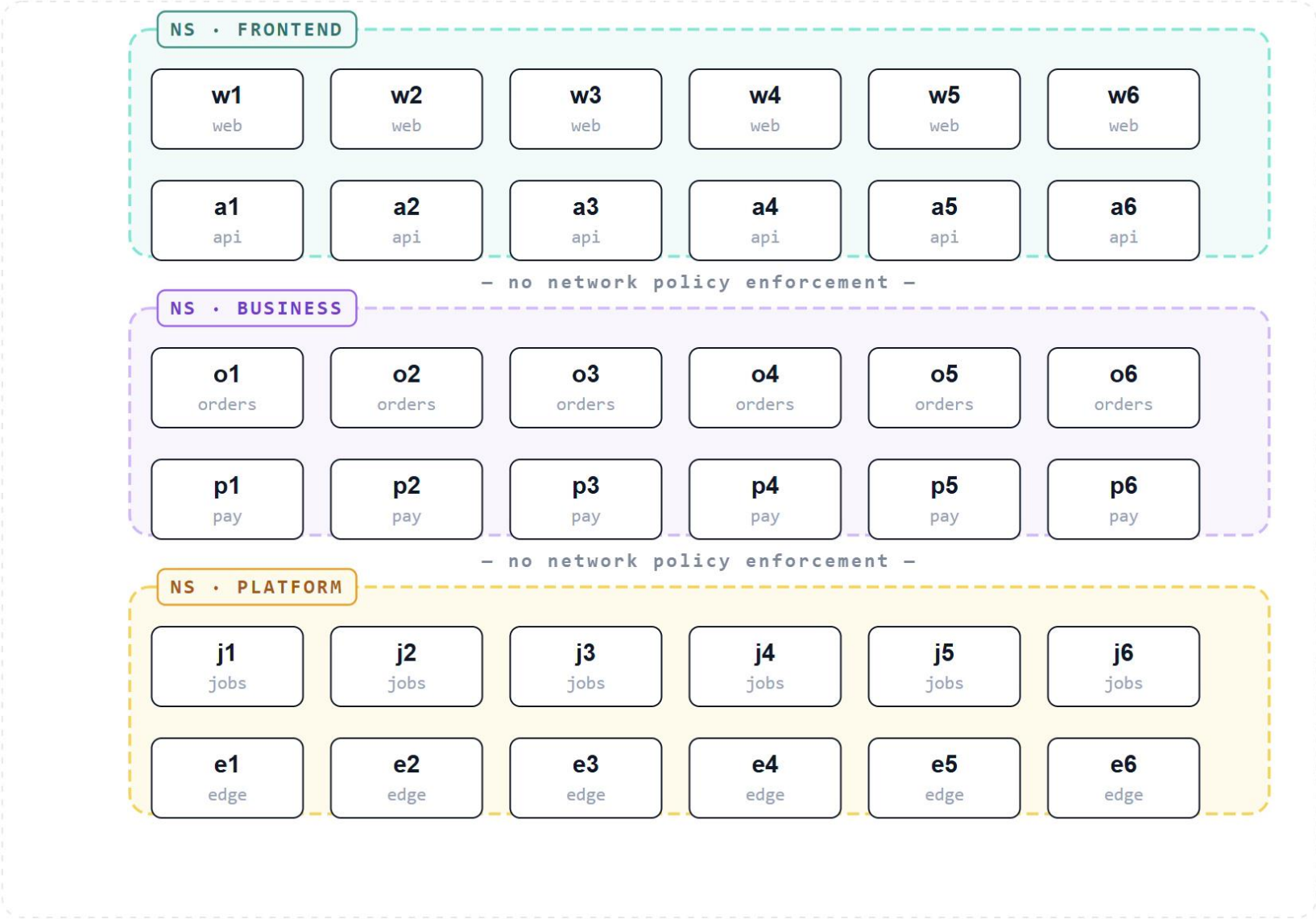

- Hubble
- L3/L4 + L7
- No sidecar

CLUSTER \u00b7 FLAT NETWORK \u00b7 3 NAMESPACES



# Isolation

A default Kubernetes cluster is a flat network. Every pod can reach every other pod, across namespaces, across tiers, across trust boundaries. One compromised workload sees the whole house.

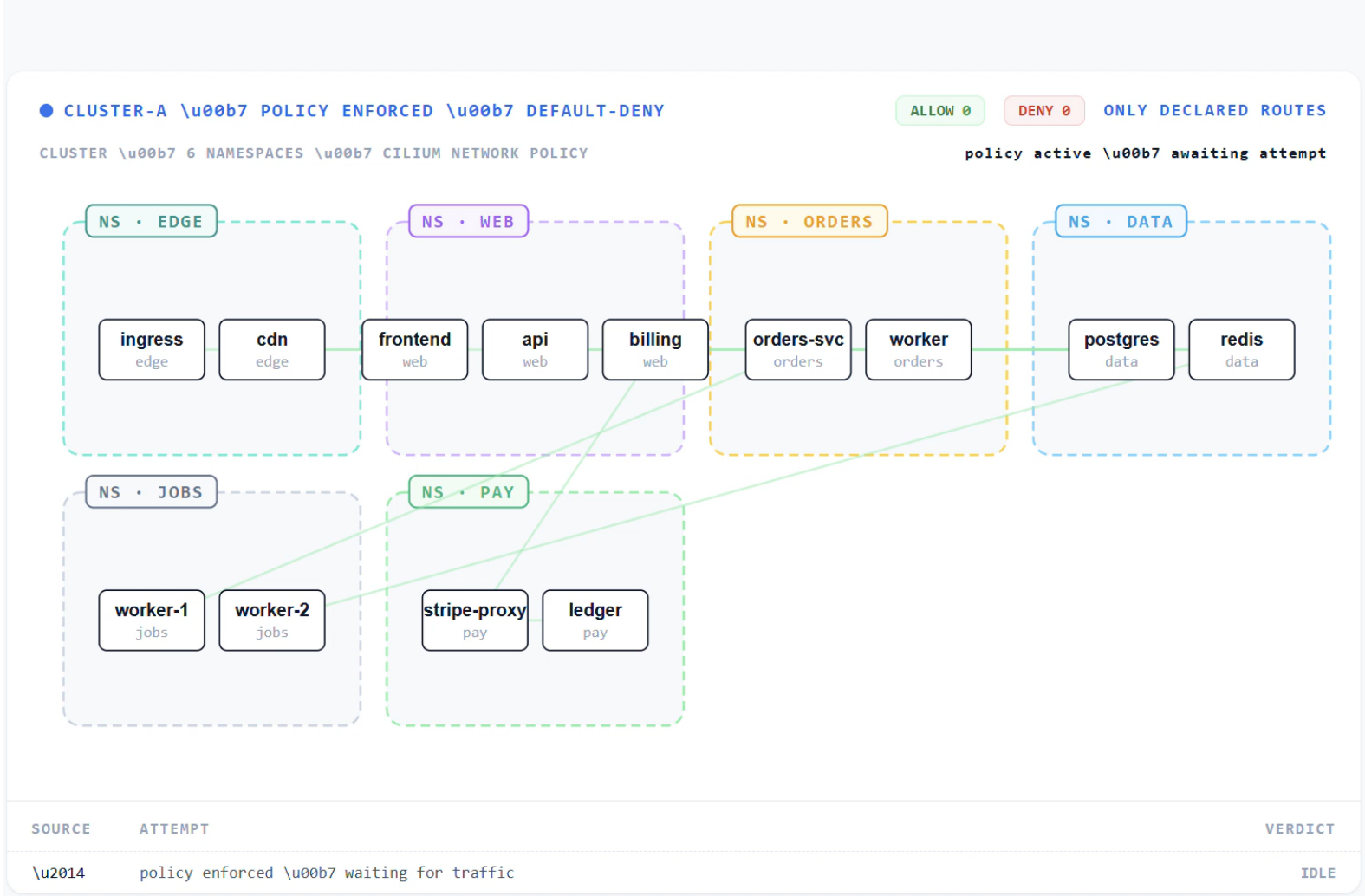



- flat network
- no segmentation
- no default-deny

# Isolation

Namespace per soft boundary, cluster per hard one, and **default-deny** as hygiene. Only declared routes work everything else hits a wall.

```
# CiliumNetworkPolicy — L7 aware
kind: CiliumNetworkPolicy spec: endpointSelector: { app: api }
 egress: - toEndpoints: [{ app: orders }]
toPorts: [ http: [{ method: GET, path: /api/orders }] ]
```

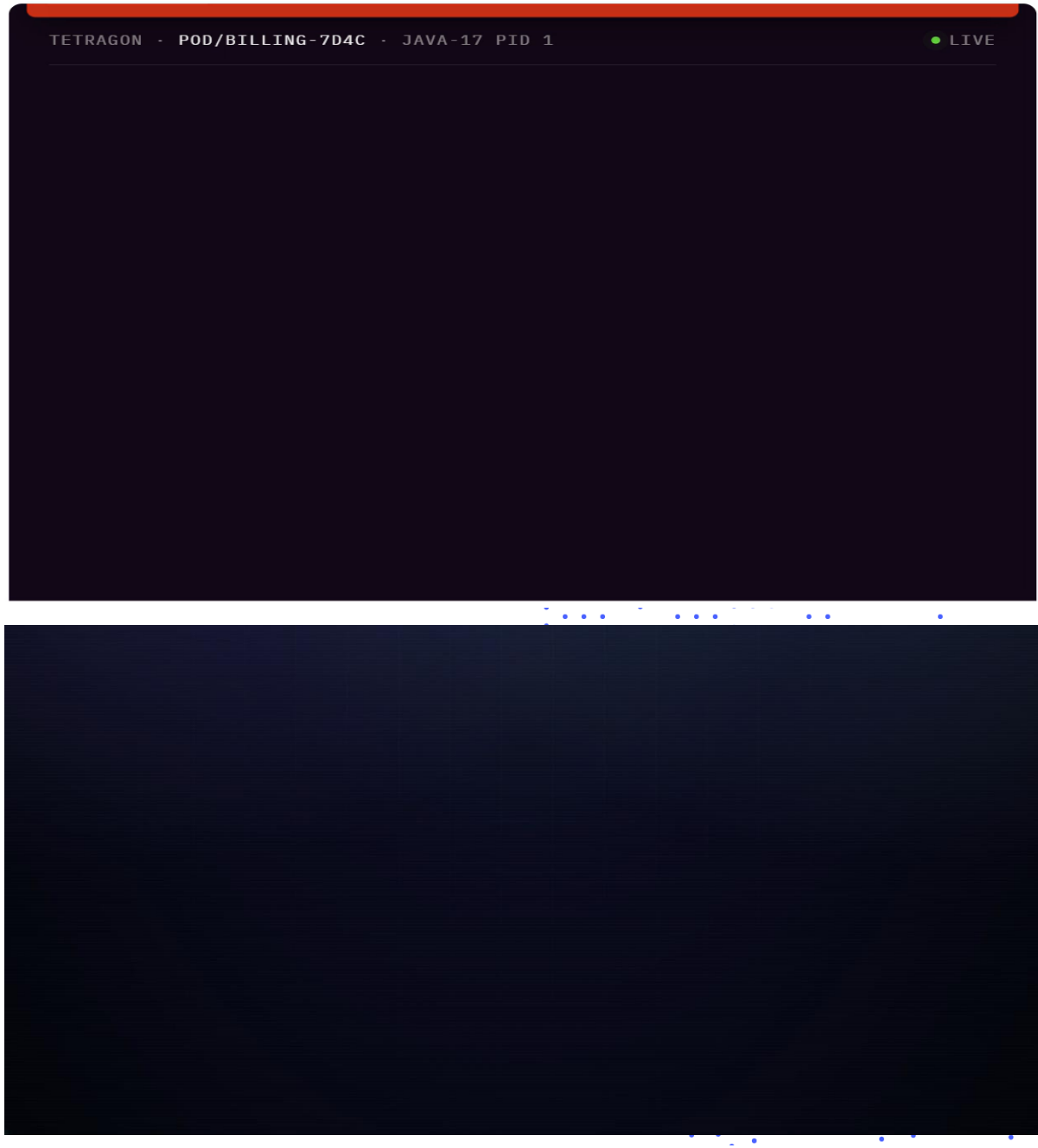



Conscia

# Runtime Security

Sometimes the compromise lives inside a legitimate pod processes no network policy will ever see. Tetragon pushes enforcement to the kernel: syscalls, file I/O, process exec, network calls.

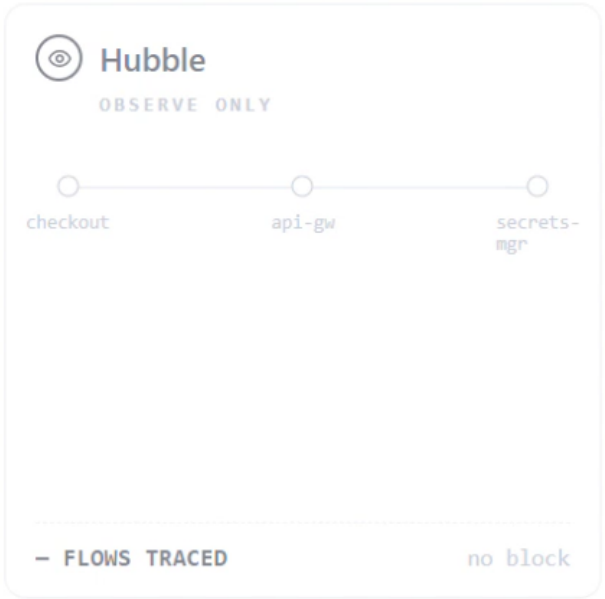
Runtime envelope ships with the workload. Security becomes a property of the artifact, not a prod-time afterthought.




- allow declared
- block & alert
- CI/CD shipped

# Four things.

01




## Observability is a security control.

Not a convenience. In an incident, it is the first thing you will wish you had.

02


## A Source Of Truth

Declare the cluster in Git. Reconcile continuously. Drift becomes an event, not a mystery.

03


## Isolation is designed in, not bolted on.

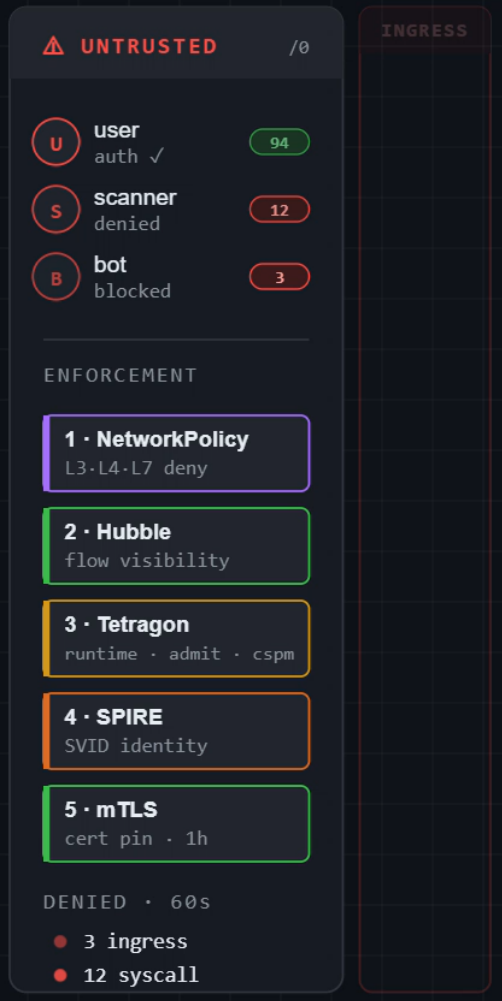
Default deny is not paranoia  it is hygiene. Express it as code; enforce it by default.

04


## Runtime closes the last gap.

The network tells you who is talking. Runtime policy tells you what is actually running. You need both.

# Zero Trust for Kubernetes (NIST 800-207)


⚠ UNTRUSTED
/0
U
user
auth ✓
94
S
scanner
denied
12
B
bot
blocked
3
ENFORCEMENT
1 · NetworkPolicy
L3·L4·L7 deny
2 · Hubble
flow visibility
3 · Tetragon
runtime · admit · cspm
4 · SPIRE
SVID identity
5 · mTLS
cert pin · 1h
DENIED · 60s
3 ingress
12 syscall

INGRESS

# Three minutes is all an attacker needs.

Make sure your cluster doesn't give it to them.

# Thank you

Fadi Dasus
fad@conscia.com

Anders Pedersen
anp@conscia.com

Conscia
Secure progress

Conscias årlige sikkerhedskonference

# Conscia Momentum 2026

## - Resilience in motion

2. september 2026

Hangaren, Kastrup

Læs mere og
tilmeld dig her:

Conscia
Secure progress