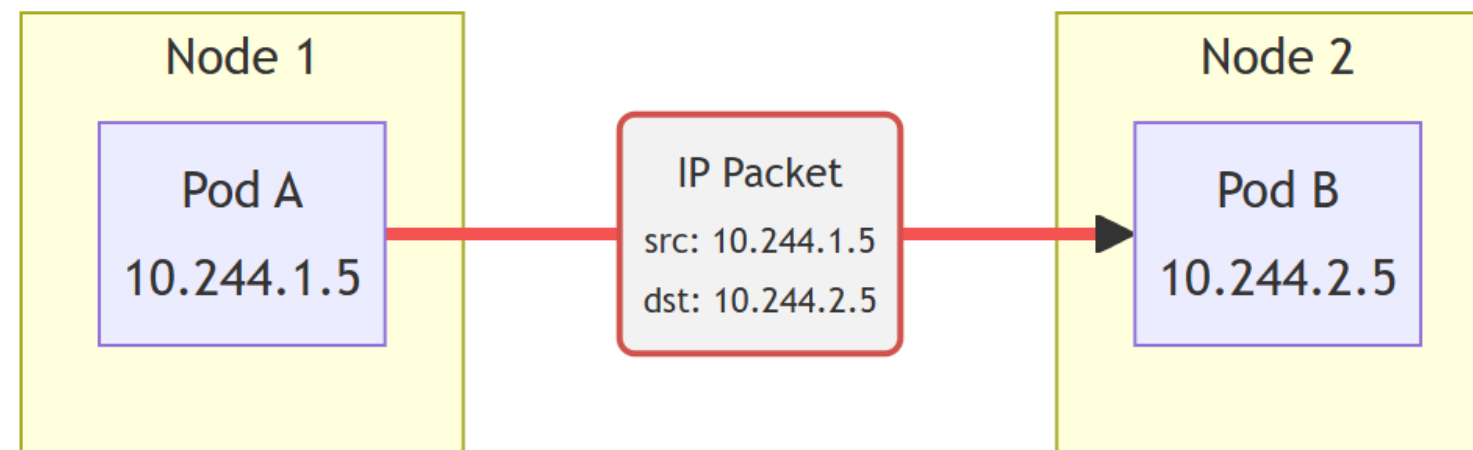


Demystifying the Kubernetes Network Stack

From Pod to Pod - A Deep Dive into How Packets Really Travel

Simone Rodigari · Microsoft

KubeCon + CloudNativeCon Europe 2026



Kubernetes Networking: Follow the Packet

You apply a manifest. Pods appear. They talk.

But what happens at the packet level?

The Gap

Most engineers don't see the packet path.

The Risk

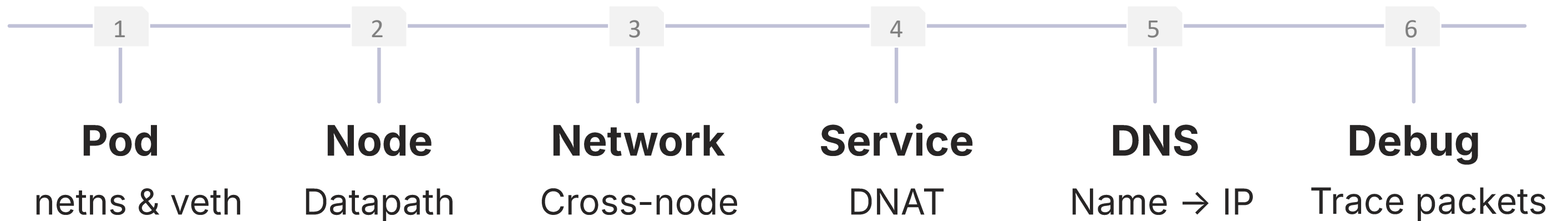
When it breaks, guessing is slow.

The Goal

A mental model for tracing packets.

Talk Roadmap

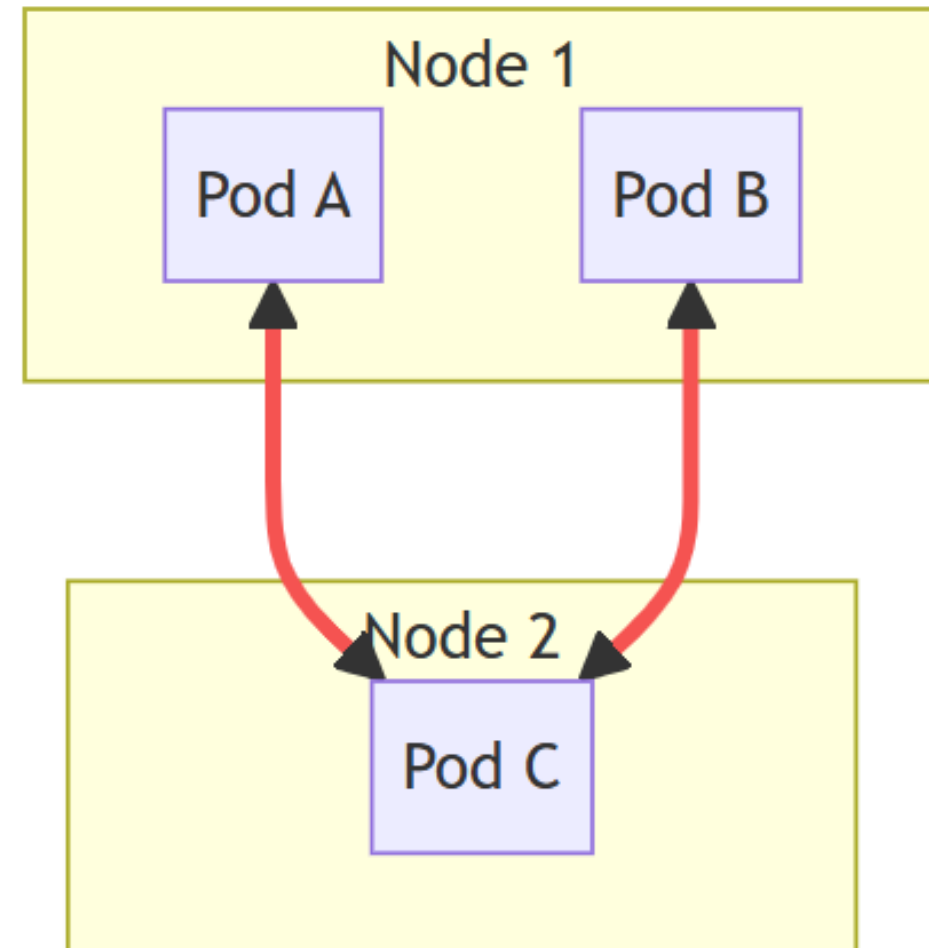
One packet. Six steps.



Baseline: Kubernetes Networking

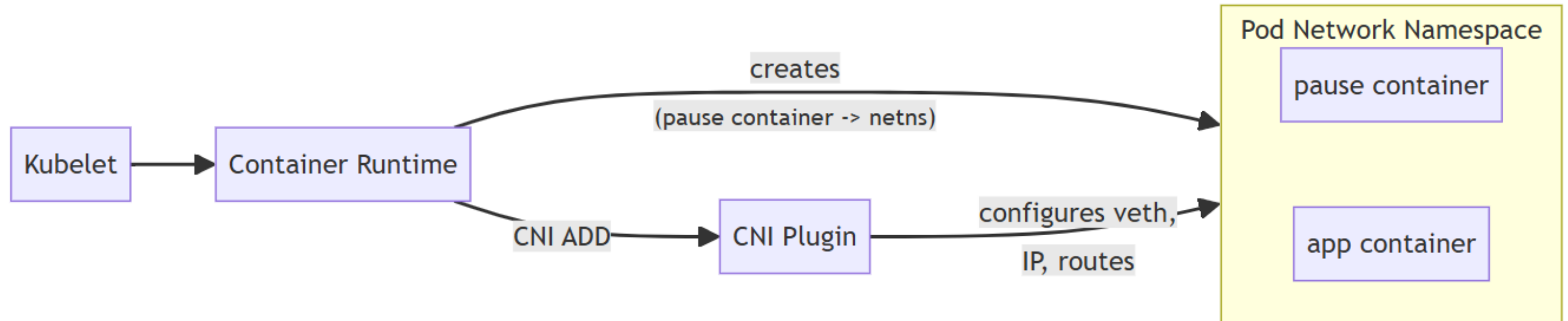
Before tracing a packet, we need the ground rules.

- Pod = unique IP
- No NAT between Pods
- Flat L3: any Pod → any Pod



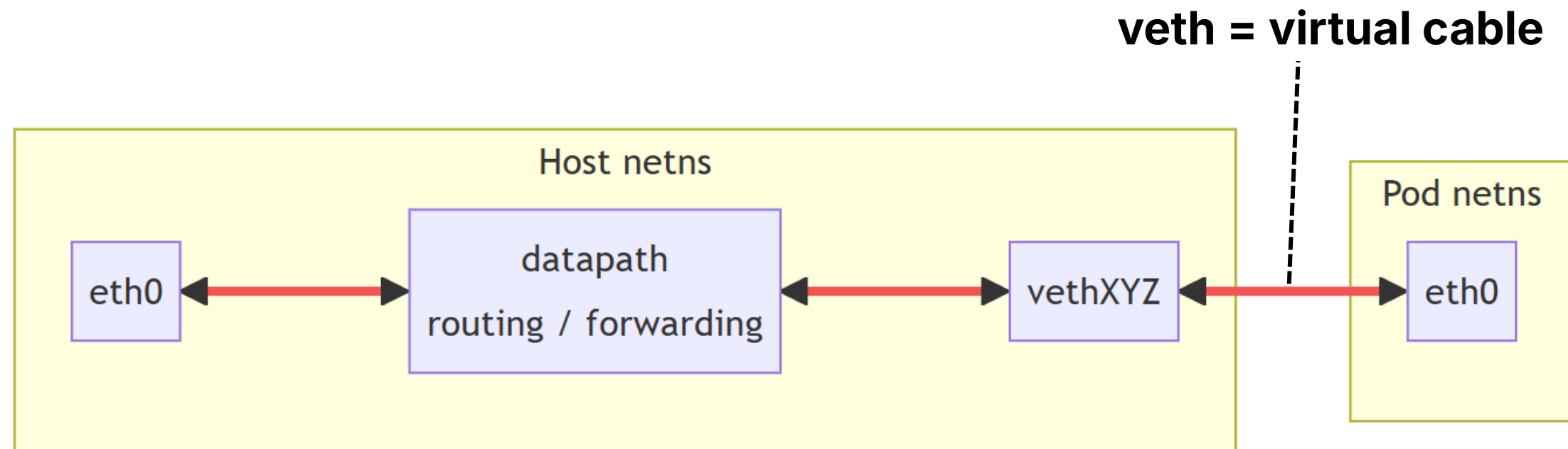
How Pod Networking is set up

- Pod networking is set up before containers start
- CNI wires the pod into the node



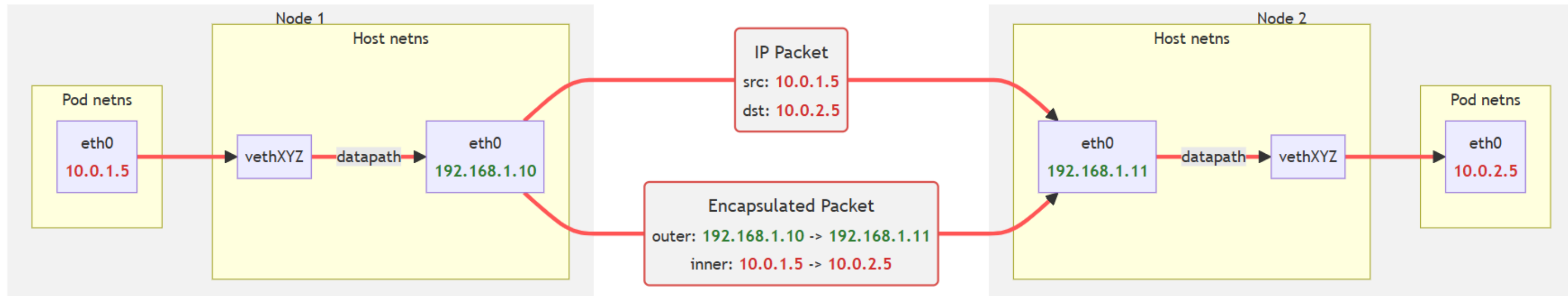
How a Pod Connects to the Node

- Pod runs in its own netns
- Connected to the host via a veth pair
- Host routes and processes traffic



How Node Talks: Overlay vs Underlay

Pod → Pod across nodes



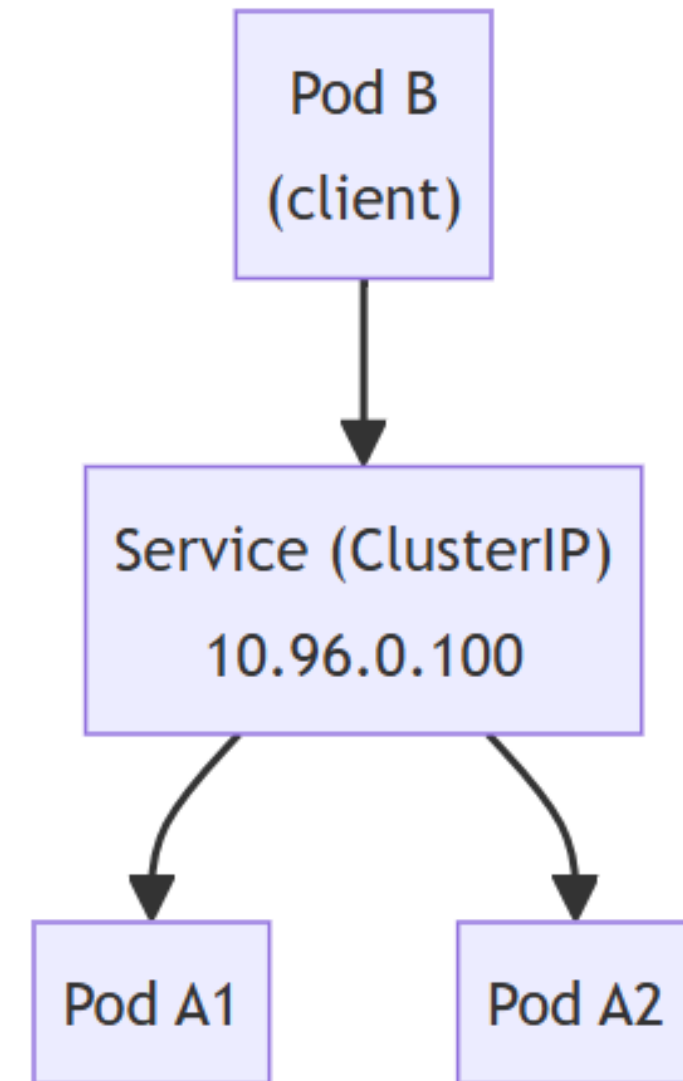
Overlay = encapsulation over node network, portable
Underlay = native routing, high performance

Stable Identity: Why Services Exist

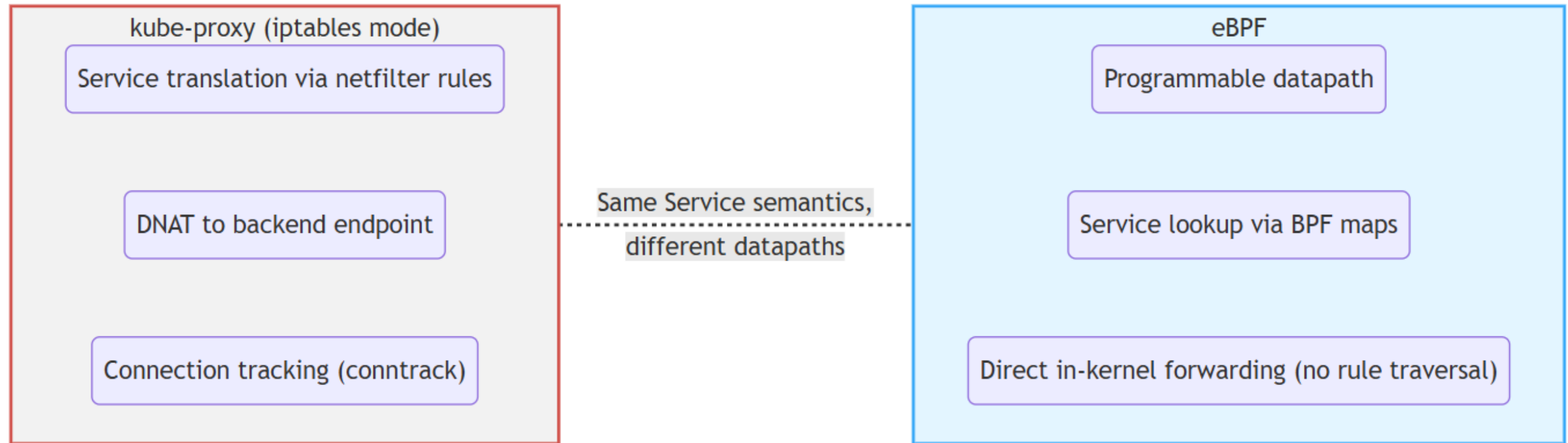
- Pods are ephemeral – their IPs change
- Direct pod-to-pod communication is unreliable

Services provide a stable virtual IP (ClusterIP) and load balance across ready Pods.

Implementation: Service datapath via kube-proxy (iptables) or eBPF-based CNIs.

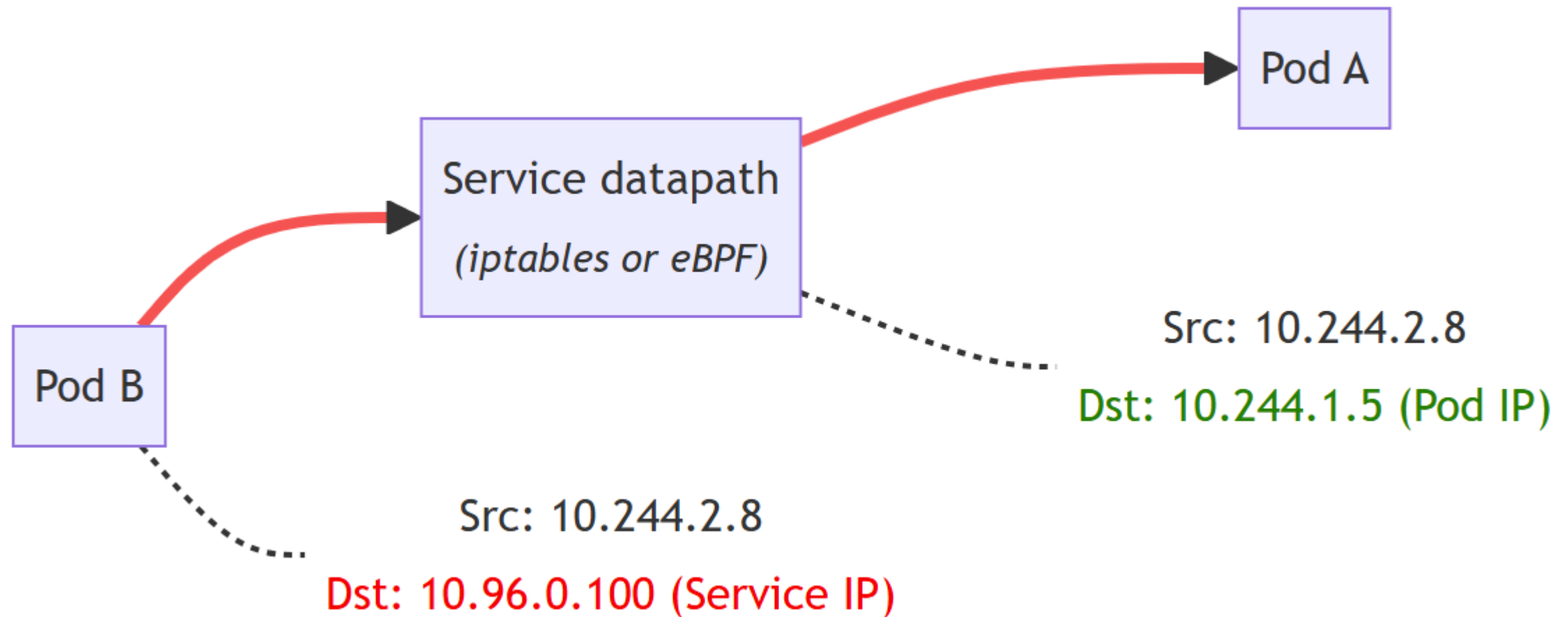


Service Datapath: kube-proxy vs eBPF



Service Datapath: ClusterIP

- Services provide load balancing and DNAT to selected backend
- DNAT rewrites the destination IP while preserving the source IP

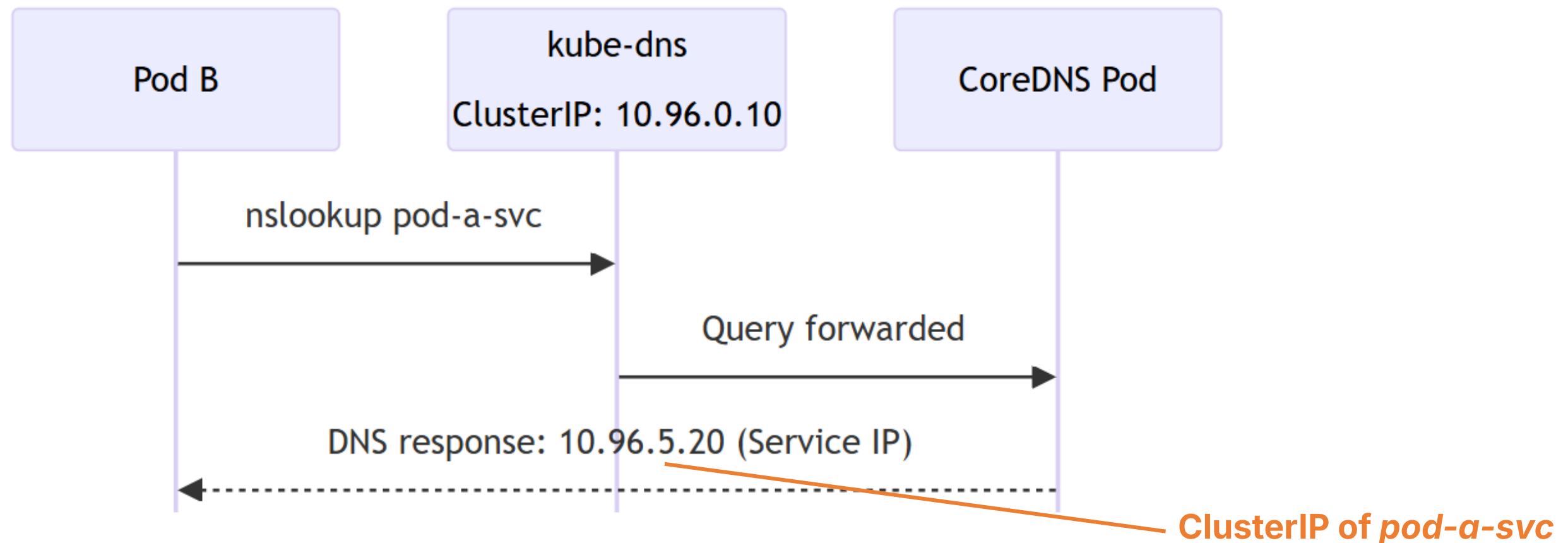


DNS and Service Discovery

DNS maps names to
Service IPs.

Services → stable IPs
DNS → stable names

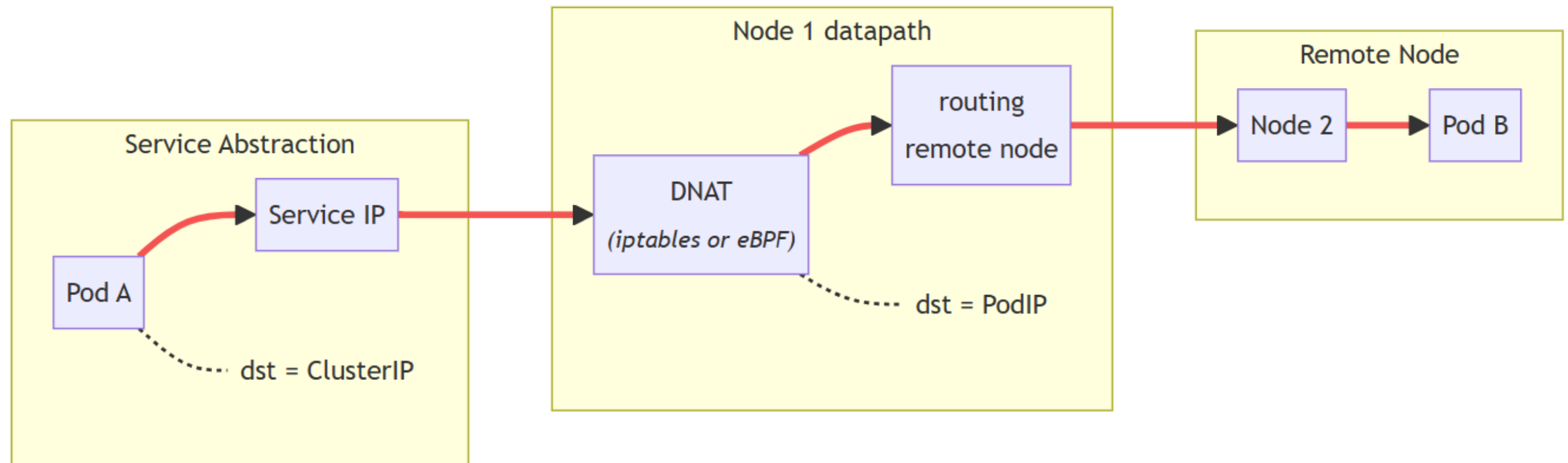
CoreDNS watches
Services and EndpointSlices



Tracing a Packet through Kubernetes

Scenario: Pod → ClusterIP Service → Pod (cross-node)

DNS (svc → ClusterIP) occurs before this flow



Debugging Kubernetes Networking

Follow the packet: interface → route ↔ translate → observe

Interfaces `ip addr` / `ip link`

Routing `ip route`

Translation `iptables` / `eBPF`

Observe `tcpdump`

DNS `nslookup` / `dig`

Access `kubectl debug`

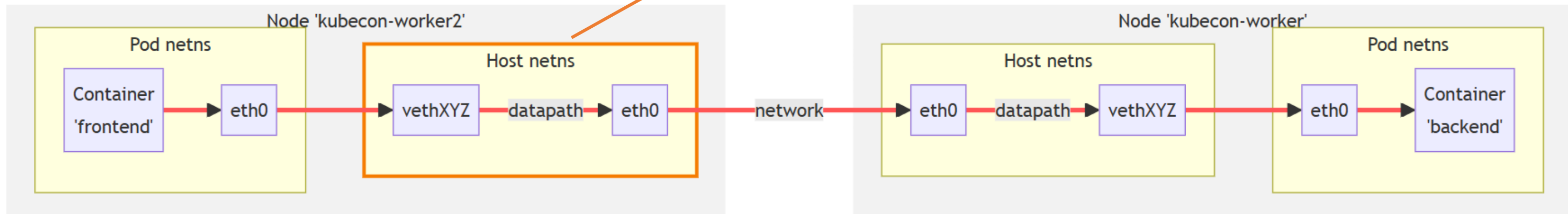
Demo Setup and Walkthrough

kind cluster (2 worker nodes)

CNI: kindnet

kube-proxy: iptables mode

- Inspect IPs
- Pod-to-Pod traffic
- Service (DNAT)
- **Packet tracing**
- DNS



~

>

08:50:18

~

>

08:48:45

~

>

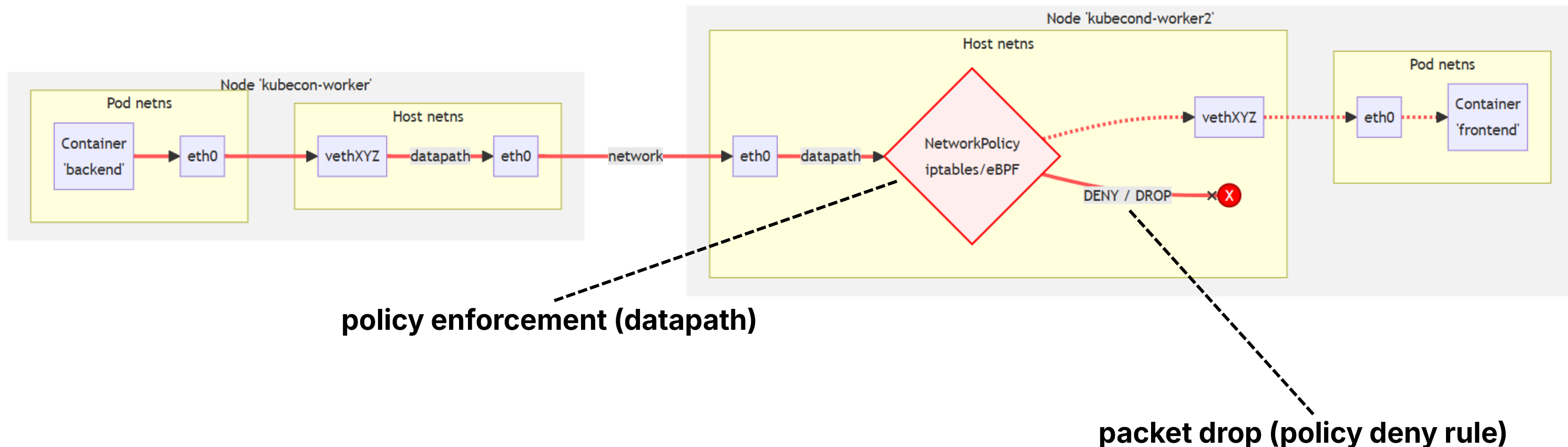
08:48:50

[Next] Policy: Path $\neq$ Permission

Default: any Pod $\leftrightarrow$ any Pod

NetworkPolicy: L3/4 firewall for Pods

Enforced in node datapath: iptables or eBPF



Key Takeaways

Built on Linux

Namespaces, veth, routing

Services = stable Identity

Stable IP over ephemeral Pods

Debug = follow packet

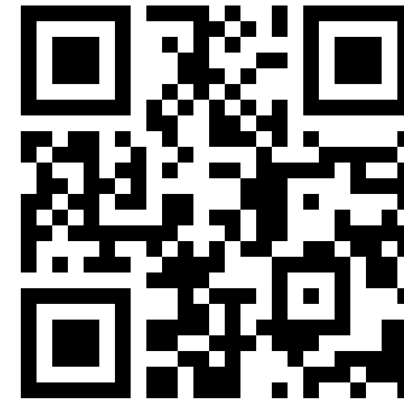
Path = truth

Questions?

Simone Rodigari · Microsoft
KubeCon + CloudNativeCon Europe 2026



Deep dive blog series



Leave feedback