Understanding the Kubernetes CNI and Network Policy

July 27, 2022

TIGERA

What is Project Calico?

The Project Calico community develops and maintains Calico, an open source networking and network security solution for containers, virtual machines, and host-based workloads.

TIGERA

PROJECT CALICO

https://projectcalico.org

@projectcalico

https://github.com/projectcalico/community

https://slack.projectcalico.org

https://discuss.projectcalico.org

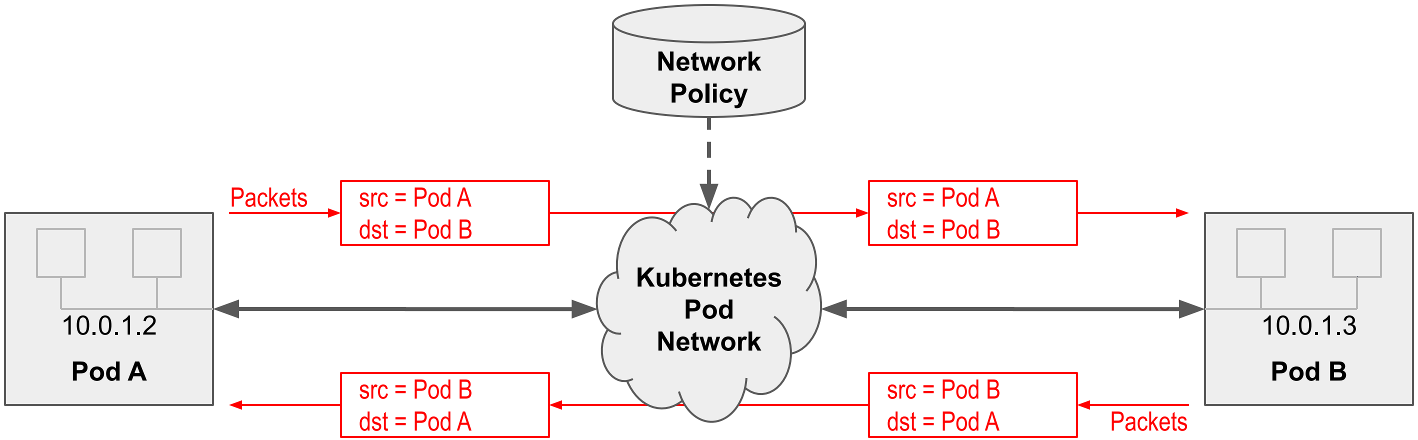
6000+
Slack channel members

150+
Contributors

1,000,000+
Nodes powered by Calico every day

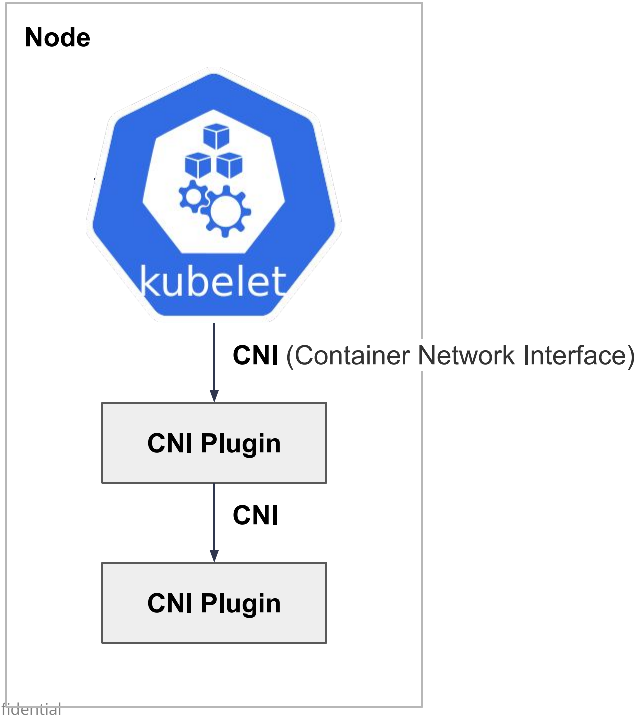
TIGERA

The Kubernetes Network Model

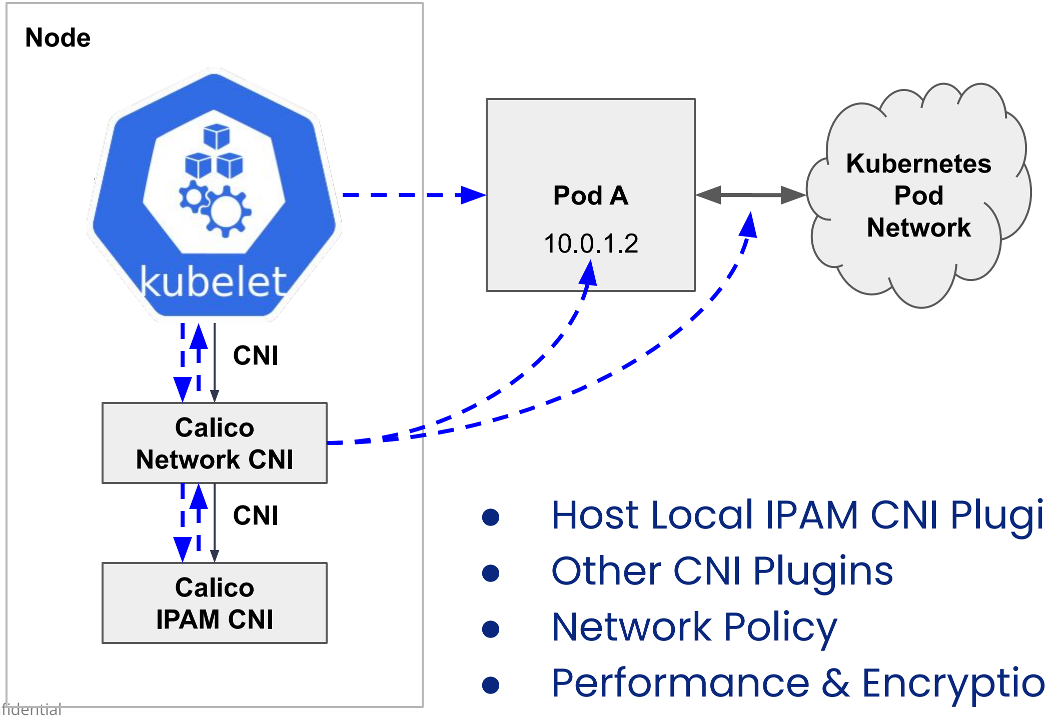


- IP per Pod -> VMs & Processes ≅ Pods & Containers
- Isolation with Network Policy -> simple “flat” network

Kubernetes Network Implementations



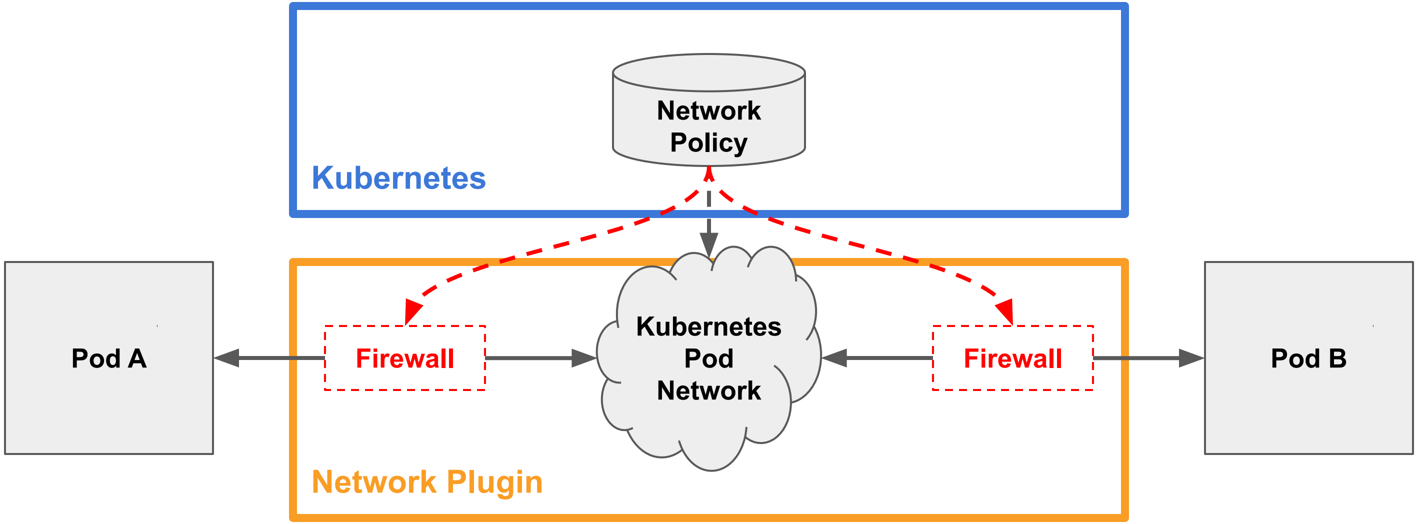
Kubernetes Network Implementations



- Host Local IPAM CNI Plugin
- Other CNI Plugins
- Network Policy
- Performance & Encryption

Calico Quickstart Demo

What is Network Policy?



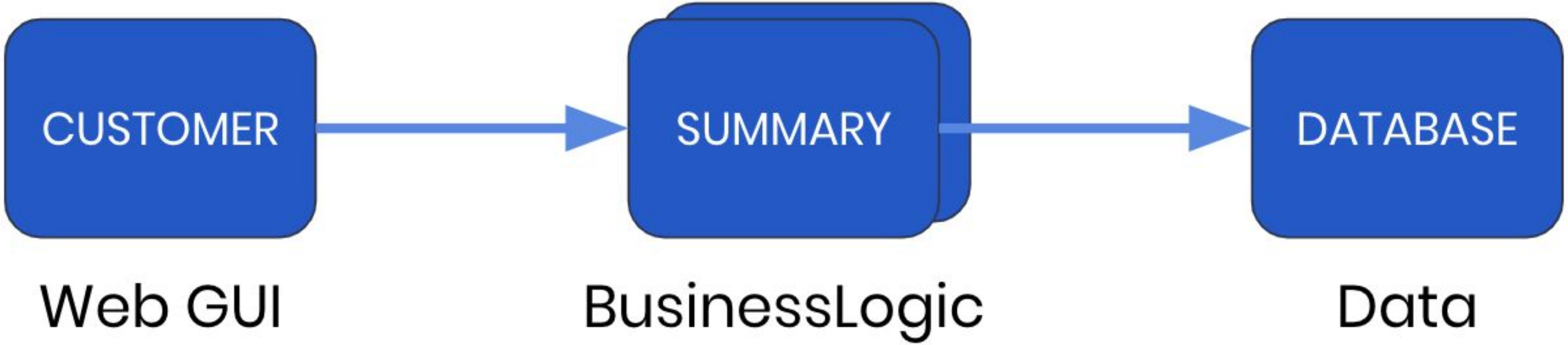
- Simple “flat” network
- Isolation is not defined by the structure of the network

Kubernetes Network Policy

```yaml
apiVersion: networking.k8s.io/v1
kind: NetworkPolicy
metadata:
  name: blue-policy
  namespace: production
spec:
  podSelector:
    matchLabels:
      color: blue
  ingress:
  - from:
    - podSelector:
        matchLabels:
          color: red
    ports:
      - port: 80
```

Introduction to the Sample Application

Yet Another Online Bank (YaoBank)



Kubernetes Network Policy Quick Demo

Network Policy Support

Kubernetes Network Policy

- Ingress & egress rules
- Pod selectors
- Namespace selectors
- Port lists
- Named ports
- IP blocks & excepts
- TCP, UDP, or SCTP

Calico Network Policy

- Namespaced & global scopes
- Deny and log actions
- Policy ordering
- Richer matches, including:
  - ServiceAccounts
  - ICMP
- Istio integration, including:
  - Cryptographic identity matching
  - Layer 5-7 match criteria

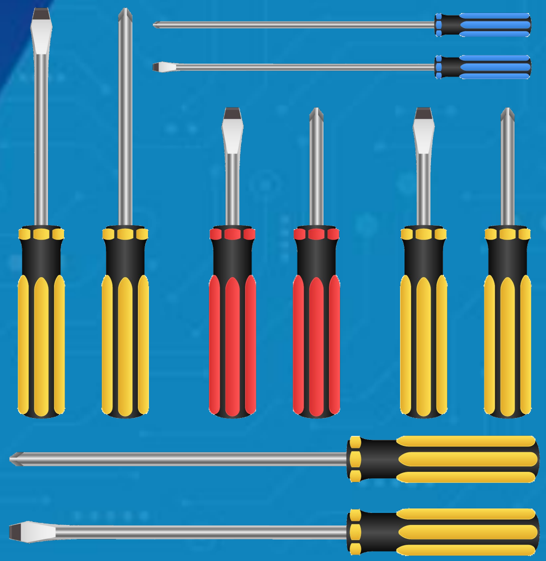
Calico Network Policy

```yaml
apiVersion: projectcalico.org/v3
kind: NetworkPolicy
metadata:
  name: blue-policy
  namespace: production
spec:
  order: 50
  selector: color == 'blue'
  ingress:
  - action: Allow
    protocol: TCP
    source:
      selector: color == 'red'
    destination:
      ports:
        - 80
```

```yaml
apiVersion: projectcalico.org/v3
kind: GlobalNetworkPolicy
metadata:
  name: red-policy
spec:
  order: 100
  selector: color == 'red'
  ingress:
  - action: Deny
    source:
      selector: color == 'blue'
  - action: Allow
    source:
      serviceAccounts:
        selector: color == 'green'
```

Calico Network Policy Quick Demo

Calico cluster and the eBPF data plane



TIGERA

Screwdriver Vectors by Vecteezy -> https://www.vecteezy.com/free-vector/screwdriver

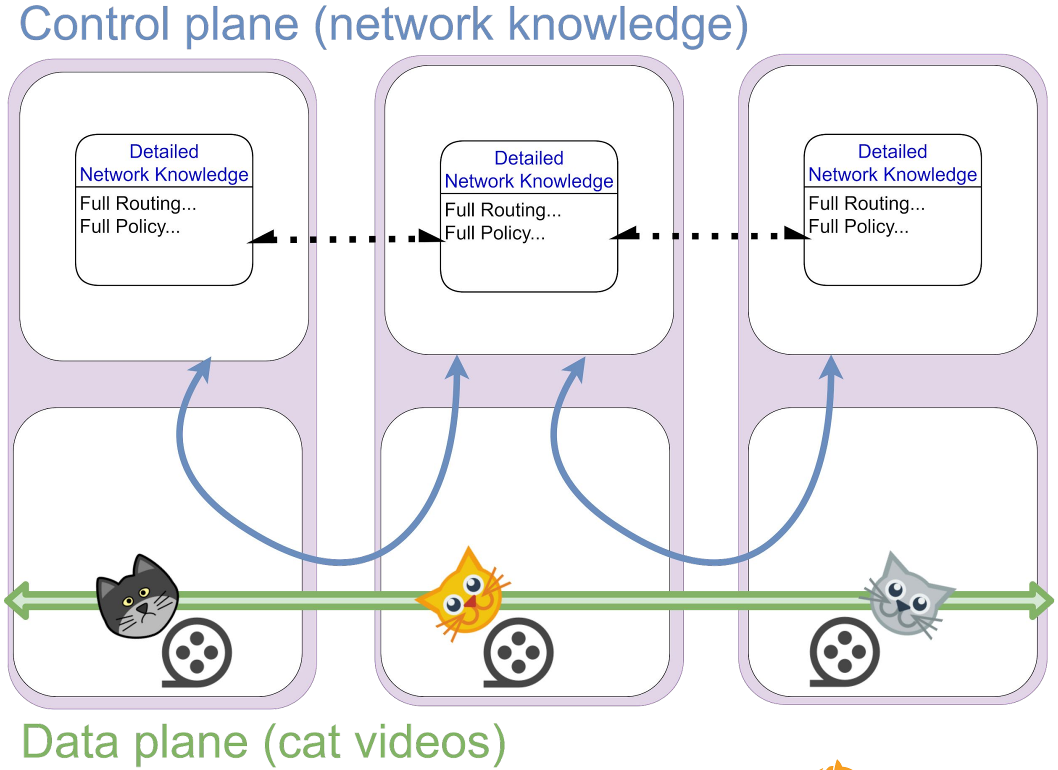
Networking Software

In networking, devices/networks usually each have an architecture of:

- A control plane
- A data plane

The blue arrows represent device and network state.

The green arrow represent "user" network traffic.


Control plane (network knowledge)
Detailed Network Knowledge
Full Routing...
Full Policy...
Detailed Network Knowledge
Full Routing...
Full Policy...
Detailed Network Knowledge
Full Routing...
Full Policy...
Data plane (cat videos)

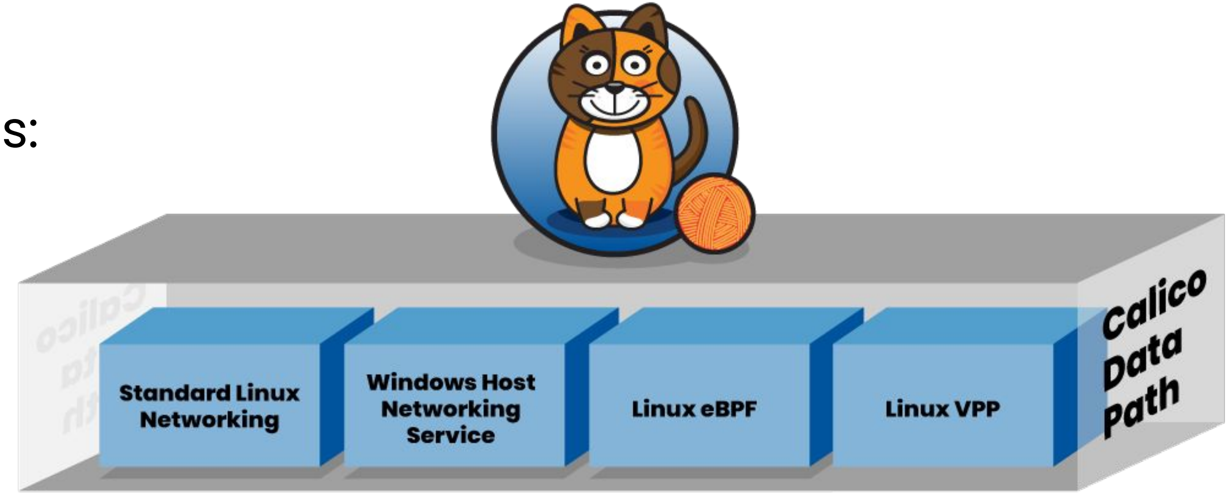
What is eBPF?

- A Linux kernel feature that lets you run small programs inside the Linux kernel
- Allows small programs to be loaded into the kernel, and attached to hooks
- Event driven
- Does not require kernel source code change or loading kernel modules
- Generic kernel v5.4.0+ or RH kernel v4.18.0-193+

About Pluggable Dataplanes

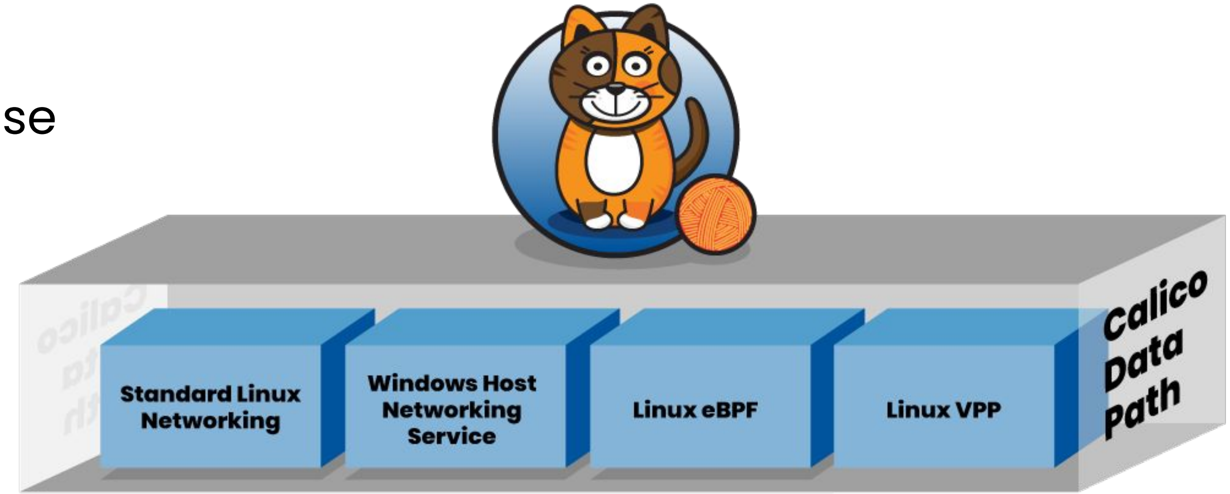
- Today, Calico offers 4 pluggable data planes:
  - Linux iptables
  - Windows HNS
  - Linux eBPF
  - Linux VPP*

(*currently tech-preview)



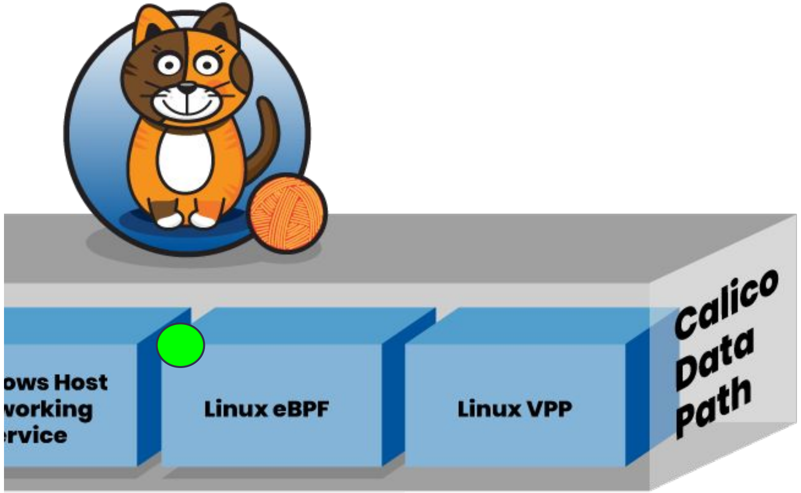
About Pluggable Dataplanes

- Control plane code reuse
- Specialised, minimal dataplane code
- Targeted feature set
- Future-proofing
- Agility (for everyone!)

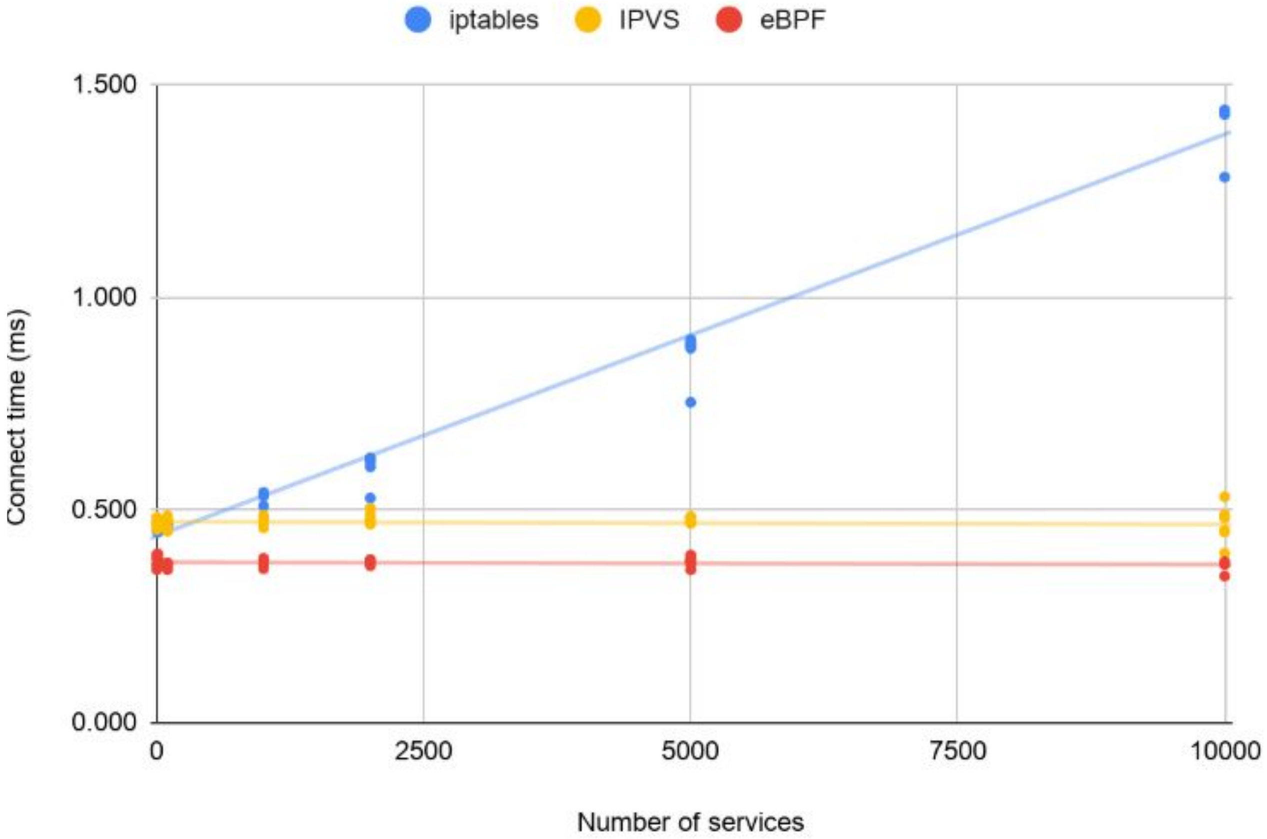


About the Linux eBPF Data Plane

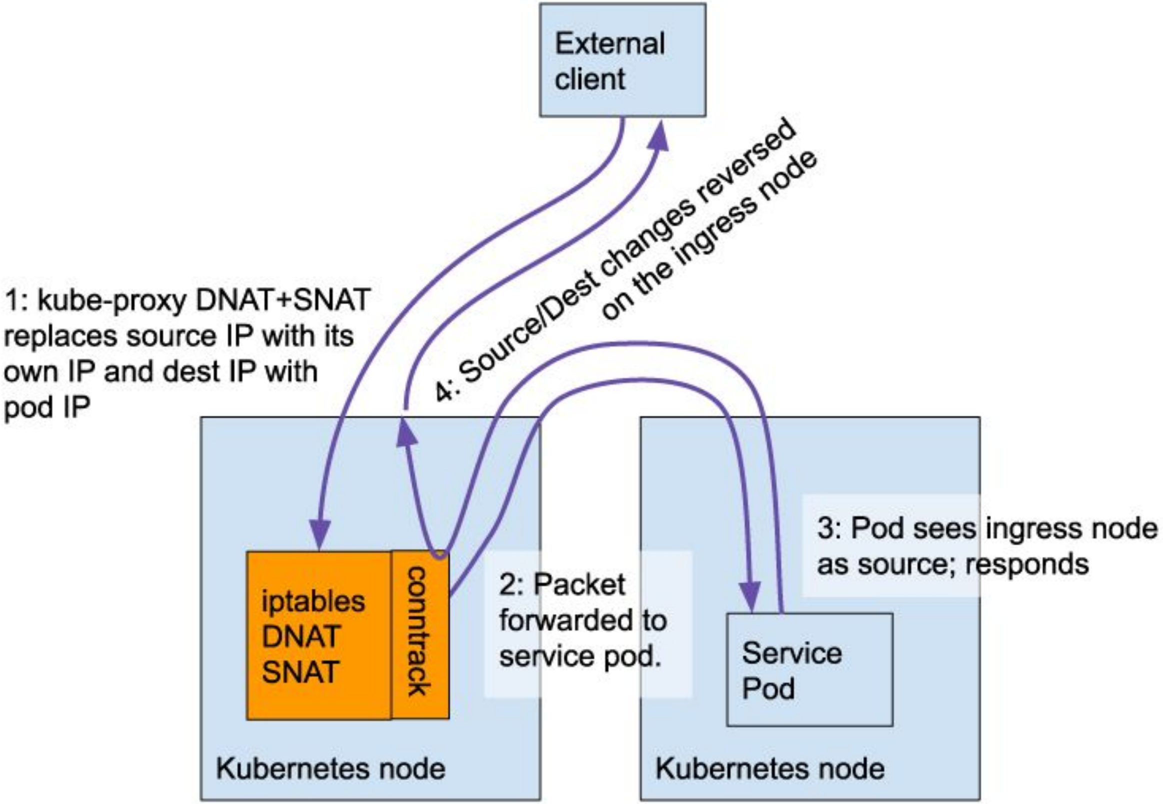
- Scales to higher throughput
- Uses less CPU per Gigabit
- Has native support for Kubernetes services:
    - Preserves external client source IP addresses all the way to the pod
    - Supports DSR (Direct Server Return) for better efficiency
    - Uses less CPU than kube-proxy



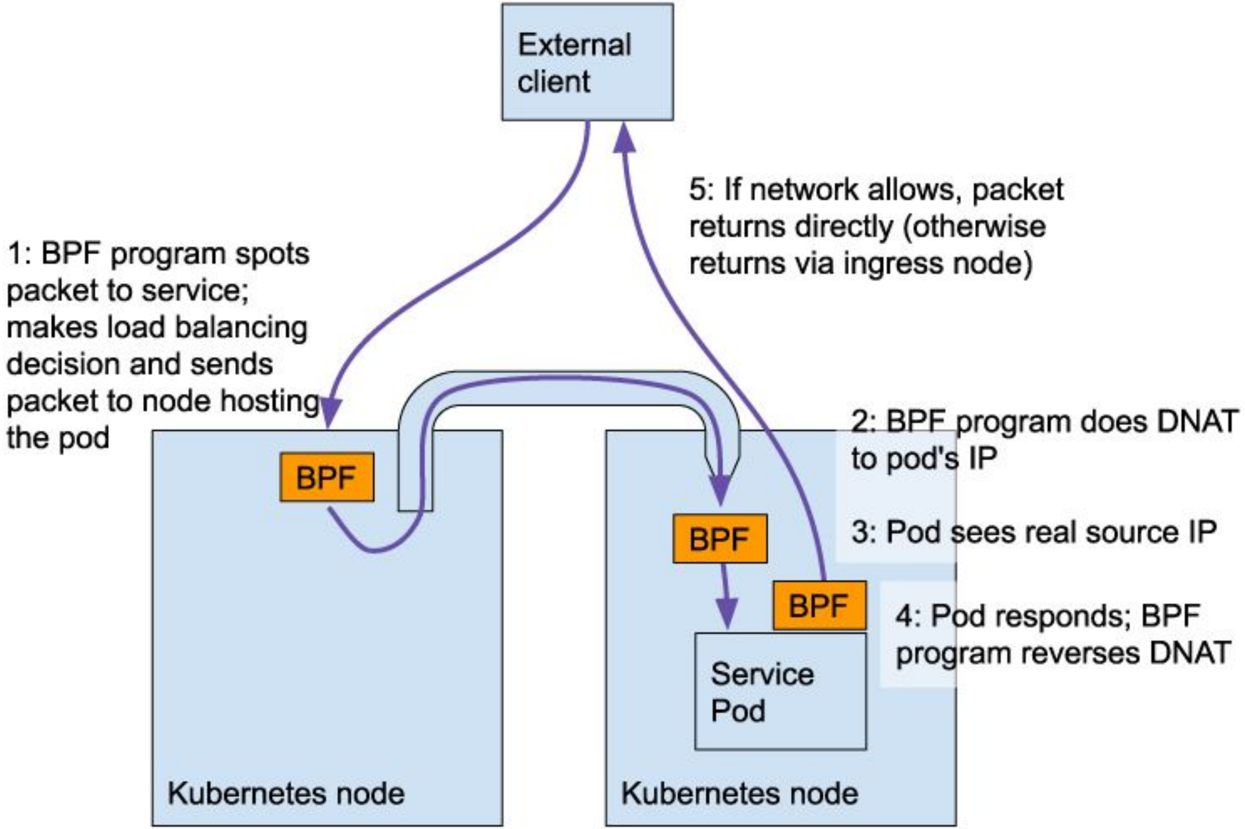
TCP Connect Time



With Kube-Proxy (non-eBPF)


External client
1: kube-proxy DNAT+SNAT replaces source IP with its own IP and dest IP with pod IP
4: Source/Dest changes reversed on the ingress node
iptables DNAT SNAT
conntrack
2: Packet forwarded to service pod.
3: Pod sees ingress node as source; responds
Service Pod
Kubernetes node
Kubernetes node

With Calico eBPF



1: BPF program spots packet to service; makes load balancing decision and sends packet to node hosting the pod

2: BPF program does DNAT to pod's IP

3: Pod sees real source IP

4: Pod responds; BPF program reverses DNAT

5: If network allows, packet returns directly (otherwise returns via ingress node)

Calico eBPF Quick check Demo

Recap of eBPF dataplane

- Alternative Calico dataplane for Linux
- Higher throughput = lower CPU/GBit
- Lower latency (especially for services)
- Preserves source IP from external clients
    - 📃Good for web server logs
    - 🎉Good for policy, can match on source IP
- kube-proxy replacement
- DSR on supported fabrics

big quality of life improvement

Check your understanding – Q1

Select all that may apply:

The key principles of K8s’ network model are:

✅ Every pod gets its own IP address

✅ Containers within a pod share the pod IP address and can communicate freely with each other

3. Pods are in the same subnet

4. Pods can communicate with other pods in the cluster directly without NAT

✅ Pods are in an overlay network

6. Network isolation is defined using network policies

✅ Pods communicate with workloads outside the cluster without NAT

Check your understanding - Q2

Select all that may apply:

Kubernetes Network Security:

☑ Assumes a flat pod network

☑ Is defined using network policies

☑ Is abstracted from the network using labels and selectors

☑ Relies in network plugins to enforce network policy

5. Relies on the capabilities of the underlying network

Check your understanding – Q3

Select all that may apply:

eBPF…:

✅ Is a mechanism for running small programs in a virtual machine within the Linux Kernel

2. Is a new way to write loadable kernel modules

✅ Is a safe way to run code in the kernel due to sandboxing

4. Is restricted to only access networking related kernel functionality

Thank you

Pick a channel

https://projectcalico.org

@projectcalico

https://github.com/projectcalico/community

https://slack.projectcalico.org

https://discuss.projectcalico.org

TIGERA

Follow us on: