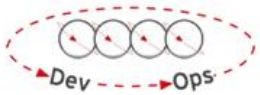
Cloud Native Networking with FD.io/VPP

Frank Brockners
Distinguished Engineer, Chief Technology and Architecture Office, Cisco

March 17, 2018

The way Applications are developed and deplo changed…..

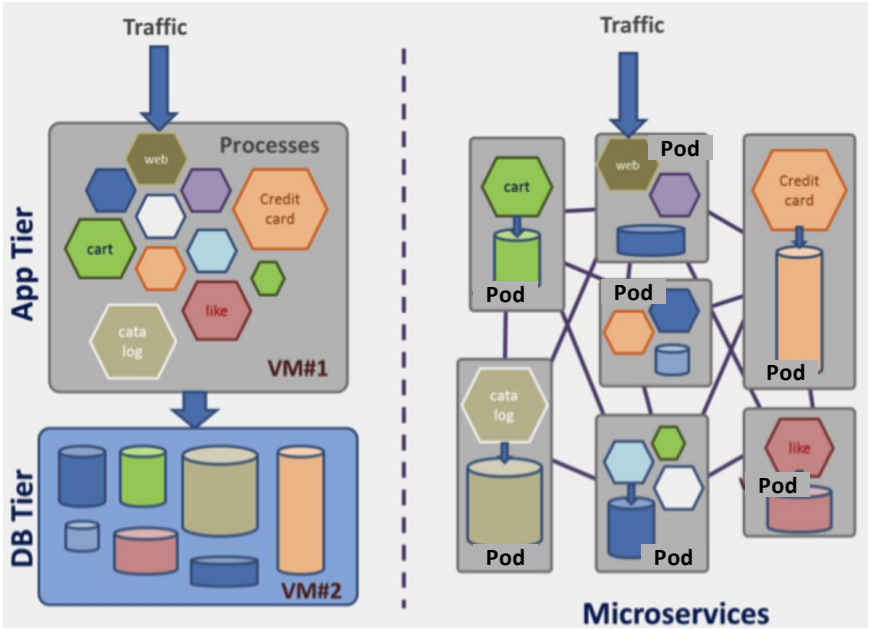
Development Process	Application Architecture	Deployment & Packaging	Application Infrastructure
Waterfall	Monolithic	Physical Servers	Datacenter
Agile	N-Tier	Virtual Servers	Hosted
DevOps	**Microservices**	**Containers**	**Serverless / FaaS**



Microservices & Containers have changed many things..

Applications are being developed and deployed very differently today

- Microservices allow you to split an application into many modular pieces, the network is how you stitch the pieces back together.
- The interconnection of the pieces results in a more complex application network which consumes lots of resources
- The performance of the cloud native network is crucial to the behavior of the overall application.

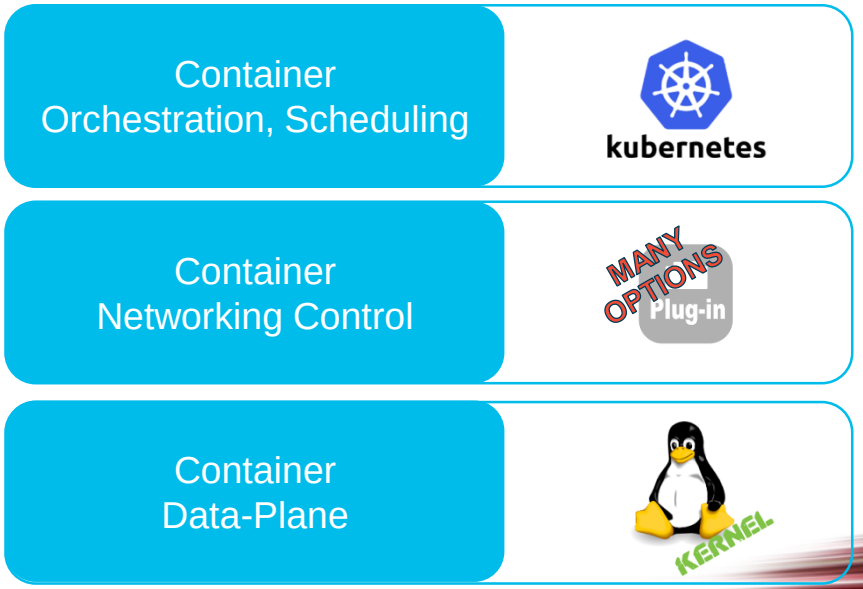


It’s crucial we get ”Container Networking” right! Lets not get “Openstacked”

What Container Network Stacks Provide Today.

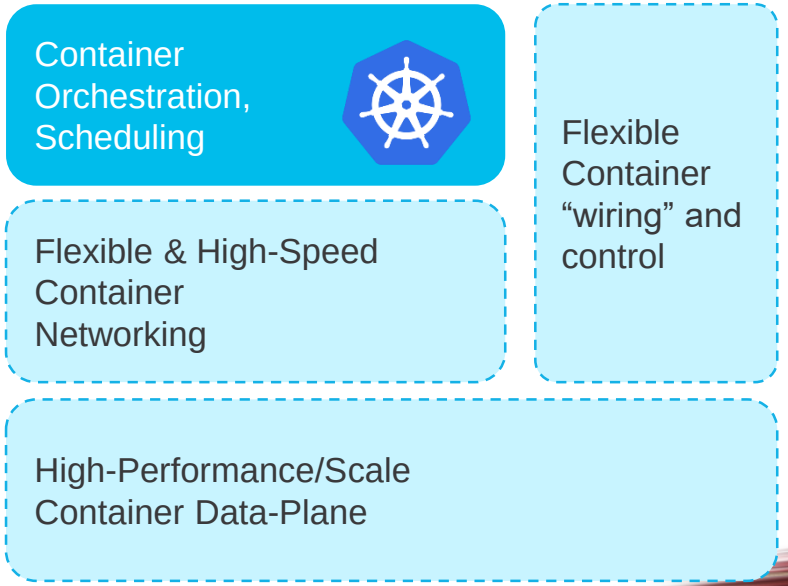
- Lifecycle management for application Containers
- Overlay connectivity for application Containers:
  - NAT communication with external world
  - Policy controlled overlay, may extend policy control to DC fabric in some cases
  - Network policy addresses security/connectivity
  - Designed for Data Center applications / use cases

Good start, but not sufficient for NFV use cases (Firewalls, Customs VNFs)



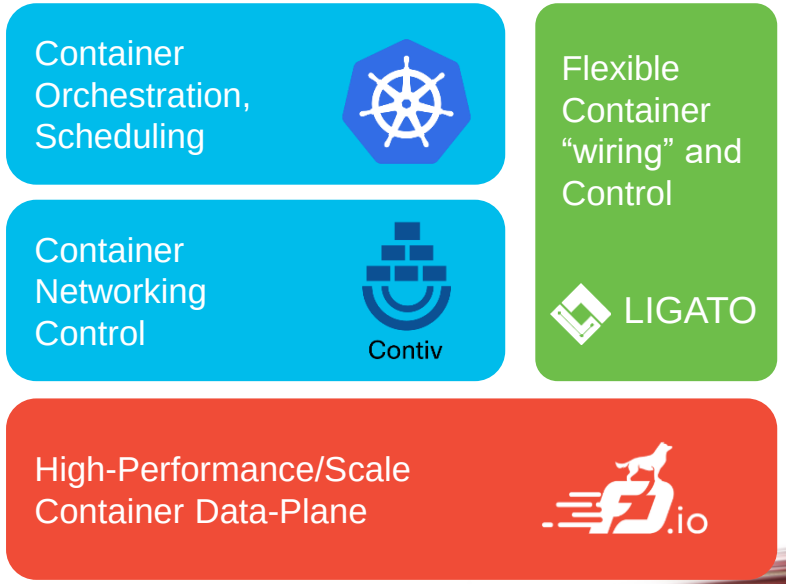
What Container Networks Stacks Lack for NFV Use Cases:

- NVF-specific policy APIs (e.g. QoS, placement considering network resources, etc.)
- Networking:
  - NAT is a bottleneck, not suitable for NFV use cases
  - VNFs often require more than 1 interface / IP address
  - No support for high-speed wiring of NFVs:
    - To the outside world; To application containers; Between NFV containers; Creation of Service Function Chains (mixed physical & virtual)
- Management/Control:
  - Containerized NVFs not really in the data plane (except for the vswitch)
  - No support for cloud-native, high-performance NVFs
- Forwarding:
  - Kernel or OVS used for forwarding



Cloud-Native Network Function Needs: Summary

- Container-based
- Container stack lifecycle (same as application containers)
- 12-factor app design for management/control
- High-performance forwarding
- High-performance networking
- Seamless virtual/physical world interworking
- Common policy control with application containers (as much as possible)
- Must solve 3 key problems:
  - Lifecycle management
  - High-Performance Networking (wiring/config, operations)
  - Easy installation, operation – policy, security



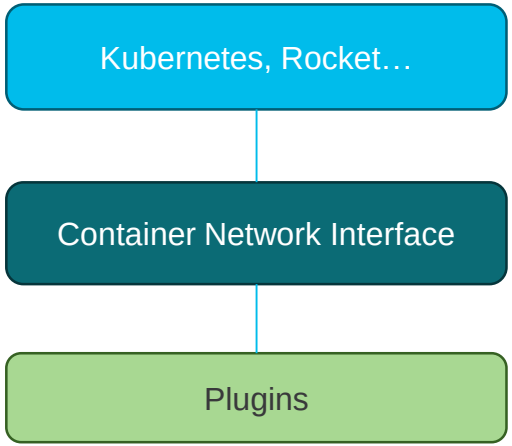
Container Networking moving from Kernel to Userspace

- Userspace enables rapid upgradability, highly available (doesn't bring down node), no system call overhead, no dependency on Linux kernel networking community for features, higher performance and scale
- FD.io (dataplane), DPDK (network), SPDK (Storage) are examples
- Cloud Native apps are all connected by the network – lots of network end points to be managed, userspace offers lower overhead and higher performance
- Meltdown/Spectre bugs add a new tax for kernel networking

Kubernetes Networking – Key Communication patterns

4 distinct communication patterns:

Highly-coupled container-to-container communications	→	Pod concept and localhost communications
Pod-to-Pod communications	→	Network Plugins
Pod-to-Service communications	→	Kubernetes services concept
External-to-Service communications	→	Kubernetes services concept



Kubernetes Network Plugins

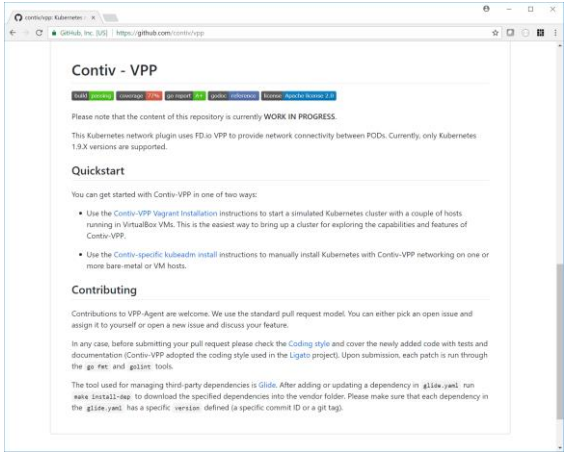
- Kubernetes assumes seamless connectivity between pods, wherever it decides to place them. A networking plugin is needed to abstract the network
- Kubernetes fundamental requirements for any networking implementation
  - all containers can communicate with all other containers without NAT
  - all nodes can communicate with all containers (and vice-versa) without NAT
  - the IP that a container sees itself as is the same IP that others see it as
- Network Plugins provide various levels of sophistication – from simple to highly feature rich and performant

Examples

- Cilium
- Contiv
- Contrail
- Flannel
- Google Compute Engine (GCE)
- Kube-router
- L2 networks and linux bridging
- Multus
- NSX-T
- Nuage Networks VCS
- OpenVSwitch
- OVN (Open Virtual Networking)
- Project Calico
- Romana
- Weave Net from Weaveworks
- CNI-Genie from Huawei

Container Networking Control: Contiv-VPP https://github.com/contiv/vpp

- Contiv is a networking plugin for Kubernetes that:
  - Allocates IP addresses to Pods (IPAM)
  - Programs the underlying infrastructure it uses (Linux TCP/IP stack, OVS, VPP, …) to connect the Pod's to other Pods in the cluster and/or to the external world.
  - Implements K8s network policies that define which pods can talk to each other.
  - Implements K8s services; a service exposes one or more (physical) service instances implemented as K8s pods to the other pods in the cluster and/or to external clients as a virtual instance (e.g. as a virtual "service" IP address).
- Contiv is a user-space based, high-performance, high-density networking plugin for Kubernetes. Contiv-VPP leverages FD.io/VPP as the industry's highest performance data plane



Ligato – Accelerate Cloud-Native Development

Platform and code samples for development of cloud native VNFs

LIGATO provides a mechanism for delivering and managing agents for cloud-native Network Functions to enable them to become part of the application service topology - all in user-space.

Components:

- Cloud-Native Infrastructure – a Golang platform for building cloud-native microservices
- VPP-agent - Golang implementation of a control/management plane for FD.io/VPP based cloud-native Virtual Network Functions (VNFs)
- SFC-Controller - Service Function Chain (SFC) Controller for stitching virtual and physical networks
- BGP-Agent - BGP Agent is a BGP information provider

https://ligato.github.io/

Network Functions can now be delivered as containers

- Take advantage of the low overhead of containers to deliver higher performance.
  - Using cloud native technologies to build the Network Functions so they run in the same network and user-space as the applications.
- Network functions become part of the service topology
  - Network Functions are truly just another service and can be developed/deployed using the same tools as the applications with the same velocity.
- Leverage cloud-native development & deployment velocity – #devployfaster
- Eliminates the overlay tax for services

Putting it all together

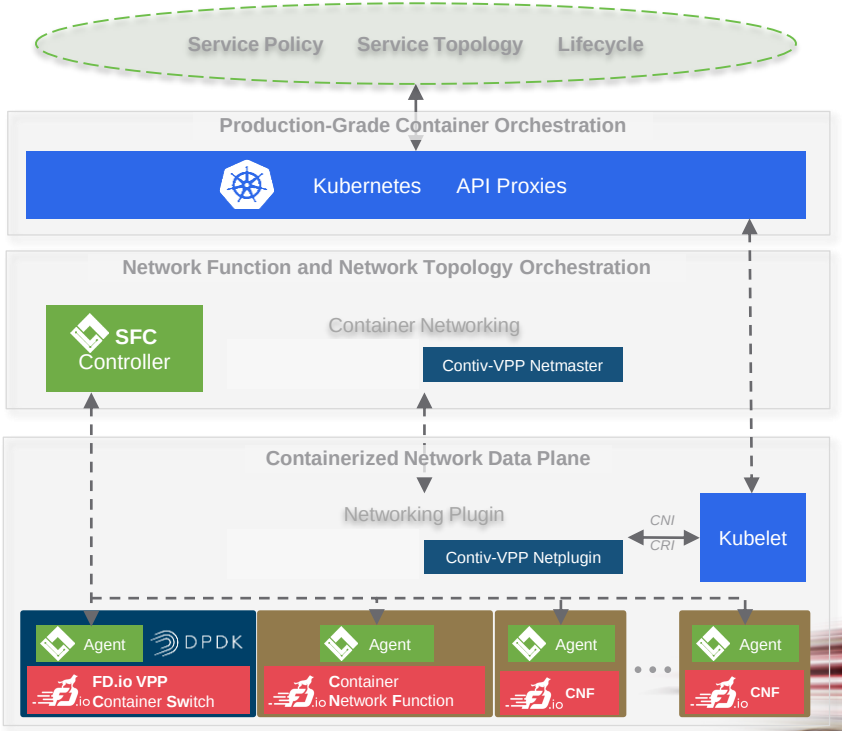
Enabling Production-Grade Native Cloud Network Services at Scale

kubernetes — Production-Grade Container Orchestration

Contiv — Performance-Centric Container Networking

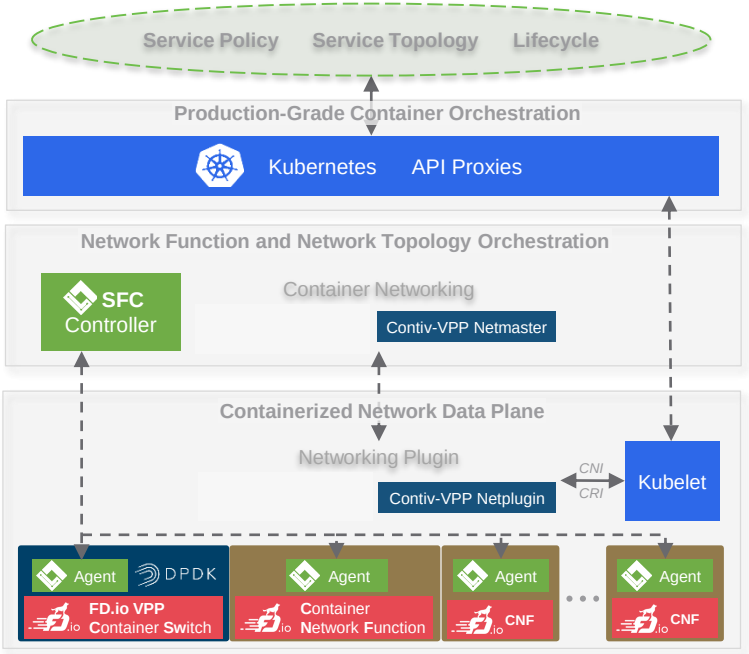
LIGATO — Cloud-native NF Orchestration
Cloud-native NF Agent platform

fd.io — Containerized Fast Data Input/ Output

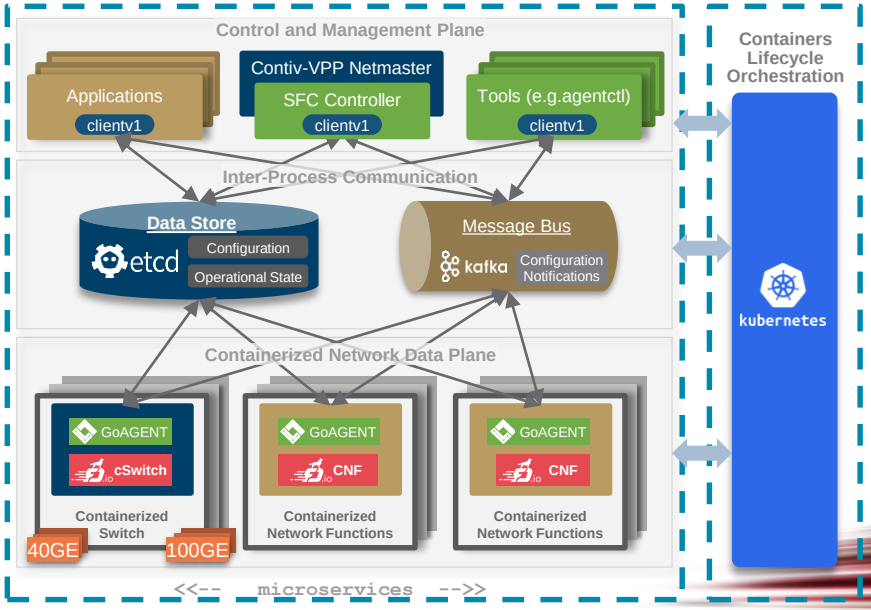


Cloud-native Network Micro-Services Putting It All Together Now – The System Design

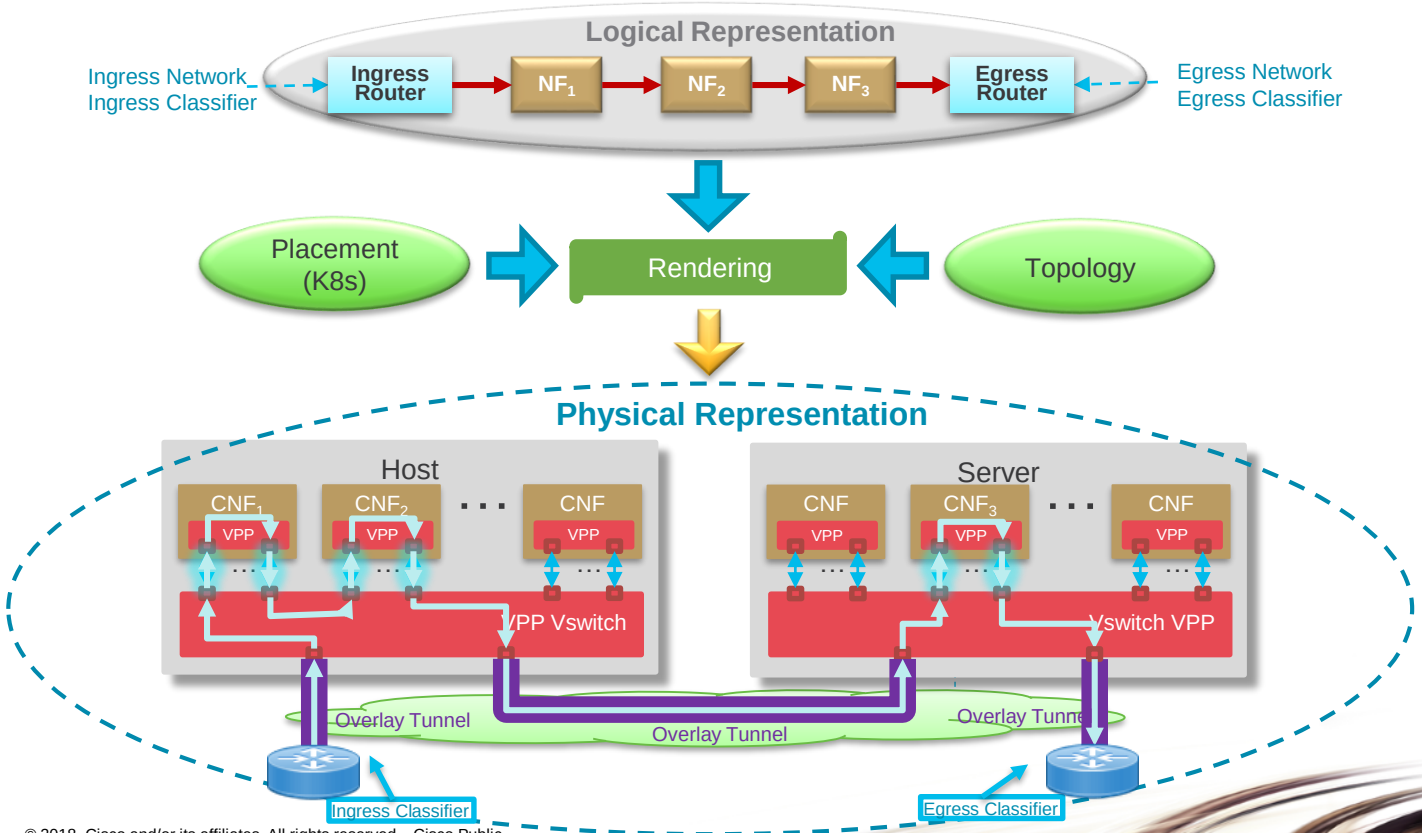
Functional Layered Diagram



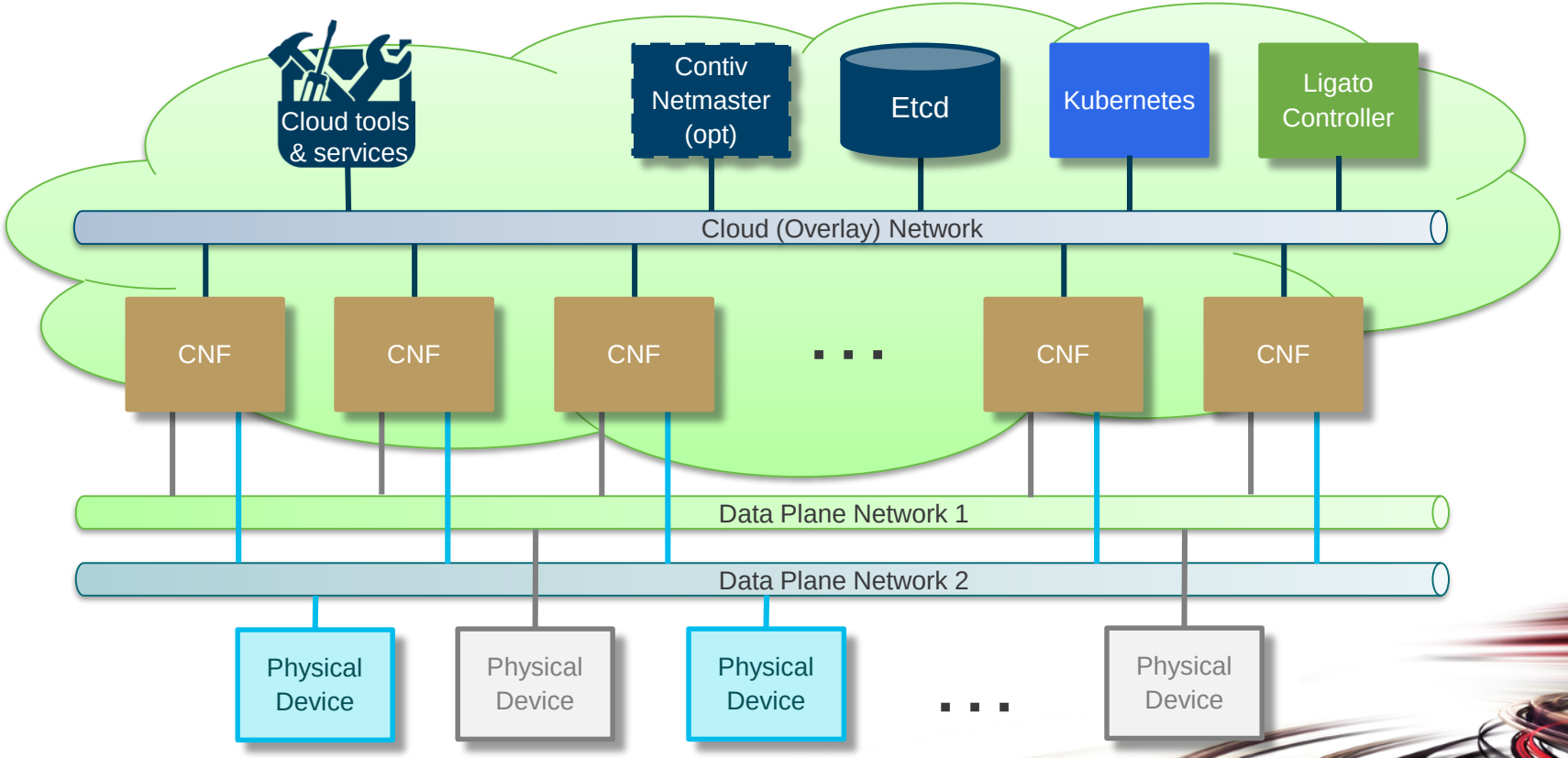
System Components



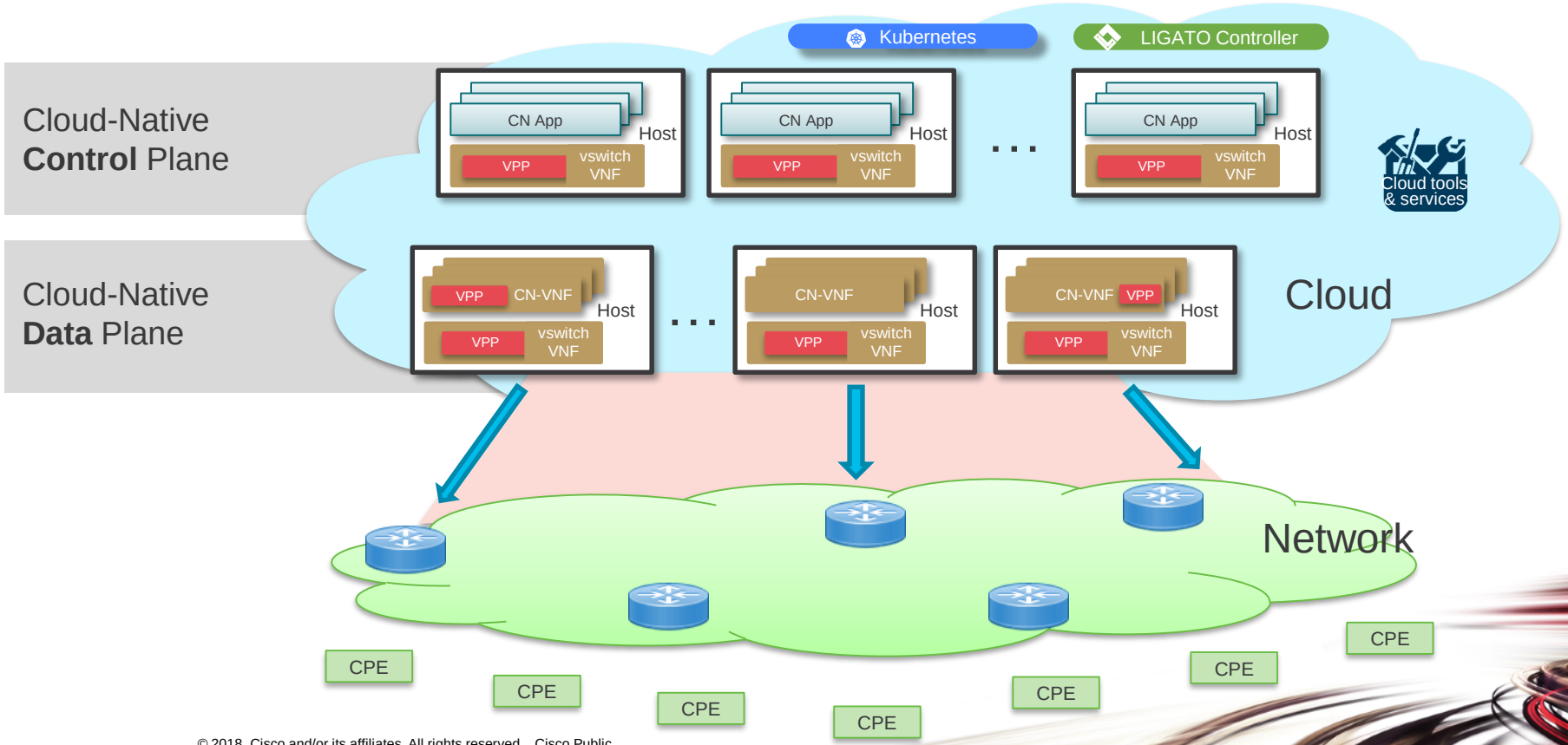
Network Micro-Service Use Case: Service Function Chaining with Cloud-Native NFs



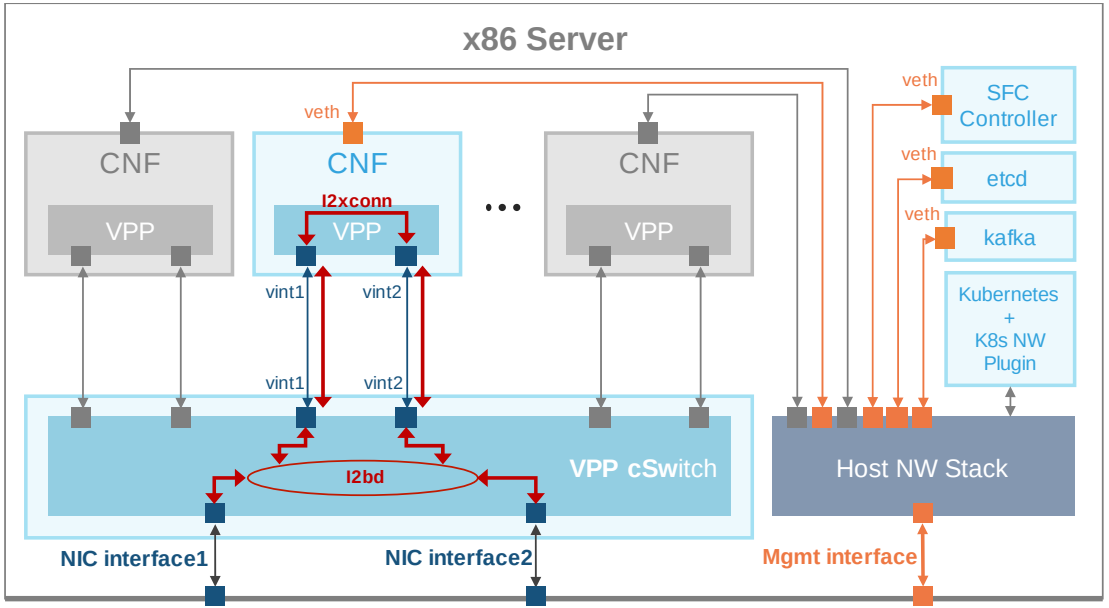
Example: Distributed, multi-cloud CUPS architecture with Cloud Native Network Functions



Use Case: Cloud-Native CUPS Architecture
Edge compute application and network service orchestration



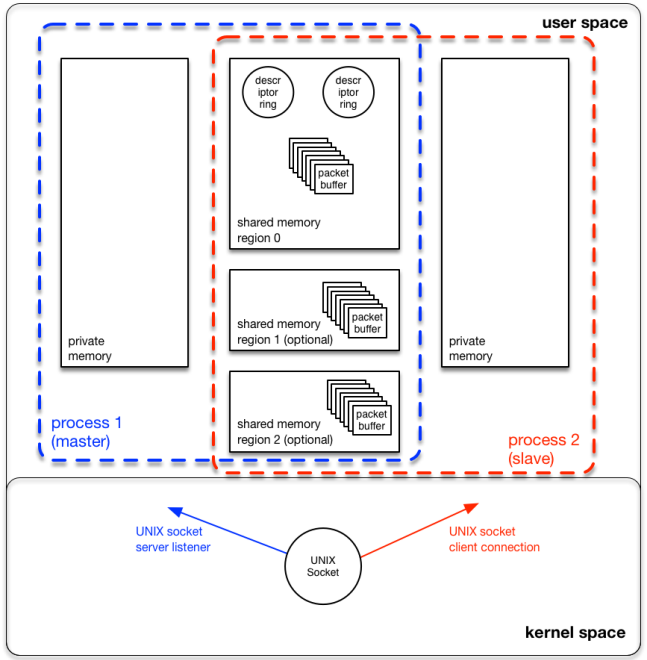
A glimpse at performance: Example topology



Topology: Containerized Switch with one or more Containerized Network Functions

memif: Shared Memory Packet Interface for VPP

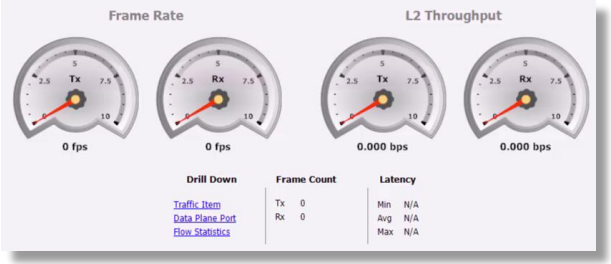
- Packet based shared memory interface for user-mode application
- Container friendly (no privileged containers needed)
- Polling and interrupt mode operation:
  - Interrupts simulated with linux eventfd infrastructure
  - Support for interrupt masking in polling mode
- 3rd-party library - allows easy creation of applications which communicate over memif
  - vpp-to-vpp, vpp-to-3rd-party and 3rd-party-to-3rd-party operation
- Support for multiple queues (incl. asymmetric configurations)
- Jumbo frames support (chained buffers)
- Secure



Example Benchmark: memif 10x faster than veth

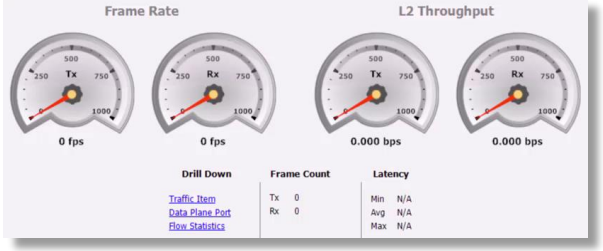
Memif – cSwitch 1core

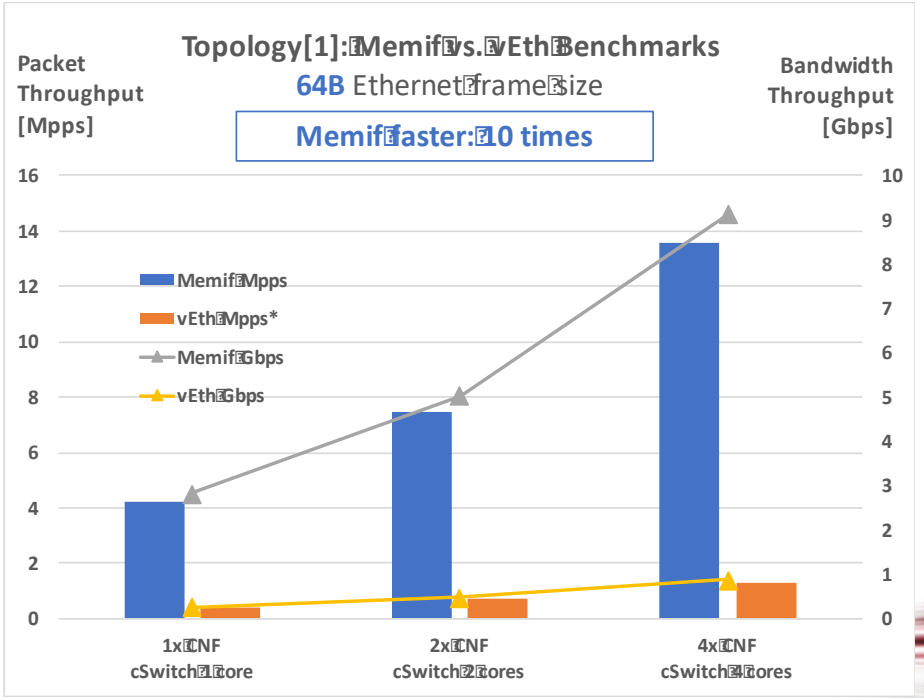
64B Ethernet frame size – *No Frame Loss*



vEth – cSwitch 1core

64B Ethernet frame size – *No Frame Loss*





One more thing…

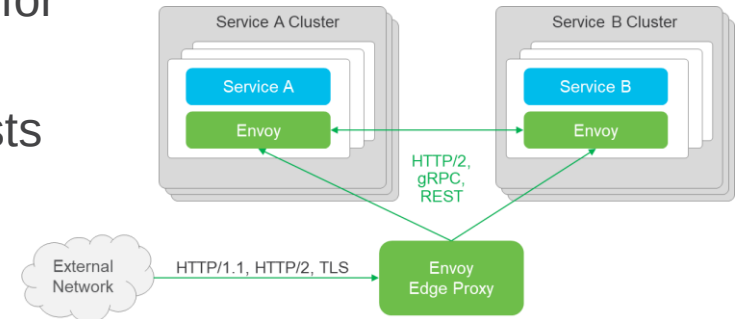
Consistent Network & Security Control across cloud environments

Application Driven Service Mesh to enable Policy enforcement & Security

- Agents deployed as part of the app code to enable policy and security enforcement at the app level
- Provides a way to express application security consistently to support mutli-cloud application deployments
- Enterprise can get back consistent Networking & control when leveraging cloud-native and multi-cloud technologies Security Controls

Istio & Envoy

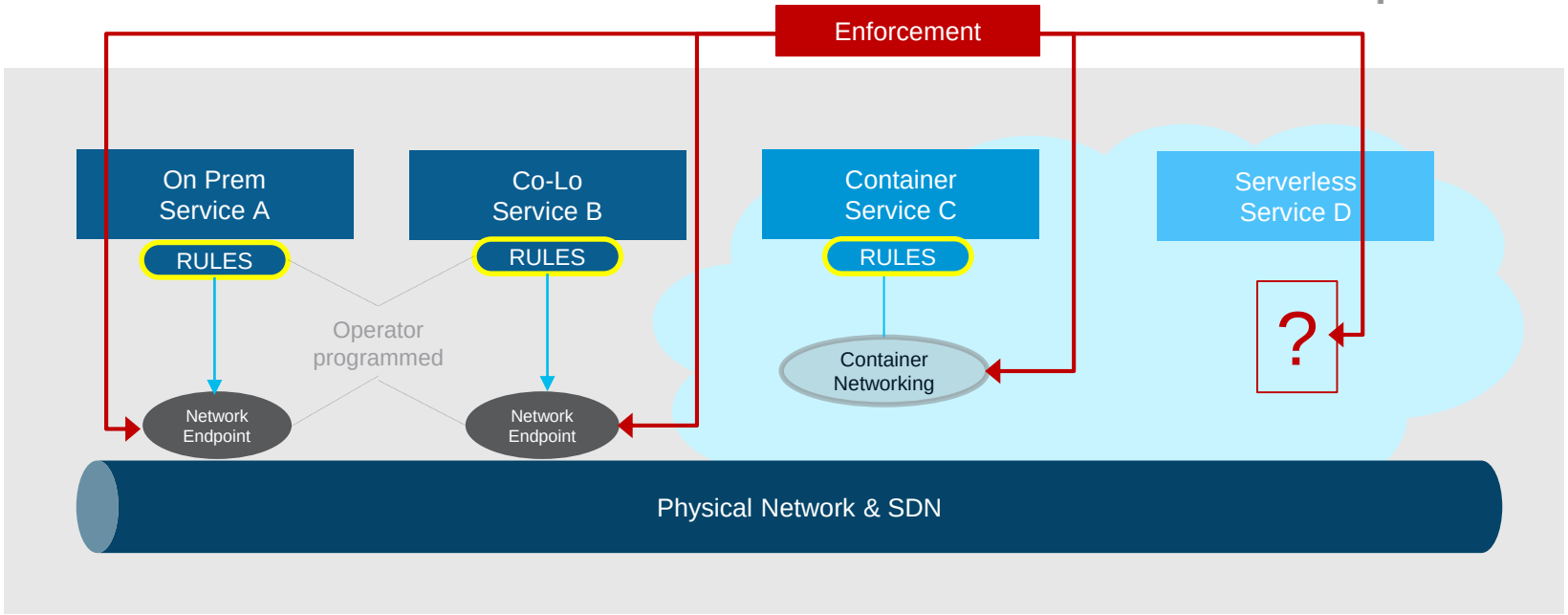
- Istio provides a dedicated infrastructure layer for handling service-to-service communication
- Responsible for the reliable delivery of requests through the complex topology of services that comprise a modern, cloud native application
- Deployed as a sidecar container with the application
- Platform agnostic, runs where ever the containers run
- Envoy agent in the sidecar provides, routing, load balancing, traffic steering, stats collection, flow data – leveraging VPP for high performance



Istio Service Mesh with Envoy Agent

- Intelligent routing & Load Balancing
- Resilience Across Languages and Platforms
- Fleet-Wide Policy Enforcement
- In-Depth Telemetry and Reporting

Multi-cloud environments are extremely challer



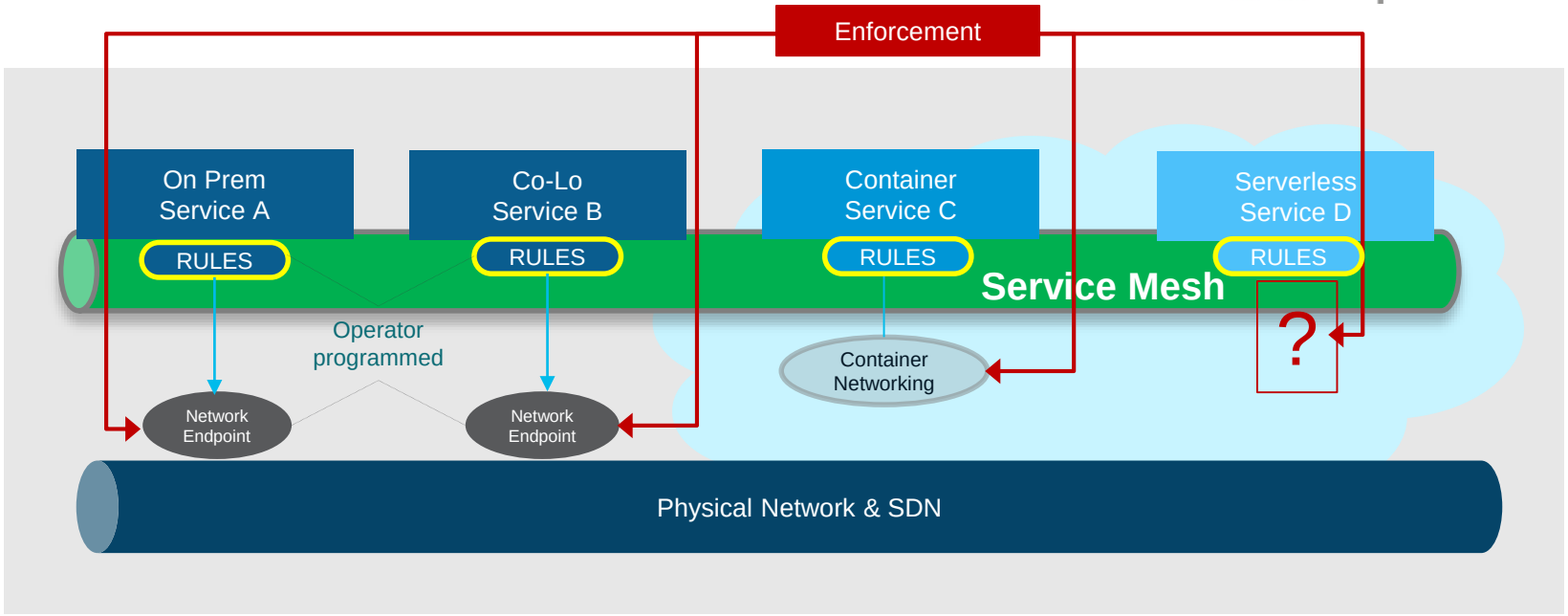
Rules programmed by Network Operator

Rules programmed by DevOps

No controls available

Public Cloud & Serverless - Networking & Security controls are limited/non existent

Application Driven Service Mesh



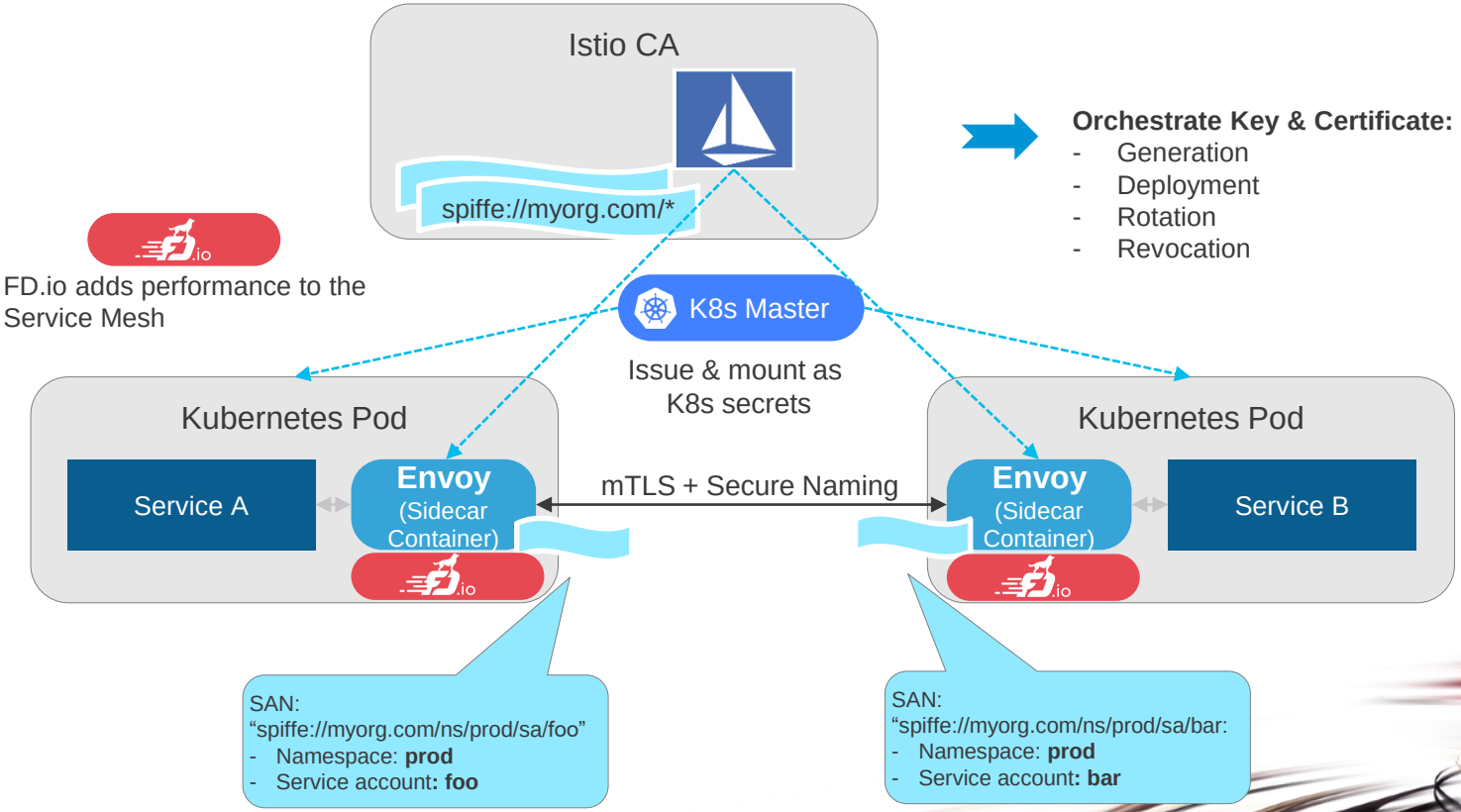
Rules programmed by Network Operator

Rules programmed by DevOps

No controls available

Agents deployed part of the app –
Enterprise get back consistent Networking & Security Controls

Security at Scale with Istio



Complete Cloud Native Network Stack

kubernetes

Production-Grade Container Orchestration

Contiv

Performance-Centric Container Networking

LIGATO

Cloud-native Network Function Orchestration

.io

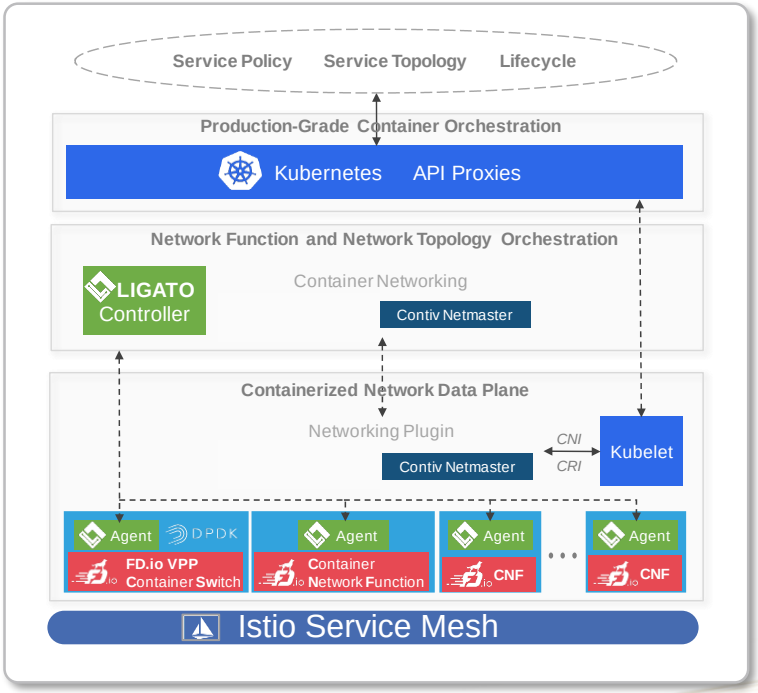
Containerized Fast Data Input/ Output

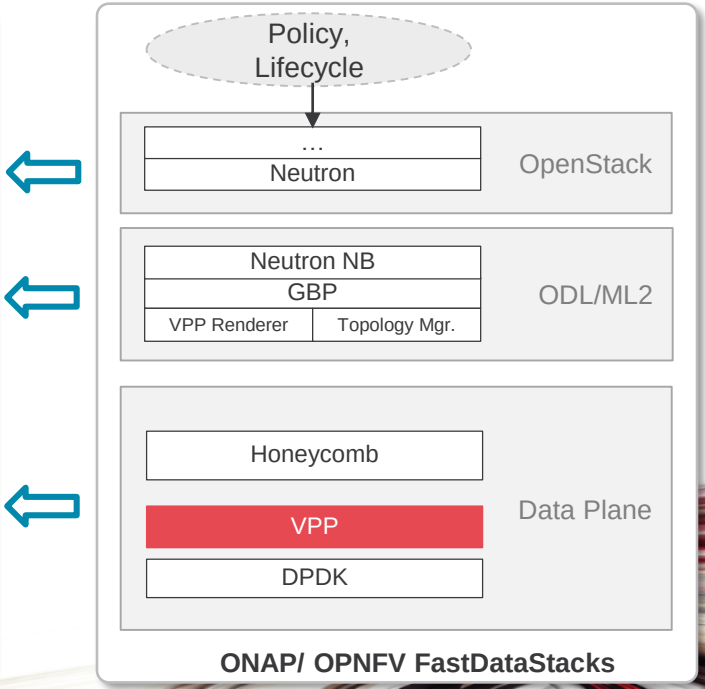
intel

IT大咖说 知识共享平台

Application Service Mesh

Enabling Production-Grade Native Cloud Network Services at Scale





What's left to do ?

- CNFs: Need to have cloud-native network functions in order to meet the needs of cloud-native applications. Kernel networking is not appropriate for cloud-native environments.
- Userspace Networking: Finish getting the kernel out of the way with userspace host stacks
- Userspace/Cloud Native Storage: Storage needs to start down the Userspace/Cloud Native road networking is walking: SPDK/Ceph/VPP integration
- Unified IO: Networking/Storage: Start thinking of blocks/packets as interchangeable units of IO with well integrated processing paths.
- Security Policy Orchestration: Integrate Application Driven security into the development toolchain for DevOps.

Summary

Production-Grade Container Workload Scheduling and Orchestration

Secure overlay network with granular policy control using a “sidecar” (L7 proxy)

Performance-Centric Container Networking

Cloud-native NF Orchestration
Cloud-native NF Agent platform

Data-Plane: Containerized Fast Data Input/ Output

Towards an orchestrated multi-cloud from edge to centralized and on-Prem to Co-lo to public cloud

Thank you

References

- FD.io/VPP
  - https://fd.io/
  - memif:
    - https://docs.fd.io/vpp/17.10/libmemif_doc.html
    - https://docs.google.com/presentation/d/1KrpOUUD7oHML6Plge_4E3-oq5YBun5T8pfoDiCBVk74
- Contiv-VPP
  - https://github.com/contiv/vpp
- Ligato
  - https://ligato.github.io/