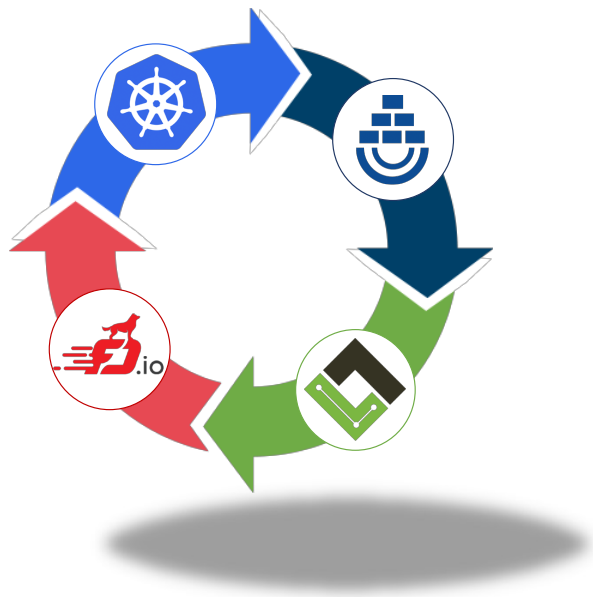
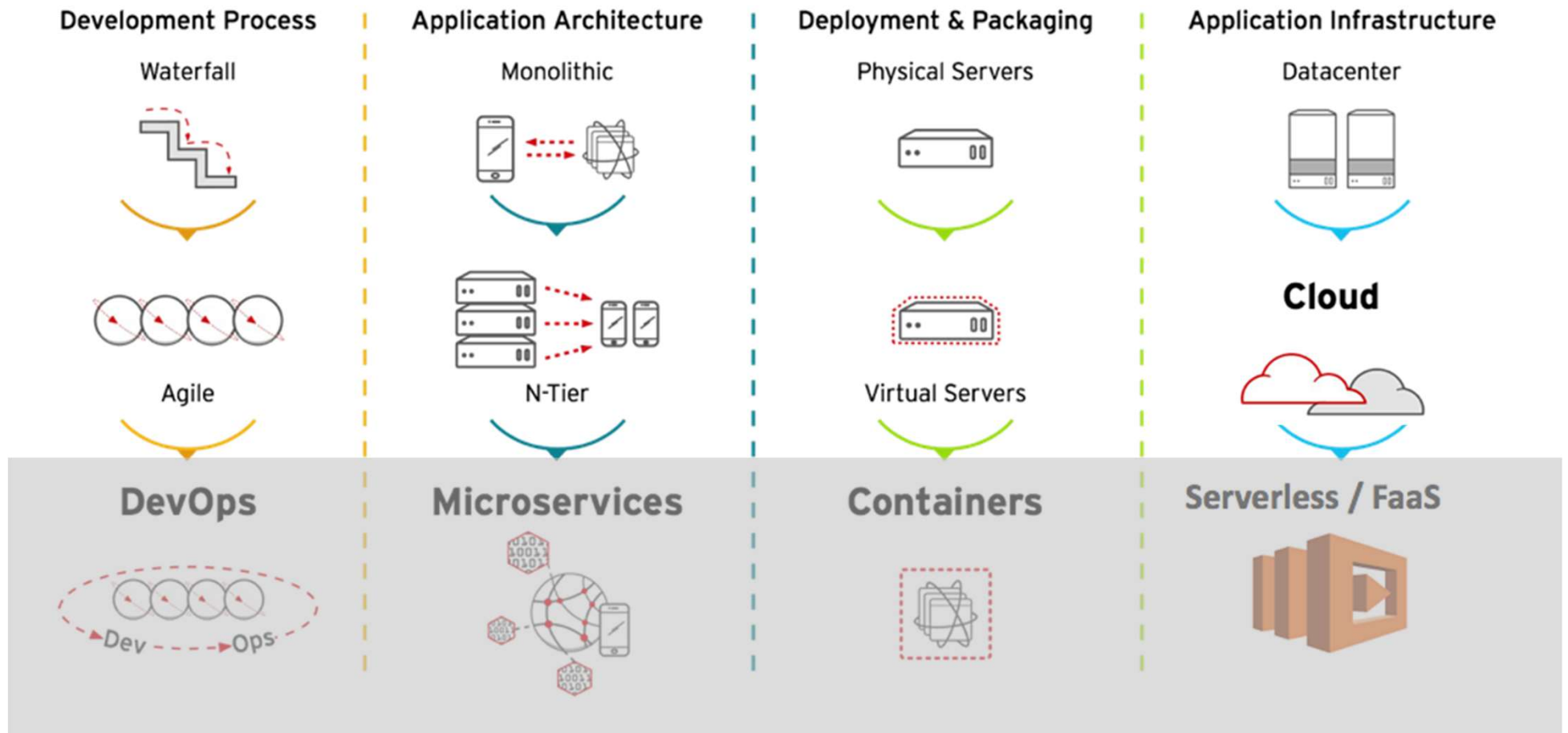


Ligato

A Platform for Development of Cloud-Native VNFs



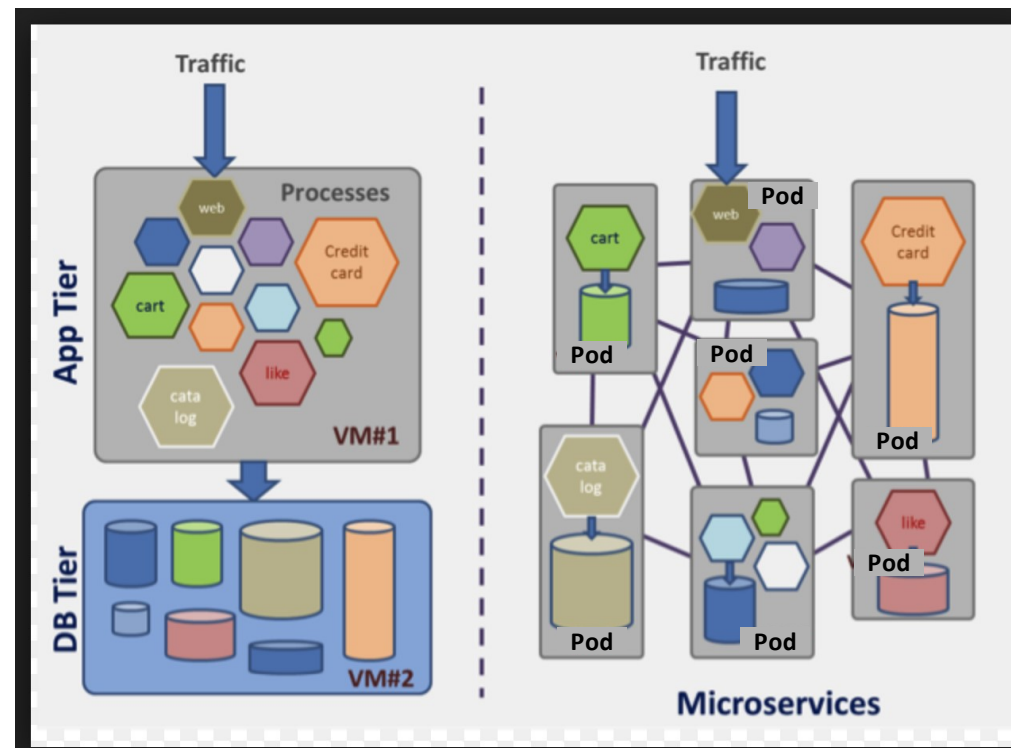
The way Applications are developed & deployed... has changed.....



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.

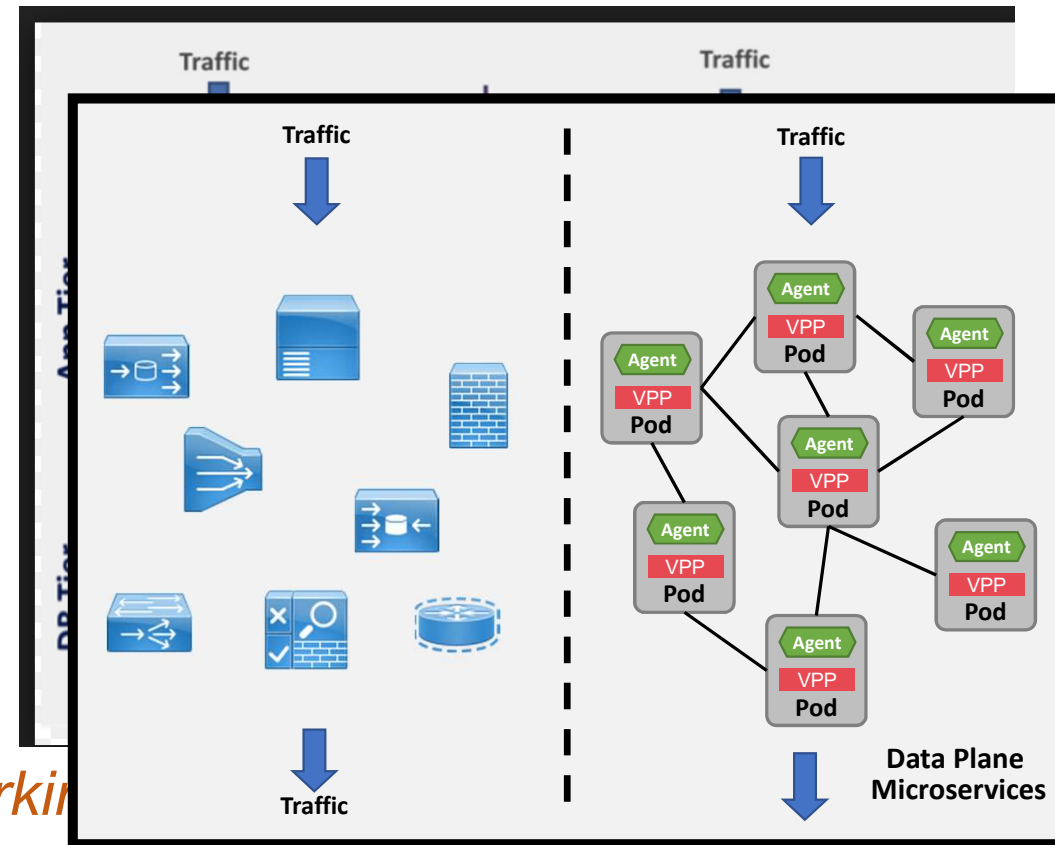


It's crucial we get "Container Networking" right!
Lets not get "**Openstacked**"

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"
Let's not get "**Openstacked**"

Solution #1

Move Cloud Native Networking out of the Kernel to Userspace

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

Solution #1

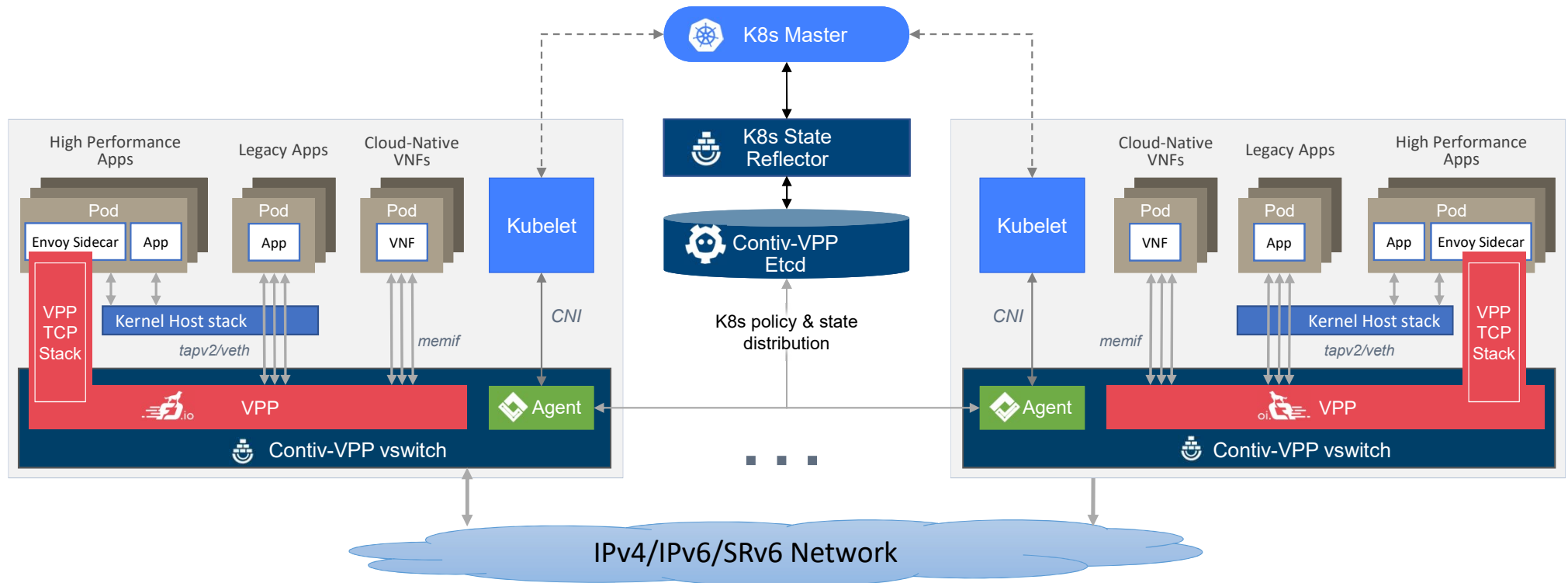
Kubernetes & Contiv-VPP

Contiv-VPP

- Kubernetes assumes seamless connectivity between pods, wherever it decides to place them. A networking plugin is needed to abstract the network
- 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 Pods 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 - leveraging FD.io/VPP as the industry’s highest performance data plane

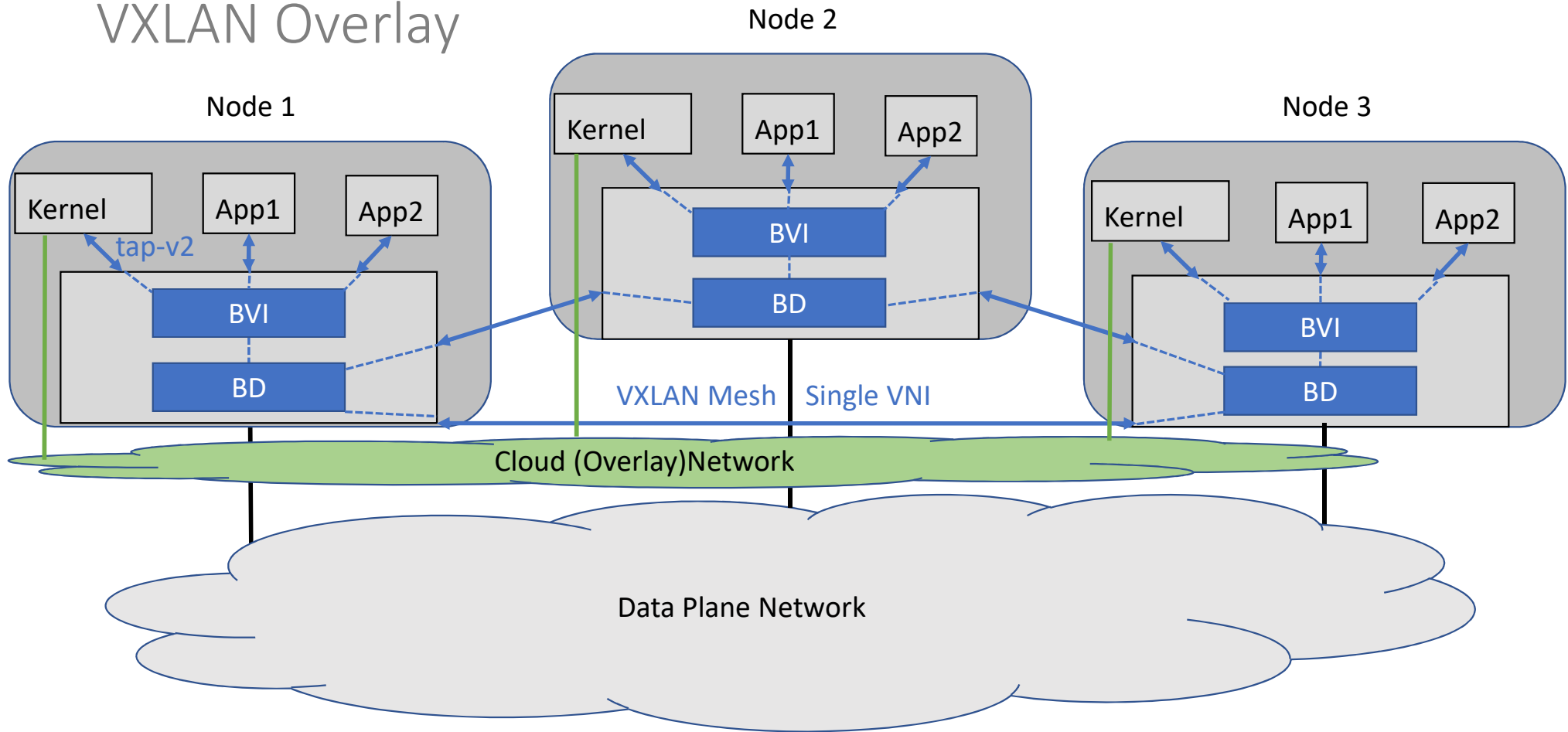
Contiv-VPP Architecture

- Can deliver complete container networking solution entirely from userspace
- Replace all eth/kernel interfaces with memif/userspace interfaces.
- Apps can add VCL library for Higher Performance (bypass Kernel host stack and use VPP TCP stack)
- Legacy apps can still use the kernel host stack in the same architecture

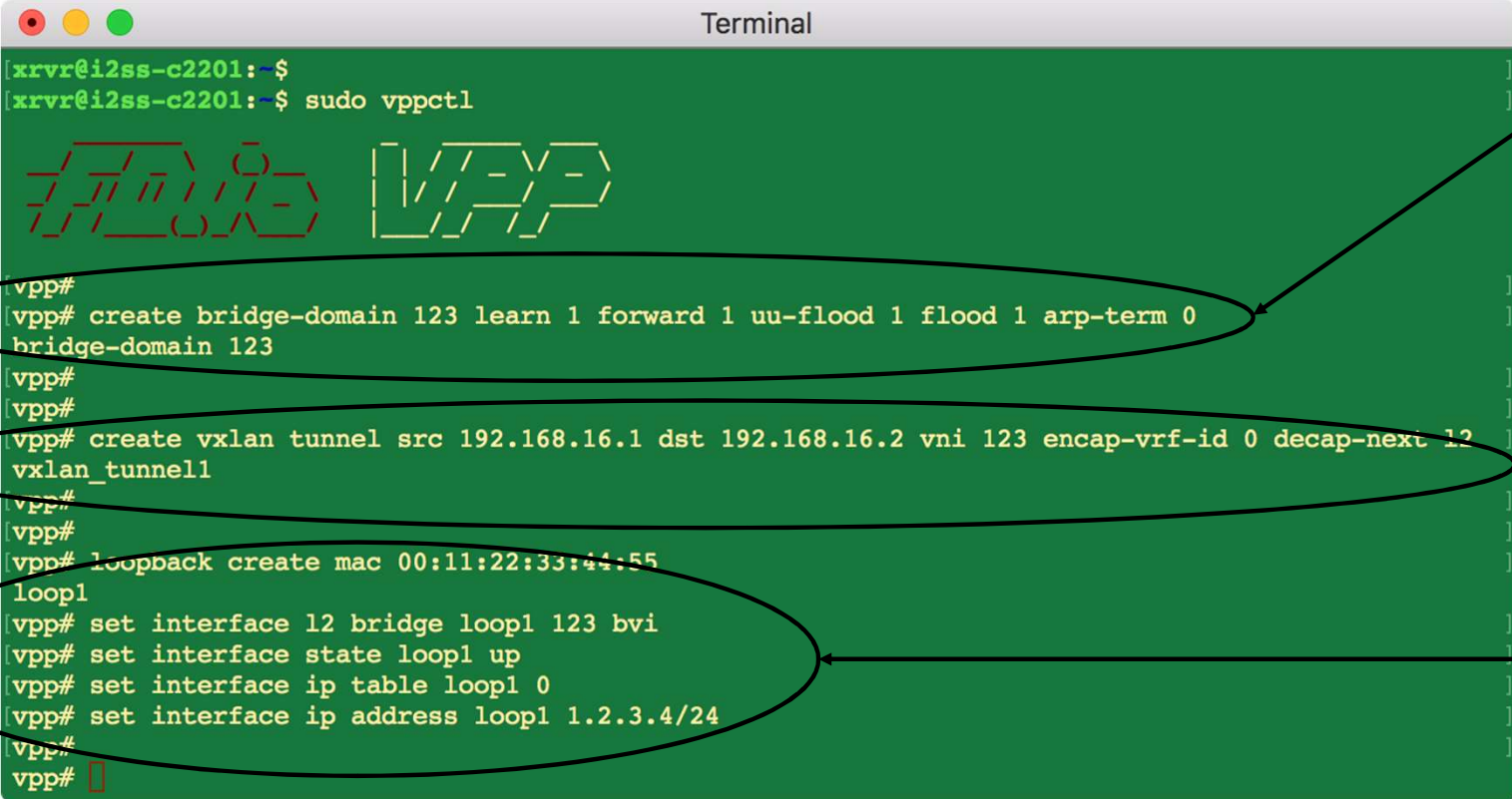


Contiv-VPP Rendering

VXLAN Overlay



Creating BD with BVI



```
Terminal
xrvr@i2ss-c2201:~$
xrvr@i2ss-c2201:~$ sudo vppctl

vpp#
vpp# create bridge-domain 123 learn 1 forward 1 uu-flood 1 flood 1 arp-term 0
bridge-domain 123
vpp#
vpp#
vpp# create vxlan tunnel src 192.168.16.1 dst 192.168.16.2 vni 123 encap-vrf-id 0 decap-next 12
vxlan_tunnel1
vpp#
vpp#
vpp# loopback create mac 00:11:22:33:44:55
loop1
vpp# set interface 12 bridge loop1 123 bvi
vpp# set interface state loop1 up
vpp# set interface ip table loop1 0
vpp# set interface ip address loop1 1.2.3.4/24
vpp#
vpp#
```

Create BD

Create VXLAN Tunnel
(one per rmt node)

Create BVI

Solution #2

Cloud-Native VNFs

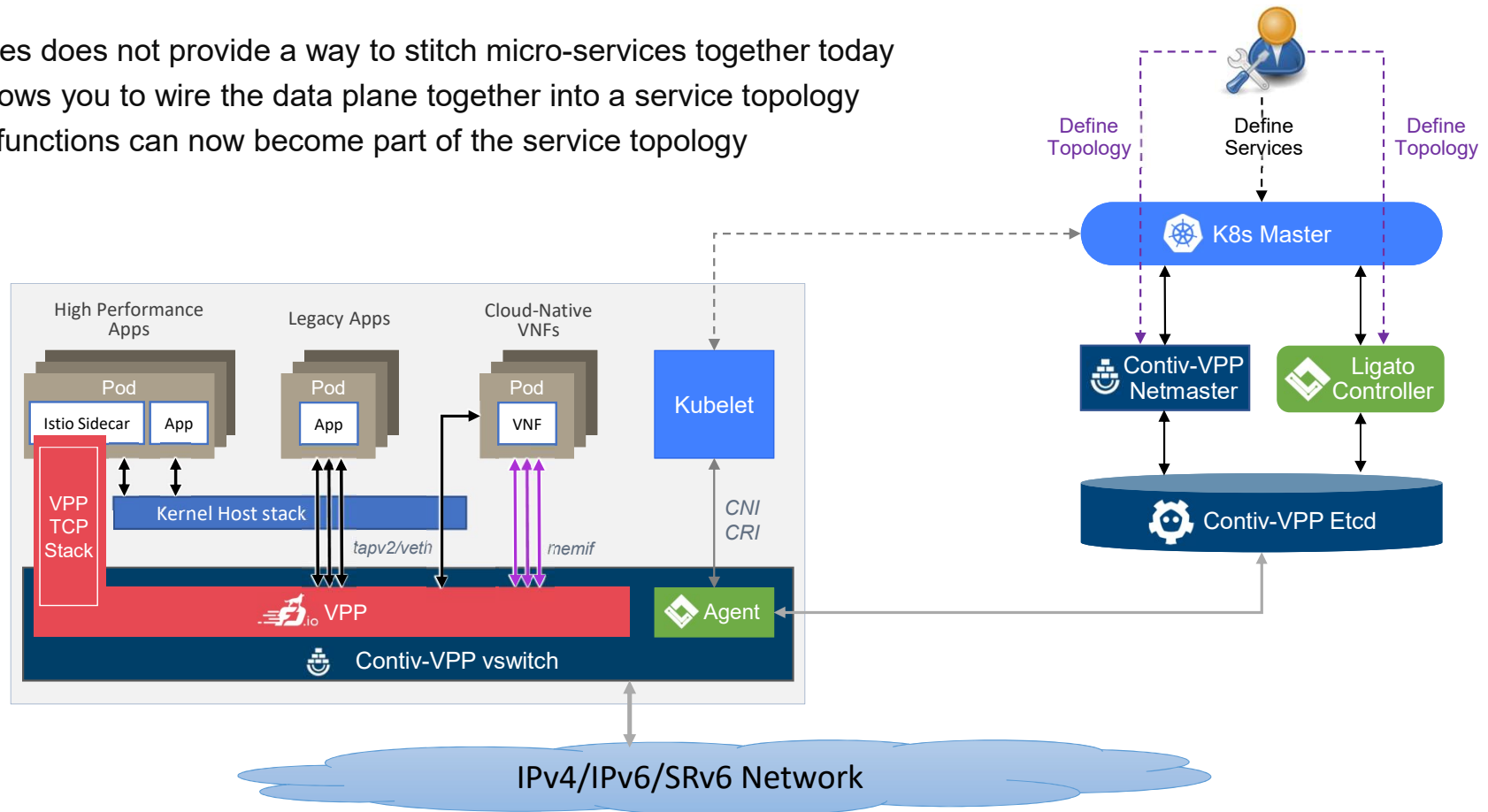
What Container-Networking Lacks for NFV Use-Cases:

- NFV-specific policy APIs (e.g. QoS, placement considering network resources)
- Networking:
 - HTTP or NAT-based load balancing isn't suitable for NFV use-cases
 - No support for high-speed wiring of NFs:
 - To the outside world
 - To application containers
 - Between NFV containers
 - Creation of Service Function Chains (mixed physical and virtual – virtual a mix of VM and container)
- Management/Control:
 - Containerised NFs not really in the data plane (except for the vSwitch)
 - No support for cloud-native, high-performance NFs
- Forwarding:
 - Kernel used for forwarding – not sufficiently performance orientated (except for Contiv-VPP!)

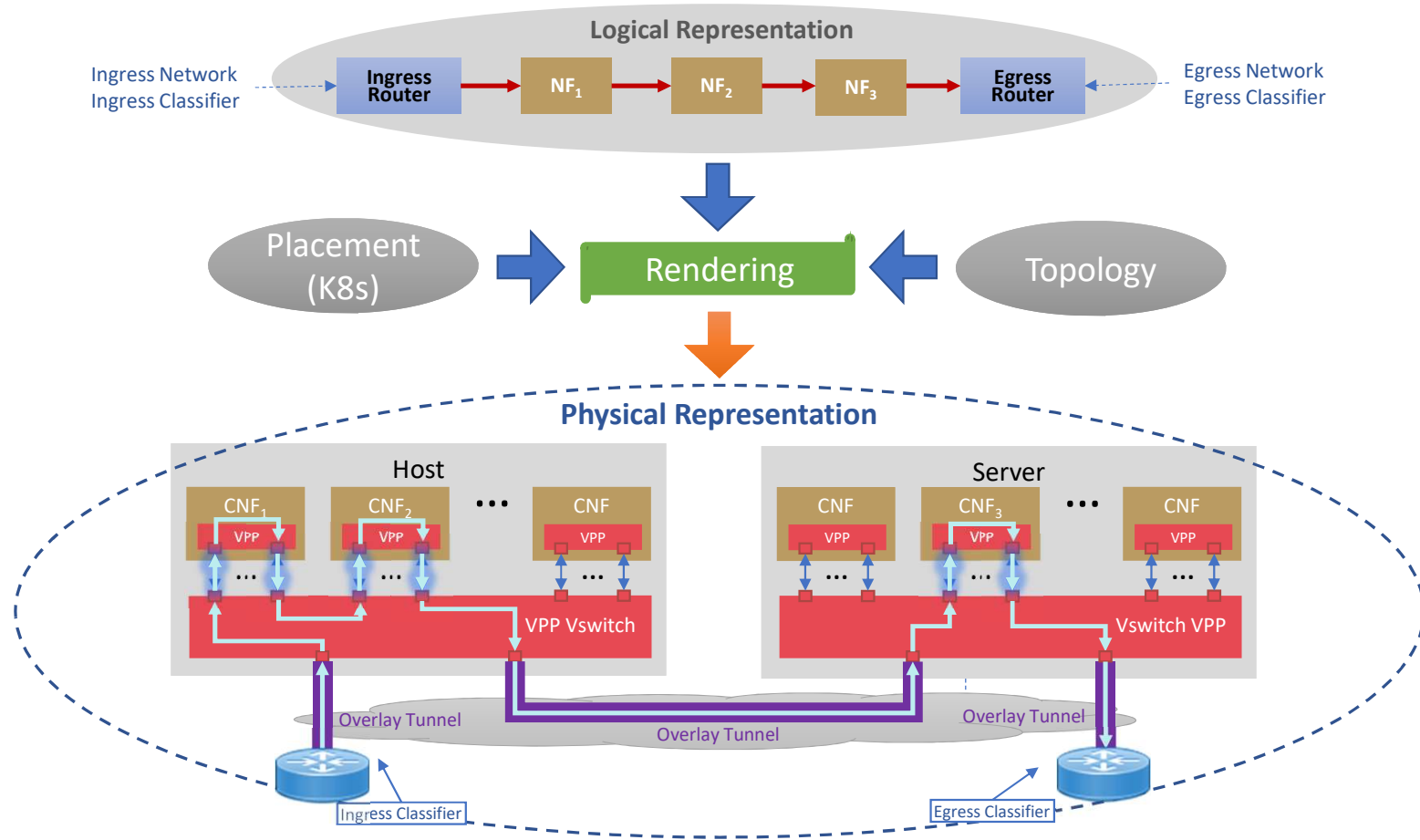
Solution #2

Cloud-Native VNFs

- Kubernetes does not provide a way to stitch micro-services together today
- Ligato allows you to wire the data plane together into a service topology
- Network functions can now become part of the service topology

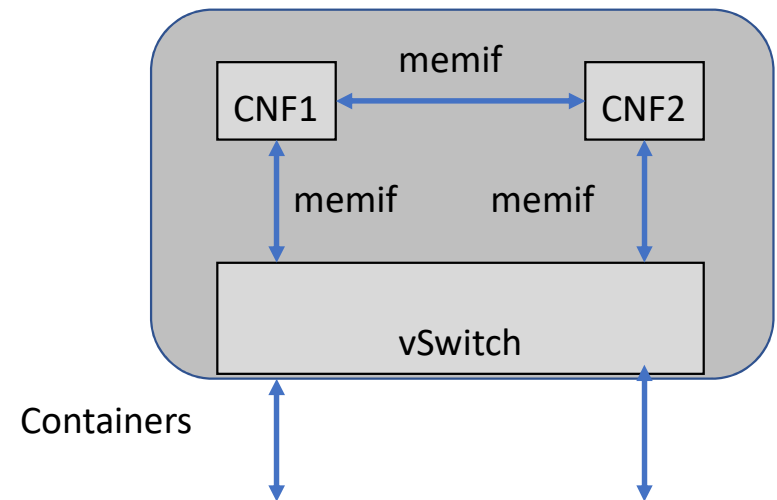
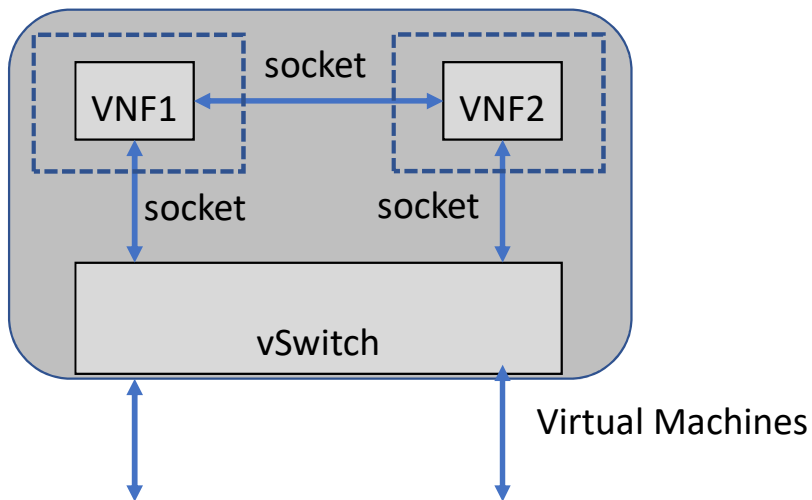


Service Function Chaining with Cloud-Native VNFs



Accelerating NFV Using Containers

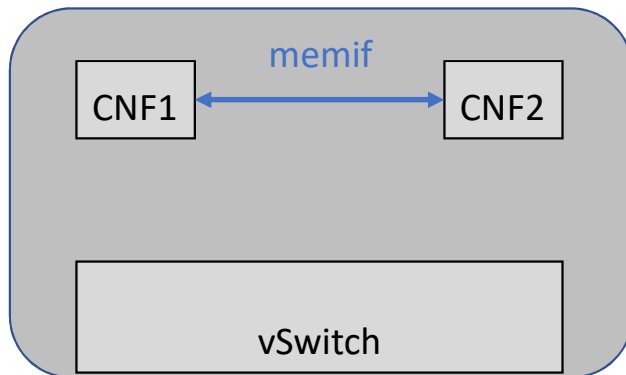
- In VM case have to copy via the kernel
- With containers we use a shared memory interface (memif)
 - Key is to chain between NFs on the same server
 - Containers are “cheap” so can have dedicated chain per tenant service



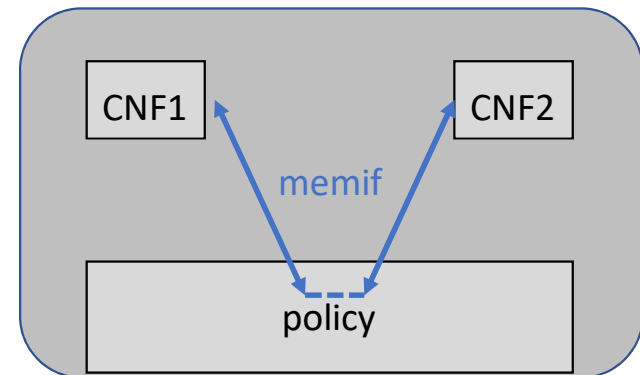
Intra-Server Rendering

Point to Point – 2 options based on policy

Direct East/West Memif

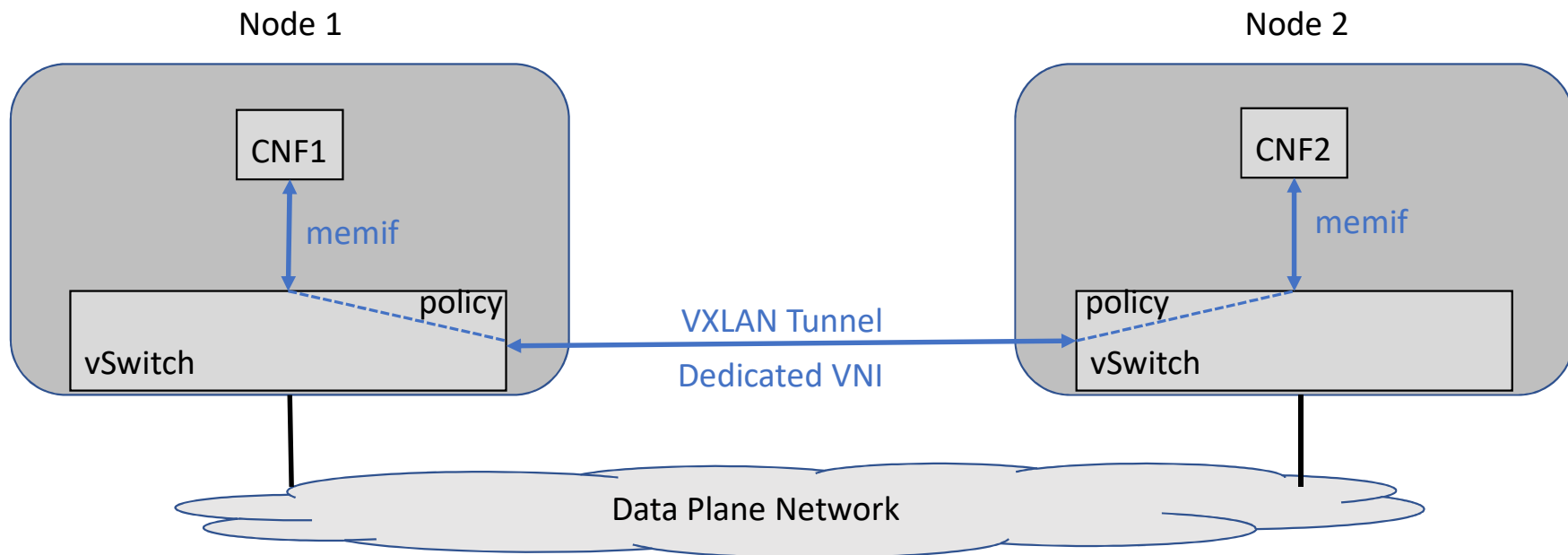


Memif via vSwitch



VXLAN Rendering

Point to Point



Creating VXLAN xConnect

```
Terminal
[rxvr@i2ss-c2201:~]$
[rxvr@i2ss-c2201:~]$
[rxvr@i2ss-c2201:~]$ sudo vppctl

  [VPP]
  [VPP]

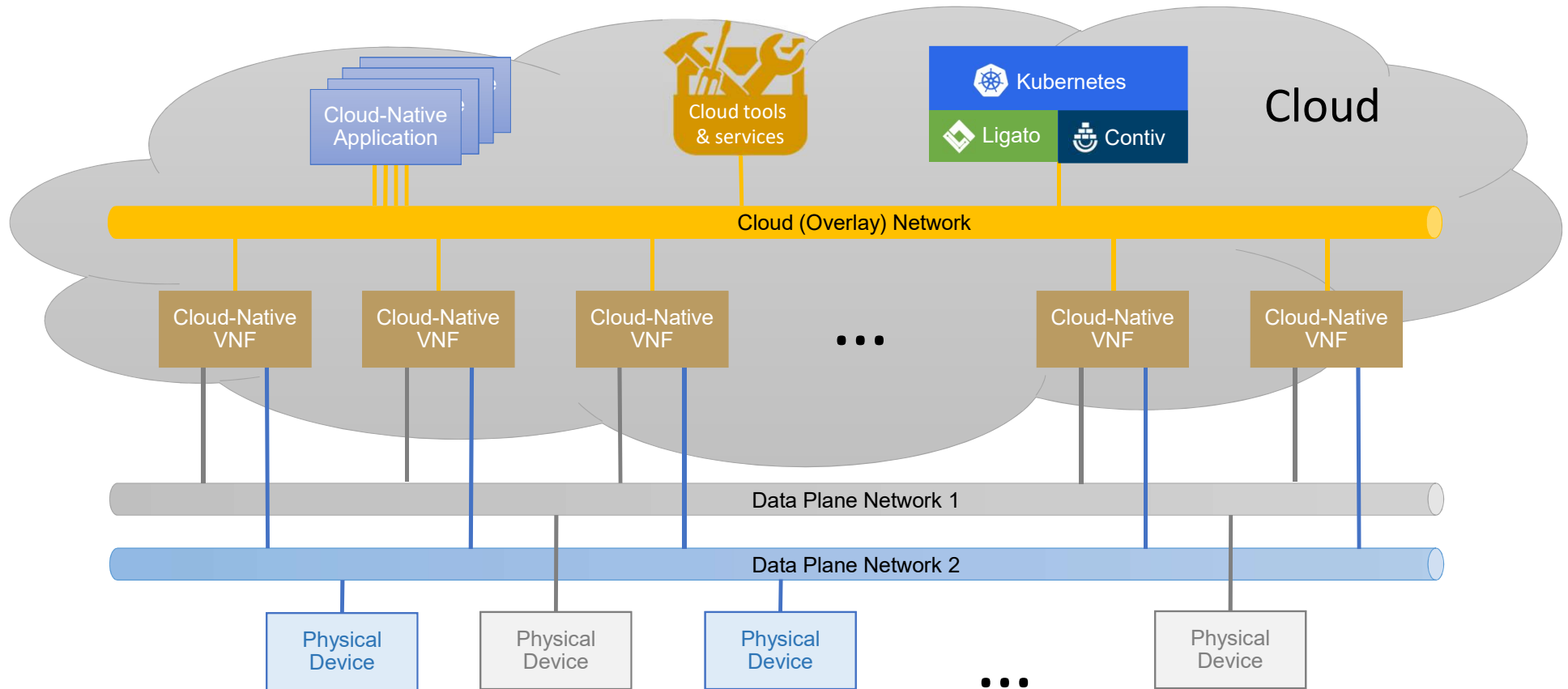
vpp#
vpp# create interface memif id 0 master
vpp#
vpp#
vpp# create vxlan tunnel src 192.168.16.1 dst 192.168.16.2 vni 321 encap-vrf-id 0 decap-next 12
vxlan_tunnel2
vpp#
vpp#
vpp# set interface 12 xconnect memif0/0 vxlan_tunnel2
vpp# set interface 12 xconnect vxlan_tunnel2 memif0/0
vpp#
vpp#
```

Create memif

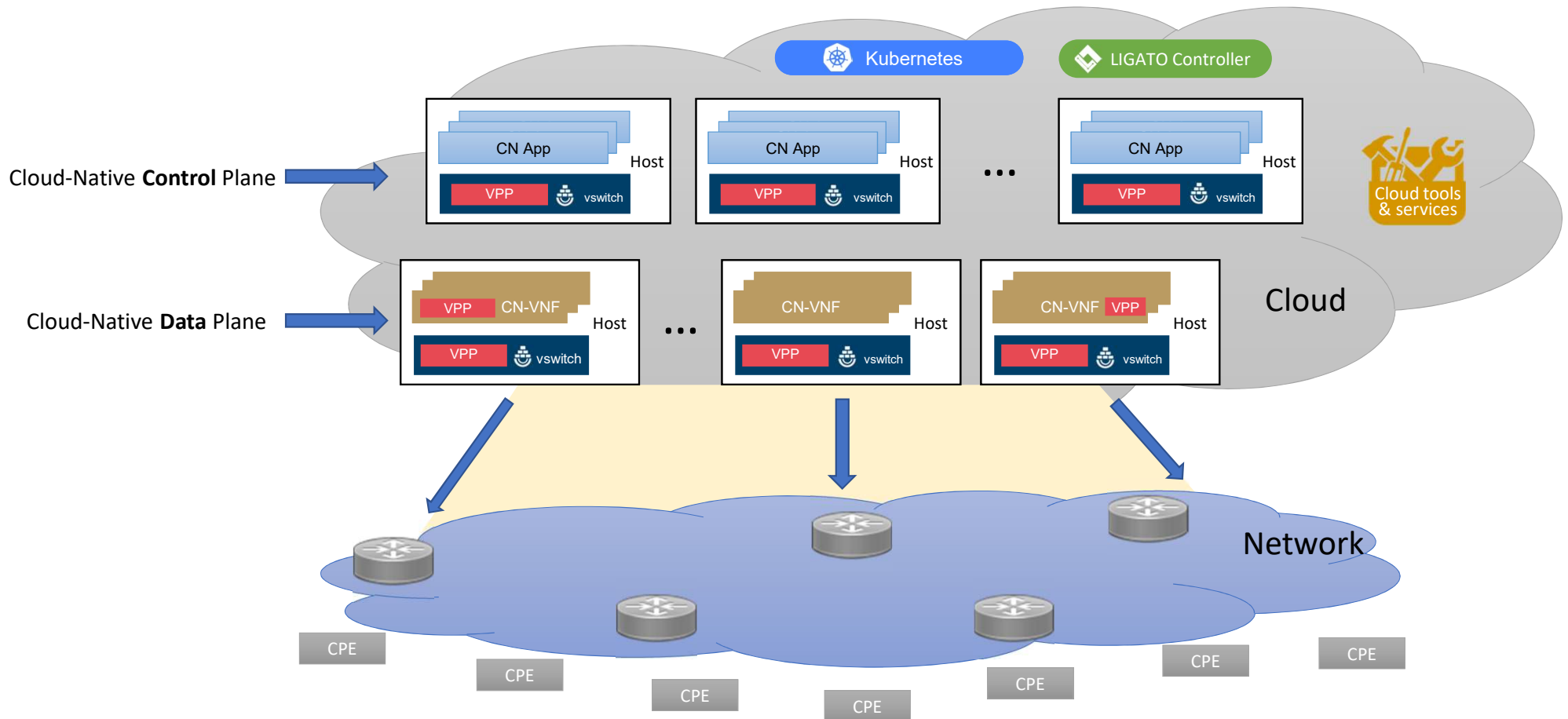
Create VXLAN Tunnel

Create xConnect

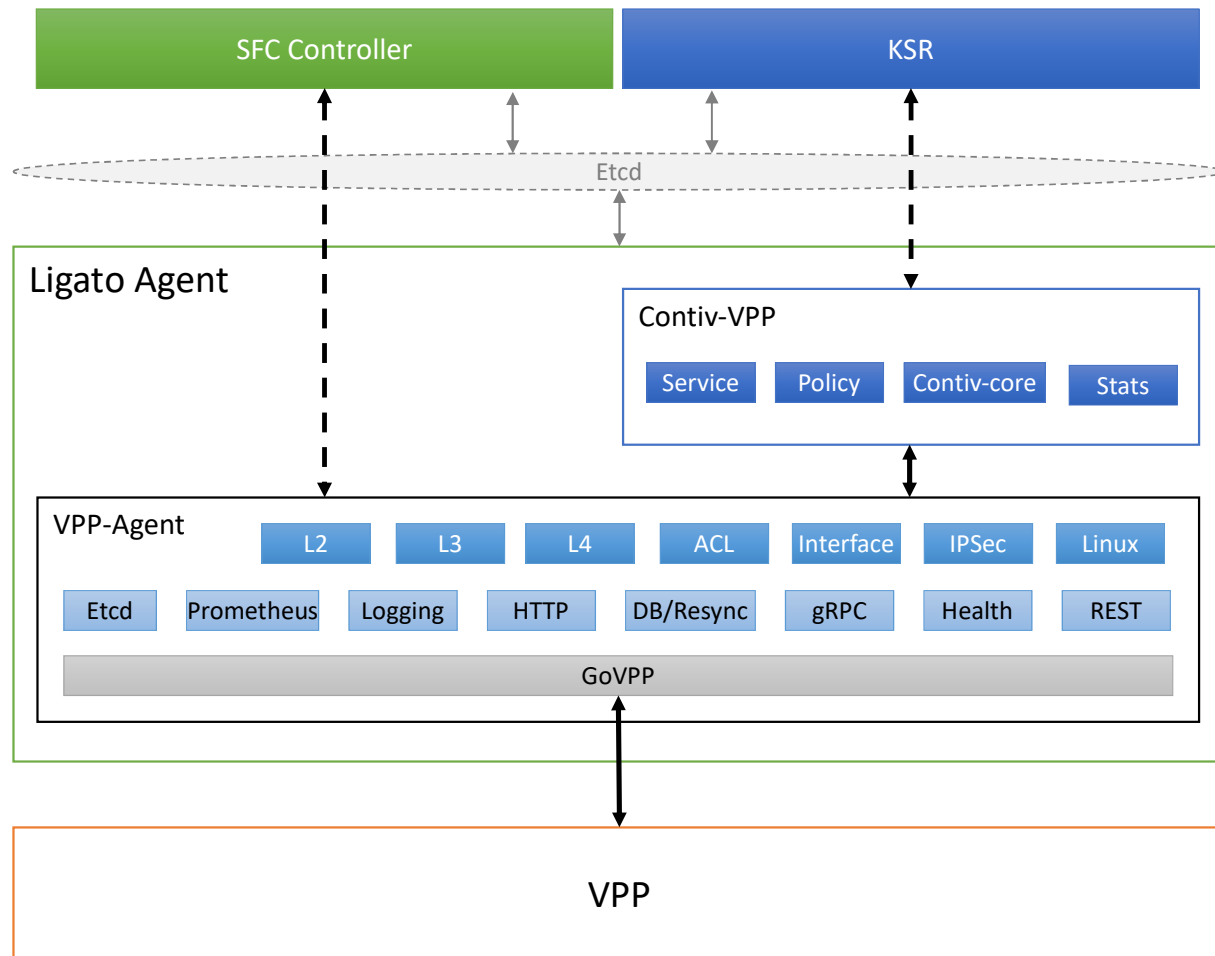
A VNF Cloud



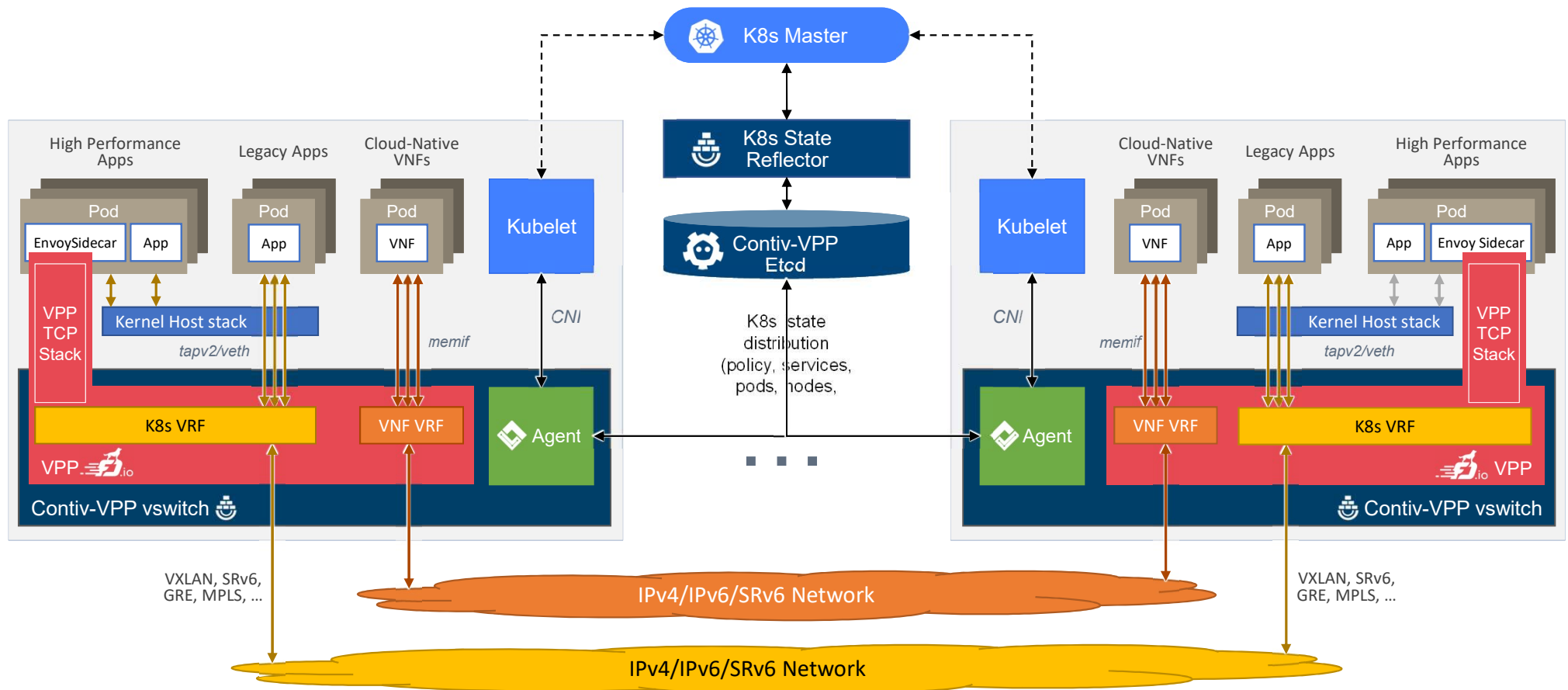
A VNF Cloud: Data and Control Planes



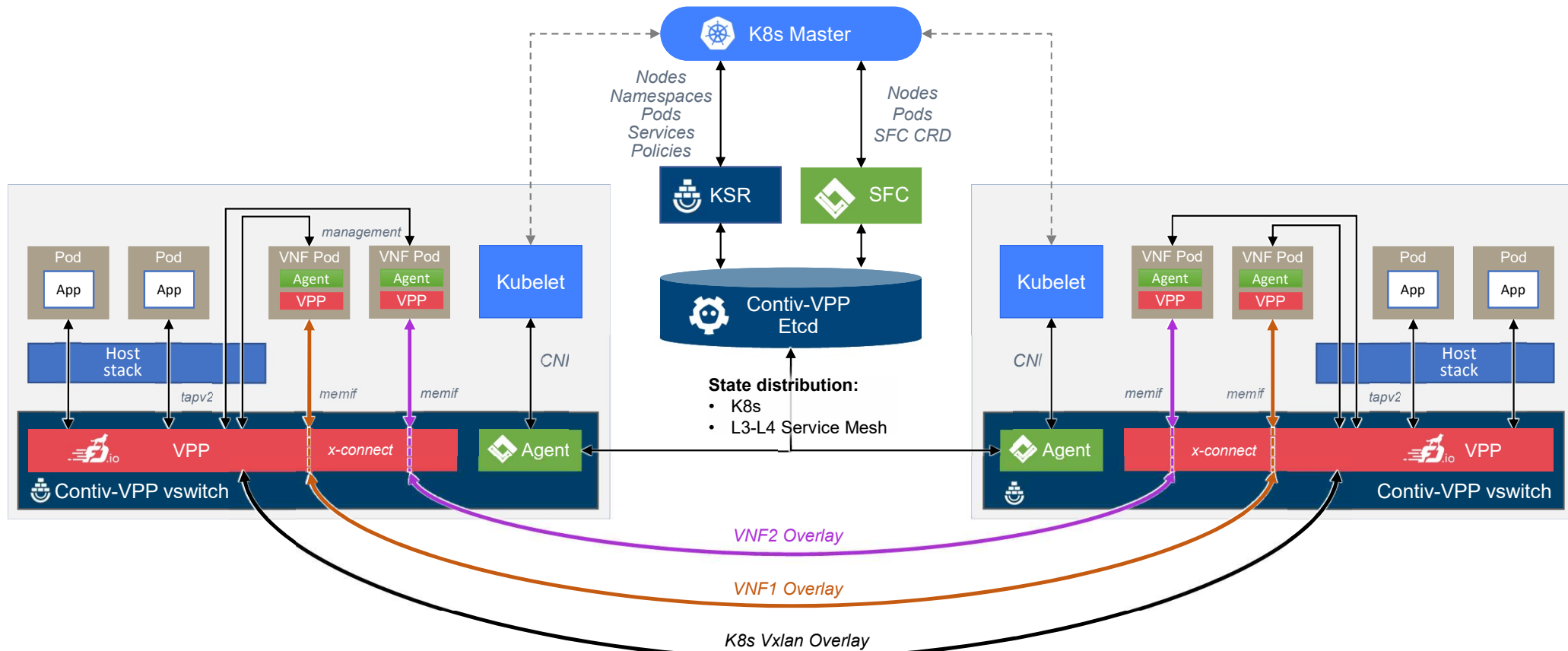
Ligato and Kubernetes Control and Data Planes



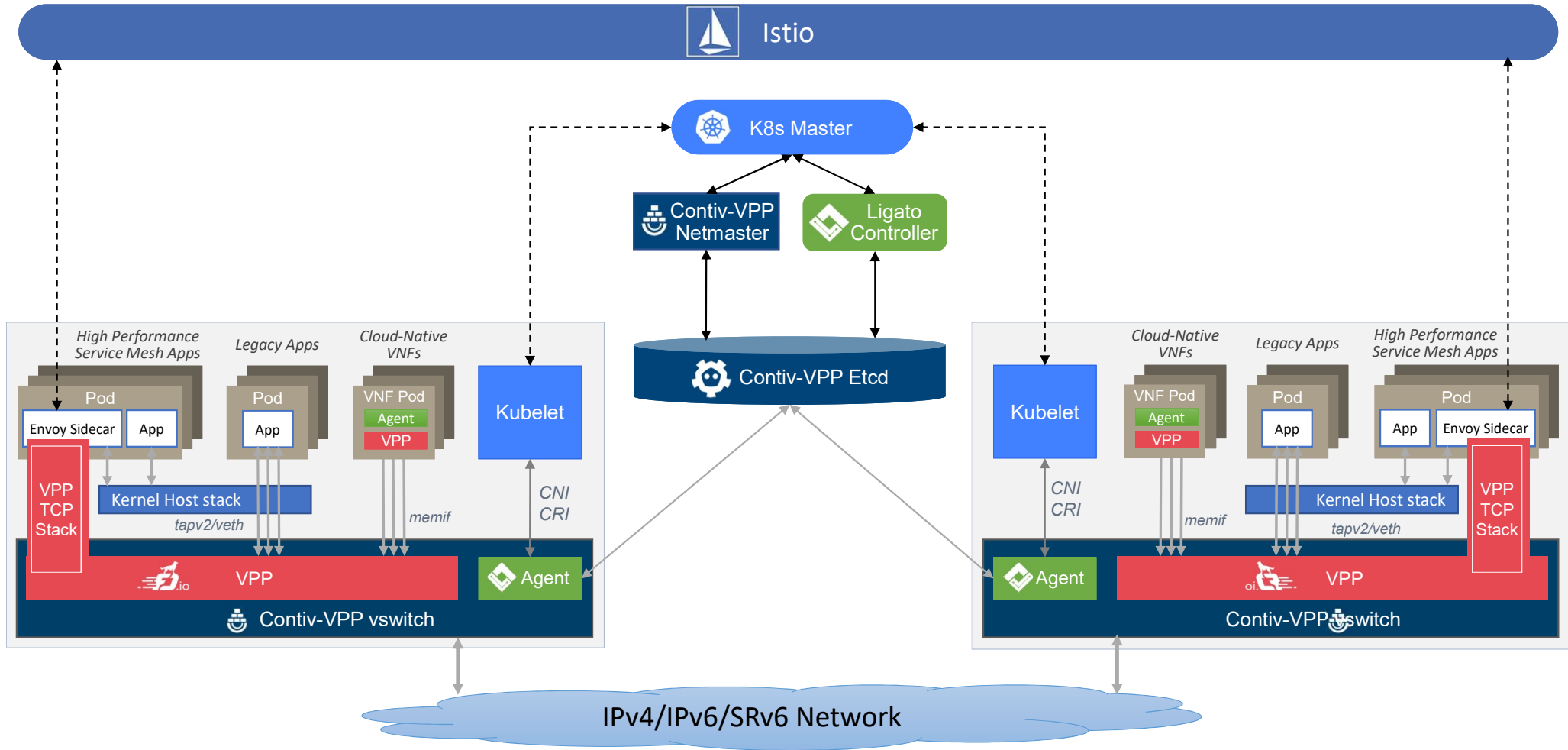
VNFs & K8s Networking




ONS Demo



Putting it All Together...



Ligato on Github

**Ligato**
<https://ligato.github.io>

Repositories 8 **People** 18 **Teams** 1 **Projects** 1 **Settings**

Pinned repositories Customize pinned repositories

cn-infra
A platform for developing cloud-native VNFs
Go ★ 13 🍴 17

vpp-agent
cn-infra based VNF agent for VPP (FD.io)
Go ★ 18 🍴 23

sfc-controller
Service Function Chain (SFC) Controller for stitching virtual and physical networking
Go ★ 4 🍴 4

Type: All **Language:** All New


vpp-agent
cn-infra based VNF agent for VPP (FD.io)
cloud-native vpp vnf fdio cn-infra
Go ★ 18 🍴 23 Apache-2.0 Updated 8 hours ago

cn-infra
A platform for developing cloud-native VNFs
microservices cloud-native vnf cn-infra
Go ★ 13 🍴 17 Apache-2.0 Updated a day ago

sfc-controller
Service Function Chain (SFC) Controller for stitching virtual and physical networking
Cisco Confidential
Go ★ 4 🍴 4 Apache-2.0 Updated 4 days ago

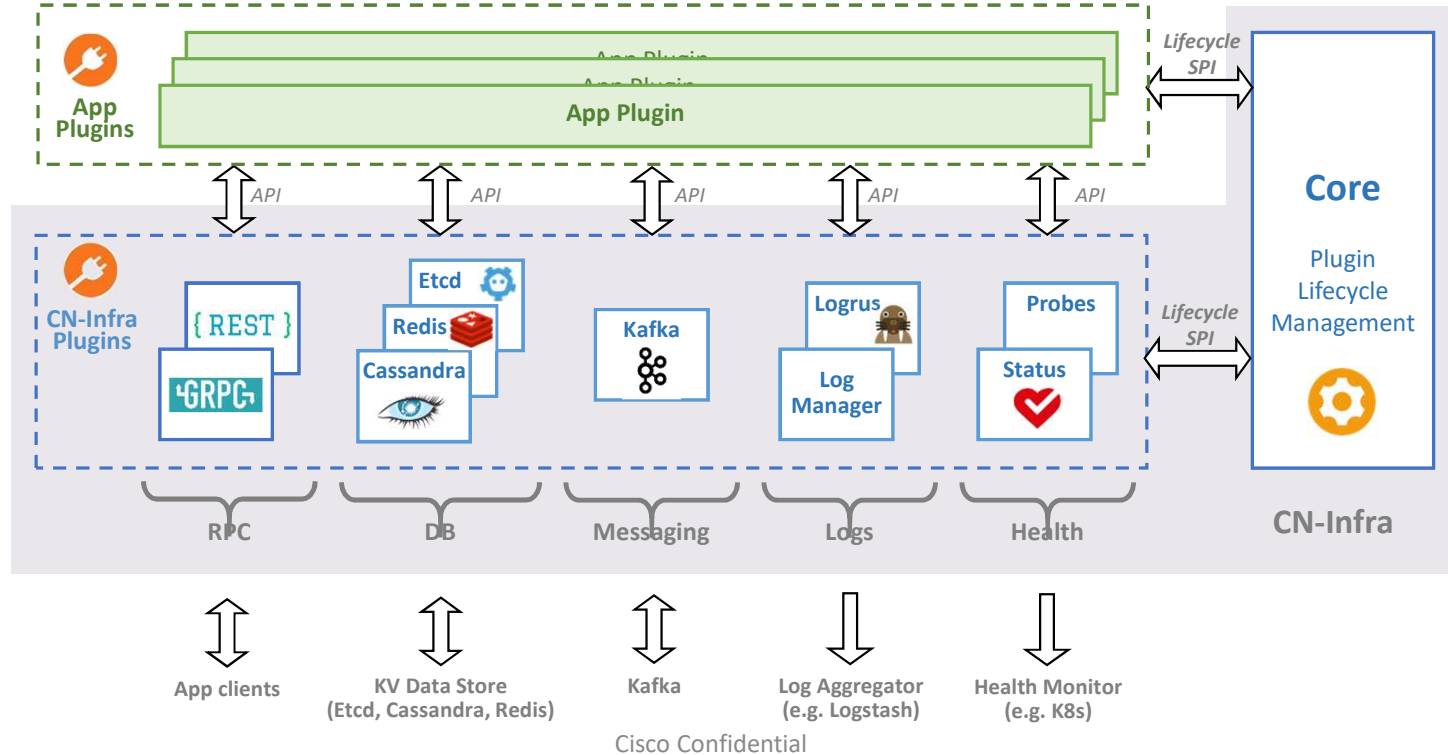
Top languages
Go CSS Makefile

Most used topics Manage
cloud-native cn-infra vnf

People 18 >

R

Ligato CN-Infra: a CNF Development Platform

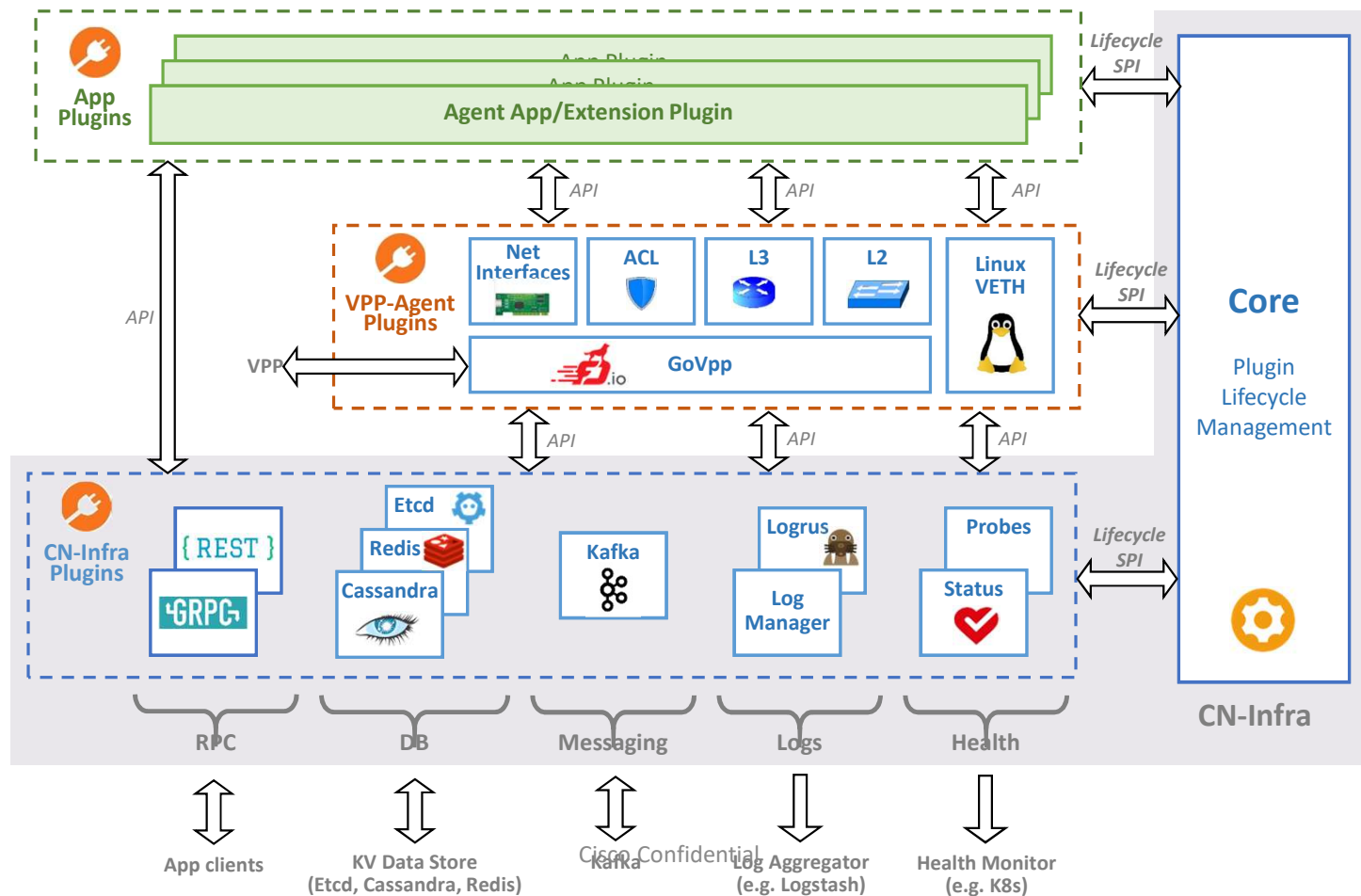
www.github.com/ligato/cn-infra



Cisco Confidential

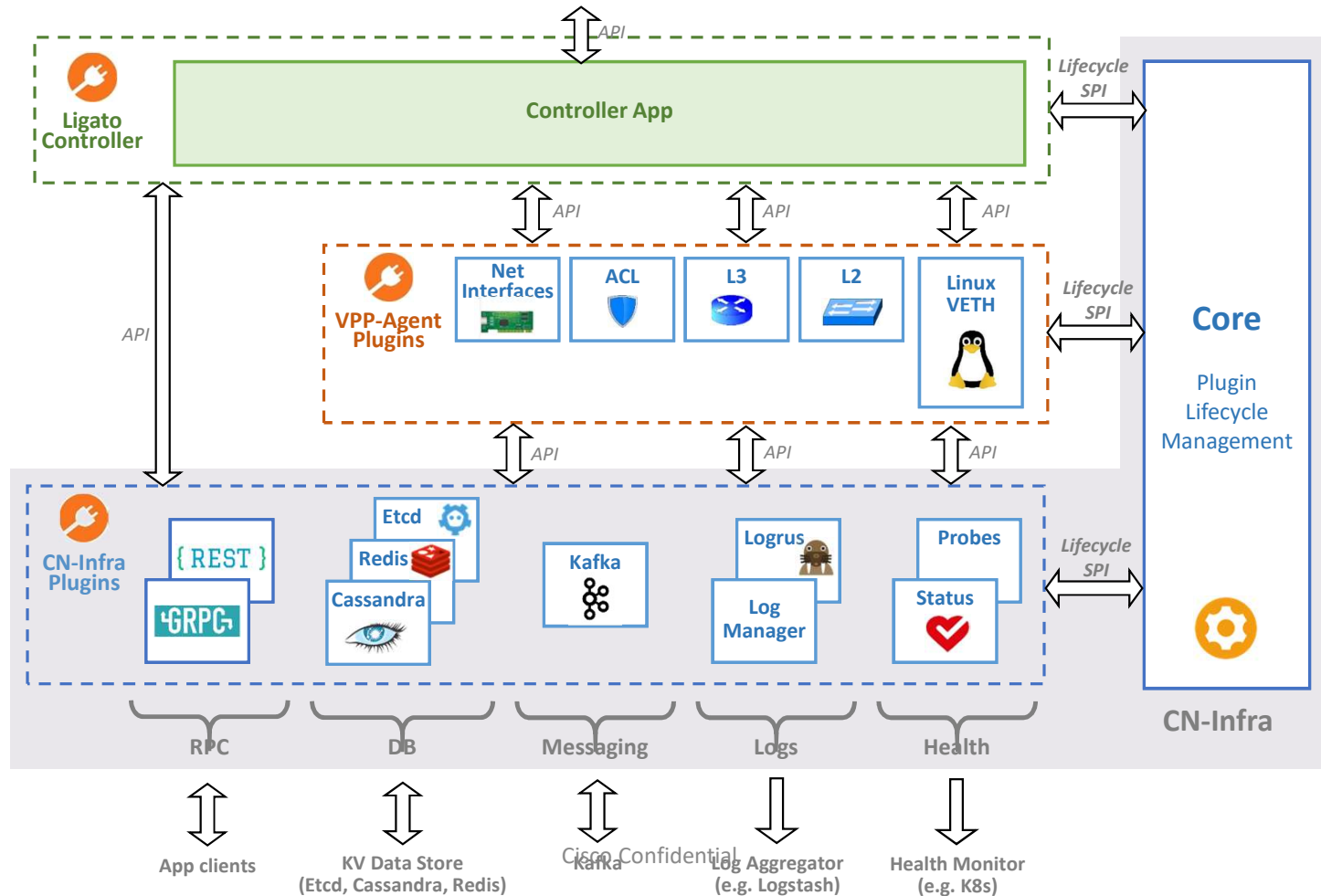
Ligato VPP Agent: a CNF Management Agent

www.github.com/ligato/vpp-agent



Ligato Controller: a CNF Deployment Platform

www.github.com/ligato/sfc-controller



Backup

Network Micro-Service Use Case:

Service Function Chaining with Cloud-Native NFs

