ons
NORTH AMERICA
OPEN NETWORKING //
Enabling Collaborative
Development & Innovation

Cloud Native Edge App & NFV Stack

(Goal – deploy all kinds of workloads – VNFs, CNFs, VM-Apps, Container-Apps, functions)

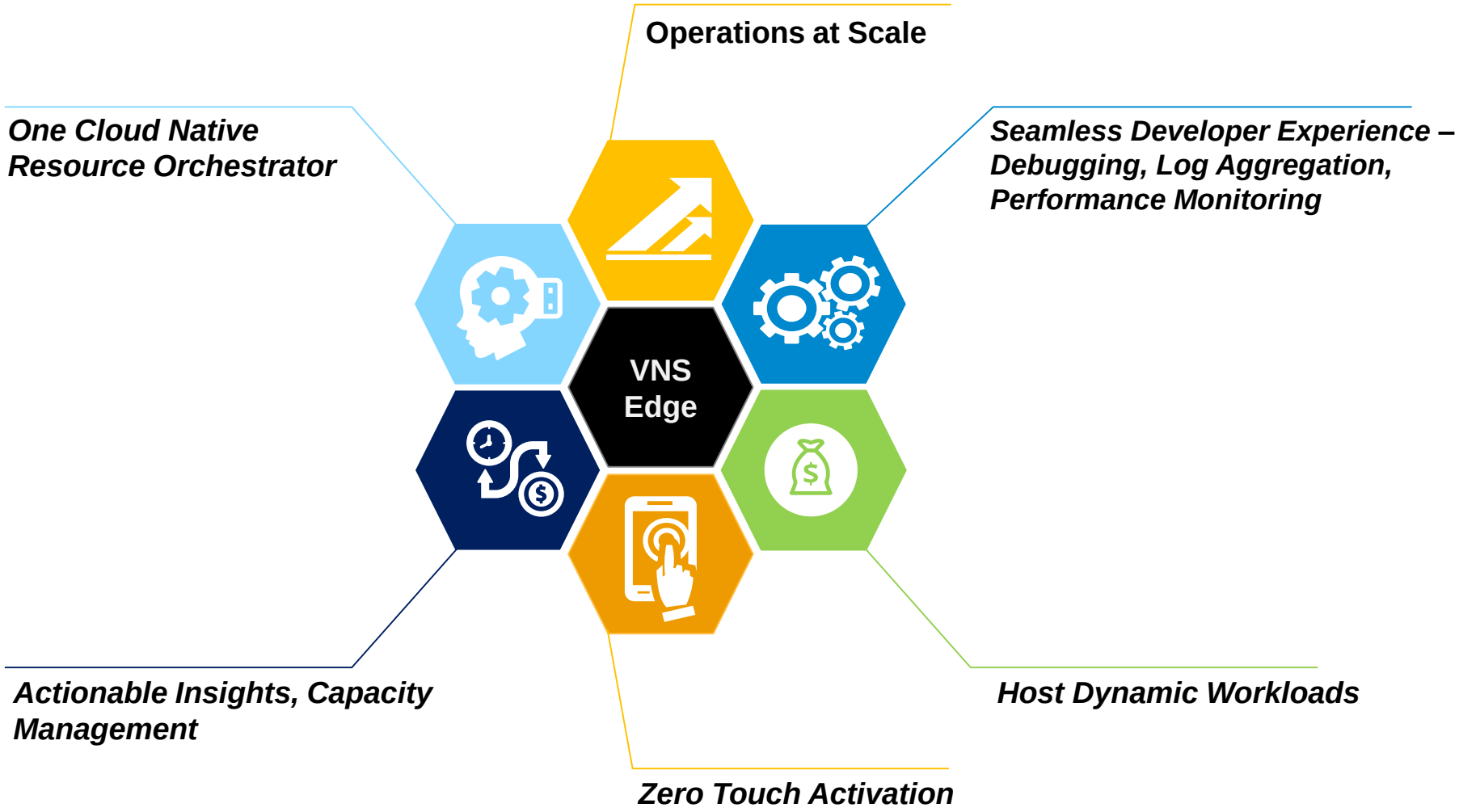
Srinivasa Addepalli (Srinivasa.r.addepalli@intel.com)
Ravi Chunduru (ravi.chunduru@Verizon.com)

Hosted By
THE LINUX FOUNDATION | LF NETWORKING

Agenda

- Why Cloud native – An operator/usecase view
- Cloud native NFV stack needs
- NFV specific requirements
- Open source projects that make Cloud native NFV stack real
- Cloud native end-to-end NFV stack – One opinionated stack
- Q&A

Requirements



© Verizon 2019, All Rights Reserved. Information contained herein is provided AS IS and subject to change without notice.
All trademarks used herein are property of their respective owners.

Approaches - Resource Management

"Co-Existing Kubernetes and Openstack"

- *Run Workloads with Kubernetes independent of Openstack*
- *Good Workload performance*
- *Easy to Support*
- *Lacks Unified View of System Resources causes problems with planning*
- *Operational challenges to debug any performance issues*

"Kubernetes Running in a VM Powered by Openstack"

- *Quick extension to Openstack Ecosystem*
- *Fully featured multi tenancy and Security*
- *Lacks Performance with Workloads*
- *Additional workflows to manage the VMs that are hosting Kubernetes*

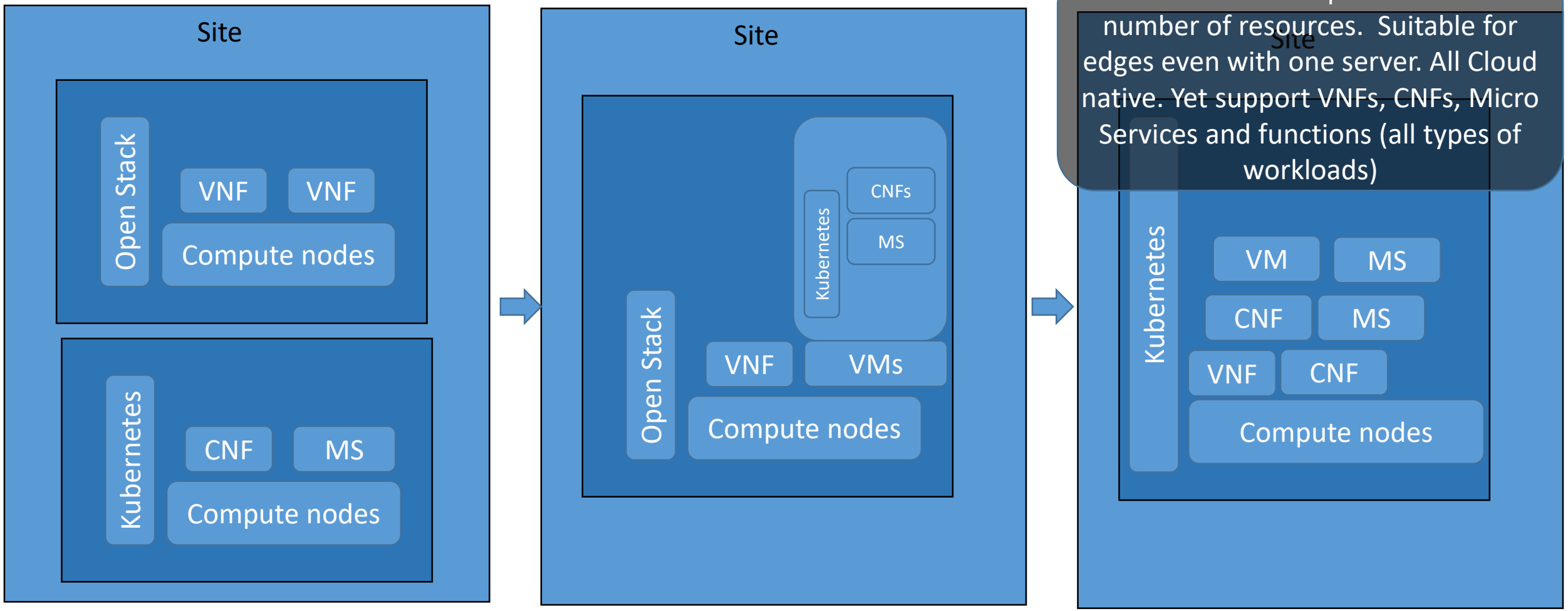
"Kubernetes replacing Openstack"

- *One stack to manage VMs and Containers*
- *Workloads take complete advantage of HW accelerators, Smart NICs etc.,*
- *Offer Integrated VNS solutions with Container Workloads*
- *Need improved Networking capabilities like in Openstack (SFC, Provider Networks, Segmentation)*

verizon

© Verizon 2019, All Rights Reserved. Information contained herein is provided AS IS and subject to change without notice.
All trademarks used herein are property of their respective owners.

Transformation journey (to Kubernetes & Cloud native)

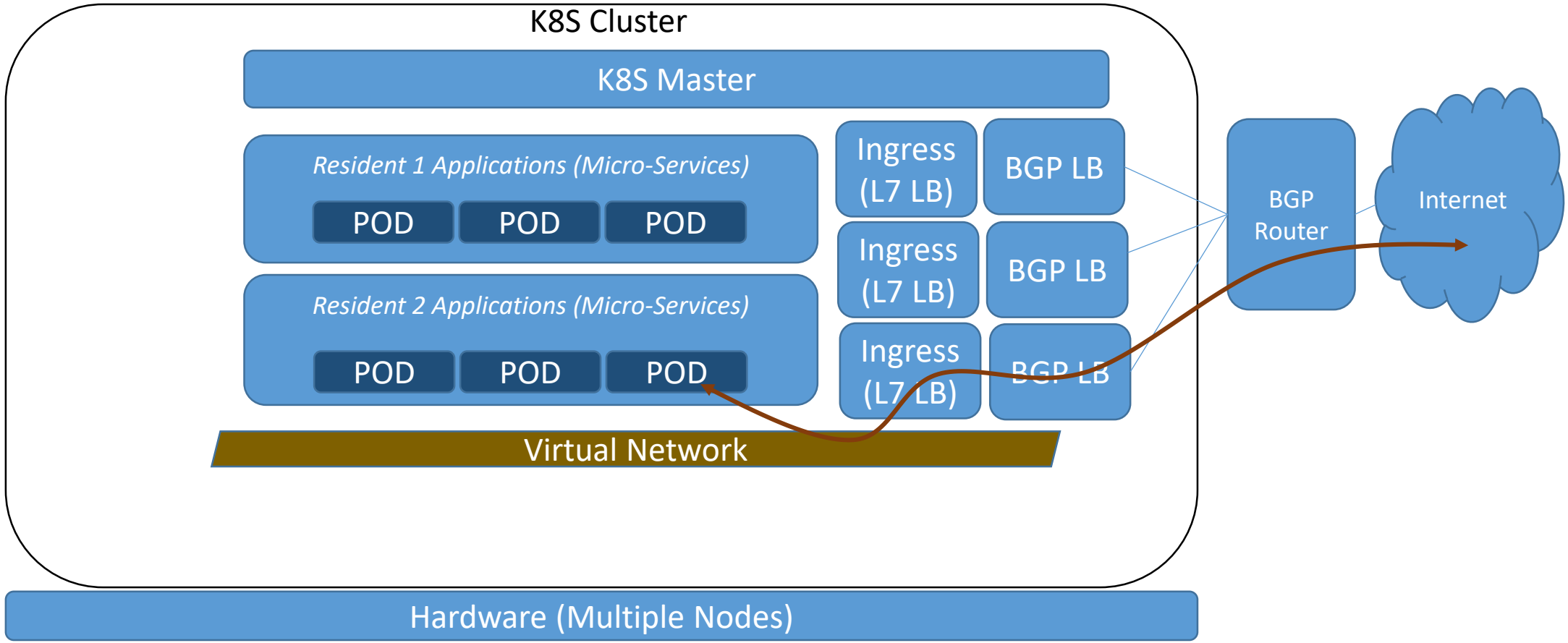


Two different resource orchestrators
Compute nodes are divided

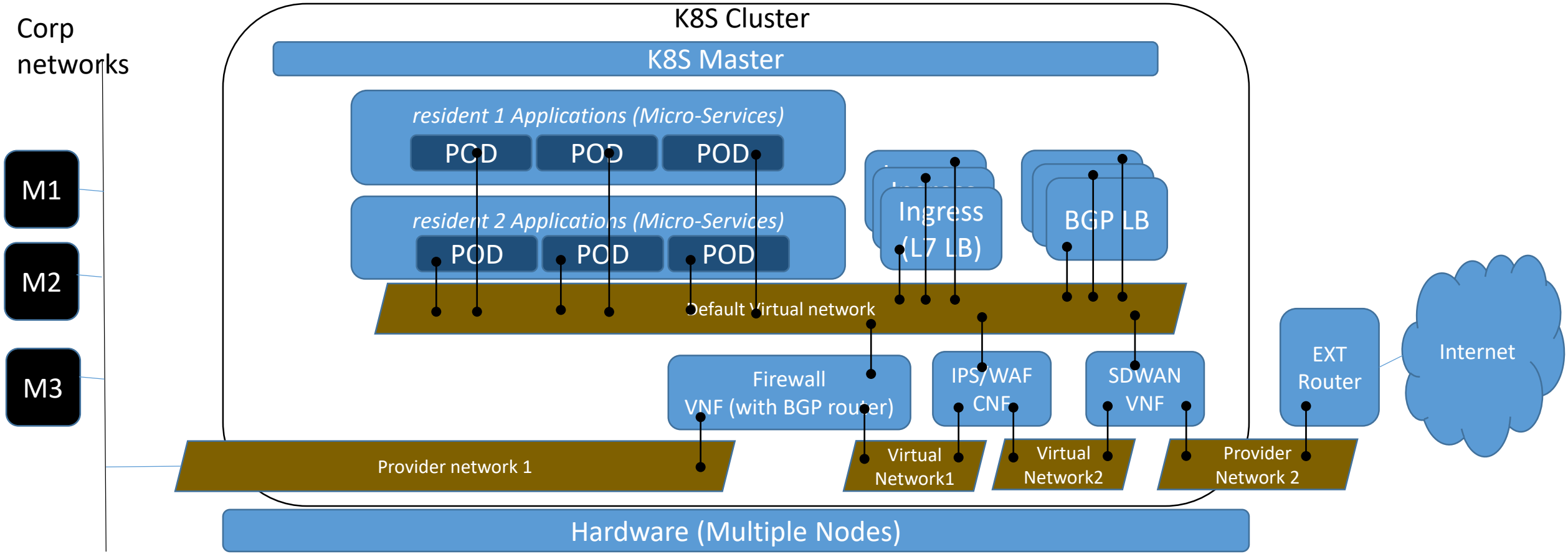
Openstack for VNFs and VMs
K8S on set of VMs.

Run all on bare-metal, one resource orchestrator

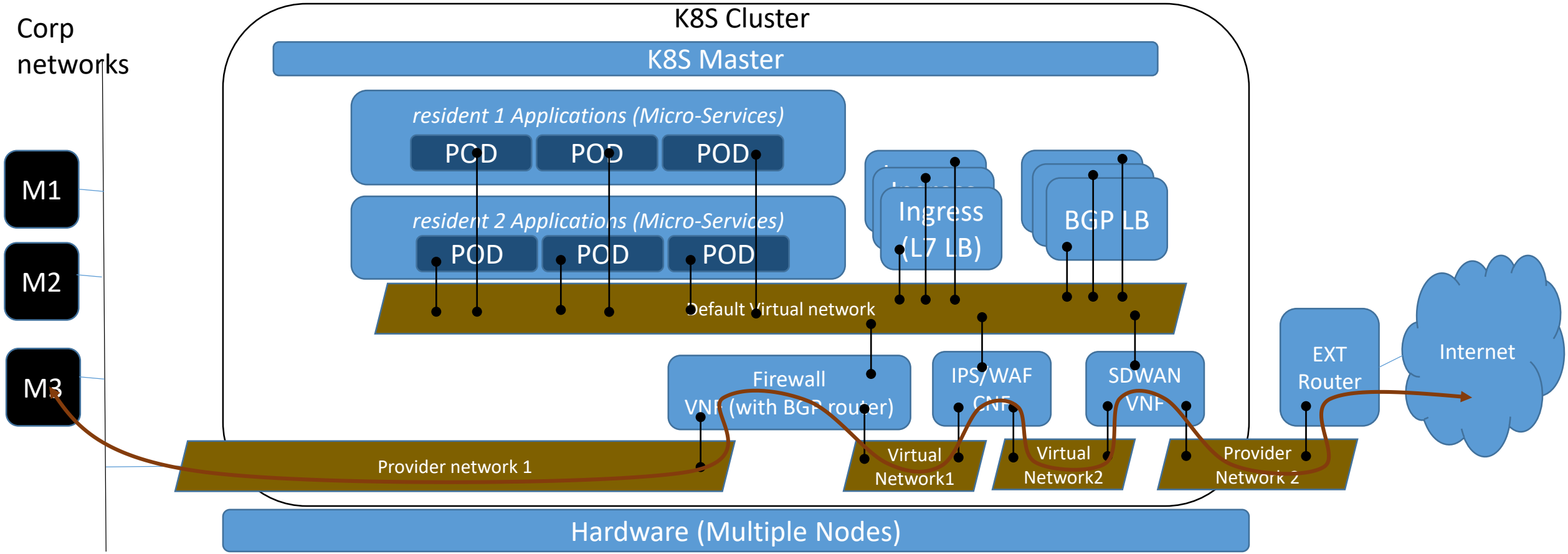
Introduction of K8S based Micro-service deployment – One example scenario



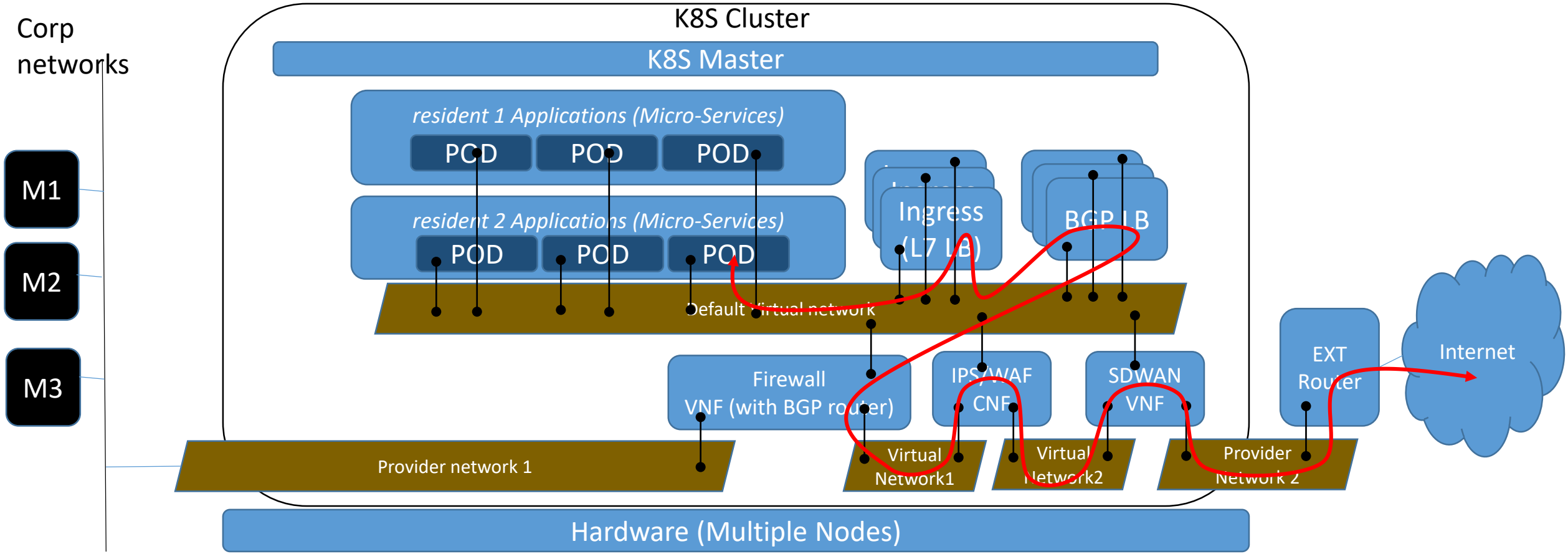
How does NFV based deployment with Cloud-native applications look like (Taking SDWAN with security NFs as an example)



Traffic flow 1 – From internal machines to Internet



Traffic flow2 – Communication between Micro-Services and Internet



Cloud native NFV stack – Requirements Summary

Network requirements
Co-existence of Network functions and applications
Network Functions as VNFs and CNFs
Multiple Virtual networks
Provider networks
Network function chaining – Inserting new functions dynamically and auto reconfiguration

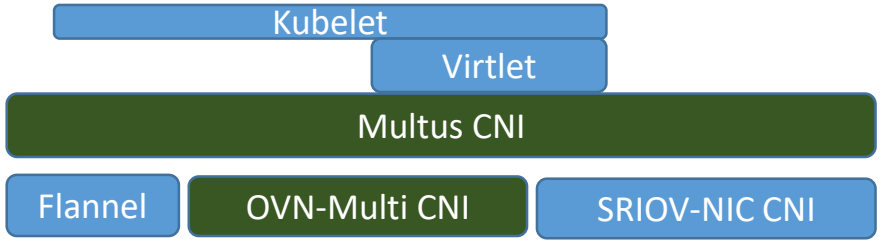
Performance requirements
High throughput with respect to PPS & bps
Low latency and low jitter
Performance determinism (even with noisy neighbors)

Security
Attestation & Verifications of infrastructure
Private key and password protection, IP protection

Generic requirements
Network Service Orchestration at a central locations across multiple K8S clusters & Openstack locations
Multi-residency support (soft isolation)
Telemetry & Monitoring
Cloud Native NFV provisioning system that deploys not only K8S components, but also NFV related components

Network requirements – Open source projects addressing them

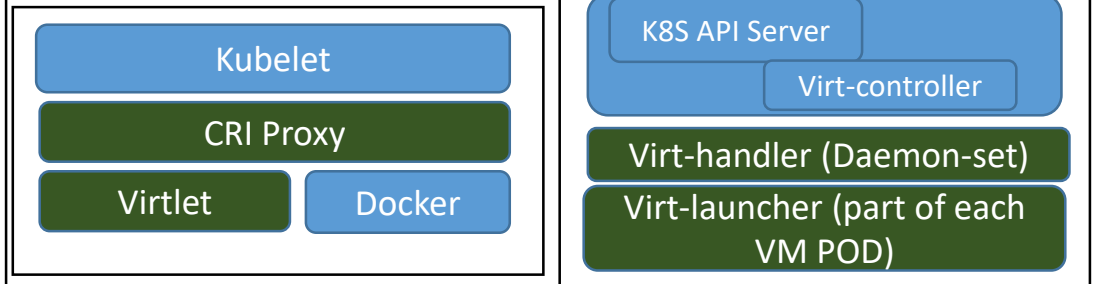
Requirements:
Multiple Virtual networks
Dynamic creation/termination
Static service function chaining
Provider network support
Programmable (for future extendability)



Multus:
Higher level CNI and associated CRD logic to allow creation of multiple networks using various CNIs
Founded by Intel: https://github.com/intel/multus-cni

OVN Multi CNI and associated network watcher (for K8S)
Enables multiple OVN based networks and enables PODs/VNFs to sit on multiple virtual networks.
Founded by Intel in **OPNFV**:
https://gerrit.opnfv.org/gerrit/gitweb?p=ovn4nfv-k8s-plugin.git;a=summary
Route Controller (for static SFC): Being conceptualized
Dynamic SFC: TBD (Candidates: https://networkservicemesh.io/, OVNSFC)

Requirements:
Co-existence of VMs and containers
VNF and CNF support
Sharing of compute nodes
K8S for VNFs/CNFs/Micro-services
Leverage VNFs that are developed for Openstack



Virtlet based

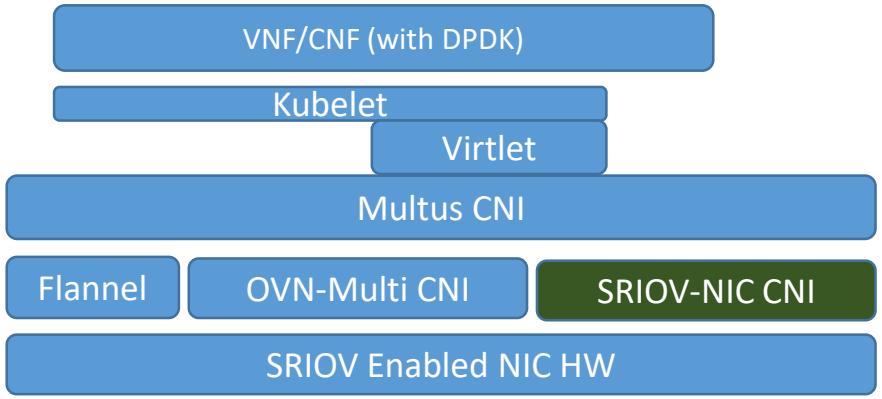
kubevirt based

VM/VNF based workload support in K8S
Virtlet:
https://github.com/Mirantis/virtlet
https://github.com/Mirantis/criproxy

Kubevirt
https://github.com/kubevirt

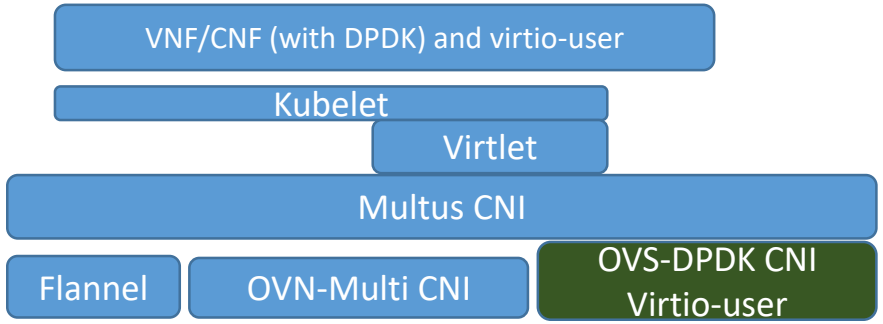
Performance requirements – Container networking acceleration
Open source projects addressing them

Requirements:
High throughput for packet processing applications



SRIOV-NIC CNI
Two modes:
- SRIOV NIC VFs are directly attached to VNF/CNF.
- SRIOV NIC VFs are attached to host Linux kernel and then the connection through veths.

https://github.com/intel/sriov-cni



OVS-DPDK CNI
Instead of vEth pair of connecting virtual ports, use familiar virtio-user/vhost-user interface to connect to OVS-DPDK based virtual switch.

Useful only if all the workloads in the node are packet processing based applications as this can reduce the performance of other workloads that don't receive packets via virto.

https://github.com/intel/userspace-cni-network-plugin

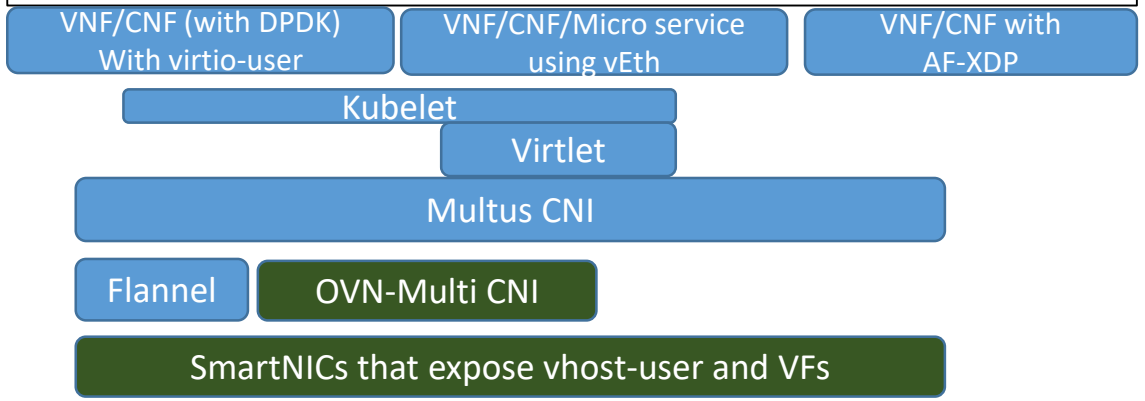
Performance requirements – Container networking acceleration with smartNICs

Requirements:

High throughput for packet processing applications

Performance determinism

Network namespace offloads (TCP session search etc…)



Kubernetes friendly smartNIC

OVN uses OVS which can be offloaded to smartNIC.
SmartNICs exposing virtio interface can send packets to VNFs/CNFs that are packet processing based
OVN with SmartNIC to enable normal containers that are not packet processing based.
Enable VNF/CNFs that leverage AF-XDP.
Open source: Community help

Performance requirements – Compute related open source projects

Requirements	Project	Descriptions
Performance determinism even when there are noisy neighbors (using affinity and isolation of cores)	https://github.com/intel/CPU-Manager-for-Kubernetes CMK – CPU Manager for Kubernetes	• If high performance PODs cores also can be dedicated. • Also, it enables affining the shared cores
Performance determinism by reducing paging via hugepages	https://github.com/kubernetes/kubernetes/pull/50859 (WIP - Hugepages feature) https://github.com/kubernetes/kubernetes/pull/50072 (WIP – Hugetlbfs via volume plugin)	• Accounting of huge pages • Relinquishing huge pages upon unexpected termination
Placement of VNFs/CNFs/Micro-services based on their hardware requirements	github.com/kubernetes-incubator/node-feature-discovery (Node feature discovery)	• Discovers the hardware capabilities on each node and advertises via labels vs nodes. • Allows VNF/CNF description to have hardware requirements via lables.

Performance requirements – Accelerator related open source projects (As Device plugins)

Requirements	Project	Descriptions
Improve performance of Crypto and compression operations of VNFs/CNFs/Micro-services by leveraging hardware	https://github.com/intel/intel-device-plugins-for-kubernetes/tree/master/cmd/qat_plugin QAT - Plugin	• QAT Device Plugin discovers QAT cards on a node and the number of VFs configured, advertises this to the node and allocates VFs based on workload resource requests
Improve AI/Media performance using FPGA and GPU	https://github.com/intel/intel-device-plugins-for-kubernetes/tree/master/cmd/fpga_plugin (FPGA) https://github.com/intel/intel-device-plugins-for-kubernetes/tree/master/cmd/gpu_plugin (GPU)	• Make GPU/FPGA available to VNFs/CNFs/Micro-Services as and when they request them.

Generic requirements – Create K8S Cluster with all NFV related SW

Requirements:
Install/Provision all software needed for NFV

KRD (Deploy Kubernetes for NFV) – Being done in ONAP

- Set of Ansible scripts
- Enhanced Kubespray
- Installs/configures following software:
  - Kubernetes
  - NFD (Node feature discovery)
  - Flannel, OVN, OVN4K8SNFV (CNI & Watcher)
  - ISTIO
  - Mutlus
  - Virtlet
  - Telemetry – CollectD, Prometheus
  - Ceph/Rook
  - kNative
  - Future: Kubevirt, CMK, SRIOV-NIC CNI, FPGA, GPU plugs, Route configurator etc…

https://github.com/onap/multicloud-k8s

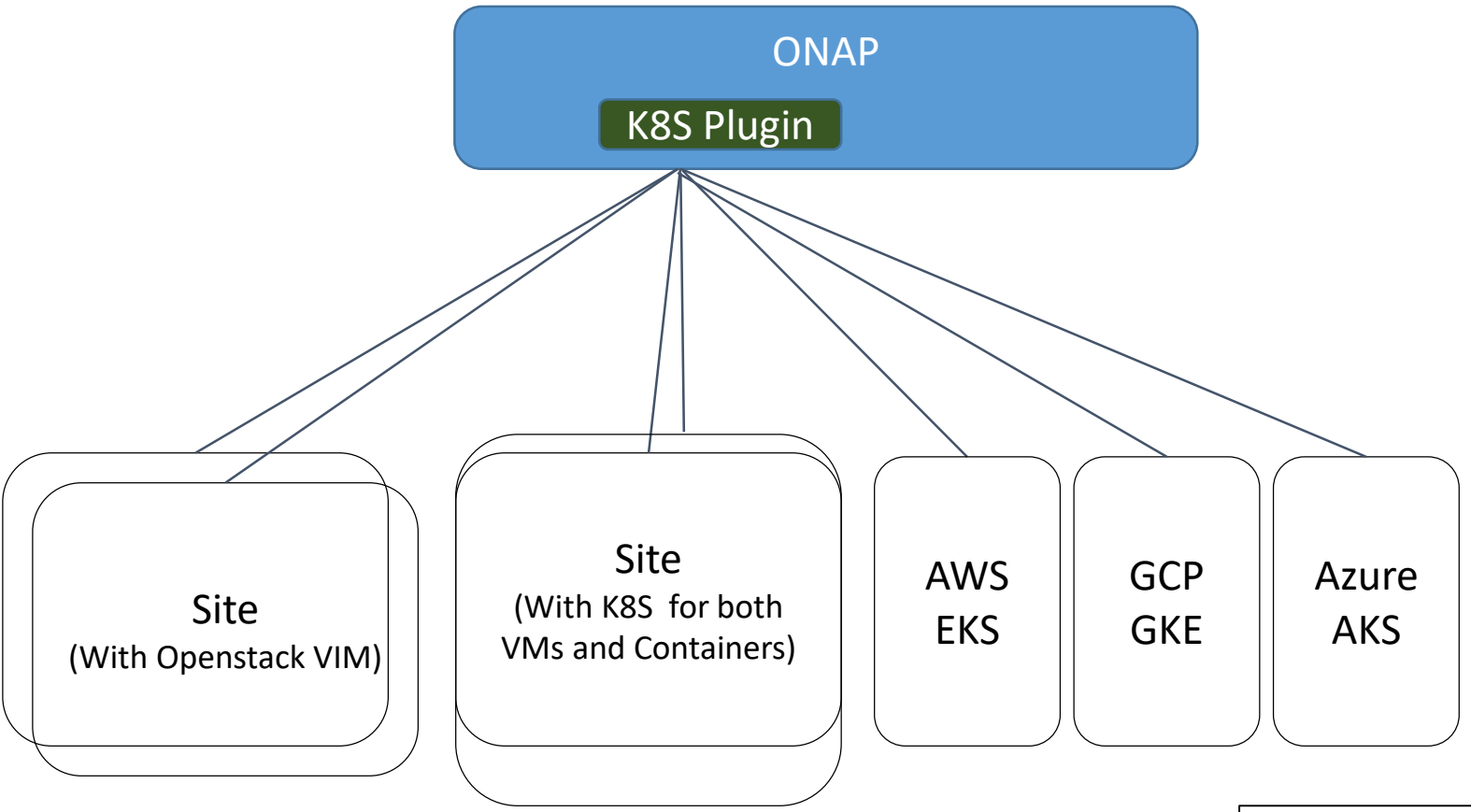
Requirements:
Bare Metal provisioning (with Linux OS and system SW)

Digital Rebar/Ironic (Being explored)

- API Driven server provisioning
- Golang based
- DHCP/PXE/TFTP based provisioning
- Workflow system

https://github.com/digitalrebar/provision

Generic requirements – Orchestrator to Orchestrators (ONAP)



K8S Plugin in ONAP (WIP for R4)

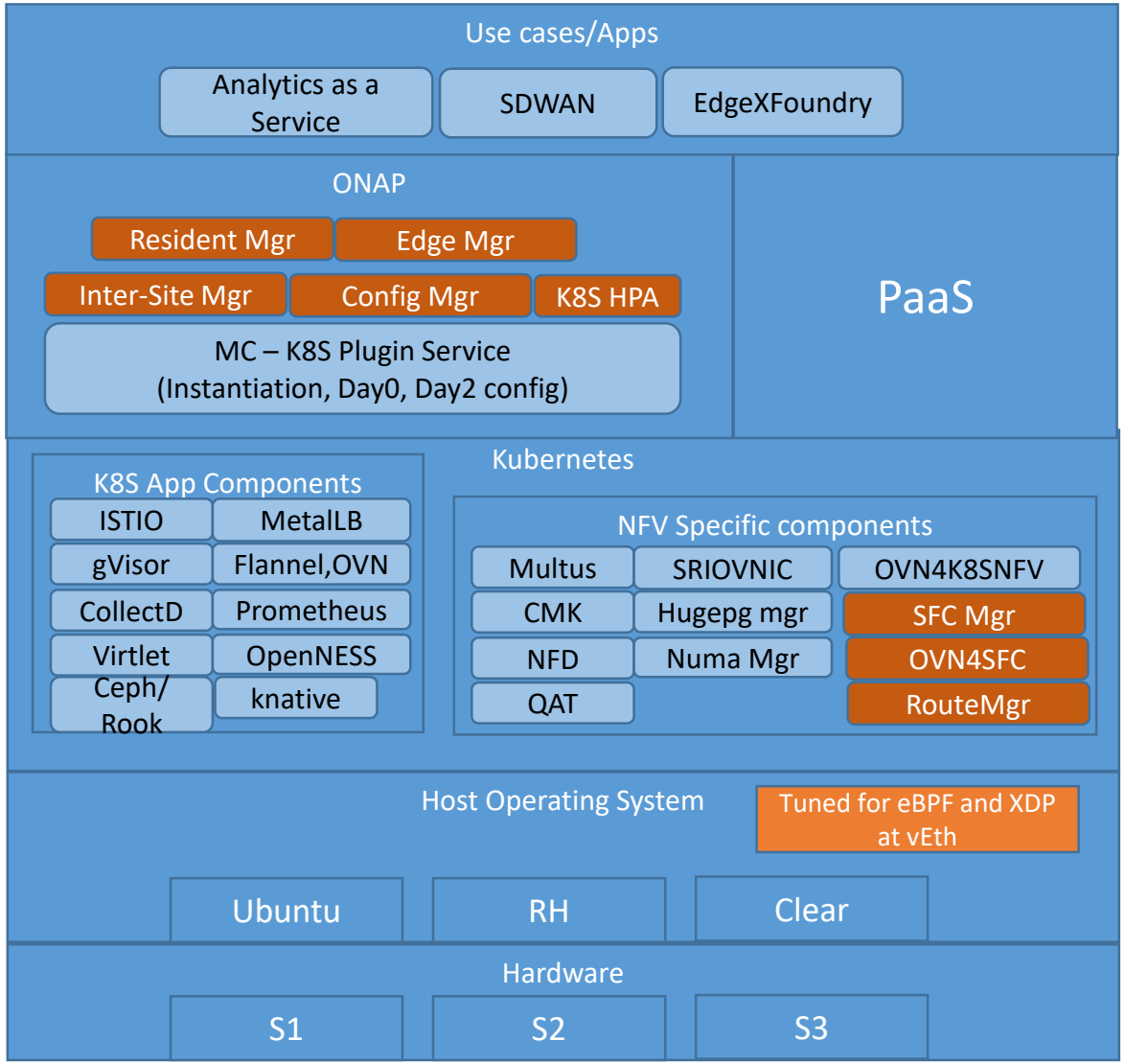
- Orchestrate workloads (VNF/CNF/Micro-Services) across multiple sites.
- Orchestrate network services that span across multiple sites
- Day0 configuration profiles
- Day2 configuration (Incremental configuration)

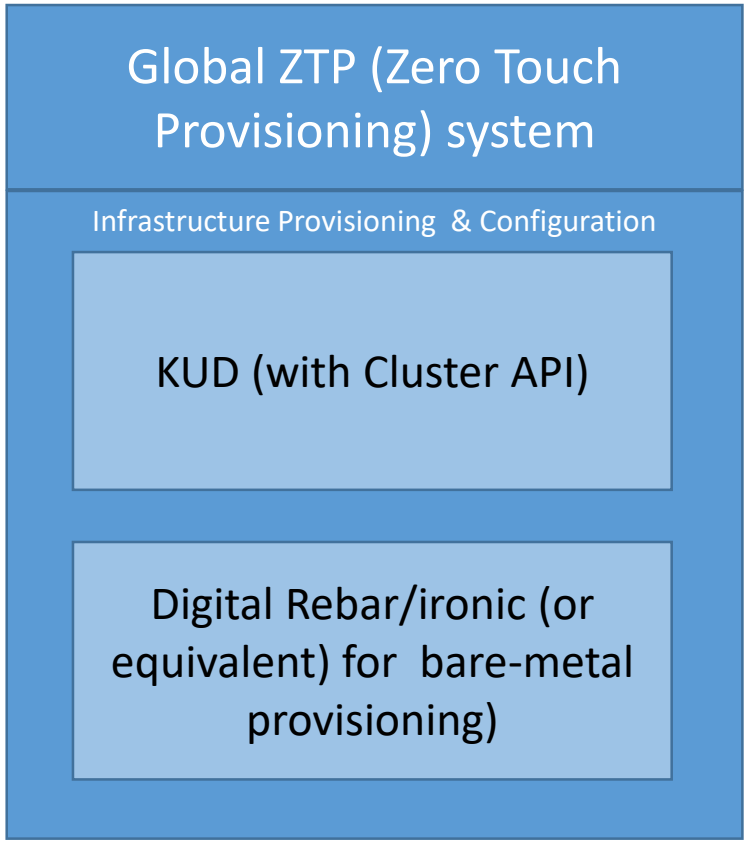
TBD:

- Hardware Platform Awareness
- Daemon set for multiple locations.
- Bulk deployment
- DAG based flow
- VNFFG and/or NSM integration

https://github.com/onap/multicloud-k8s

Cloud Native NFV Stack – Putting it all together

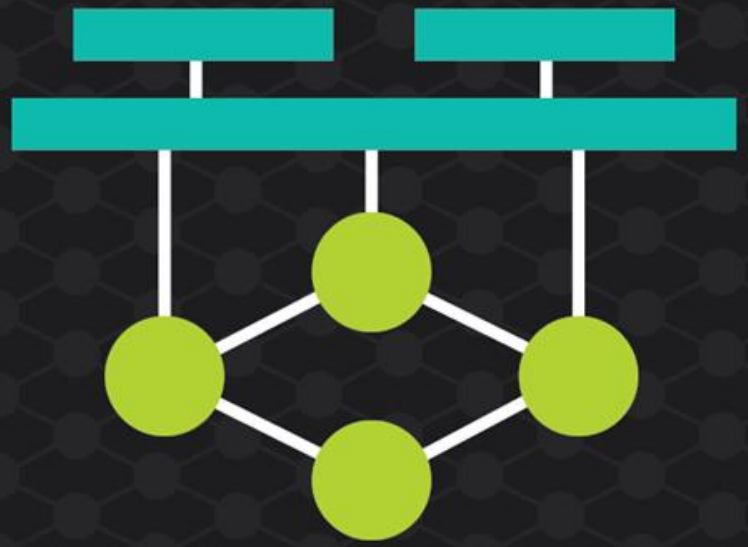




Summary

- Kubernetes (with its minimal footprint) is becoming choice of site orchestrator in Edges.
- Quite a bit of work going on in Open source (LFN, CNCF) to make K8S choice for network functions
  - Virtlet, KubeVirt for VM based Network functions.
  - OVN with OVN4K8SNFV for multiple and dynamic networks.
  - Effort to bring performance of VNFs and CNFs as physical functions with SRIOV-NICs, SmartNICs
  - Performance determinism using CPU affinity/isolation, NUMA and Huge page table support
  - Multiple device plugins to expose various HW acceleration and security features.
- Single Orchestrator for VNFs/CNFs and Micro-services
- Integration is the key
- ONAP, OPNFV and Akraino will play a key role.
- Help the community to make Cloud native NFV real.

Q&A



ons

NORTH AMERICA

OPEN NETWORKING //
Enabling Collaborative
Development & Innovation

Hosted By

THE LINUX FOUNDATION | LF NETWORKING