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)
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

- One Cloud Native Resource Orchestrator
- Actionable Insights, Capacity Management
- Zero Touch Activation
- Host Dynamic Workloads
- VNS Edge

Operations at Scale
- Seamless Developer Experience – Debugging, Log Aggregation, Performance Monitoring
## 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)
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

K8s based control plane uses less number of resources. Suitable for edges even with one server. All Cloud native. Yet support VNFs, CNFs, Micro Services and functions (all types of workloads)
Introduction of K8S based Micro-service deployment – One example scenario

K8S Cluster

K8S Master

Resident 1 Applications (Micro-Services)

POD  POD  POD

Resident 2 Applications (Micro-Services)

POD  POD  POD

Ingress (L7 LB)
BGP LB

BGP LB

BGP LB

Ingress (L7 LB)

Virtual Network

Hardware (Multiple Nodes)

Internet

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

**Diagram Description:**
- **K8S Cluster**
  - **K8S Master**
  - **resident 1 Applications (Micro-Services)**
    - PCD
    - POD
    - POD
  - **resident 2 Applications (Micro-Services)**
    - POD
    - POD
    - POD
  - **Ingress (L7 LB)**
  - **BGP LB**
- **Provider network 1**
- **Hardware (Multiple Nodes)**
  - **Corporation networks**
    - M1
    - M2
    - M3
  - **Internet**
  - **Default Virtual network**
  - **Firewall VNF (with BGP router)**
  - **IPS/WAF CNF**
  - **SDWAN VNF**
  - **EXT Router**
Traffic flow 1 – From internal machines to Internet

1. From internal machines to Internet
2. Through Corp networks
3. Through Hardware (Multiple Nodes)
4. To K8S Cluster
5. Resident 1 Applications (Micro-Services)
   - PCD
   - POD
   - POD
6. Resident 2 Applications (Micro-Services)
   - POD
   - POD
   - POD
7. Through Default Virtual network
8. Through Provider network 1
9. Through Provider network 2
10. To EXT Router
11. To Internet
Traffic flow 2 – Communication between Micro-Services and Internet

K8S Cluster

K8S Master

resident 1 Applications (Micro-Services)
- PCD
- POD
- POD

resident 2 Applications (Micro-Services)
- POD
- POD
- POD

Ingress (L7 LB)

BGP LB

Firewall VNF (with BGP router)

IPS/WAF CNF

SDWAN VNF

Provider network 1

Provider network 2

Hardware (Multiple Nodes)

Corp networks

M1

M2

M3

Internet

EXT Router

Hardware (Multiple Nodes)
## Cloud native NFV stack – Requirements Summary

<table>
<thead>
<tr>
<th>Network requirements</th>
<th>Performance requirements</th>
</tr>
</thead>
<tbody>
<tr>
<td>Co-existence of Network functions and applications</td>
<td>High throughput with respect to PPS &amp; bps</td>
</tr>
<tr>
<td>Network Functions as VNFs and CNFs</td>
<td>Low latency and low jitter</td>
</tr>
<tr>
<td>Multiple Virtual networks</td>
<td>Performance determinism (even with noisy neighbors)</td>
</tr>
<tr>
<td>Provider networks</td>
<td></td>
</tr>
<tr>
<td>Network function chaining – Inserting new functions</td>
<td></td>
</tr>
<tr>
<td>dynamically and auto reconfiguration</td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Security</th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>Attestation &amp; Verifications of infrastructure</td>
<td></td>
</tr>
<tr>
<td>Private key and password protection, IP protection</td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Generic requirements</th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>Network Service Orchestration at a central locations across multiple K8S clusters &amp; Openstack locations</td>
<td></td>
</tr>
<tr>
<td>Multi-residency support (soft isolation)</td>
<td></td>
</tr>
<tr>
<td>Telemetry &amp; Monitoring</td>
<td></td>
</tr>
<tr>
<td>Cloud Native NFV provisioning system that deploys not only K8S components, but also NFV related components</td>
<td></td>
</tr>
</tbody>
</table>
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)

**Kubelet**

**Virtlet**

**Multus CNI**

**Flannel**

**OVN-Multi CNI**

**SRIOV-NIC CNI**

**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](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]:
Route Controller (for static SFC): Being conceptualized
Dynamic SFC: TBD (Candidates: [https://networkservicemesh.io](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

**Kubelet**

**CRI Proxy**

**Virtlet**

**Docker**

**K8S API Server**

**Virt-controller**

**Virt-handler (Daemon-set)**

**Virt-launcher (part of each VM POD)**

**Virtlet based**

**kubevirt based**

**VM/VNF based workload support in K8S**

**Virtlet:**
[https://github.com/Mirantis/virtlet](https://github.com/Mirantis/virtlet)
[https://github.com/Mirantis/criproxy](https://github.com/Mirantis/criproxy)

**Kubevirt**
[https://github.com/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

**VNF/CNF (with DPDK)**

- Kubelet
- Virtlet
- Multus CNI

- Flannel
- OVN-Multi CNI
- SRIOV-NIC CNI

**SRIOV Enabled NIC HW**

**OVN-Multi 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

**OVN**

- Kubelet
- Virtlet
- Multus CNI

- Flannel
- OVN-Multi CNI
- OVS-DPDK CNI Virtio-user

**SRIOV-NIC CNI**

- Kubelet
- Virtlet
- Multus CNI

- Flannel
- OVN-Multi CNI
- SRIOV-NIC CNI

- SRIOV Enabled NIC HW
Performance requirements – Container networking acceleration with smartNICs

Requirements:

- High throughput for packet processing applications
- Performance determinism
- Network namespace offloads (TCP session search etc...)

VNF/CNf (with DPDK)
With virtio-user
Kubelet
Virtlet
Multus CNI

VNF/CNf/Micro service using vEth

VNF/CNf with AF-XDP

Open source: Community help
VNF/CNf/Micro service

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

<table>
<thead>
<tr>
<th>Requirements</th>
<th>Project</th>
<th>Descriptions</th>
</tr>
</thead>
</table>
| Performance determinism even when there are noisy neighbors (using affinity and isolation of cores) | [https://github.com/intel/CPU-Manager-for-Kubernetes](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](https://github.com/kubernetes/kubernetes/pull/50859) (WIP - Hugepages feature)  
[https://github.com/kubernetes/kubernetes/pull/50072](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](https://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 labels. |
Performance requirements – Accelerator related open source projects (As Device plugins)

<table>
<thead>
<tr>
<th>Requirements</th>
<th>Project</th>
<th>Descriptions</th>
</tr>
</thead>
<tbody>
<tr>
<td>Improve performance of Crypto and compression operations of VNFs/CNFs/Micro-services by leveraging hardware</td>
<td><a href="https://github.com/intel/intel-device-plugins-for-kubernetes/tree/master/cmd/qat_plugin">https://github.com/intel/intel-device-plugins-for-kubernetes/tree/master/cmd/qat_plugin</a></td>
<td>• 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</td>
</tr>
<tr>
<td></td>
<td>QAT - Plugin</td>
<td></td>
</tr>
<tr>
<td>Improve AI/Media performance using FPGA and GPU</td>
<td><a href="https://github.com/intel/intel-device-plugins-for-kubernetes/tree/master/cmd/fpga_plugin">https://github.com/intel/intel-device-plugins-for-kubernetes/tree/master/cmd/fpga_plugin</a></td>
<td>• Make GPU/FPGA available to VNFs/CNFs/Micro-Services as and when they request them.</td>
</tr>
<tr>
<td></td>
<td>(FPGA)</td>
<td></td>
</tr>
<tr>
<td></td>
<td><a href="https://github.com/intel/intel-device-plugins-for-kubernetes/tree/master/cmd/gpu_plugin">https://github.com/intel/intel-device-plugins-for-kubernetes/tree/master/cmd/gpu_plugin</a></td>
<td></td>
</tr>
<tr>
<td></td>
<td>(GPU)</td>
<td></td>
</tr>
</tbody>
</table>
**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](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](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

<table>
<thead>
<tr>
<th>Use cases/Apps</th>
<th>K8S App Components</th>
<th>Kubernetes</th>
<th>NFV Specific components</th>
<th>Host Operating System</th>
<th>Hardware</th>
</tr>
</thead>
<tbody>
<tr>
<td>Analytics as a Service</td>
<td>ISTIO</td>
<td>MetallB</td>
<td>Multus</td>
<td>Ubuntu</td>
<td>S1</td>
</tr>
<tr>
<td>SDWAN</td>
<td>gVisor</td>
<td>Flannel, OVN</td>
<td>SROVNIC</td>
<td>RH</td>
<td>S2</td>
</tr>
<tr>
<td>EdgeXFoundry</td>
<td>CollectD</td>
<td>Prometheus</td>
<td>OVN4K8SNFV</td>
<td>Clear</td>
<td>S3</td>
</tr>
<tr>
<td>ONAP</td>
<td>Virtual</td>
<td>OpenNESS</td>
<td>CMK</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Resident Mgr</td>
<td>Virtlet</td>
<td>Hugpep mgr</td>
<td>NFD</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Edge Mgr</td>
<td>Ceph/ Rook</td>
<td>Numa Mgr</td>
<td>QAT</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Inter-Site Mgr</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Config Mgr</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>K8S HPA</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

**ONAP**
- Resident Mgr
- Edge Mgr
- Inter-Site Mgr
- Config Mgr
- K8S HPA

**Kubernetes**
- Multus
- SROVNIC
- OVN4K8SNFV
- CMK
- Hugpep mgr
- NFD
- Numa Mgr
- QAT

**NFV Specific components**
- Multus
- SROVNIC
- OVN4K8SNFV
- CMK
- Hugpep mgr
- NFD
- Numa Mgr
- QAT

**Host Operating System**
- Ubuntu
- RH
- Clear

**Hardware**
- S1
- S2
- S3

**Global ZTP (Zero Touch Provisioning) system**
- KUD (with Cluster API)

**Infrastructure Provisioning & Configuration**
- Digital Rebar/ironic (or equivalent) for bare-metal provisioning
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.