KubeCon | CloudNativeCon

Europe 2019

High Performance Networking with KubeVirt

Doug Smith, Red Hat
Abdul Halim, Intel

Introductions

@dougbtv

Doug Smith

- Member of the NFV Partner Engineering team in Red Hat's Office of the CTO
- Focus on analyzing gaps in containerized workloads for NFV, including container networking & orchestration (e.g. Kube & OpenShift)
- Blog: https://dougbtv.com

@ahalim-intel

Abdul Halim

- Cloud Software Engineer at Network Platform Group, Intel
- Focused on enabling high-performance networking solution for VNF applications in K8s
- Blog post: @ Medium

Agenda

- KubeVirt Introduction
  - The how & why
- High performance networking - SR-IOV DP, Multus, NPWG
- KubeVirt networking demo
  - Using a VM with a VoIP workload running traffic over an SR-IOV interface
  - Including a DIY at home workshop to replicate all the moving parts in the demo

Who's familiar with...

KubeVirt

A virtual machine management add-on for Kubernetes. The aim is to provide a common ground for virtualization solutions on top of Kubernetes.

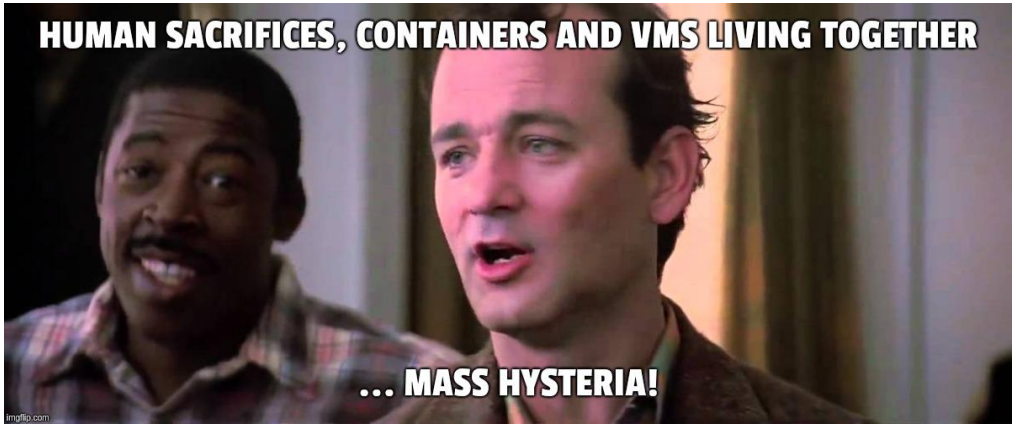
Enables you to put your VM workloads into Kubernetes utilizing a community founded & focused toolset.

We STILL need VMs!

Sometimes, you can't just containerize everything…

- Custom Kernels
- Security/Isolation
- Monolithic Applications
- Redesign/Rewrite that old application (ummm… yeah sure)
- Old legacy apps, may not have licensing, tool chains etc
- Some things just don't (or can't) fit in a Container
- Hardware abstractions are really useful in some cases
- The vendor delivered the product this way
- I just like my VM Pets, they're part of the family

VMs & Containers side by side in K8s

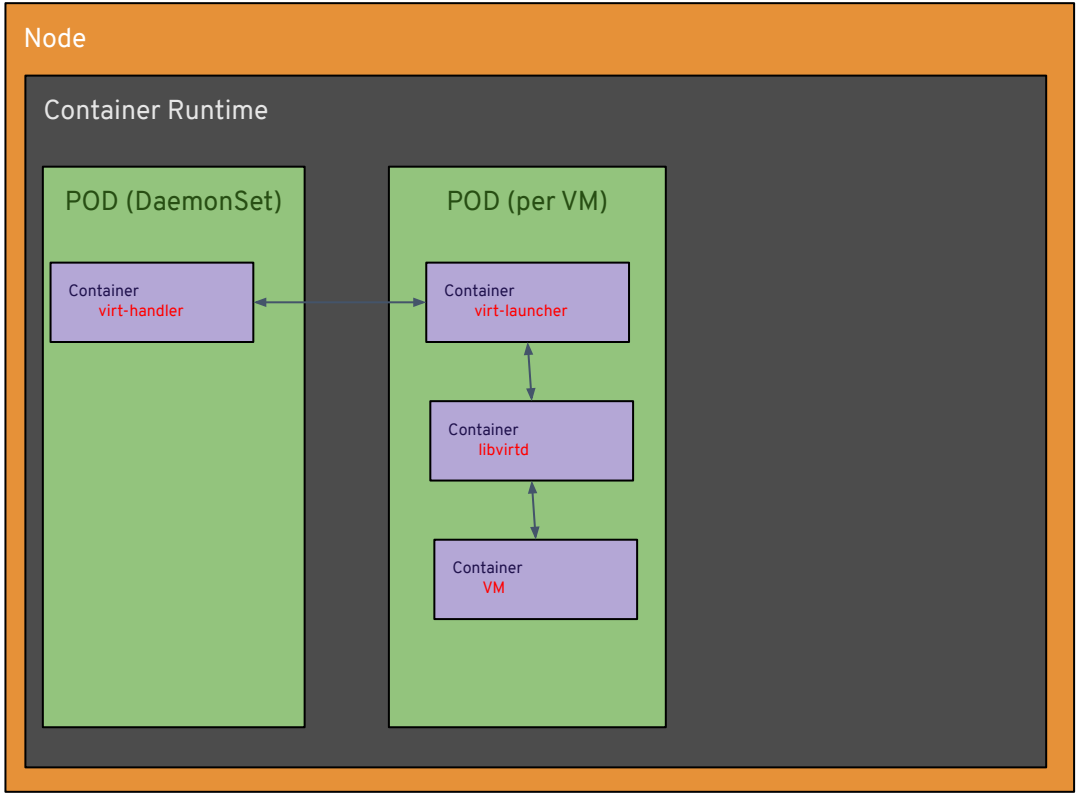


- Build, Modify and Deploy ALL THE THINGS in one way in one environment
- Single workflow for Devs/Ops
- Portability of VMs (wait.. whaaaat?)
- Only ONE environment to maintain!
- Migrate at your leisure (or don't)

A little bit of the “how”

- Extends an existing Kubernetes Cluster
- Just deploy it on your existing Cluster
  - Implemented as a Custom Resource Definition (CRD)
- Extends your K8s cluster to support VMs
- Sticks to K8s native approach as much as possible
  - Pod Networking
  - Storage supported in K8s works with your VMs
  - Just another Resource, same process using manifests etc

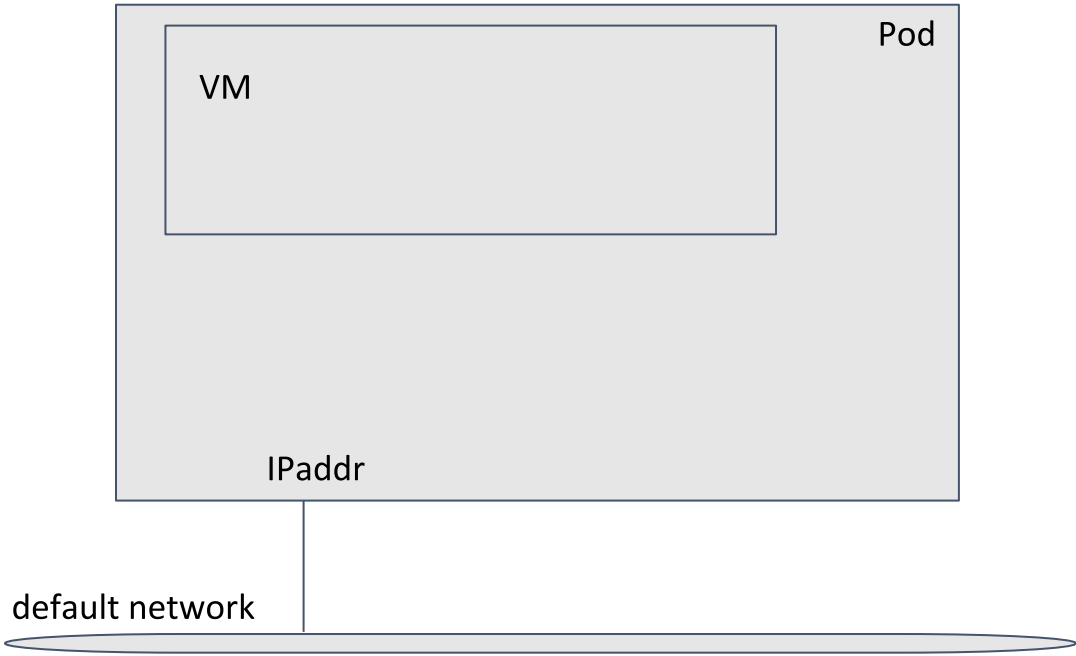
What does a KubeVirt VM look like?



KubeVirt Networking 101

Virtual Machine Instance:

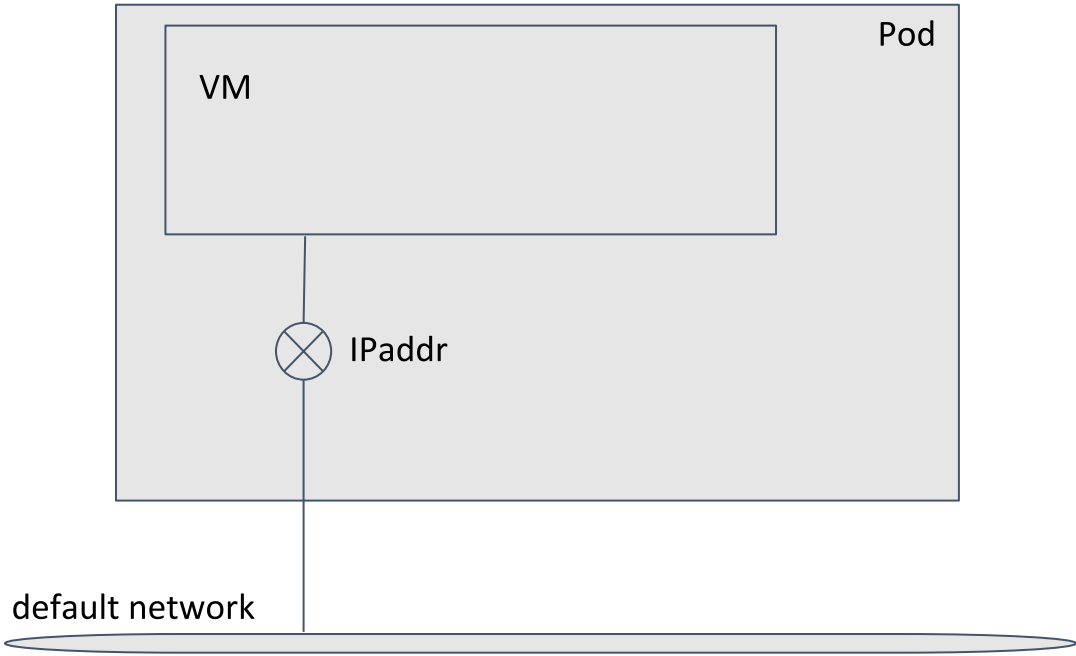
```
kind:
VirtualMachineInstance
spec:
  domain:
    devices:
      interfaces:
      - name: default
        masquerade: {}
  networks:
  - name: default
    pod: {}
```



KubeVirt Networking 101

Virtual Machine Instance:

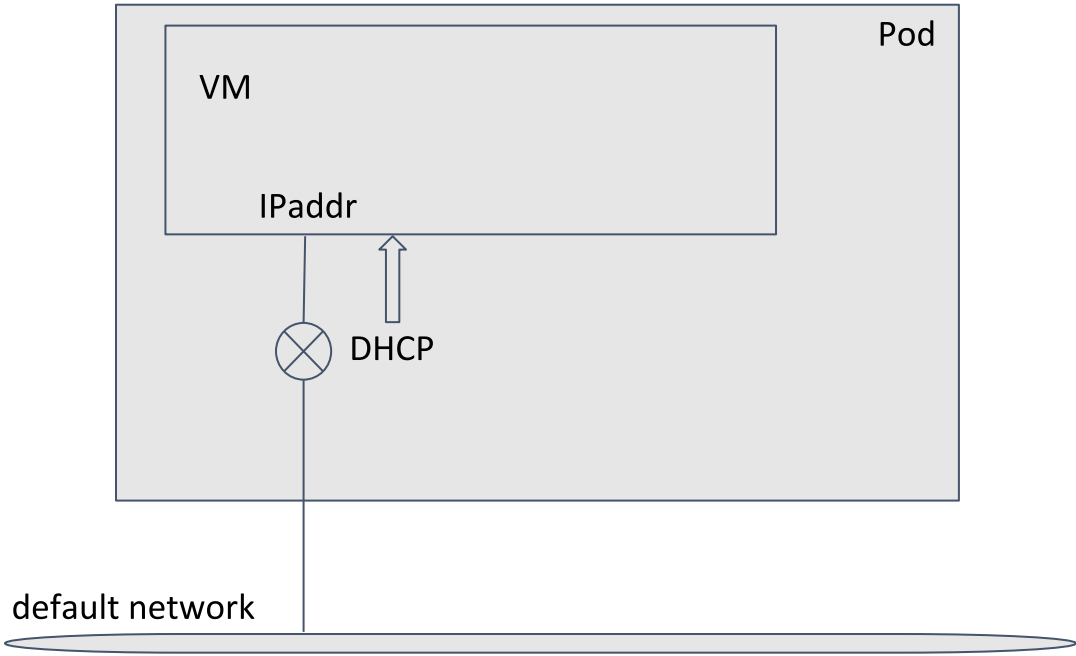
```
kind:
VirtualMachineInstance
spec:
  domain:
    devices:
      interfaces:
      - name: default
        masquerade: {}
  networks:
  - name: default
    pod: {}
```



KubeVirt Networking 101

Virtual Machine Instance:

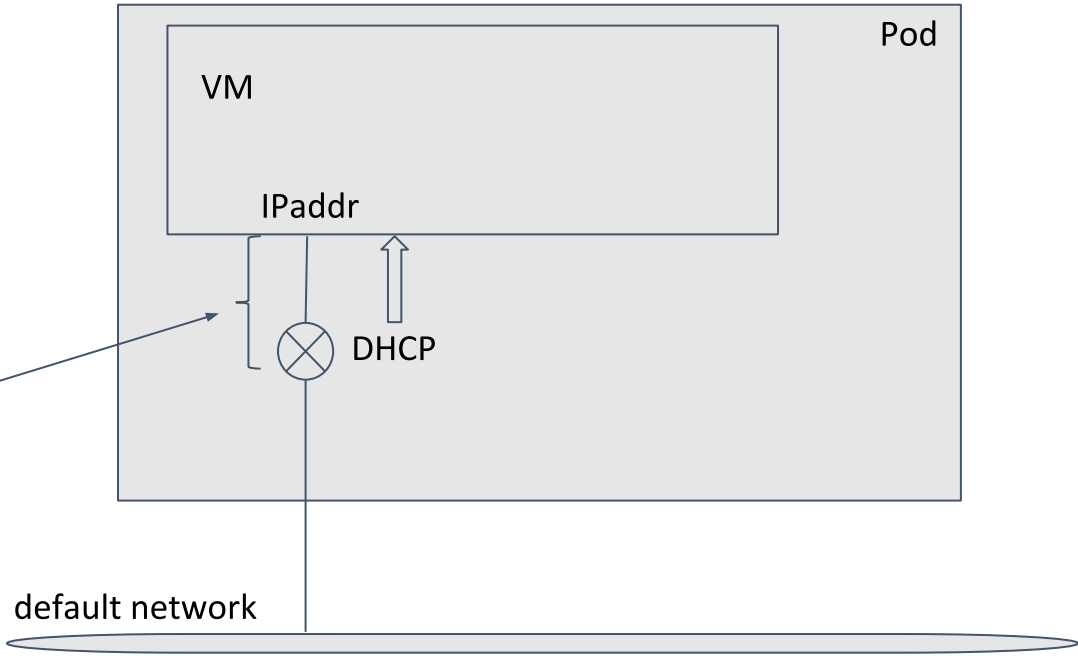
```
kind:
VirtualMachineInstance
spec:
  domain:
    devices:
      interfaces:
      - name: default
        masquerade: {}
  networks:
  - name: default
    pod: {}
```



KubeVirt Networking 101

Virtual Machine Instance:

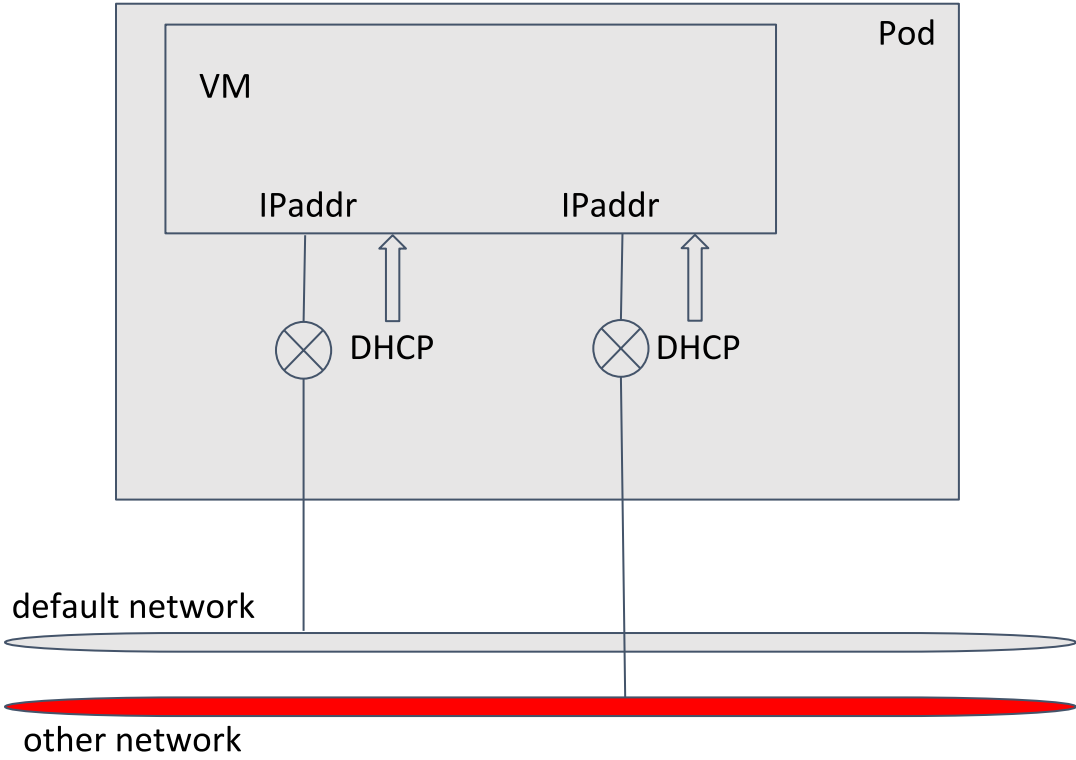
```
kind:
VirtualMachineInstance
spec:
  domain:
    devices:
      interfaces:
      - name: default
        masquerade: {}
  networks:
  - name: default
    pod: {}
```



KubeVirt Networking 101

Virtual Machine Instance:

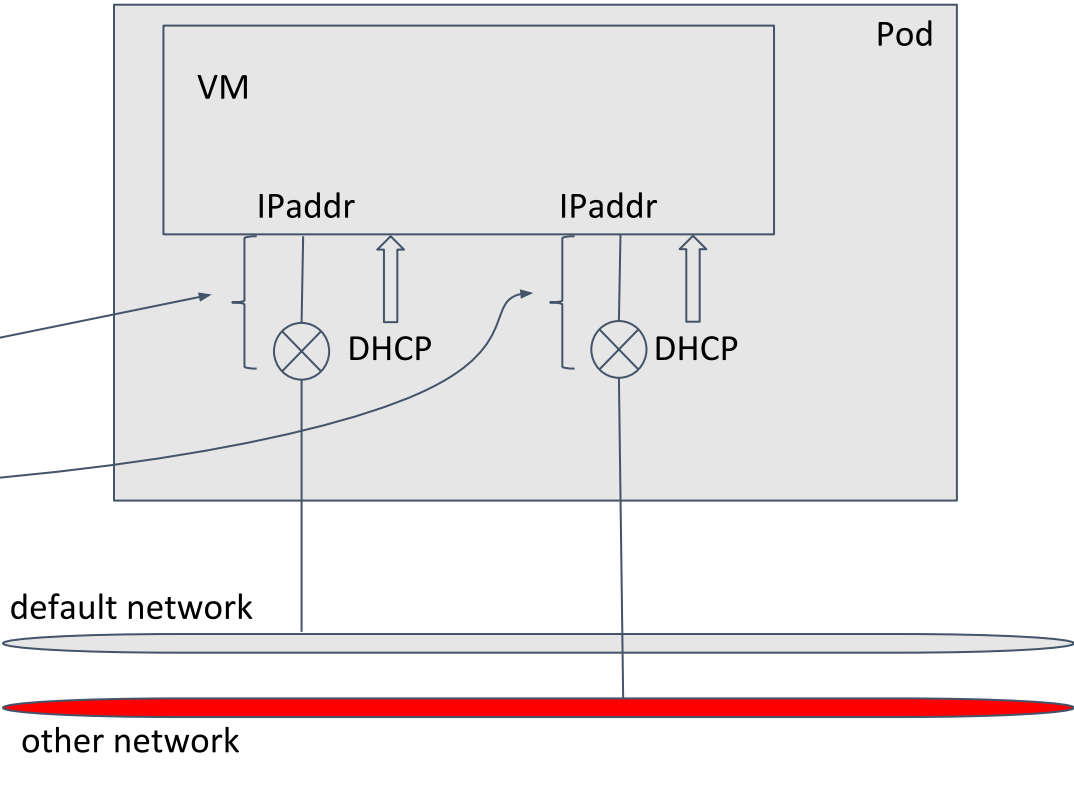
```
kind:
VirtualMachineInstance
spec:
  domain:
    devices:
      interfaces:
      - name: default
        masquerade: {}
      - name: other
        bridge: {}
  networks:
  - name: default
    pod: {}
  - name: other
    multus:
      networkName: other
```



KubeVirt Networking 101

Virtual Machine Instance:

```
kind:
VirtualMachineInstance
spec:
  domain:
    devices:
      interfaces:
      - name: default
        masquerade: {}
      - name: other
        bridge: {}
  networks:
  - name: default
    pod: {}
  - name: other
    multus:
      networkName: other
```



Leveraging many community technologies

KubeVirt employs a variety of technologies developed across the open source ecosystem…

- libvirt/KVM
- Standardized method of attaching multiple network interfaces using the Network Plumbing Working Group's specification
- Reference CNI plugins for Linux bridge connectivity, MAC addresses, …
- Operators framework
- SR-IOV Device Plugin

Network Plumbing Working Group

Founded by interested parties during Kubecon 2017 in Austin to address specifically multi-network requirements and related components.

- Currently focused on multiple network attachments
  - In an out-of-tree solution.
- Gather use-cases and propose standard
- Implement reference meta plugin
- Expand to further related advanced networking use-cases
  - Multiple IP addresses per pod, overlapping IP addresses, service abstraction, and so on.

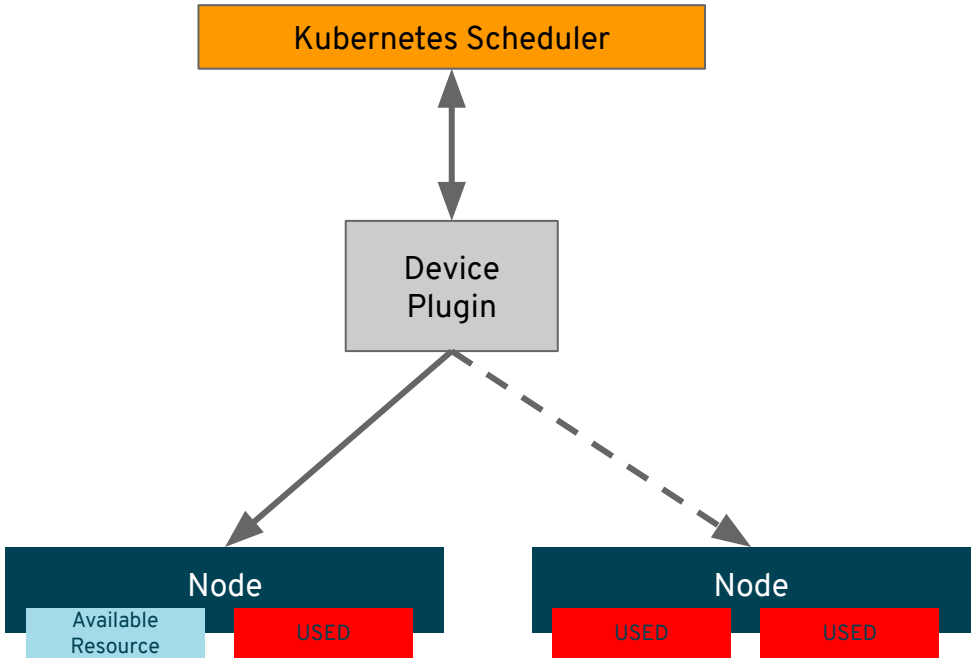
Multus CNI

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

Intel & Red Hat in collaboration with the community have developed Multus as a reference implementation of the Network Plumbing Working Group specification.

Multus CNI is a “meta plugin” for Kubernetes CNI which enables one to create multiple network interfaces per pod. It allows one to assign a CNI plugin to each interface created in the pod.

Leveraging Device Plugins



Device plugins are used in Kubernetes as a way to give the scheduler awareness of limited resources on a given node (typically hardware resources), enabling the scheduling of workloads on nodes with available resources.

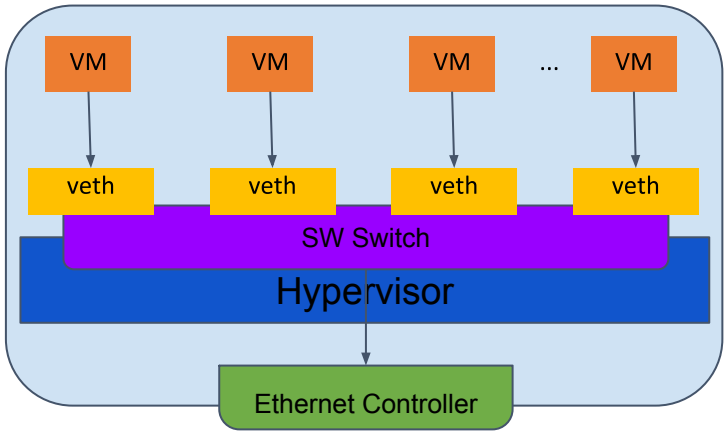
For high performance networking, KubeVirt utilizes SR-IOV devices which have a limited number of resources, devices plugins allow KubeVirt to put VMs on nodes with SR-IOV resources available.

SR-IOV Network device plugin

https://github.com/intel/sriov-network-device-plugin

Why SR-IOV Networking?

- Workload gets a Virtual Ethernet interface
- Shared access to networking HW with an additional overhead
- Poor network performance

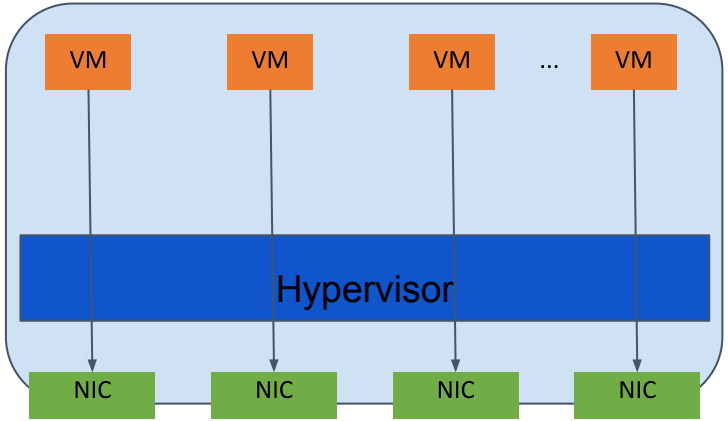


Non SR-IOV Network

Why SR-IOV Networking?

What if we could skip this additional overhead?

- Give direct access to a NIC
- Without SR-IOV, limited number of I/O ports(PCIe) → Limited scalability

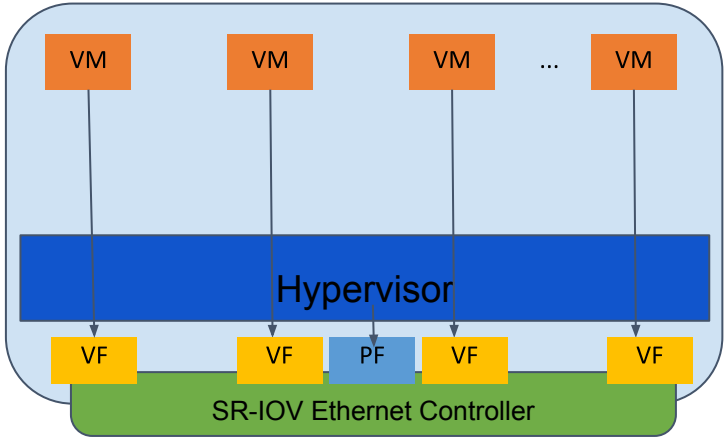


Non SR-IOV Network

Why SR-IOV Networking?

SR-IOV is a PCI-SIG standard specification to mitigate this limitation

- Introduces the PCIe **Physical Function**(PF) and **Virtual Function**(VF)
- **PF**:
  - Full featured PCIe functions
  - Have full configuration resources
  - Typically SR-IOV NIC PF has L2 sorter/switcher, link controls etc.
- **VF**:
  - Light-weight PCIe functions - no configuration resources
  - Has own BARs/registers
  - Each SR-IOV NIC VF has dedicated Tx/Rx queues
  - Can move data in and out of DMA



SR-IOV Network

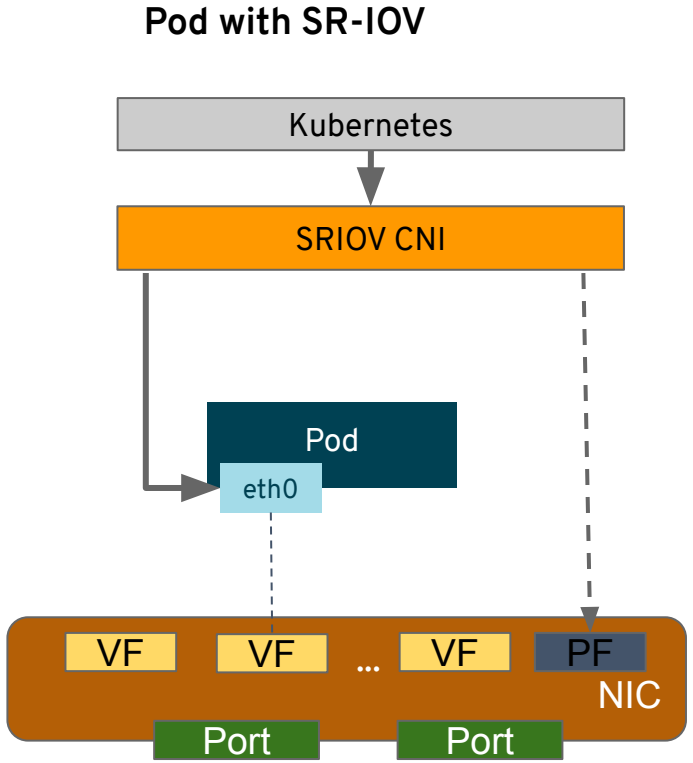
Evolution of SR-IOV Networking in K8s

The [sriov-cni] plugin created by Tencent is a first CNI plugin to enable SR-IOV networking in K8s.

- Adds a VF in a Pod from a specific PF
- No mechanism to detect in-use and not-in-use VFs

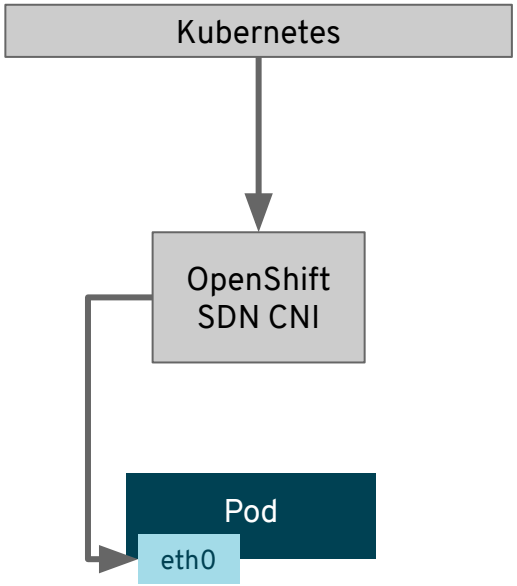
Extension:

- VF management capability
- VF with DPDK compatible driver
- https://github.com/intel/sriov-cni/

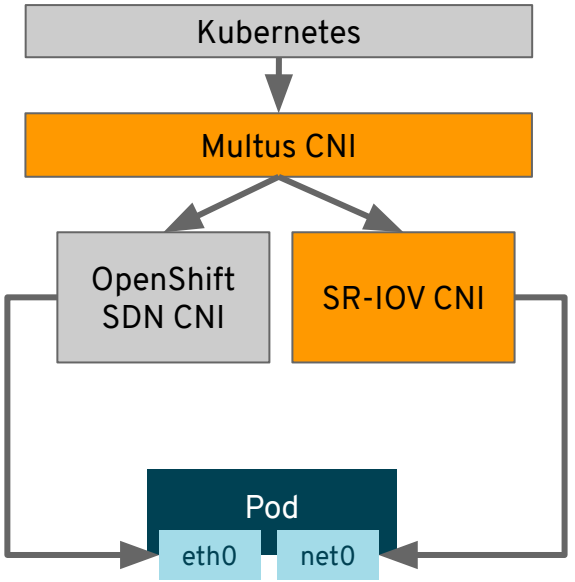


SR-IOV: with Multus & SR-IOV CNI

Pod without Multus



Pod with Multus + SR-IOV



Limitation with SR-IOV CNI

- K8s scheduler is unaware of SR-IOV network resources
- NUMA alignment with other resources(CPUs, memory etc.) is not possible
- VFs with vfio/uio drivers, cgroup isolation is not possible

Kubernetes Device plugin

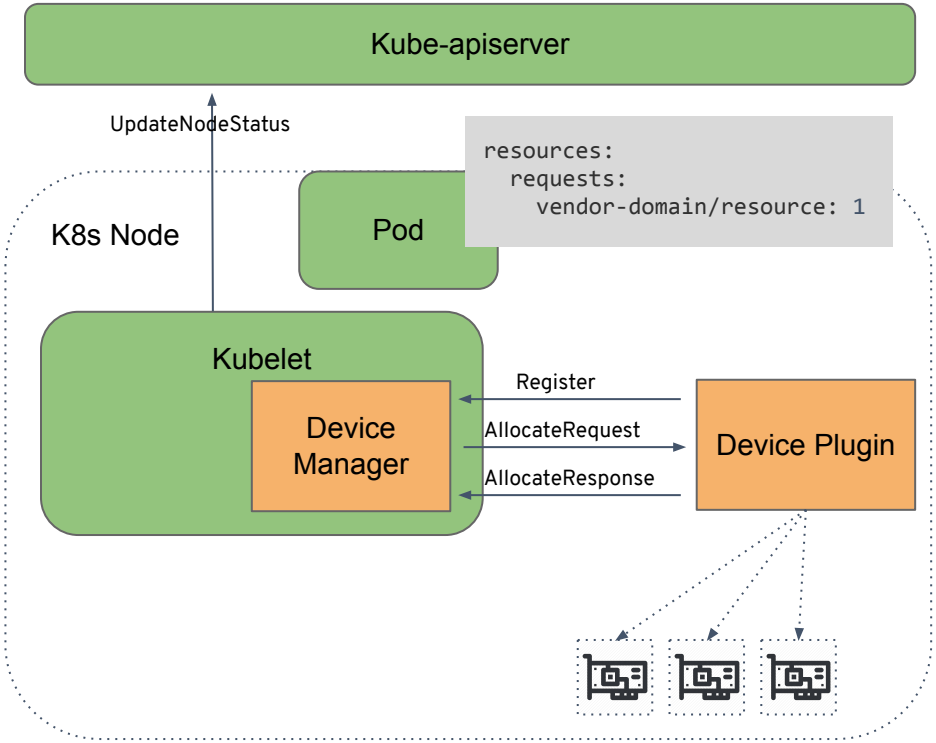
K8s device plugin framework

- Resource discovery
- Resource advertising
- Resource allocation
- Resource health-check

Device plugin:

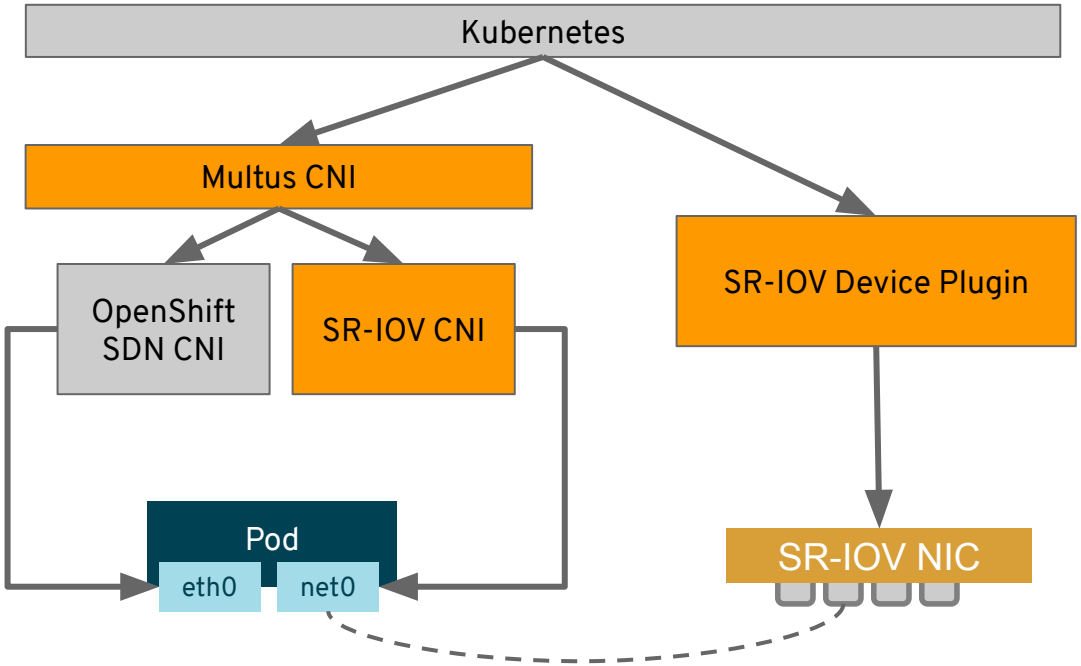
- A gRPC server
- Can be deployed manually or as a DaemonSet

- Workloads make requests for devices via resource requests
- K8s scheduler places workloads on a node that has this capacity



SR-IOV Device Plugin

- SR-IOV device plugin
  - Discovery &
  - Advertising of SR-IOV network resources
- SR-IOV CNI
  - Configuring of VFs

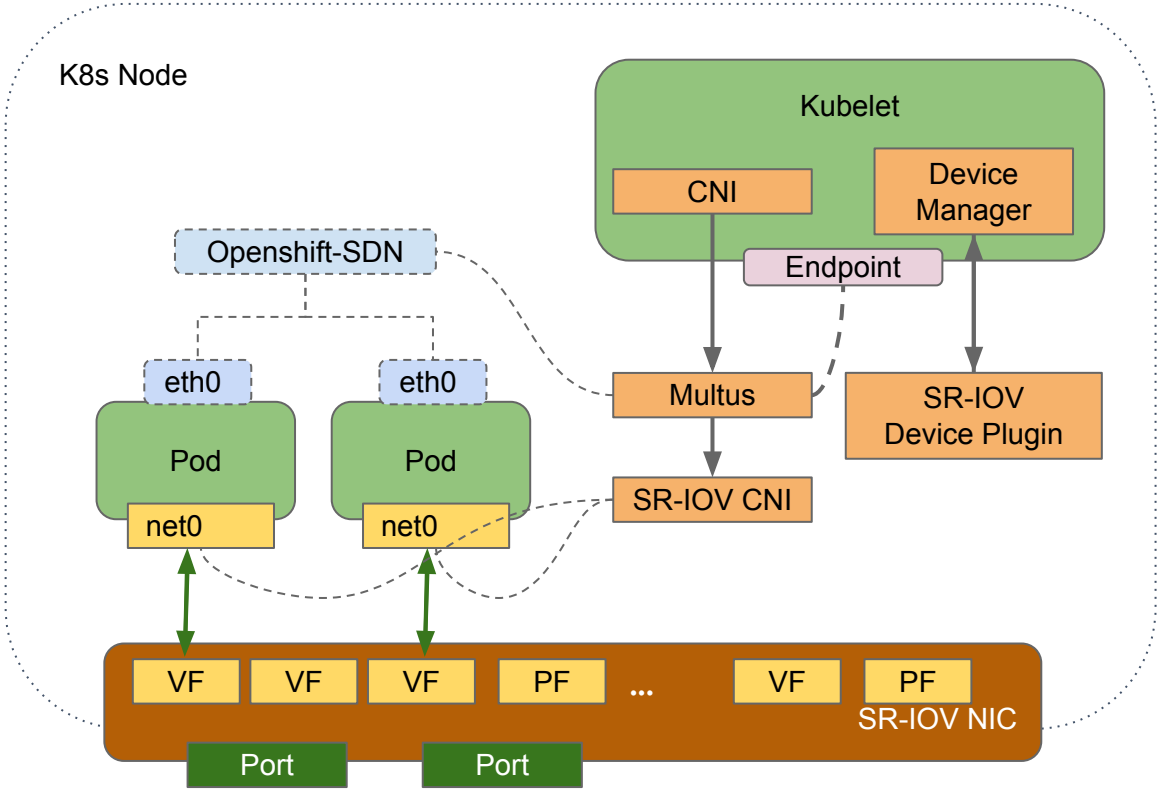


https://github.com/intel/sriov-network-device-plugin

SR-IOV Device Plugin

Pod creation:

- K8s scheduler assigns Pods to node that can satisfy SR-IOV network resource requests
- Kubelet takes care of all resource allocation and updates its internal checkpoint
- Multus retrieves device information from Kubelet using net-attach CRD
- Multus provides device ID to SR-IOV CNI for bringing it up inside that Pod



OpenShift SR-IOV Roadmap

Functionality

- SR-IOV Network Operator
- SR-IOV RDMA (RoCE) support
- SR-IOV Admission Controller
- Resource Class API

Performance

- CPU Manager (Isolated CPUs, Sibling CPUs)
- NUMA awareness (Topology Manager)

References

- https://github.com/intel/multus-cni
- https://github.com/intel/sriov-cni/
- https://github.com/intel/sriov-network-device-plugin
- SR-IOV device plugin proposal - https://docs.google.com/document/d/1Ewe9Of84GkP0b2Q2PC0y9RVZNkN2WeVEagX9m99Nrzc/
- Join us on Slack

SR-IOV Networking DEMO

Please come join us @ Intel booth(G-13)

- Dynamic SR-IOV Network device plugin deployment in action
- SR-IOV network resource orchestration
- SR-IOV network performance benchmarking results
- Intel QuickAssist© Device plugin demo

DEMO MATERIAL NOTES

- (Doug will make these notes as the demo commences)
- Uses an operator to manage the components.

KubeVirt + SR-IOV Demo

Thank you! Questions?

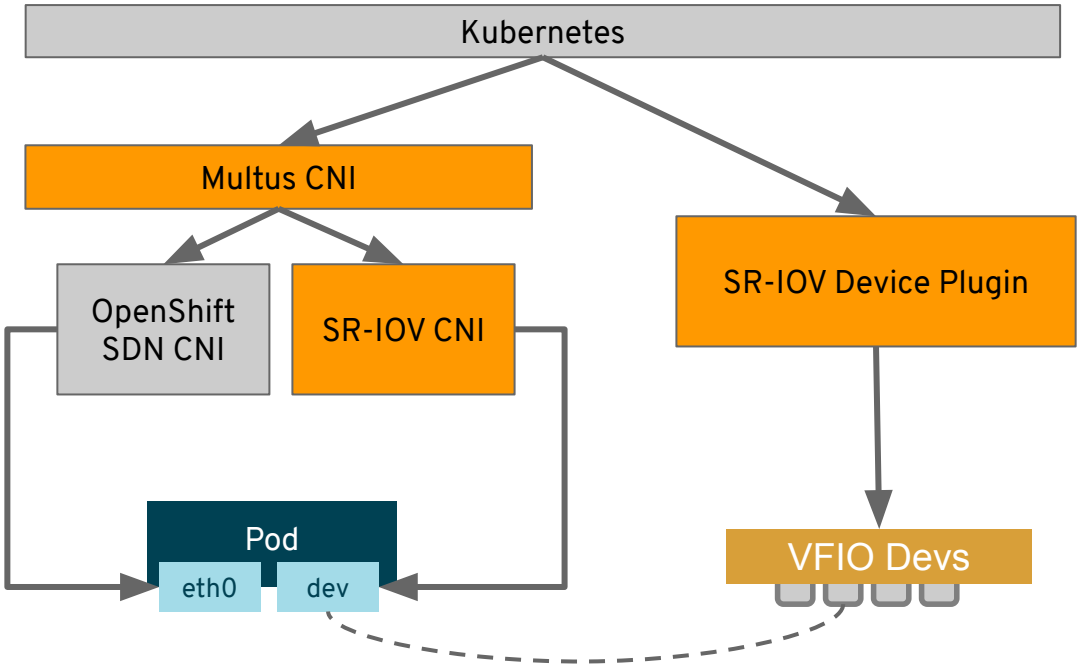
Acknowledgements

- Thanks to Ihar Hrachyshka, Petr Horacek, Sebastian Scheinkman for their assistance with KubeVirt
- John Griffith for the source material for the KubeVirt introduction slides

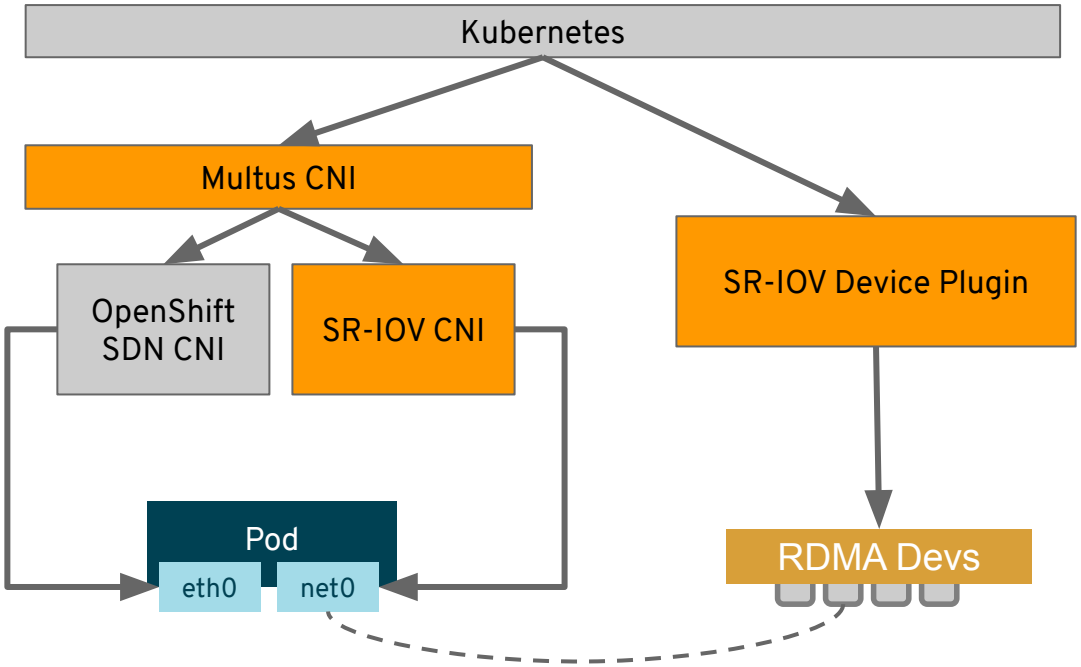
{intentionally blank}

(put reference material after this slide)

Roadmap (SR-IOV DPDK)



Roadmap (SR-IOV RDMA)



Kubevirt Reference Material

Containers are the new “cool kids”

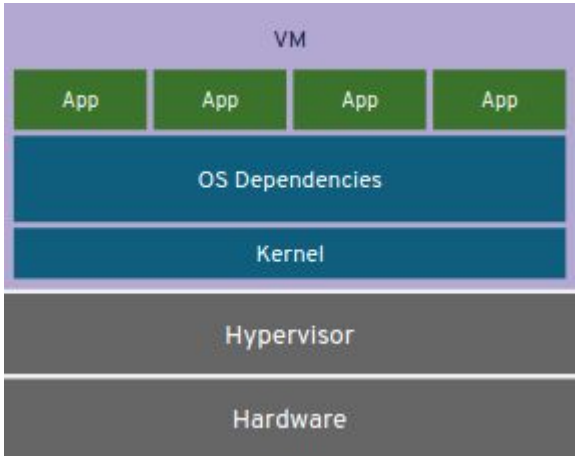
Automation and Declarative Systems RULE!

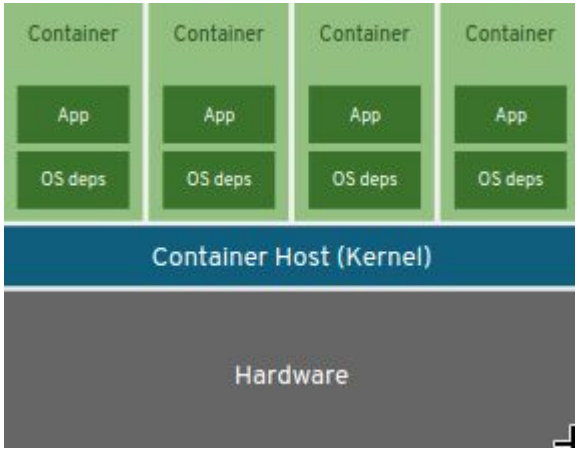
Nobody (or almost nobody) thinks:

> *“Hey, I should do a really cool presentation on how I maintain and run a custom production floor app built on Windows 2000 Server that can’t be replaced or run on anything but our customized version of Windows 2000!”*

Well, ok, that “might” be interesting… but not the type of thing you line out the door to see probably.

Quick Refresh, VM vs Container





Run two environments, problem solved!

Legacy VM Env

- VMware
- OpenStack
- CloudStack
- …

Shiny new/cool stuff

- Swarm
- Kubernetes
- …

I can have it all!!! VM's when I need them, and Containers/K8s for everything else! I'm a "Happy Puppy"!

Run two environments, now I have a new problem!

Double Trouble

- Two environments to maintain
- Two WorkFlows
- Two platforms to manage

Managing infrastructure and the platforms on top of them can be a challenge, adding more of these can make for a “sad puppy”

Run K8s on top of my Virtualization

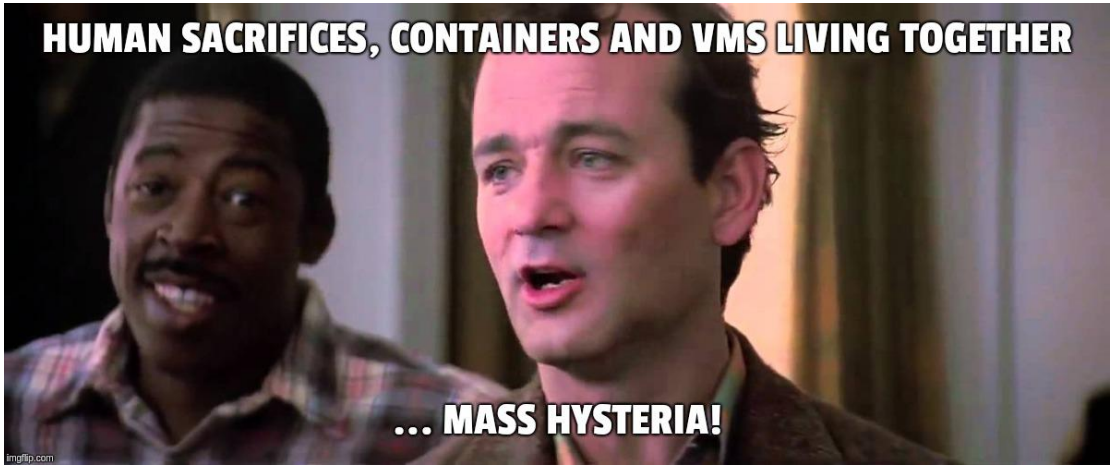
Not a bad approach

- Provides some flexibility
- Marks off the check boxes
- Fits the mental model of the Public Cloud Providers

Ummm… wait though

- I'm still managing two environments!
- I want bare metal performance
- I want K8s workflows for ALL MY THINGS
- K8s is what I want, and I want it NOW!

Running VMs AND Containers side by side in K8s



- Build, Modify and Deploy ALL THE THINGS in one way in one environment
- Single workflow for Devs/Ops
- Portability of VMs (wait.. whaaaat?)
- Only ONE environment to maintain!
- Migrate at your leisure (or don’t)

There’s a few approaches to this sort of thing

Mostly using new CRIs to control VMs (“*not that there’s anything wrong with that”*)

- RancherVM
- Kata-Containers
- Mirantis Virtlet
- Google gvisor

For now, *most* of these are focused on slightly different problems, mostly around isolation/security.

They’re cool too; you should check them out but let’s talk about a different approach...

KubeVirt

Not running Containers in VMs,instead let’s run and expose VMs in Containers!!

Then you’ll have:

- Unified platform to build, modify and deploy applications (Container or VM)
- BOTH VM workloads and Container workloads using the same automation, and K8s APIs
- Treat VMs just like any other K8's workload, while preserving its true “VMness”
- Bring your legacy VMs to a shiny new environment
- Working on leveraging virt-v2v to easily migrate existing VMs into the Cluster
- Main focus is enabling the migration of legacy apps/VMs in to a K8s world

A little bit of the “how”

- Extends an existing Kubernetes Cluster
- Just deploy it on your existing Cluster
  - Implemented as a Custom Resource Definition (CRD)
- Extends your K8s cluster to support VMs
- Sticks to K8s native approach as much as possible
  - Pod Networking
  - Storage supported in K8s works with your VMs
  - Just another Resource, same process using manifests etc

Adding it to a running K8s Cluster

```
> export VERSION="v0.7.0"
> kubectl create -f https://github.com/kubevirt/kubevirt/releases/download/$VERSION/kubevirt.yaml
.....................
.....................
.....................
> curl -L -o virtctl \
    https://github.com/kubevirt/kubevirt/releases/download/$VERSION/virtctl-$VERSION-linux-amd64
.....................
.....................
.....................
> chmod +x virtctl
```

Notice we didn't say anything about modifying your K8s Cluster or it's Nodes, you don't have to.

**In a production environment you would want to install KVM modules and enable nested virt*

Now it just works like any other K8s resource

```
> kubeclt create -f my-firstvm.yaml
virtualmachine.kubevirt.io "my-firstvm" created
virtualmachineinstancepreset.kubevirt.io "small" created
```

You can specify things in the manifest like you do with any other POD/Application, Volumes, Networking etc

Nice Extras

Kubevirt also has some extended projects like containerized-data-importer (CDI)

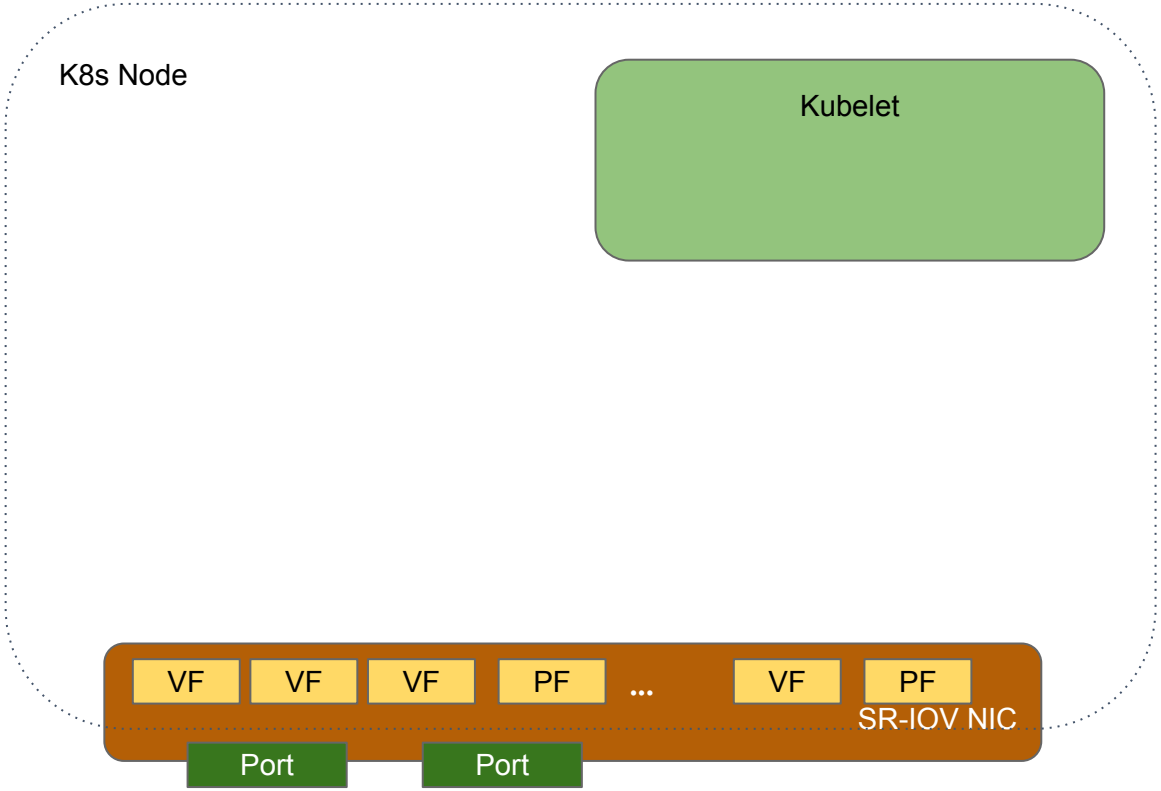
- Another Controller Add On for your Cluster
- Import existing Data or Images to PVCs in your Cluster
- Host assisted Cloning of PVs
- Basically “things to move data around more sensibly”

Backup

SR-IOV Device Plugin

Node with SR-IOV NICs:

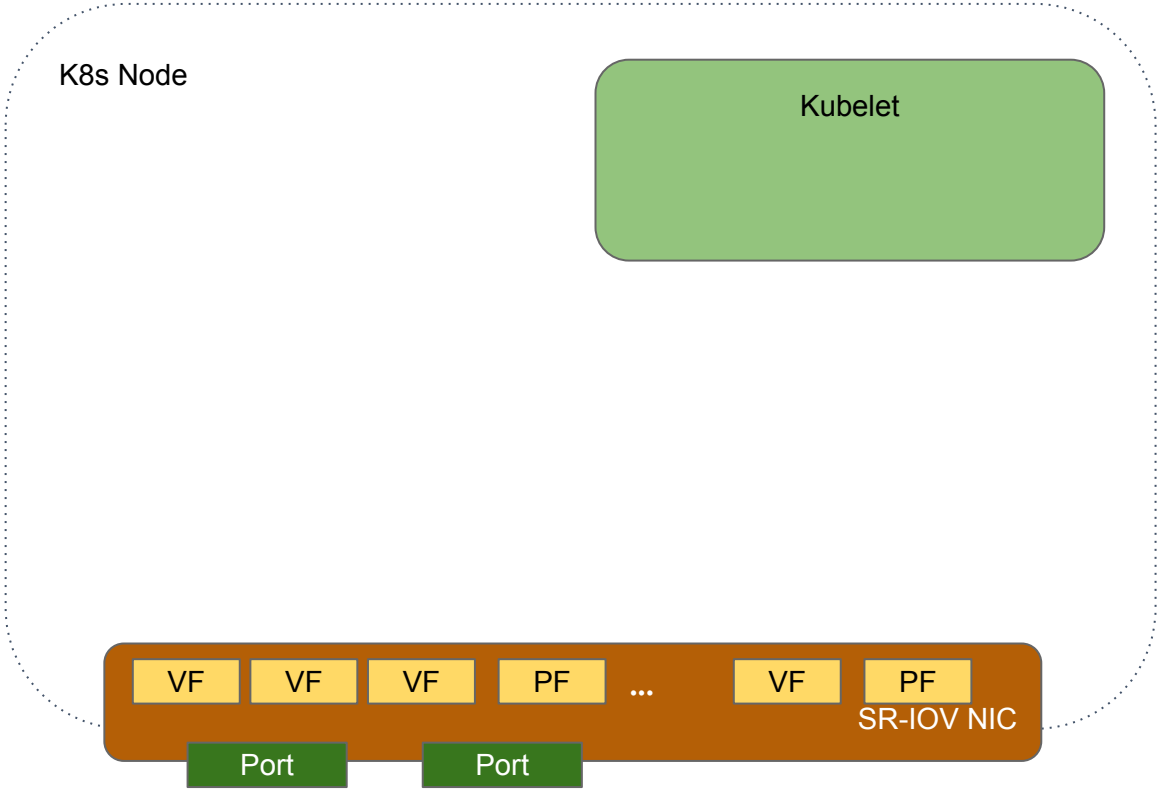
- Discovery
- Register with K8s
- Provisioning
- Attaching



SR-IOV Device Plugin

Bootstrapping SR-IOV network resources:

- Initialize SR-IOV NIC
- Device resource definition
- Deploy SR-IOV device plugin

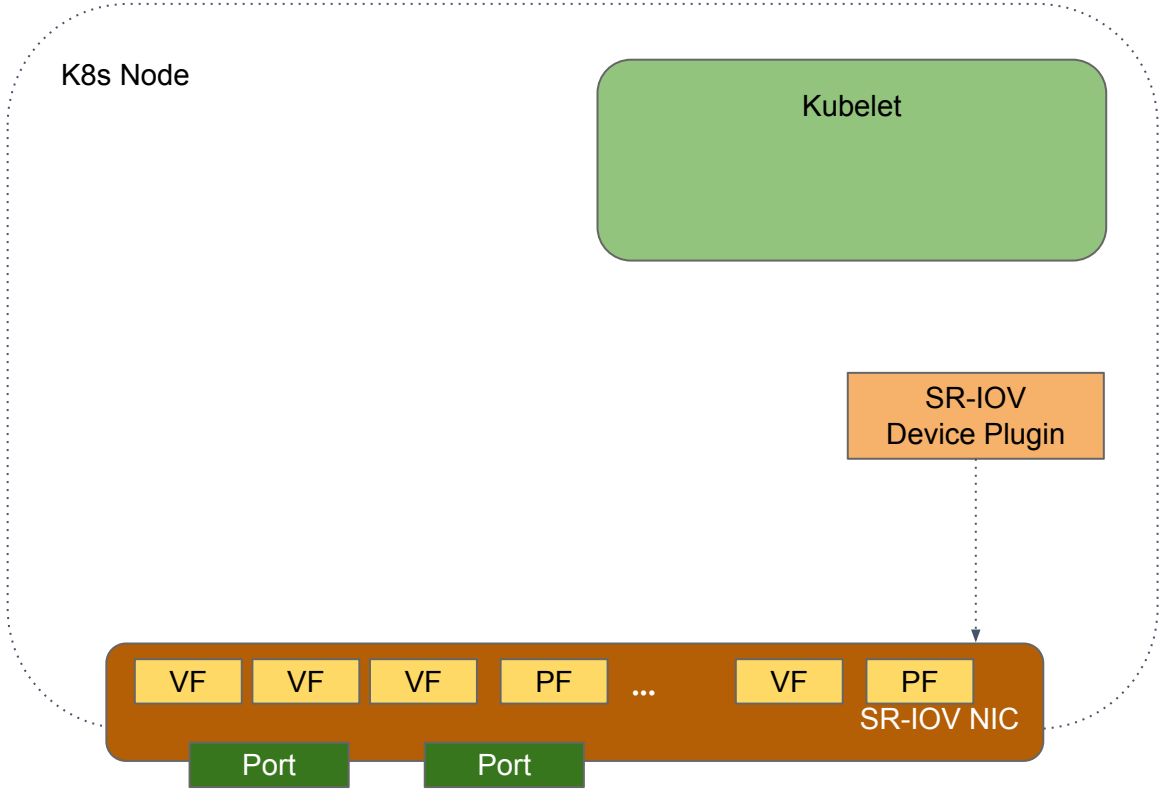


SR-IOV Device Plugin

Resource configurations:

- Define SR-IOV network resource pool
- Node specific file: /etc/pcidp/config.json

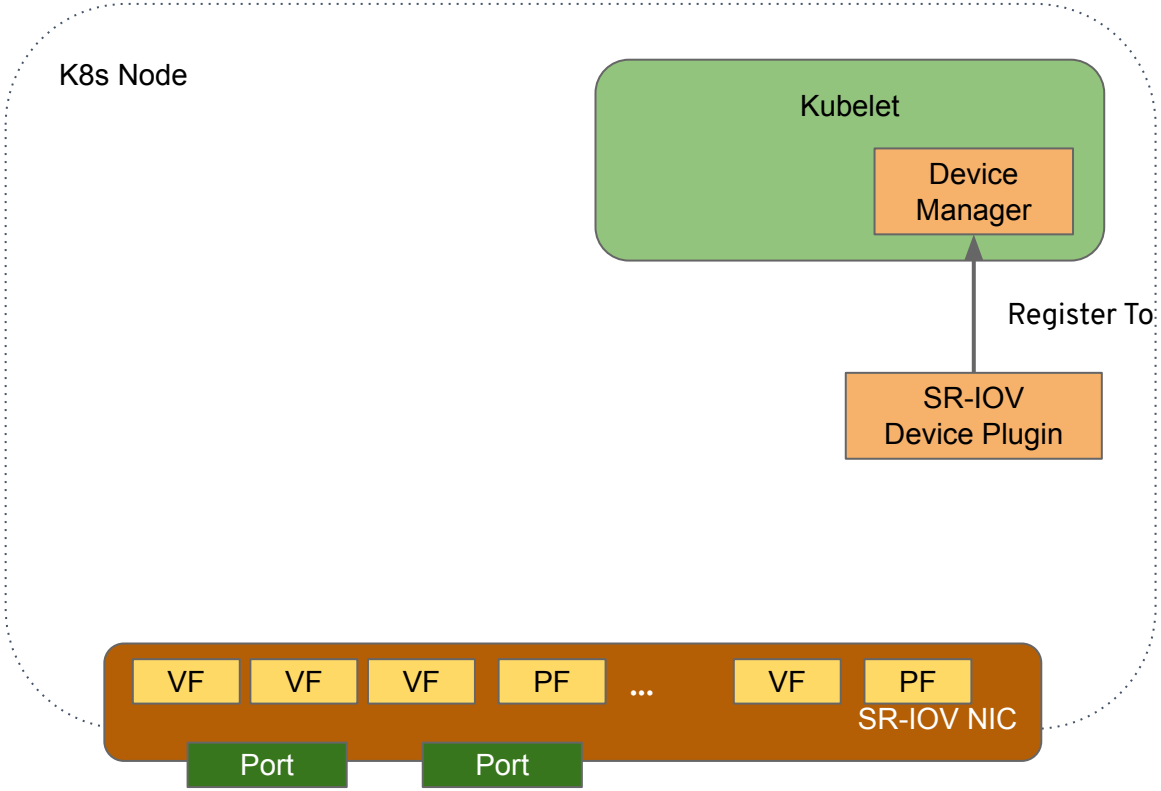
```
{
  "resourceList":
  [
    {
      "resourceName": "sriov_net_A",
      "rootDevices": ["02:00.0", "02:00.2"],
      "deviceType": "netdevice"
    },
    {
      "resourceName": "sriov_net_B",
      "rootDevices": ["02:00.1", "02:00.3"],
      "deviceType": "netdevice"
    }
  ]
}
```



SR-IOV Device Plugin

Node status

```
Name:
k8s-node1.ir.intel.com
Capacity:
 cpu:                                8
 ephemeral-storage:
184447308Ki
 hugepages-1Gi:                      0
 hugepages-2Mi:                      8Gi
 intel.com/sriov_net_A:              8
 intel.com/sriov_net_B:              4
 memory:
16371628Ki
 pods:                               1k
Allocatable:
 cpu:                                8
 ephemeral-storage:
169986638772
 hugepages-1Gi:                      0
 hugepages-2Mi:                      8Gi
 intel.com/sriov_net_A:              8
 intel.com/sriov_net_B:              4
 memory:
7880620Ki
 pods:                               1k
```



SR-IOV Device Plugin

Node status

```
Name:
k8s-node1.ir.intel.com
Capacity:
 cpu:                                8
 ephemeral-storage:
184447308Ki
 hugepages-1Gi:                      0
 hugepages-2Mi:                      8Gi
 intel.com/sriov_net_A:              8
 intel.com/sriov_net_B:              4
 memory:
16371628Ki
 pods:                               1k
Allocatable:
 cpu:                                8
 ephemeral-storage:
169986638772
 hugepages-1Gi:                      0
 hugepages-2Mi:                      8Gi
 intel.com/sriov_net_A:              8
 intel.com/sriov_net_B:              4
 memory:
7880620Ki
 pods:                               1k
```

Net-attach CRD

```
apiVersion: "k8s.cni.cncf.io/v1"
kind: NetworkAttachmentDefinition
metadata:
  name: sriov-net
  annotations:

k8s.v1.cni.cncf.io/resourceName:
intel.com/sriov_net_A
spec:
  config: '{
  "type": "sriov",
  "name": "sriov-network""
  }
}'
```

Pod Specs

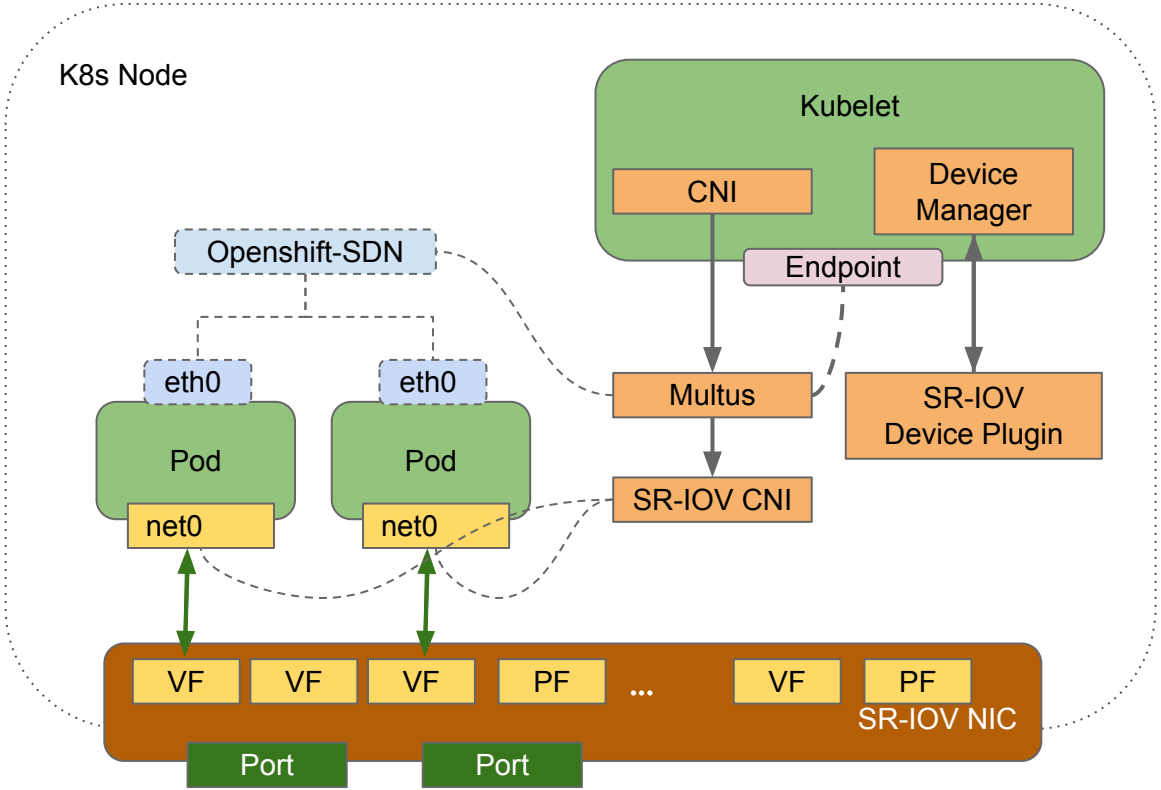
```
apiVersion: v1
kind: Pod
metadata:
  name: testpod
  annotations:

k8s.v1.cni.cncf.io/networks:
openshift-sdn, sriov-net
spec:
  containers:
  - name: appcntr1
    resources:
      requests:

intel.com/sriov_net_A: 1
      limits:

intel.com/sriov_net_A: 1
```

SR-IOV Device Plugin



KubeVirt Networking 101

Virtual Machine Instance:

```
kind:
VirtualMachineInstance
spec:
  domain:
    devices:
      interfaces:
      - name: default
        masquerade: {}
  networks:
  - name: default
    pod: {}
```