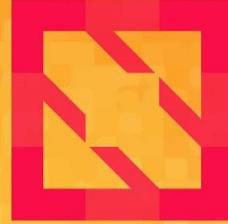




KubeCon



CloudNativeCon

---

North America 2019

---



# Virtualized GPU workloads on KubeVirt

***David Vossel***

*Principal Software Engineer  
Red Hat*

***Vishesh Tanksale***

*Senior Software Engineer  
NVIDIA*



KubeCon



CloudNativeCon

North America 2019



# What isn't KubeVirt?

Quick clarification on what KubeVirt is not.

- **NOT** involved with managing AWS or GCP instances
- **NOT** a competitor to Firecracker or Kata Containers
- **NOT** a container runtime replacement

# What is KubeVirt?



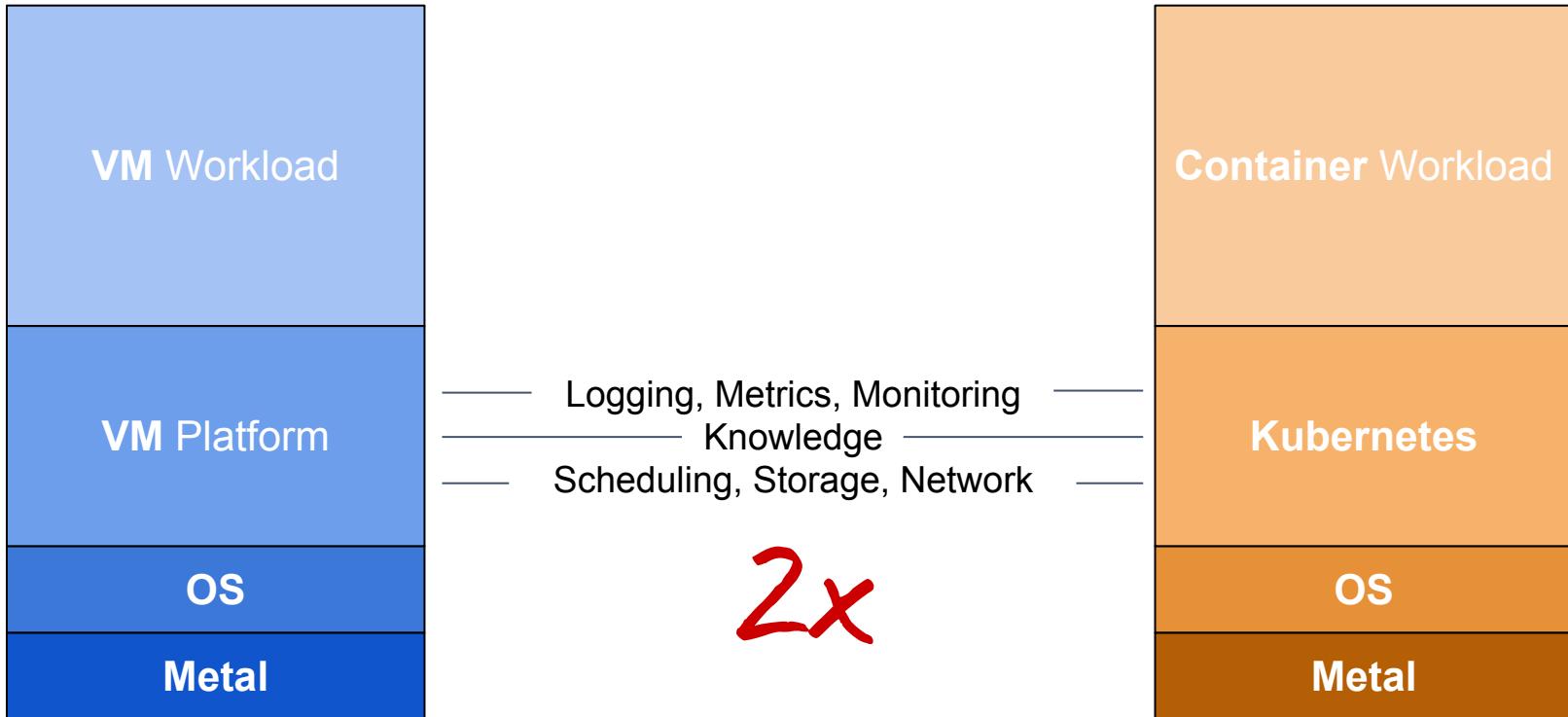
**“KubeVirt is a Kubernetes extension that allows running traditional VM workloads natively side by side with Container workloads.”**

# But... Why KubeVirt?

- Already have on-premise solutions like Openstack, oVirt
- And then there's the public cloud, AWS, GCP, Azure.
- Why are we doing this VM management stuff yet again?

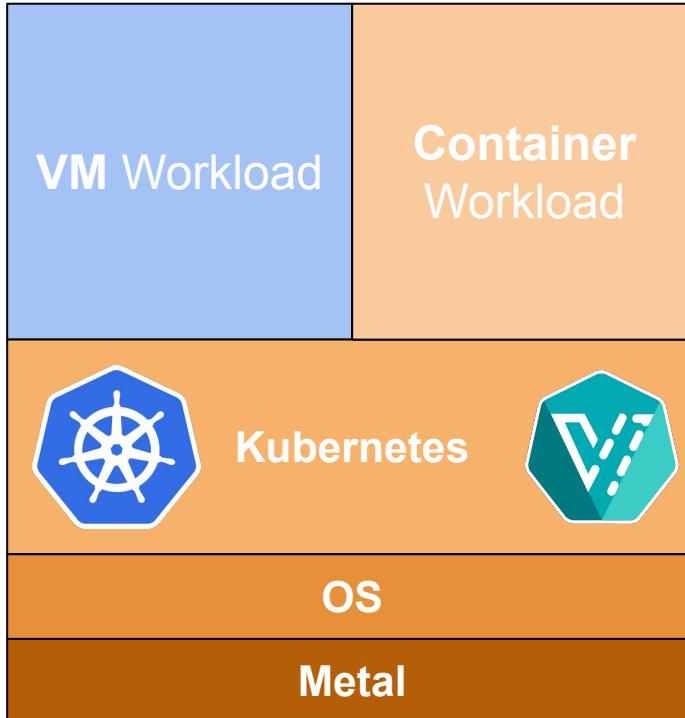
# Infrastructure Convergence

## Old Way ... Multiple Workloads, Multiple Stacks



# Infrastructure Convergence

## KubeVirt way... Multiple Workloads, One Stack



- Logging, Metrics, Monitoring
- Knowledge
- Scheduling, Storage, Network

1x

# Workflow Convergence



KubeCon

CloudNativeCon

North America 2019

- Converging VM management into **container management workflows**.
- Same tooling (**kubectl**)
- **Declarative API** for VM management (just like pods, deployments, etc...)

# Workflow Convergence

- Converging VM management into **container management workflows**.
- Same tooling (**kubectl**)
- **Declarative API** for VM management (just like pods, deployments, etc...)

```
$ cat <<EOF | kubectl create -f -
```

```
apiVersion: v1
kind: Pod
```

```
...
```

```
spec:
  containers:
    - name: nginx
      image: nginx
```

```
...
```

# Workflow Convergence

- Converging VM management into **container management workflows**.
- Same tooling (**kubectl**)
- **Declarative API** for VM management (just like pods, deployments, etc...)

```
$ cat <<EOF | kubectl create -f -
apiVersion: v1
kind: Pod
...
spec:
  containers:
    - name: nginx
      image: nginx
...
EOF
```

```
$ cat <<EOF | kubectl create -f -
apiVersion: kubevirt.io/v1alpha1
kind: VirtualMachineInstance
...
spec:
  domain:
    cpu:
      cores: 2
    devices:
      disk: fedora29
...
EOF
```



KubeCon



CloudNativeCon

North America 2019

# KubeVirt Architecture



# Virtual Machines in Pods

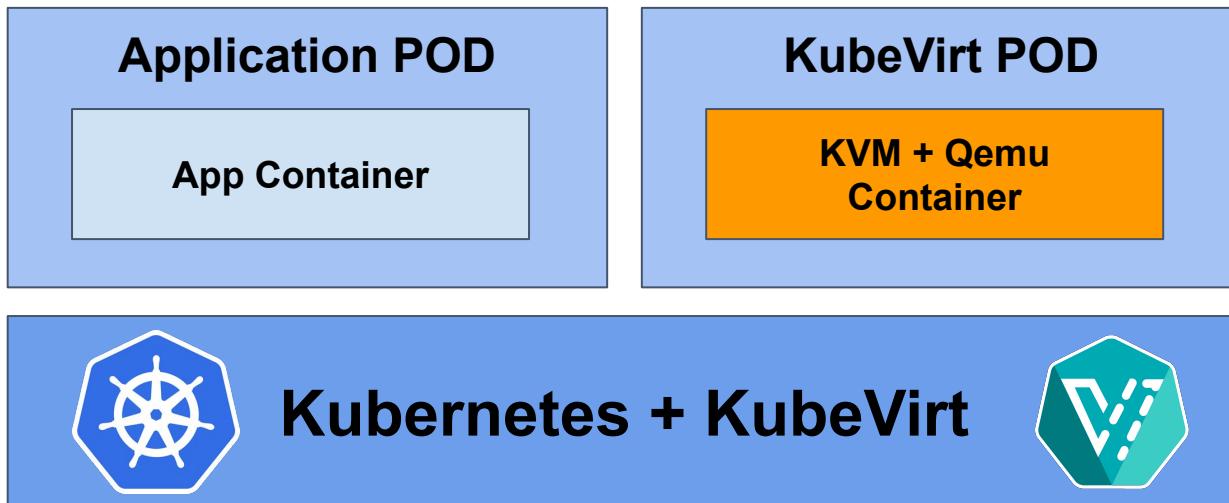


KubeCon

CloudNativeCon

North America 2019

KubeVirt VM is a KVM+qemu process running inside a pod



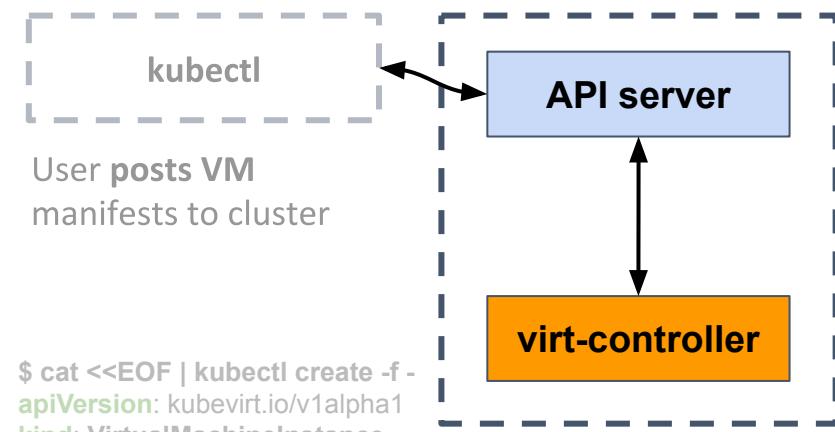
# Virtual Machine Launch Flow

kubectl

User posts VM  
manifests to cluster

```
$ cat <<EOF | kubectl create -f -
apiVersion: kubevirt.io/v1alpha1
kind: VirtualMachineInstance
...
spec:
  domain:
    cpu:
      cores: 2
    devices:
      disk: fedora29
...
EOF
```

# Virtual Machine Launch Flow



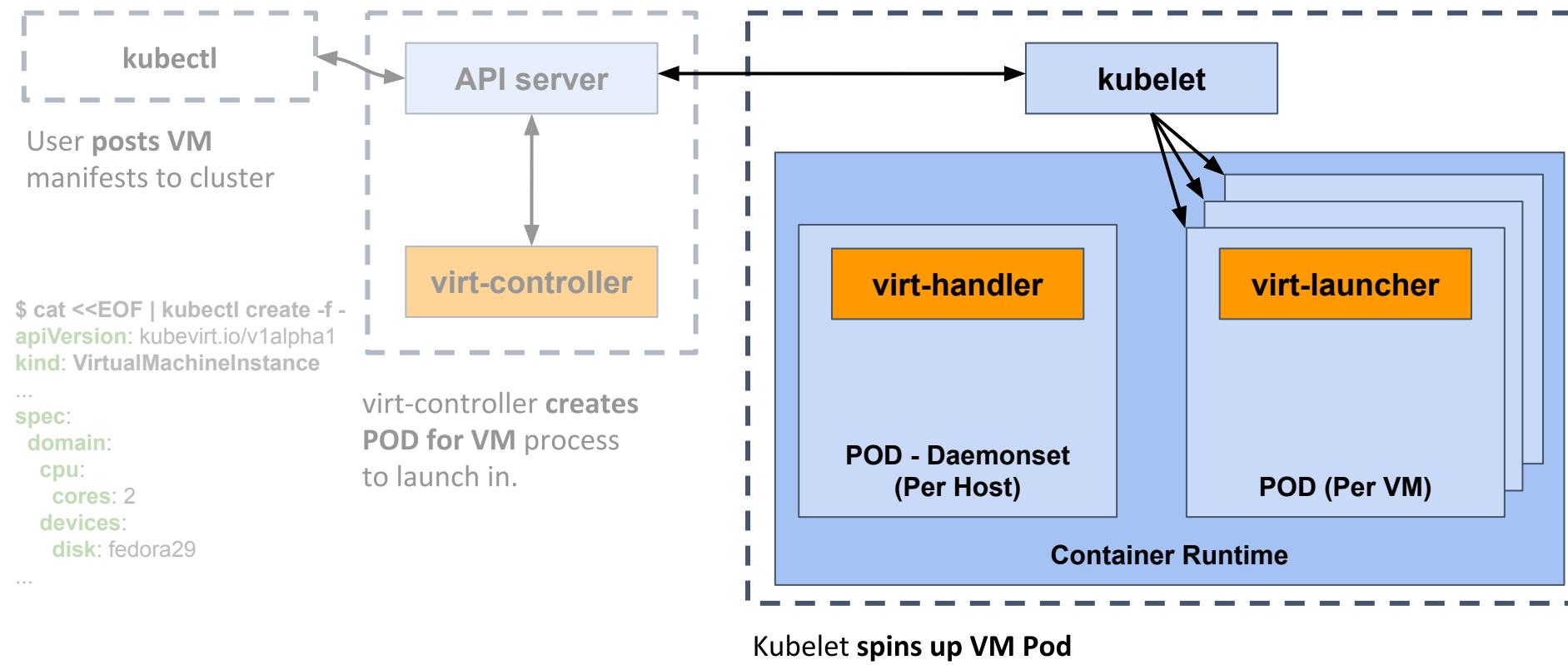
User posts VM  
manifests to cluster

```
$ cat <<EOF | kubectl create -f -  
apiVersion: kubevirt.io/v1alpha1  
kind: VirtualMachineInstance
```

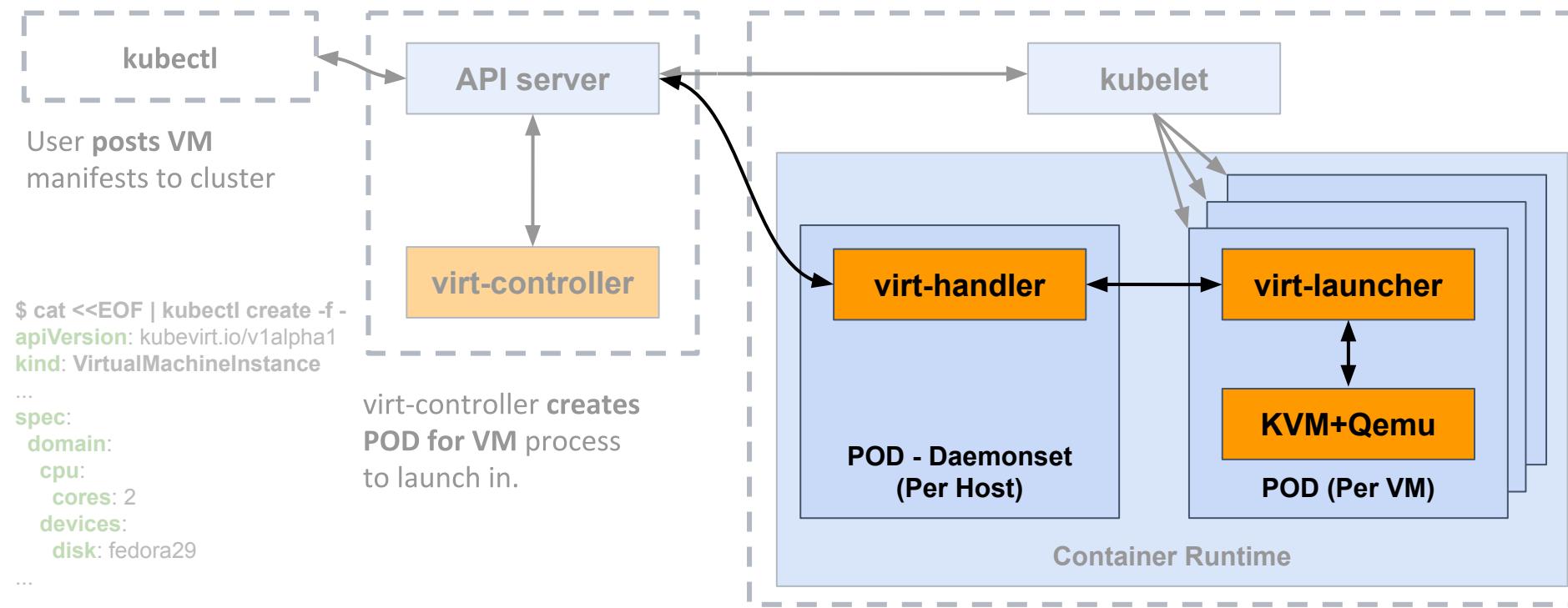
```
...  
spec:  
domain:  
cpu:  
cores: 2  
devices:  
disk: fedora29  
...
```

virt-controller **creates**  
**POD for VM** process  
to launch in.

# Virtual Machine Launch Flow



# Virtual Machine Launch Flow



Kubelet spins up VM Pod  
virt-handler instructs virt-launcher how to launch the qemu

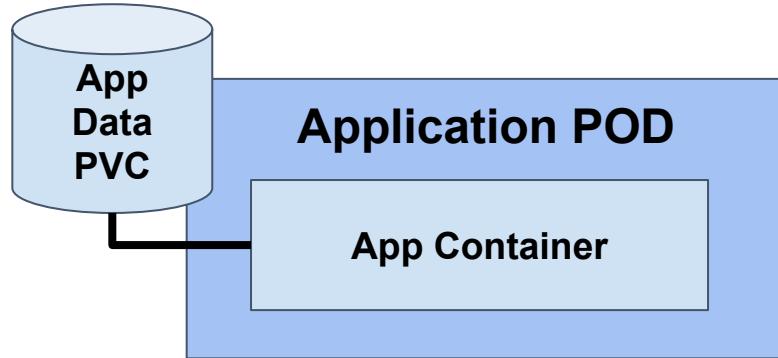
# Persistent Storage for Containers



CloudNativeCon

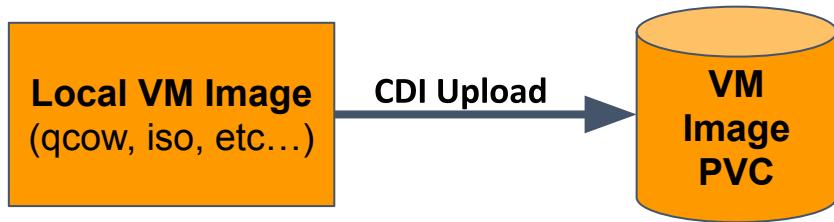
North America 2019

Attach Persistent Data for Applications using PVCs



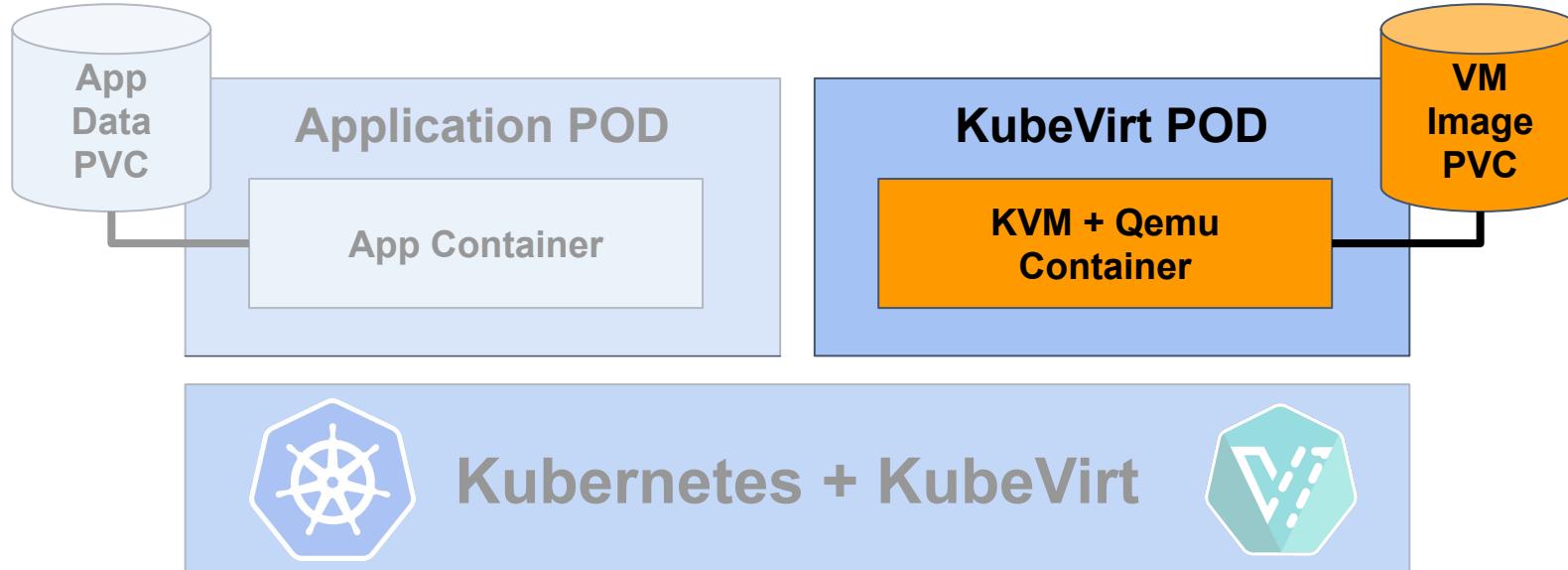
# VM Image Upload to PVC

Upload VM Images with **containerized-data-importer (CDI)**



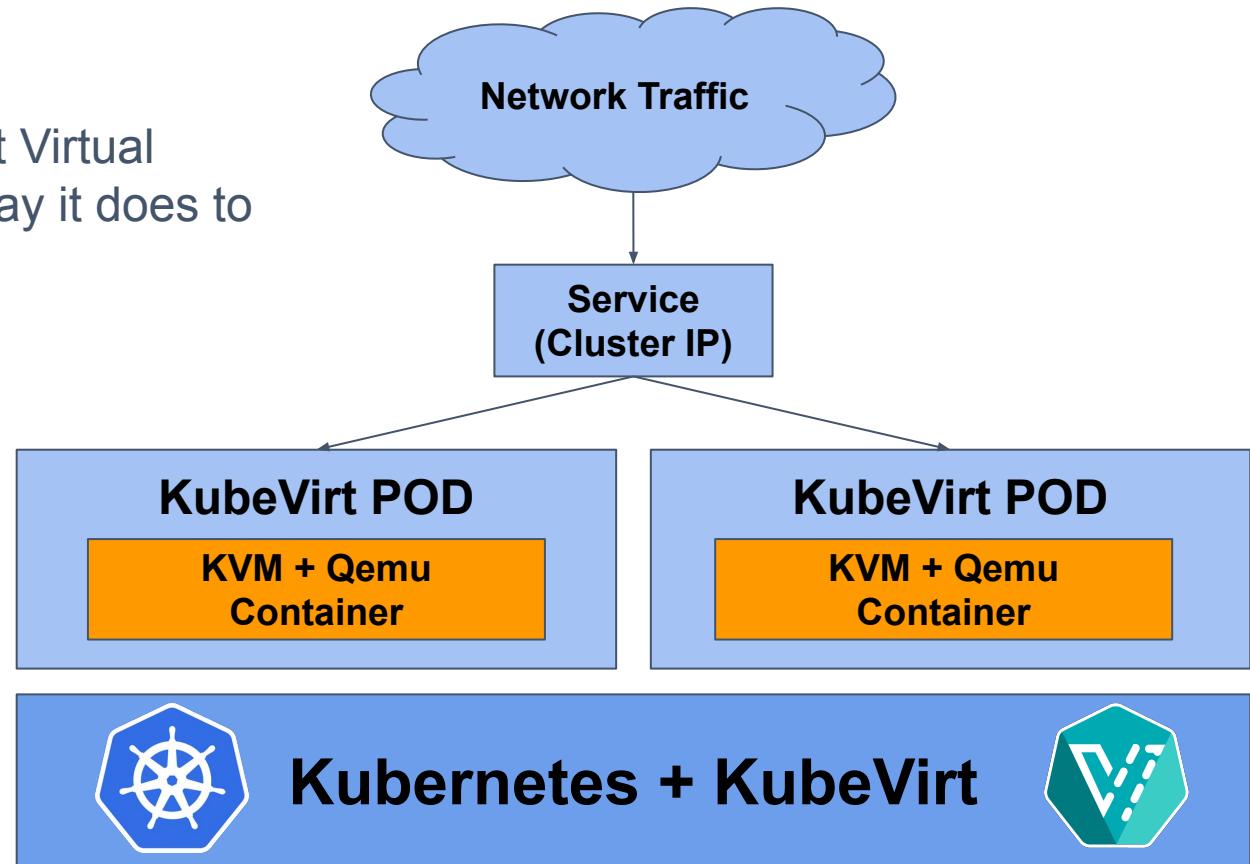
# Use of Persistent Storage

Attach VM Volumes using PVCs



# Use of Network Services

Traffic routes to KubeVirt Virtual Machines in the same way it does to container workloads



# Use of Host Resources



KubeCon

CloudNativeCon

North America 2019

- VM Guest CPU and NUMA Affinity
  - **CPU Manager**
  - **Topology Manager**
- VM Guest CPU/MEM requirements
  - **POD resource request/limits**
- VM Guest use of Host Devices
  - **Device Plugins** for access to (/dev/kvm, SR-IOV, GPU passthrough)
  - **POD resource request/limits** for device allocation

# Title

- END OF KUBEVIRT INTRO

Handing off to Vishesh to handle in depth GPU workload info.

# GPU/vGPU in Kubevirt VMs. Why?



KubeCon

CloudNativeCon

North America 2019

## K8s Containers with GPU

- Deep Learning
- AI Inferencing
- Big data

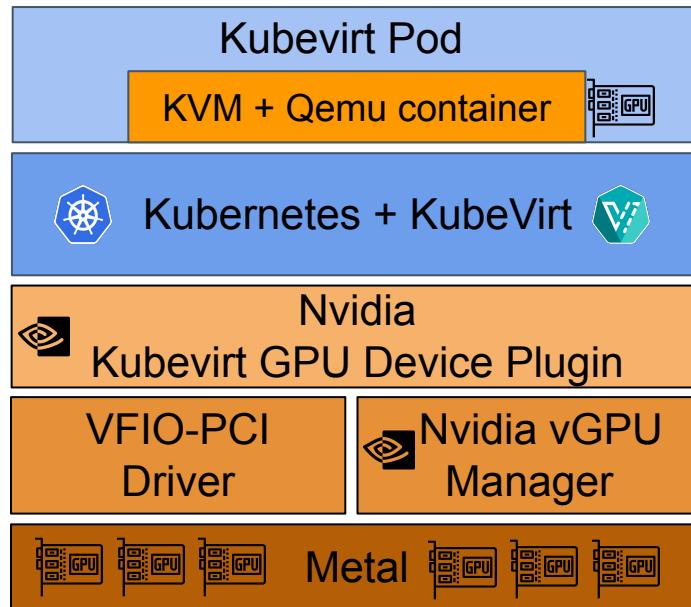
## Gaps

- Gaming
- Professional graphics



# GPU Pass Through Architecture

Using Device Plugin framework is a natural choice to provide GPU access to Kubevirt VMs



# GPU Passthrough to Kubevirt VM



KubeCon

CloudNativeCon

North America 2019

## Node Status

```
Name:Node-1  
...  
Capacity:  
cpu: 32  
devices.kubevirt.io/kvm: 110  
devices.kubevirt.io/tun: 110  
ephemeral-storage: 575019Mi  
memory: 65839284Ki  
nvidia.com/GP102GL_Tesla_P40: 5  
pods: 110  
Allocatable:  
cpu: 32  
devices.kubevirt.io/kvm: 110  
devices.kubevirt.io/tun: 110  
ephemeral-storage: 575019Mi  
memory: 65839284Ki  
nvidia.com/GP102GL_Tesla_P40: 5  
pods: 110  
...
```

## Virtual Machine Spec

```
apiVersion: kubevirt.io/v1alpha3  
kind: VirtualMachineInstance  
...  
spec:  
domain:  
devices:  
disks:  
- disk:  
bus: virtio  
name: fedora29  
gpus:  
- deviceName:  
nvidia.com/GP102GL_Tesla_P40  
name: gpu1  
...  
resources:  
requests:  
memory: 2Gi  
...
```

## Pod Status

```
Name: virt-launcher-test-gpu-vmi-plzjj  
Namespace: default  
...  
Containers:  
compute:  
Container ID:  
...  
Limits:  
devices.kubevirt.io/kvm: 1  
devices.kubevirt.io/tun: 1  
nvidia.com/GP102GL_Tesla_P40: 1  
Requests:  
cpu: 100m  
devices.kubevirt.io/kvm: 1  
devices.kubevirt.io/tun: 1  
memory: 2259016Ki  
nvidia.com/GP102GL_Tesla_P40:1  
...
```

# Device Plugin Functions



KubeCon

CloudNativeCon

North America 2019

- **GPU and vGPU device Discovery**
  - GPUs with VFIO-PCI driver on the host are identified
  - vGPUs configured using Nvidia vGPU manager are identified
- **GPU and vGPU device Advertising**
  - Discovered devices are advertised to kubelet as allocatable resources
- **GPU and vGPU device Allocation**
  - Returns the PCI address of allocated GPU device
- **GPU and vGPU Health Check**
  - Performs health check on the discovered GPU and vGPU devices

# GPU Passthrough Lifecycle

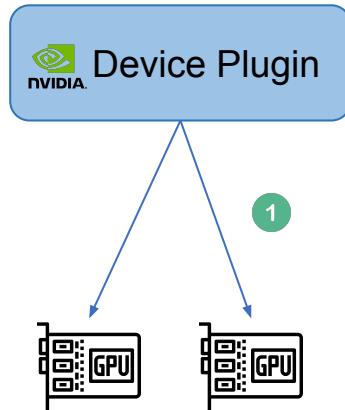


KubeCon

CloudNativeCon

North America 2019

K8s Node



DP reads PCI device  
on the host. Identifies  
the Nvidia GPUs and  
group GPUs based on  
type and driver

# GPU Passthrough Lifecycle



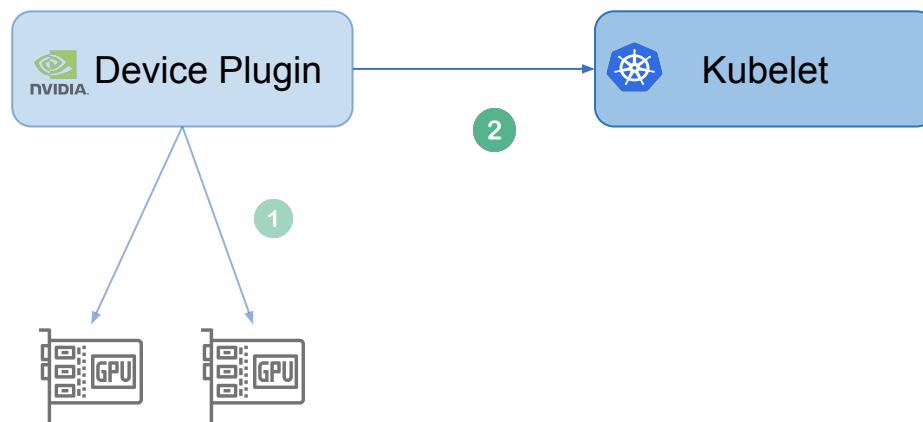
KubeCon

CloudNativeCon

North America 2019

K8s Node

DP starts a gRPC server  
for each group of GPUs. It  
also registers each group  
with Kubelet



# GPU Passthrough Lifecycle



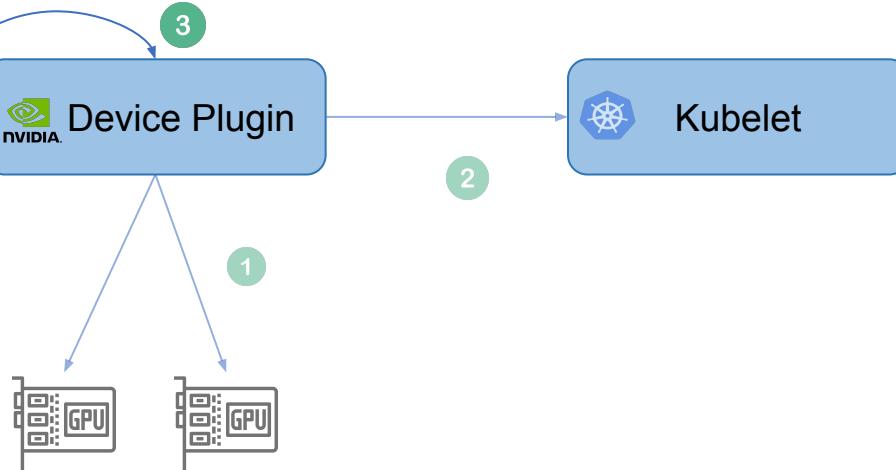
KubeCon

CloudNativeCon

North America 2019

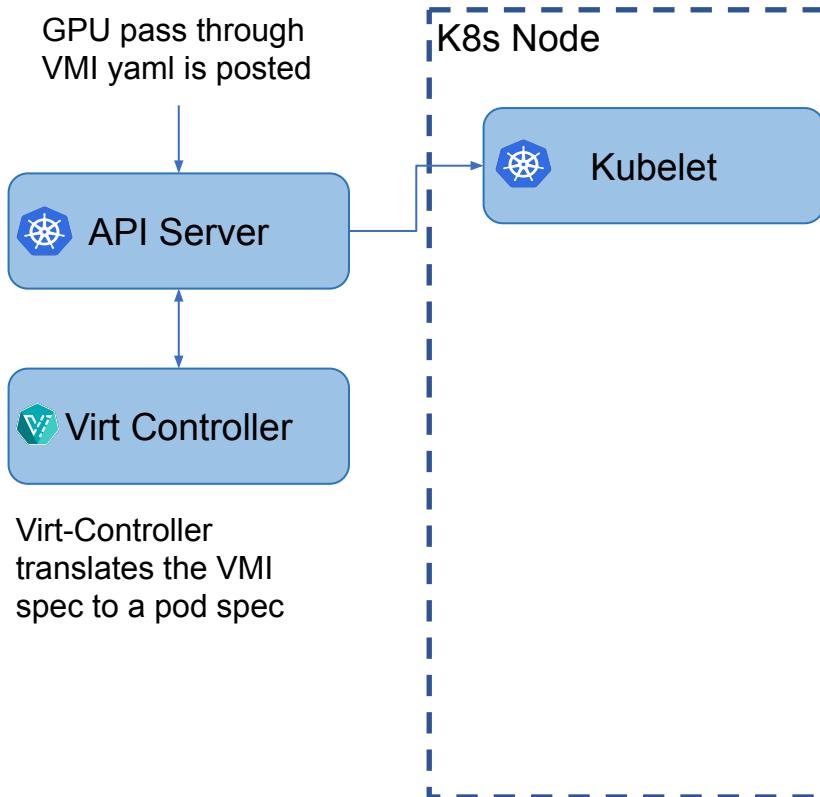
K8s Node

DP is waiting for a allocation request from Kubelet. It is also performing health check in a loop.

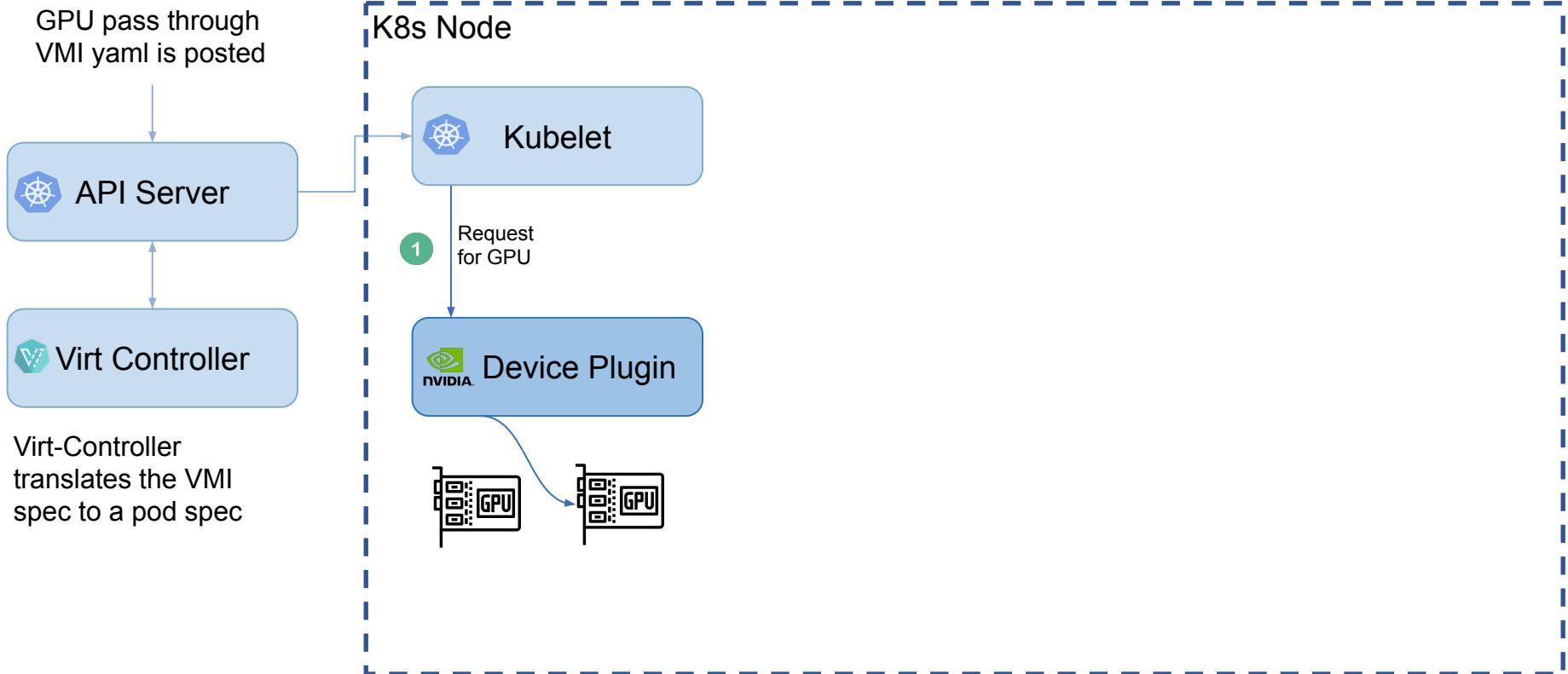


# GPU Passthrough Lifecycle

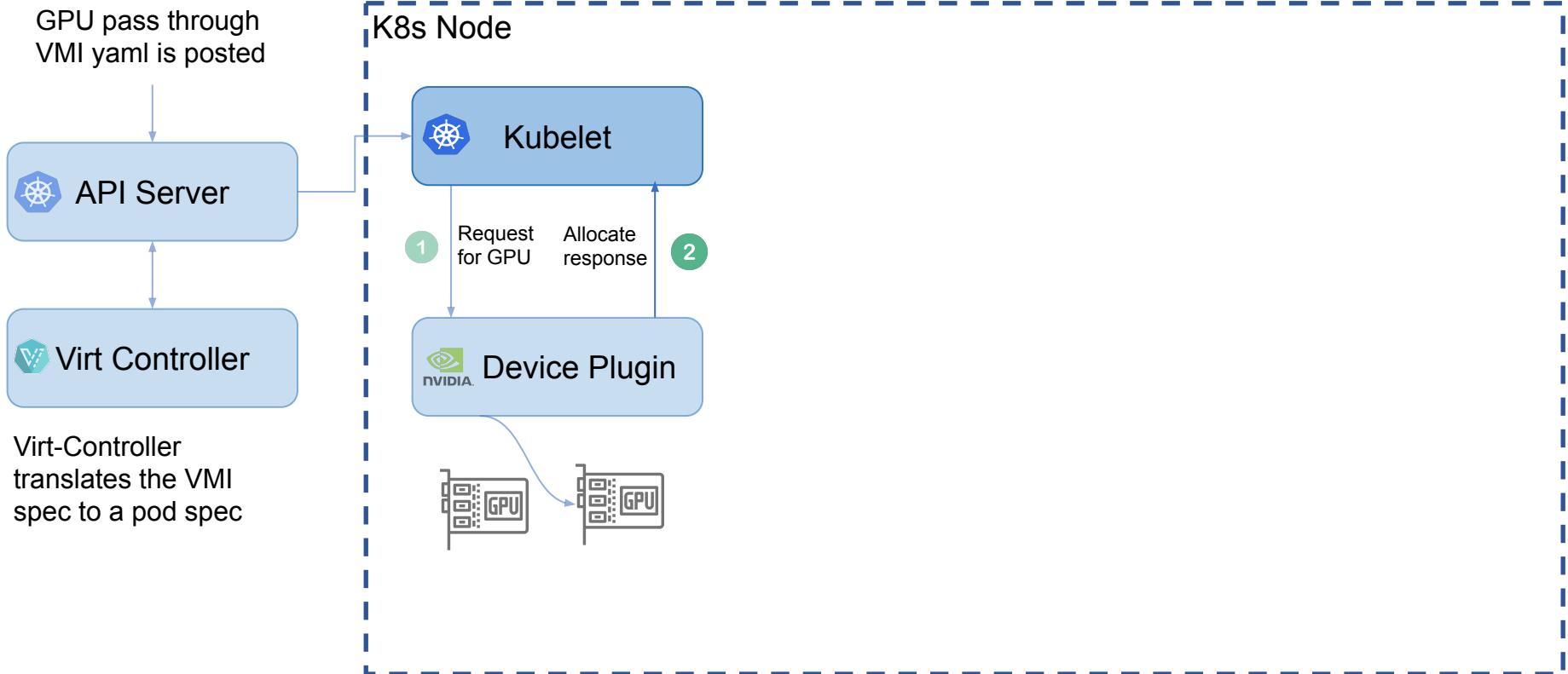
GPU pass through  
VMI yaml is posted



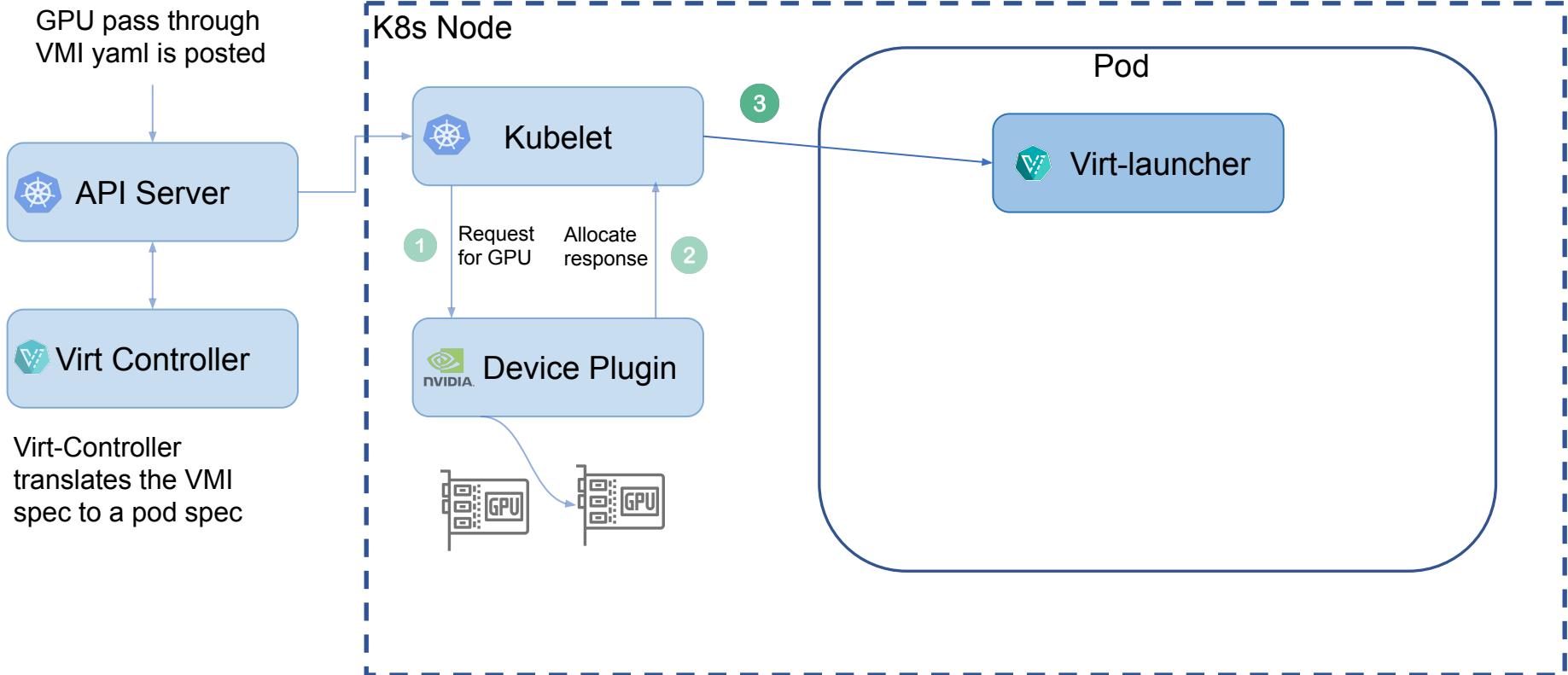
# GPU Passthrough Lifecycle



# GPU Passthrough Lifecycle



# GPU Passthrough Lifecycle



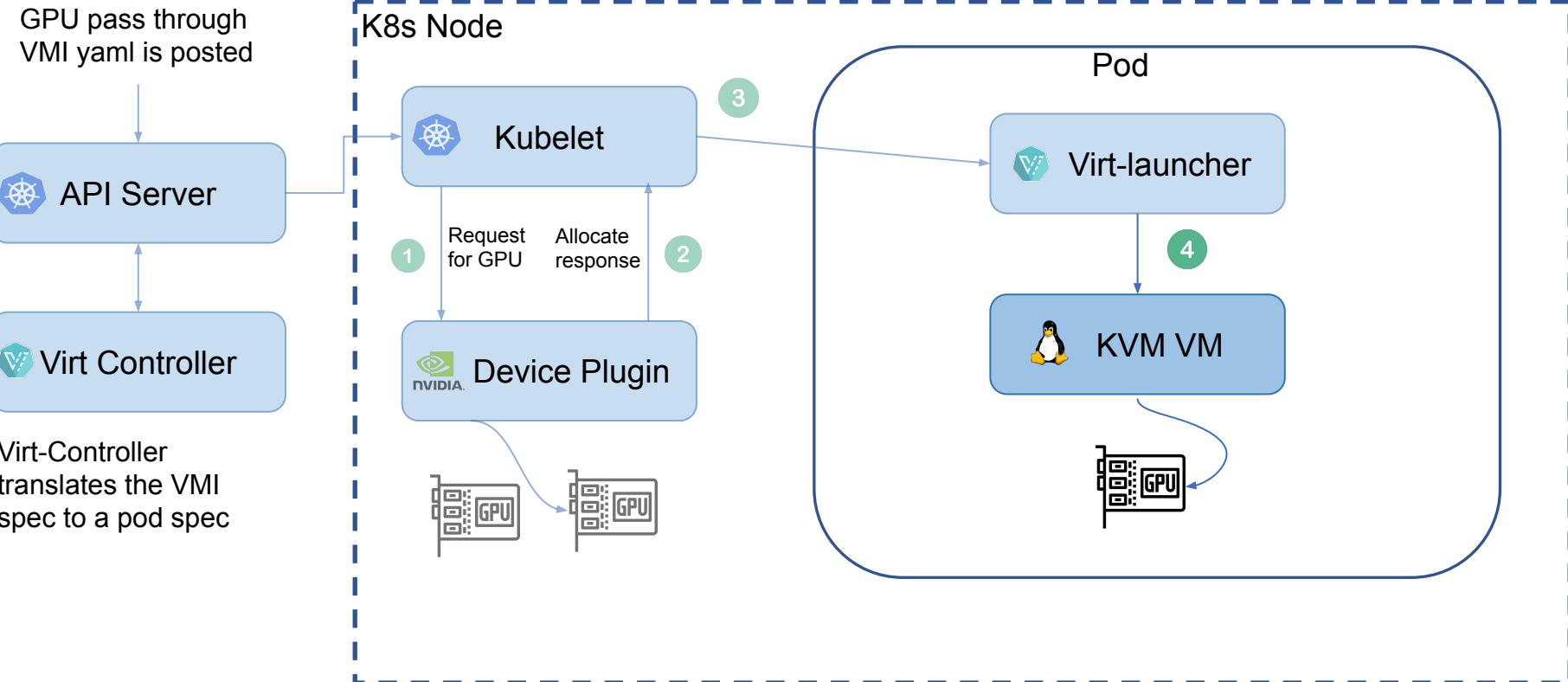
# GPU Passthrough Lifecycle



KubeCon

CloudNativeCon

North America 2019



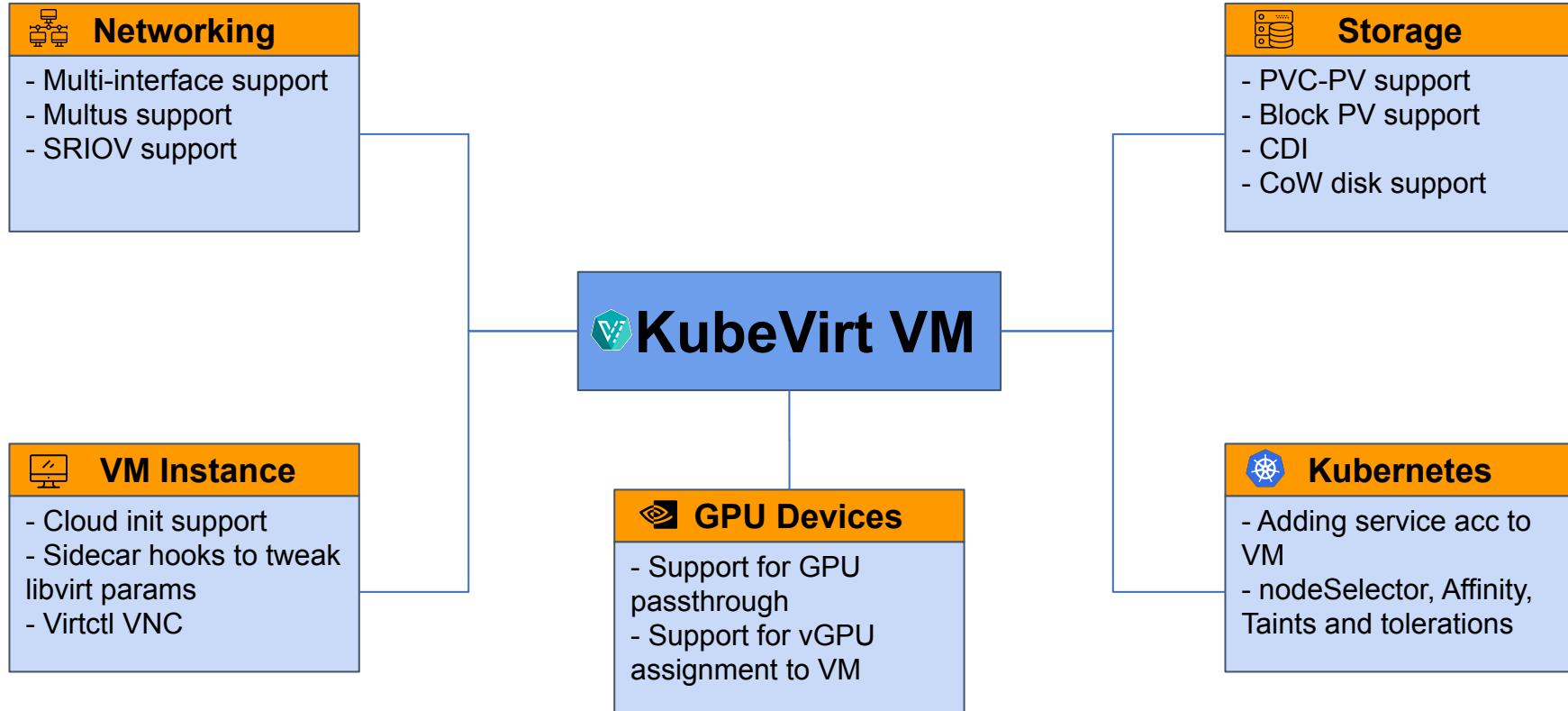
# NVIDIA Usecase: Key features



KubeCon

CloudNativeCon

North America 2019





KubeCon



CloudNativeCon

North America 2019

# Thank You

