
KubeCon
CloudNativeCon
North America 2019

Virtualized GPU workloads on KubeVirt

David Vossel
Principal Software Engineer
Red Hat

Vishesh Tanksale
Senior Software Engineer
NVIDIA

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?

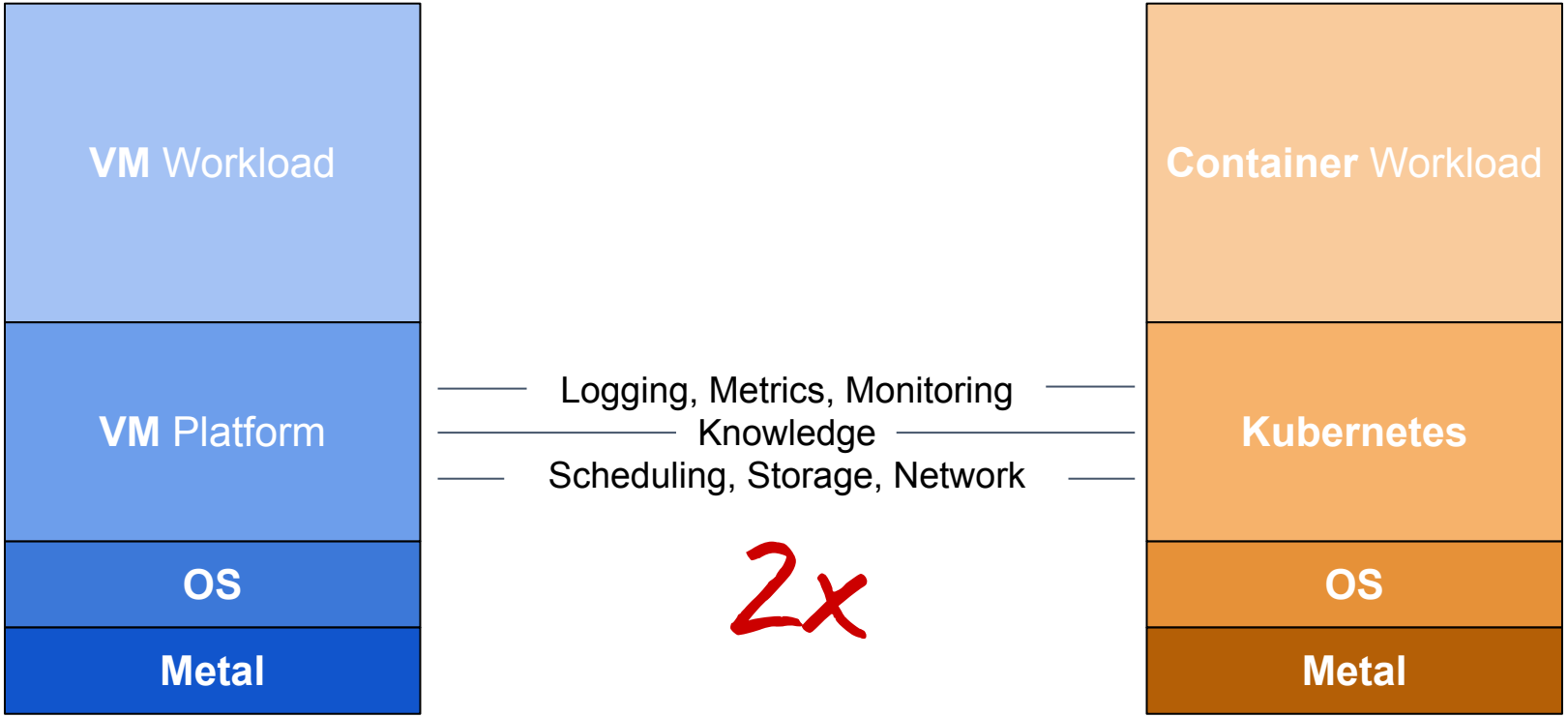
"KubeVirt is a Kubernetes extension that allows running traditional VM workloads natively side by side with Container workloads."

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?

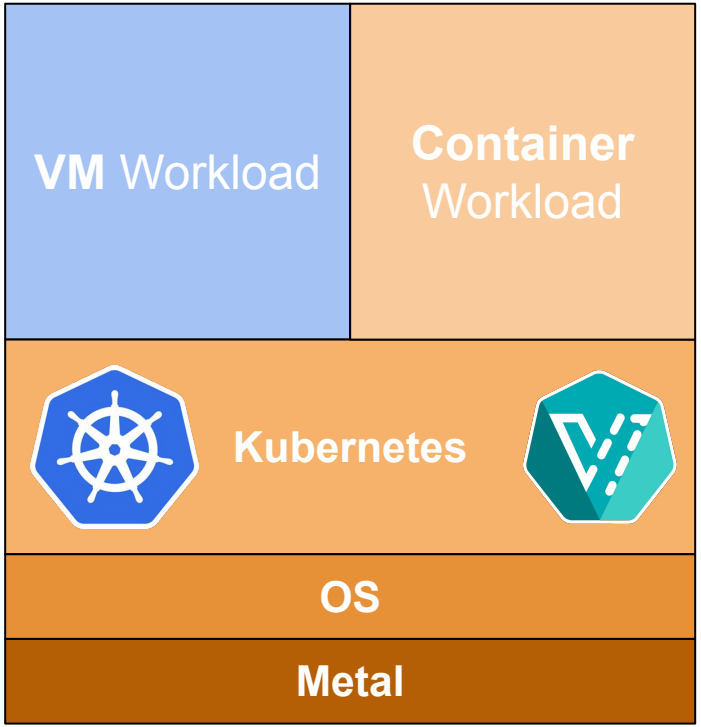
Infrastructure Convergence

Old Way … Multiple Workloads, Multiple Stacks



Infrastructure Convergence

KubeVirt way… Multiple Workloads, One Stack



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

1x

Workflow Convergence

- 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...)

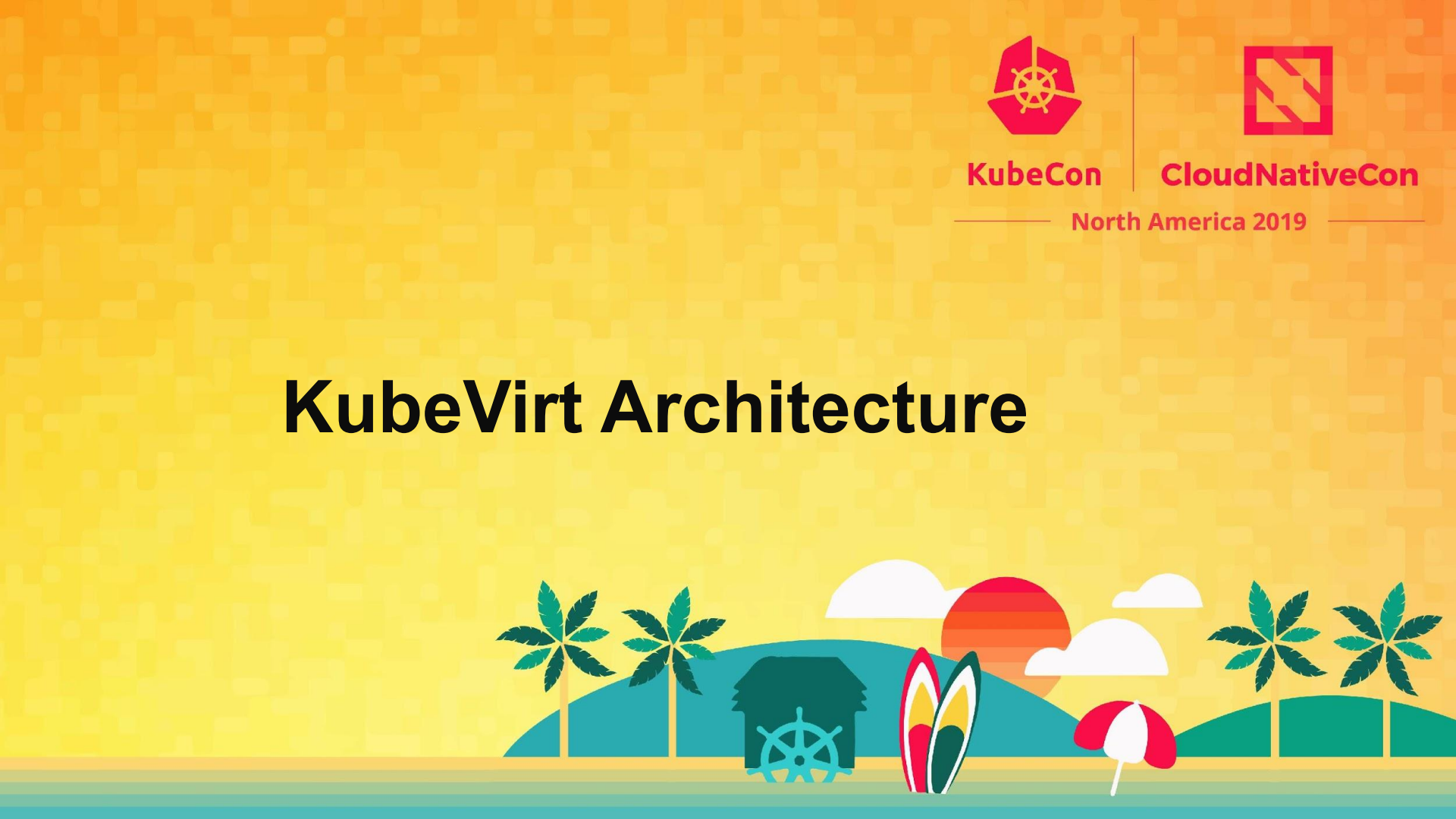
```
$ cat <<EOF | kubectl create -f -
apiVersion: v1
kind: Pod
...
spec:
  containers:
  - name: nginx
    image: nginx
...
```

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
...
```

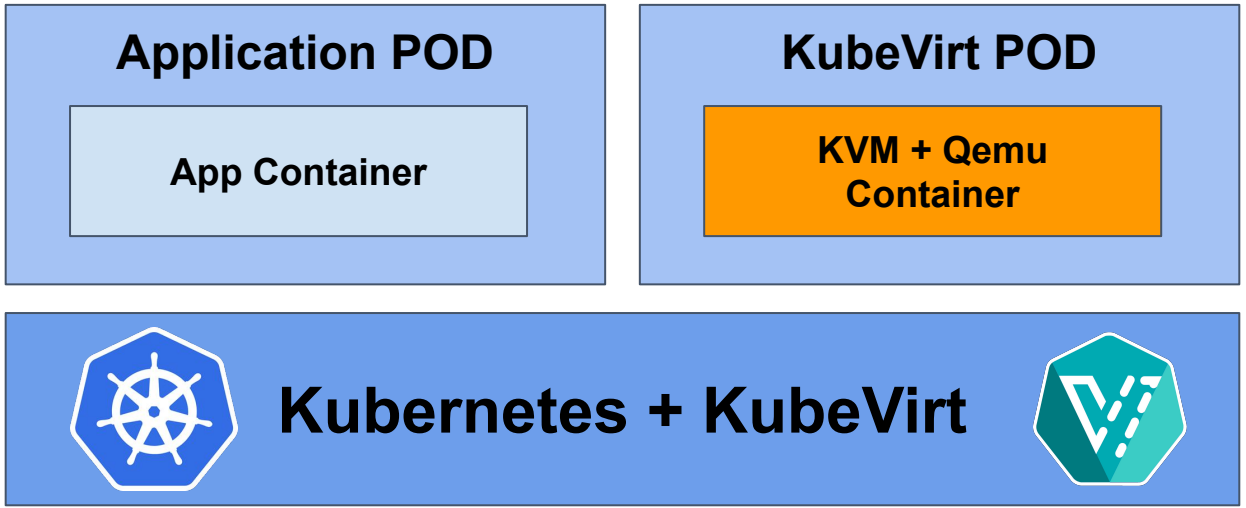
```
$ cat <<EOF | kubectl create -f -
apiVersion: kubevirt.io/v1alpha1
kind: VirtualMachineInstance
...
spec:
  domain:
    cpu:
      cores: 2
    devices:
      disk: fedora29
...
```



KubeVirt Architecture

Virtual Machines in Pods

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



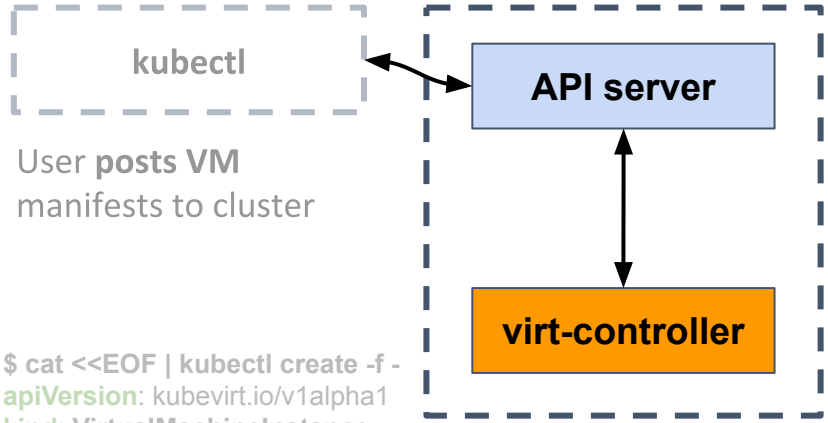
Virtual Machine Launch Flow

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
...
```

Virtual Machine Launch Flow

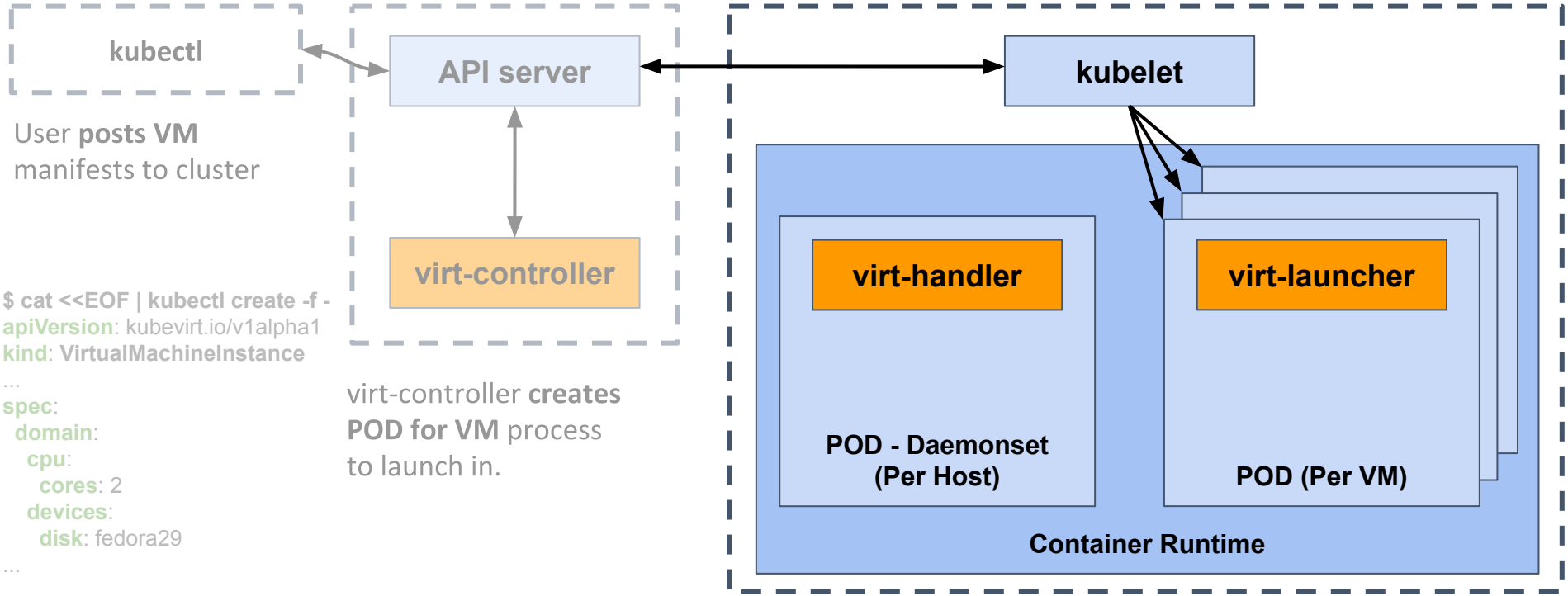


User **posts VM** manifests to cluster

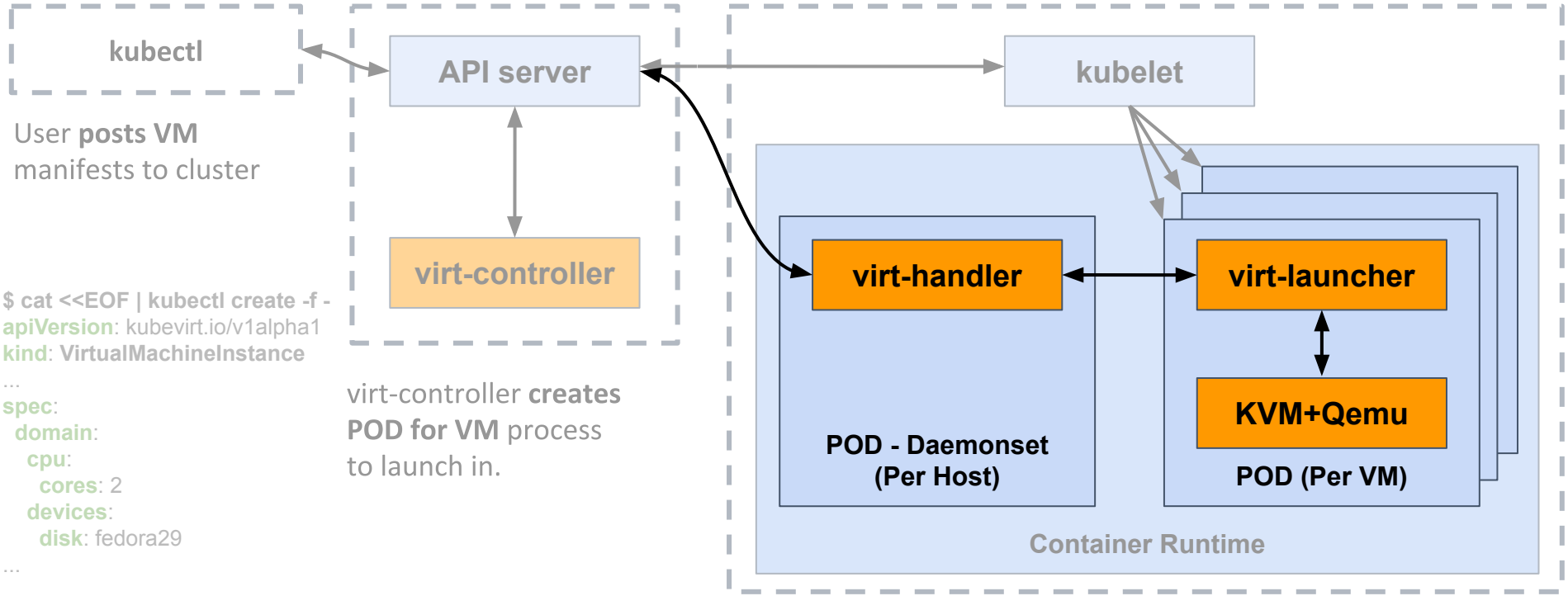
```
$ cat <<EOF | kubectl create -f -
apiVersion: kubevirt.io/v1alpha1
kind: VirtualMachineInstance
...
spec:
  domain:
    cpu:
      cores: 2
    devices:
      disk: fedora29
...
```

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

Virtual Machine Launch Flow



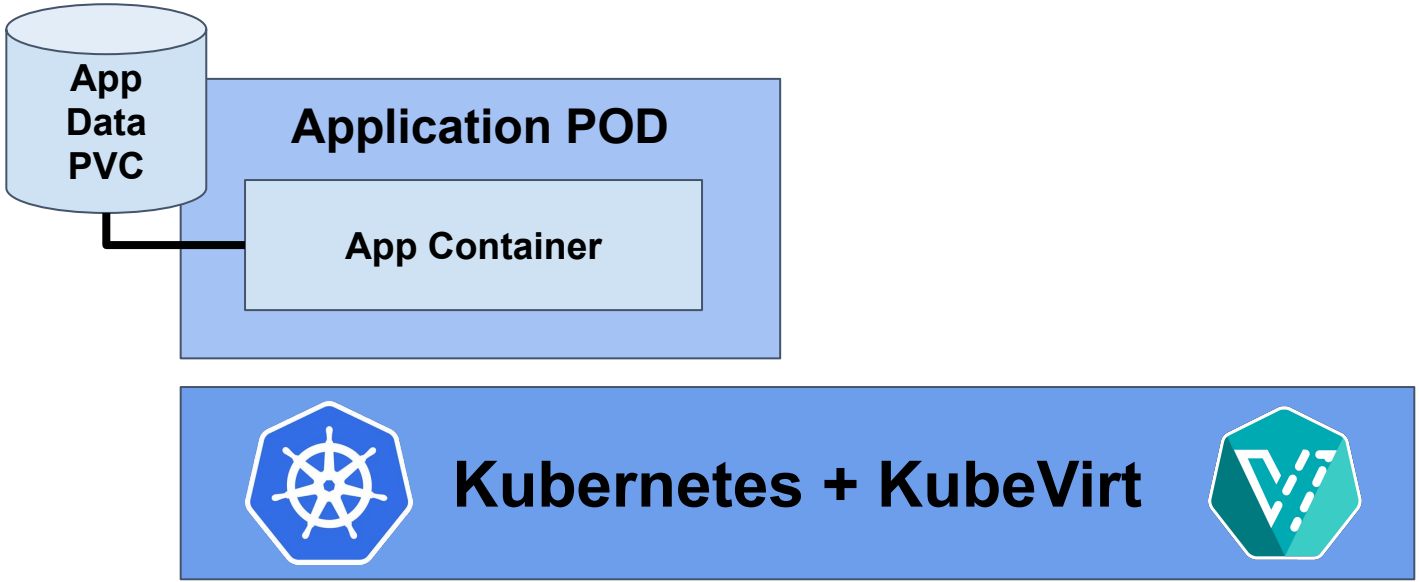
Virtual Machine Launch Flow



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

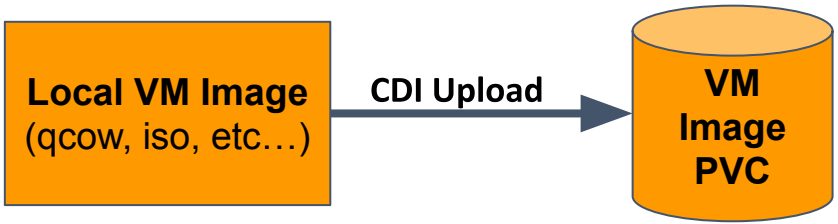
Persistent Storage for Containers

Attach Persistent Data for Applications using PVCs



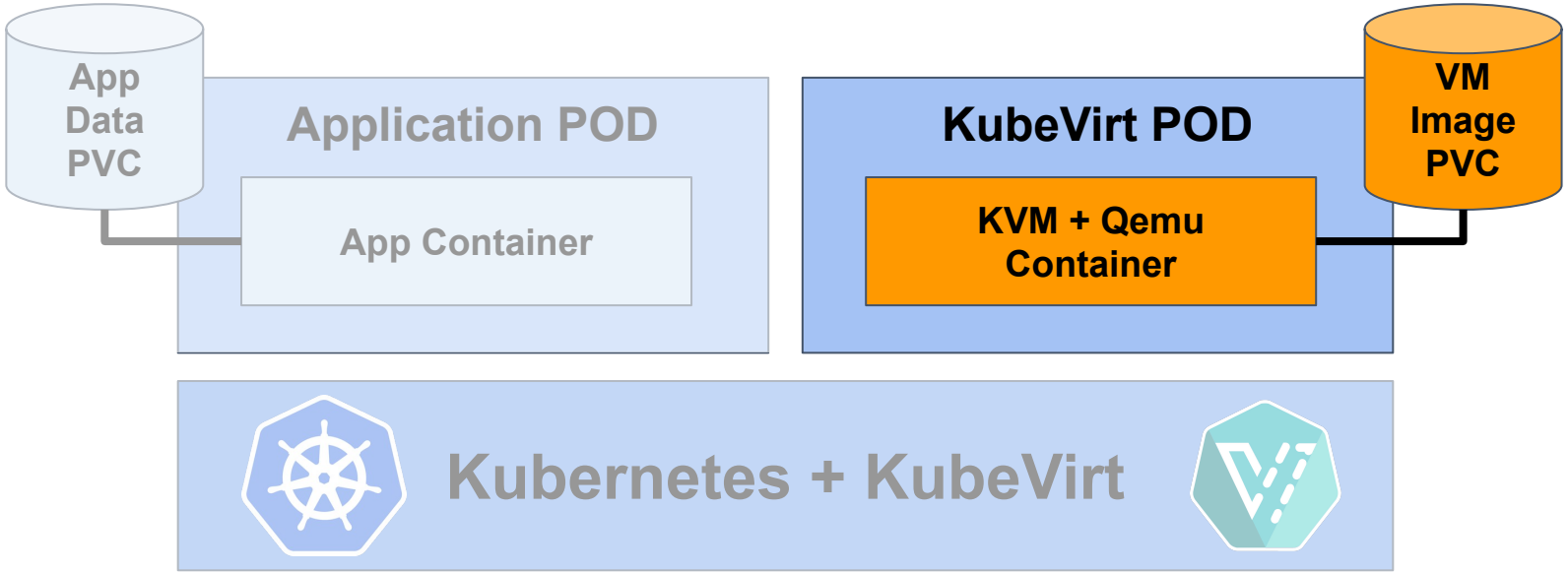
VM Image Upload to PVC

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



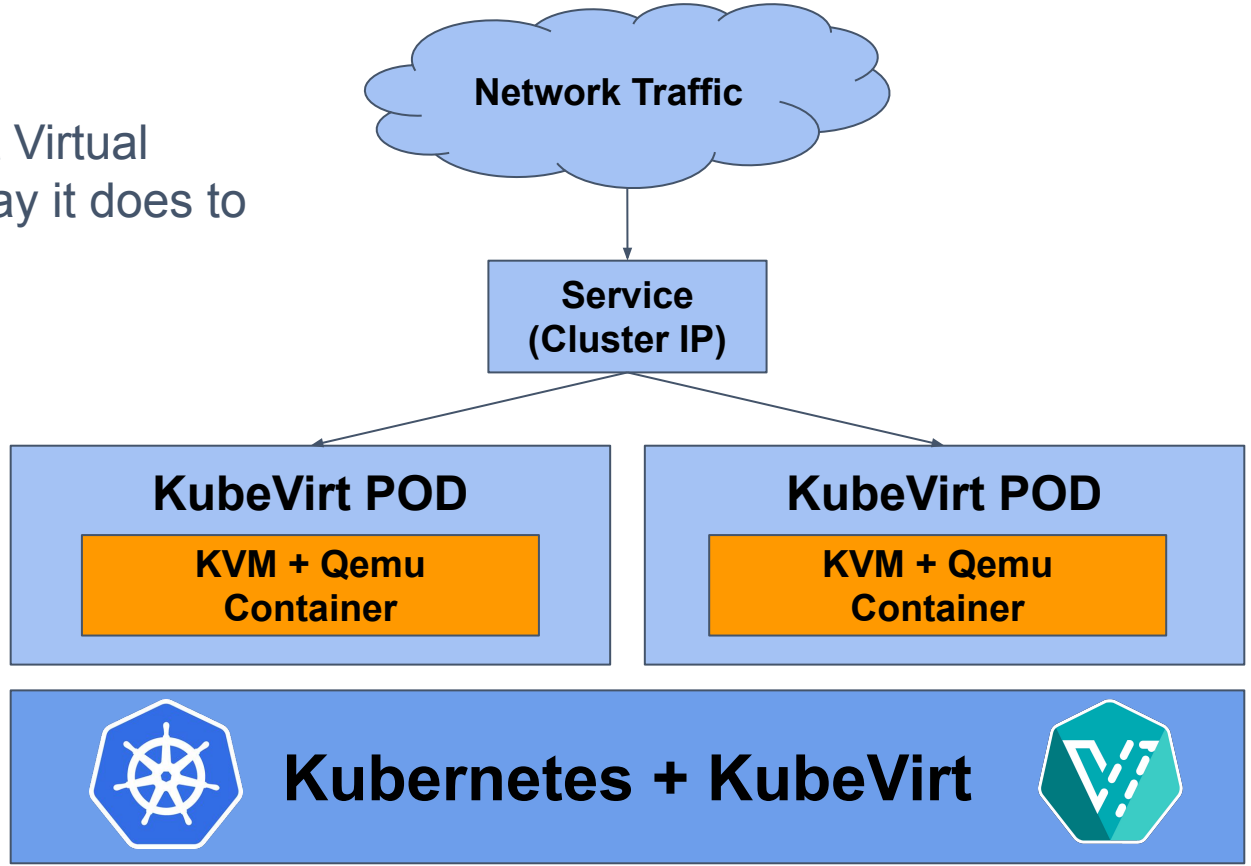
Use of Persistent Storage

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

- 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 passthough)
  - 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?

K8s Containers with GPU

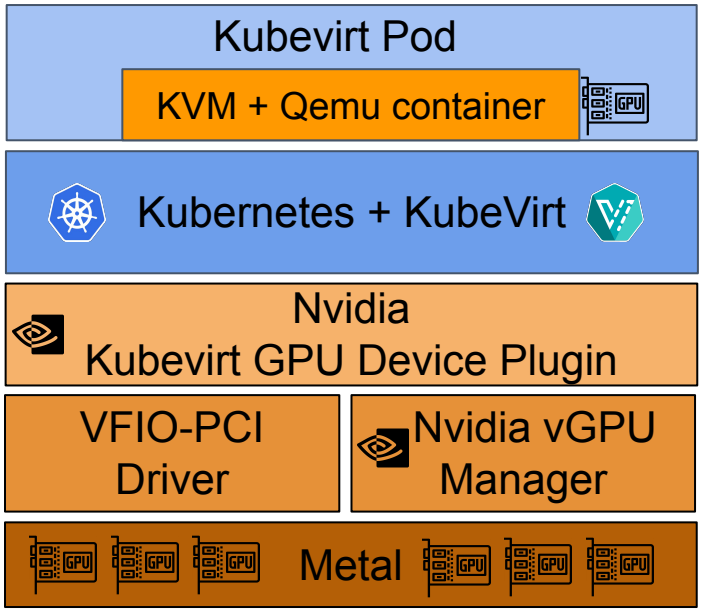
- Deep Learning
- AI Inferencing
- Big data

Gaps

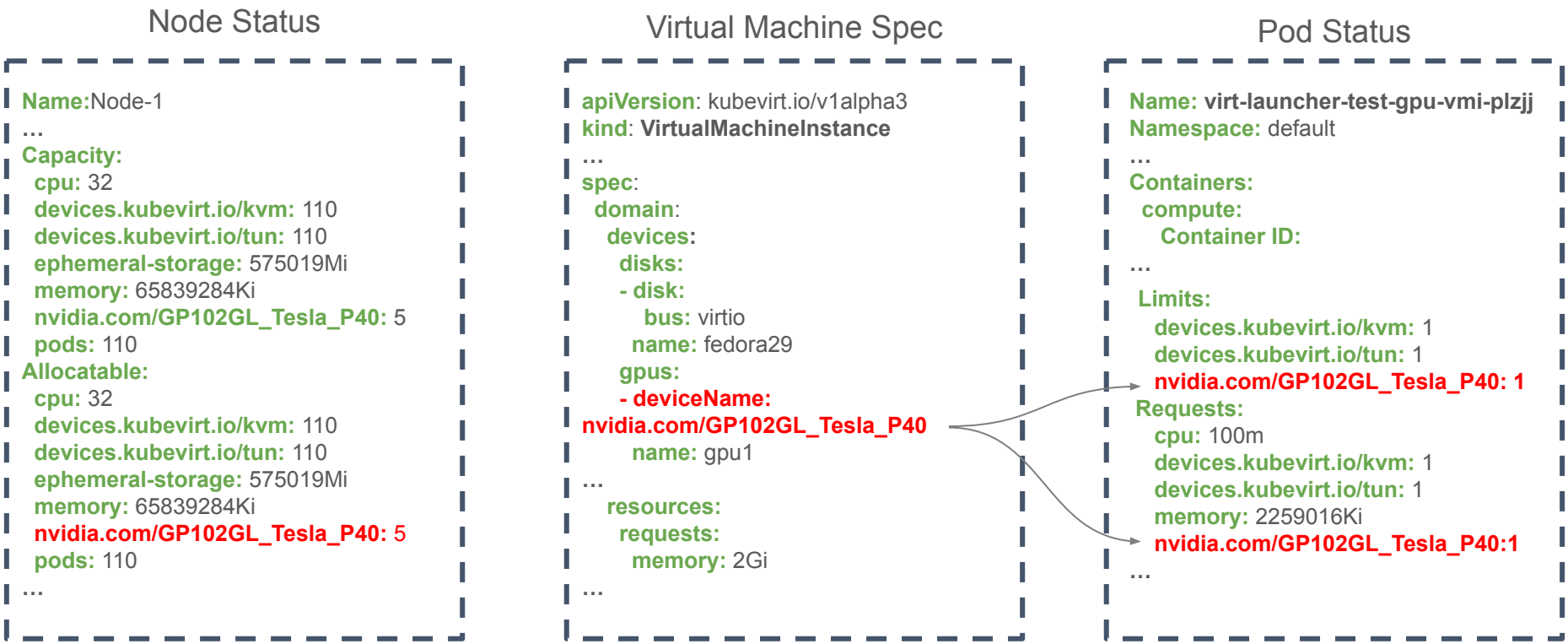
- Gaming
- Professional graphics

GPU Pass Through Architecture

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



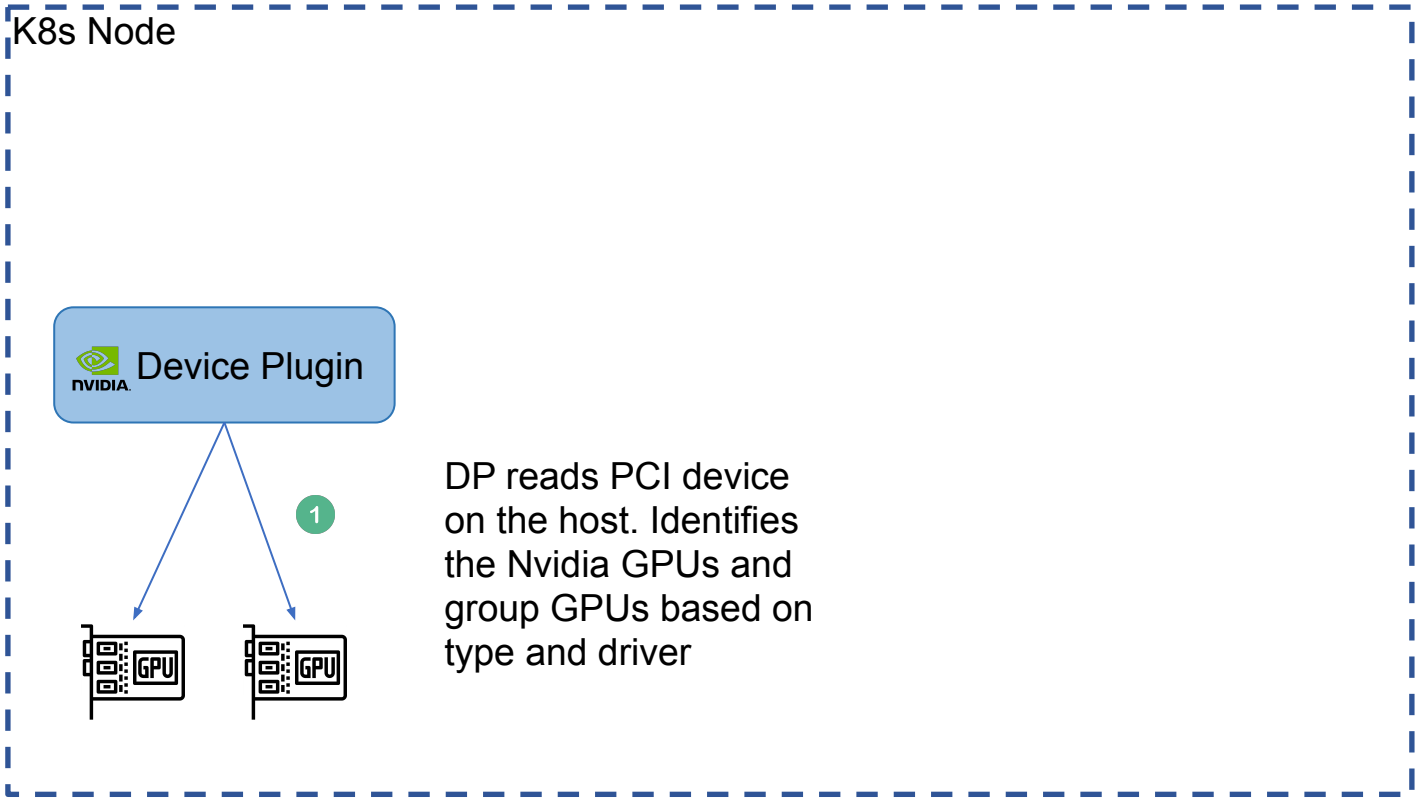
GPU Passthrough to Kubevirt VM



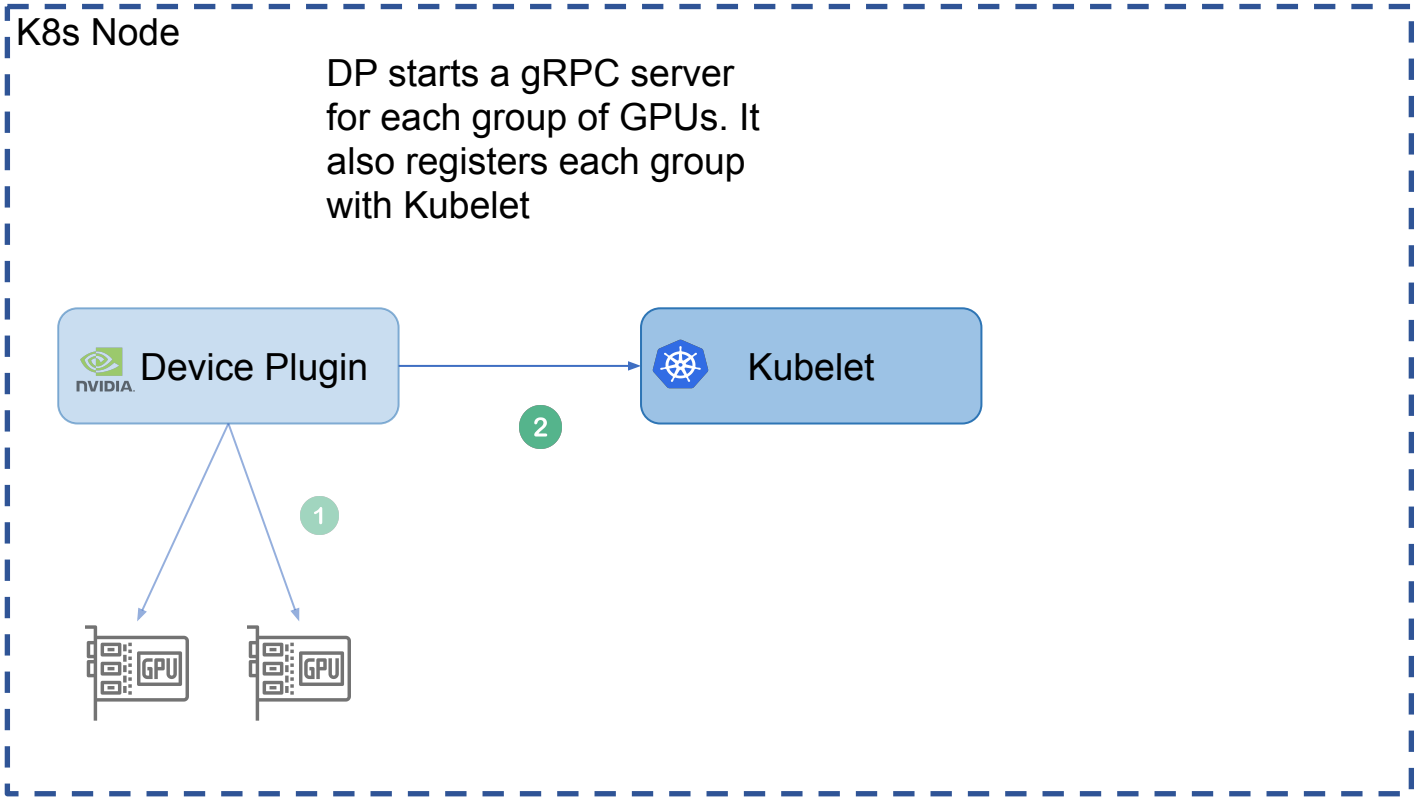
Device Plugin Functions

- **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

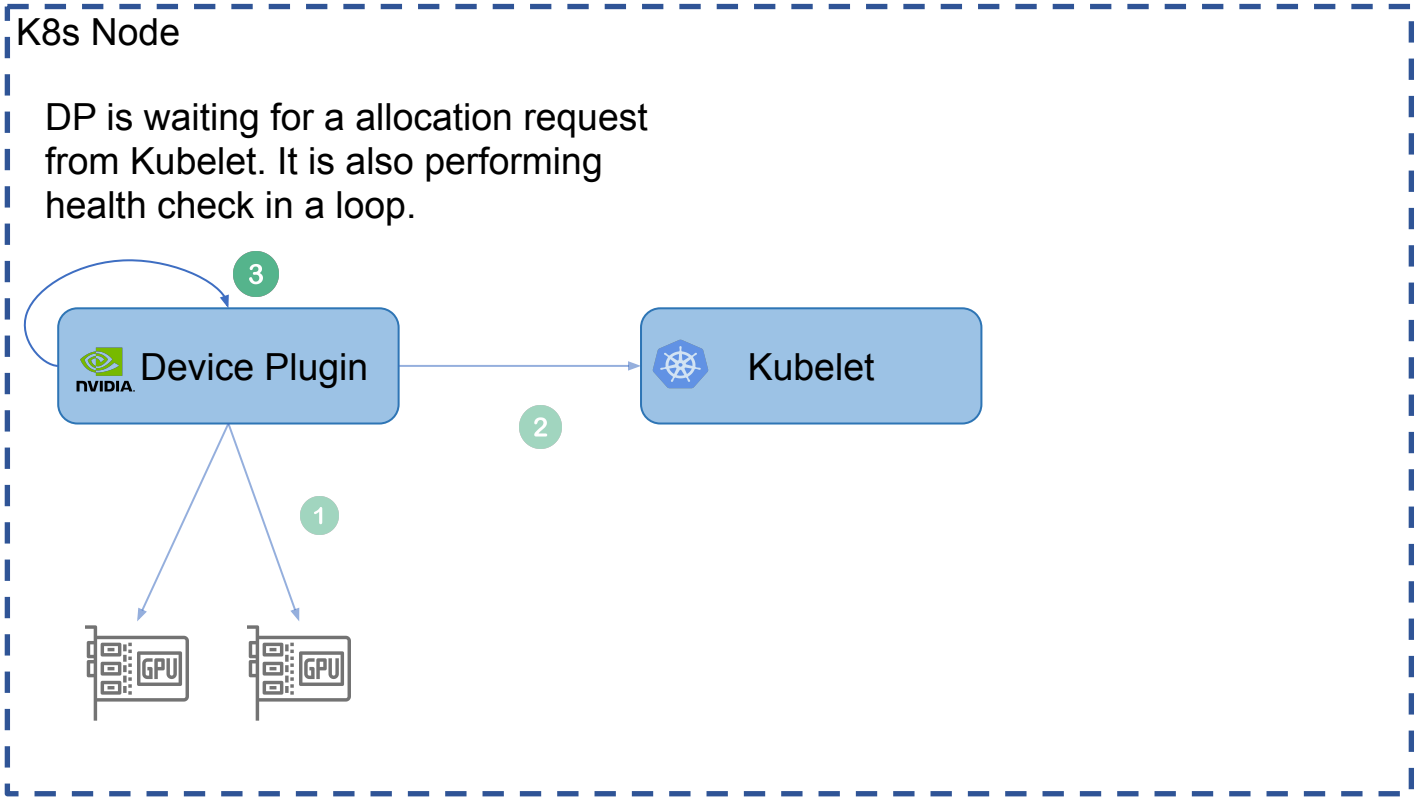
GPU Passthrough Lifecycle



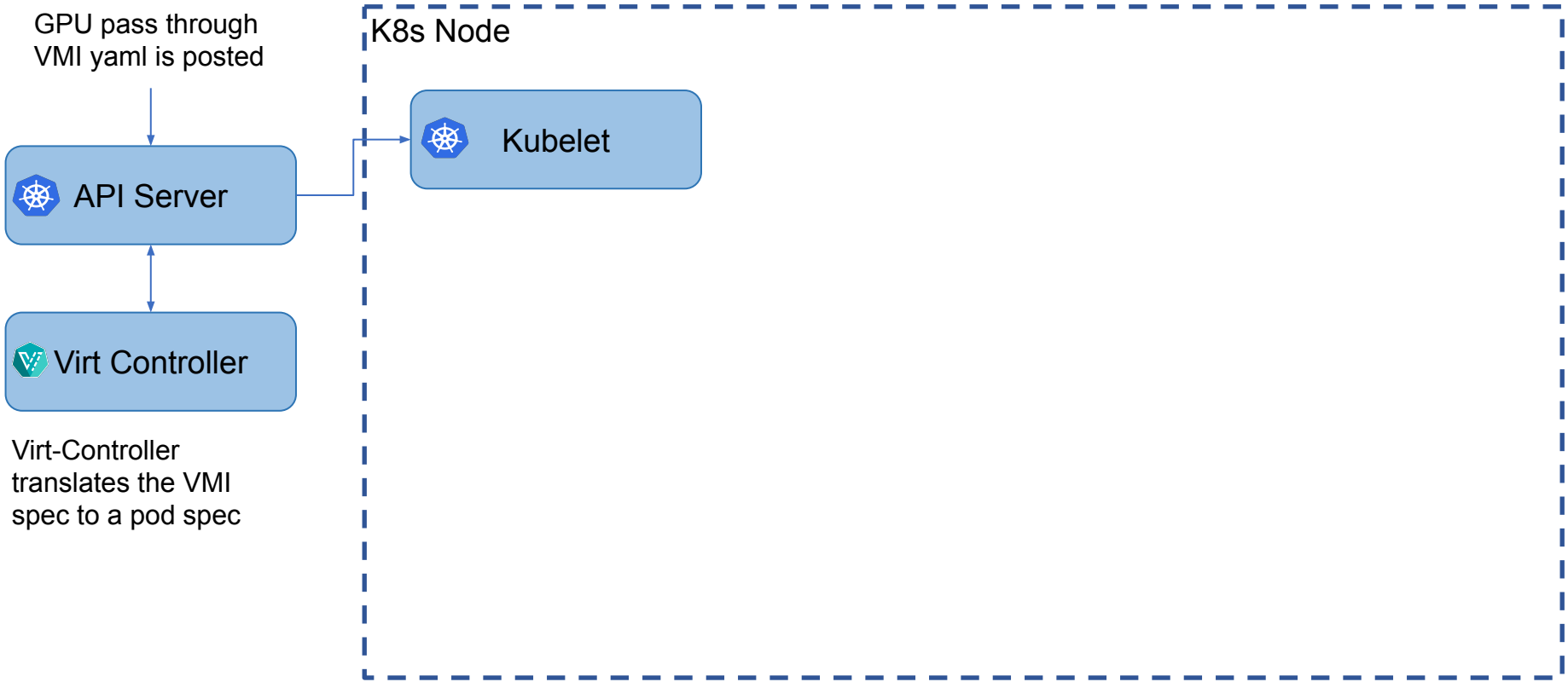
GPU Passthrough Lifecycle



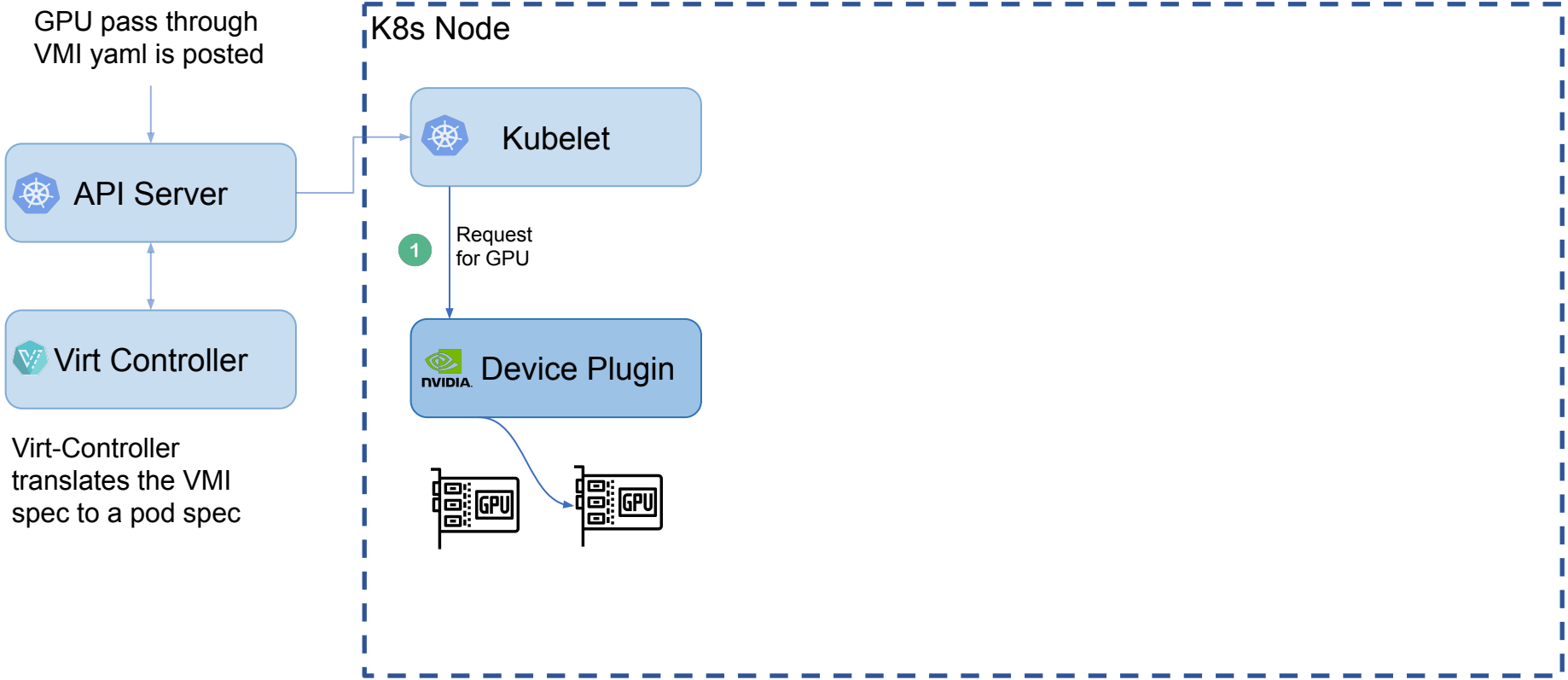
GPU Passthrough Lifecycle



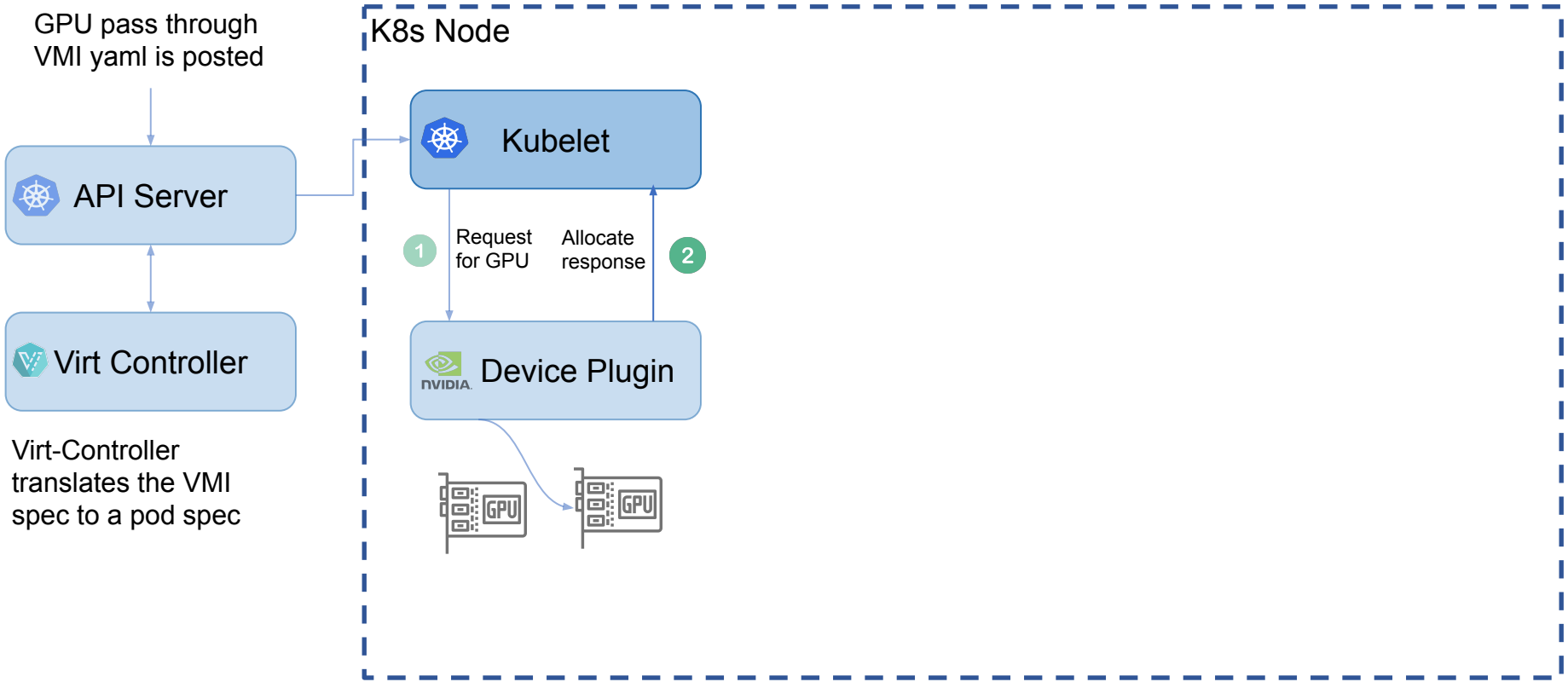
GPU Passthrough Lifecycle



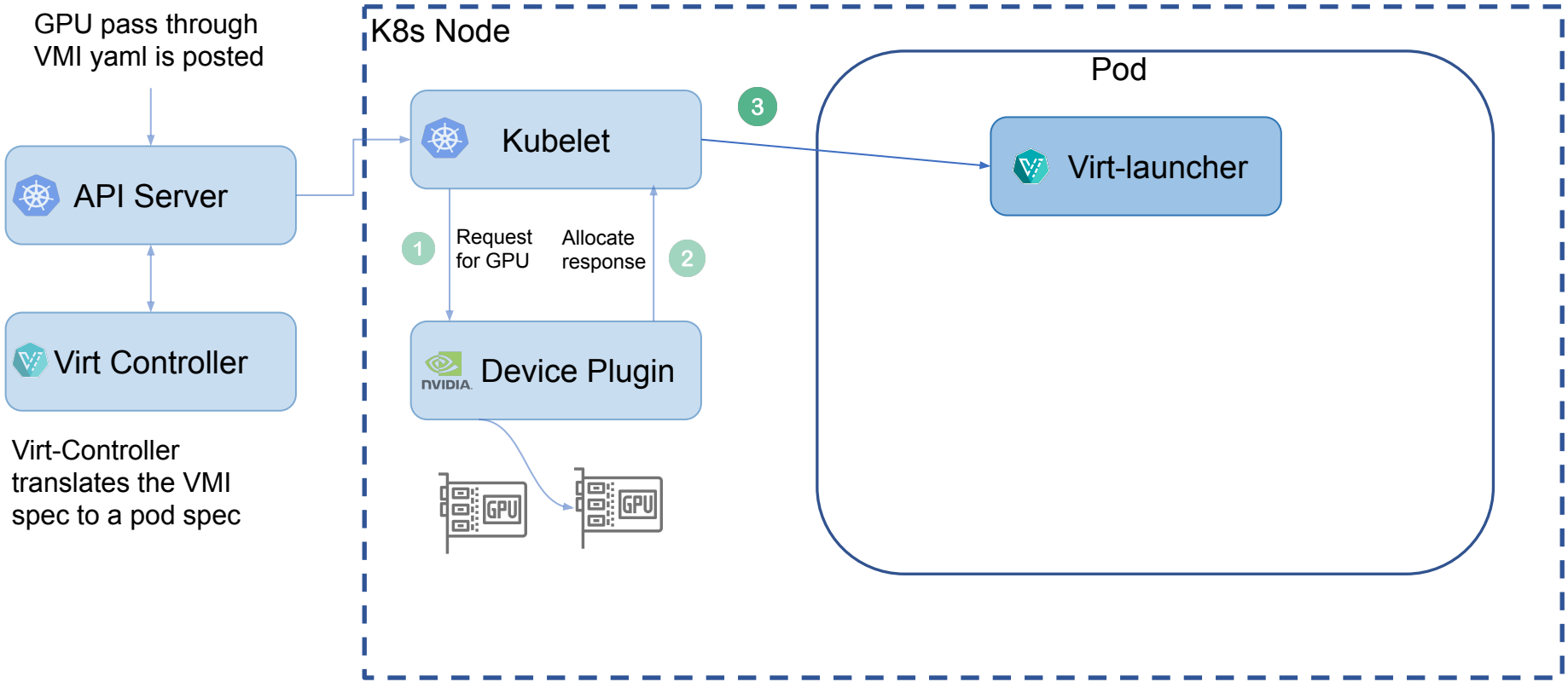
GPU Passthrough Lifecycle



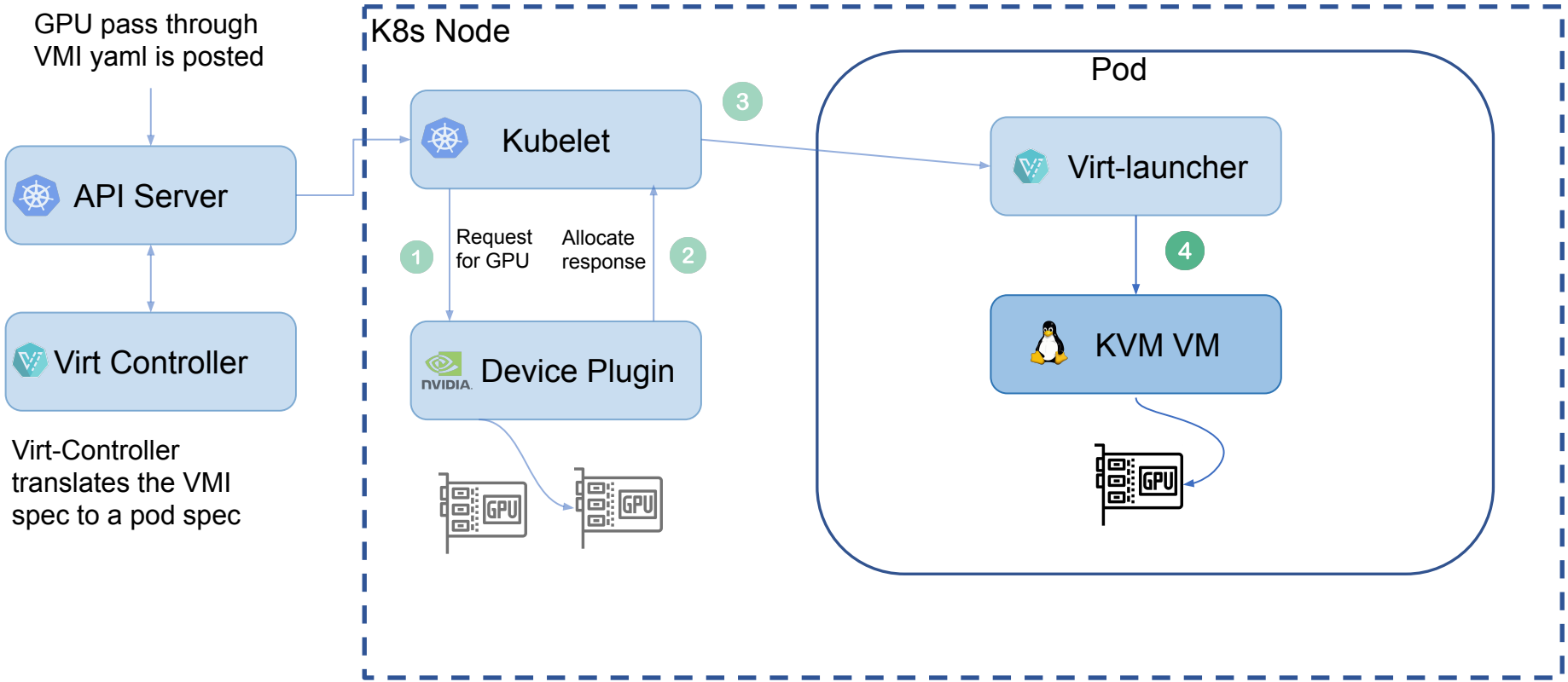
GPU Passthrough Lifecycle



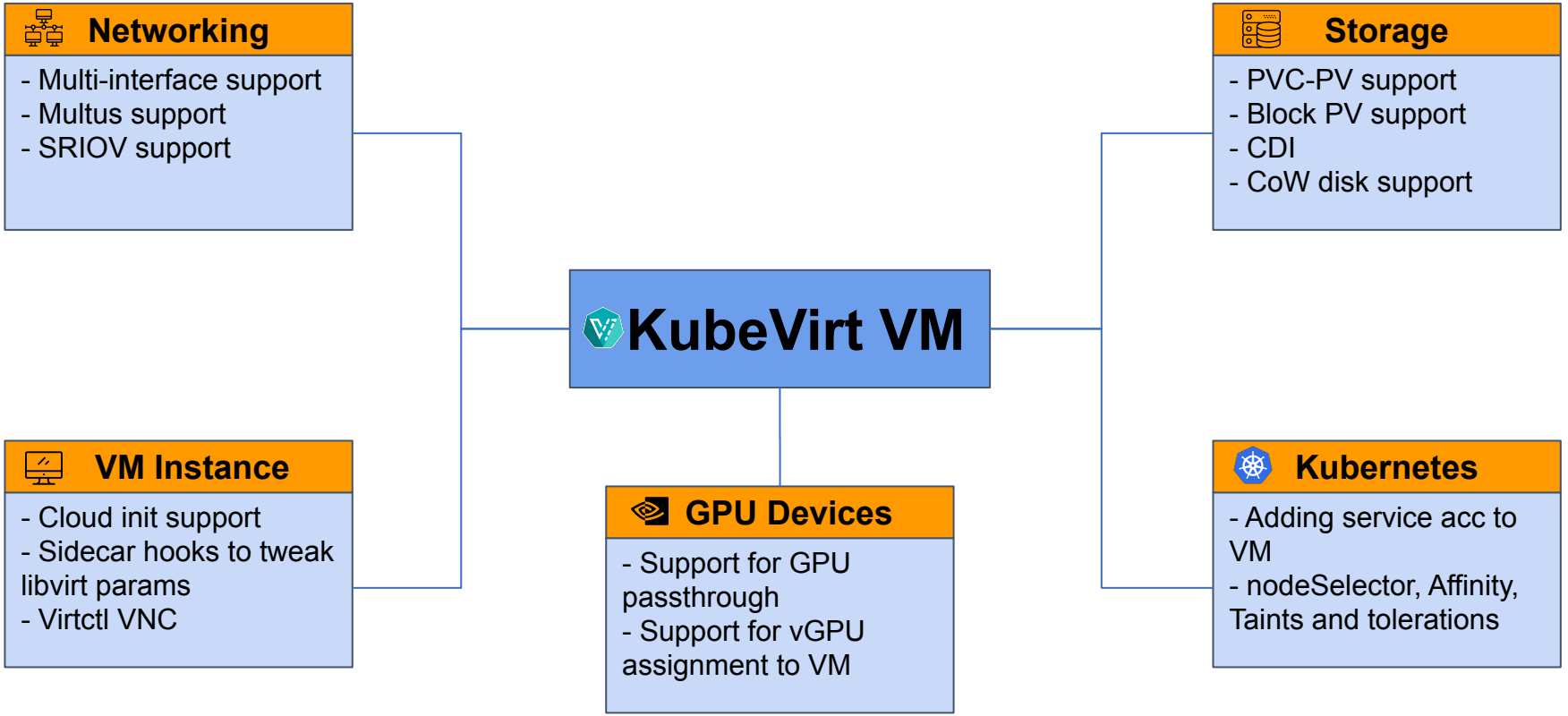
GPU Passthrough Lifecycle



GPU Passthrough Lifecycle



NVIDIA Usecase: Key features




KubeCon
CloudNativeCon
North America 2019
Thank You