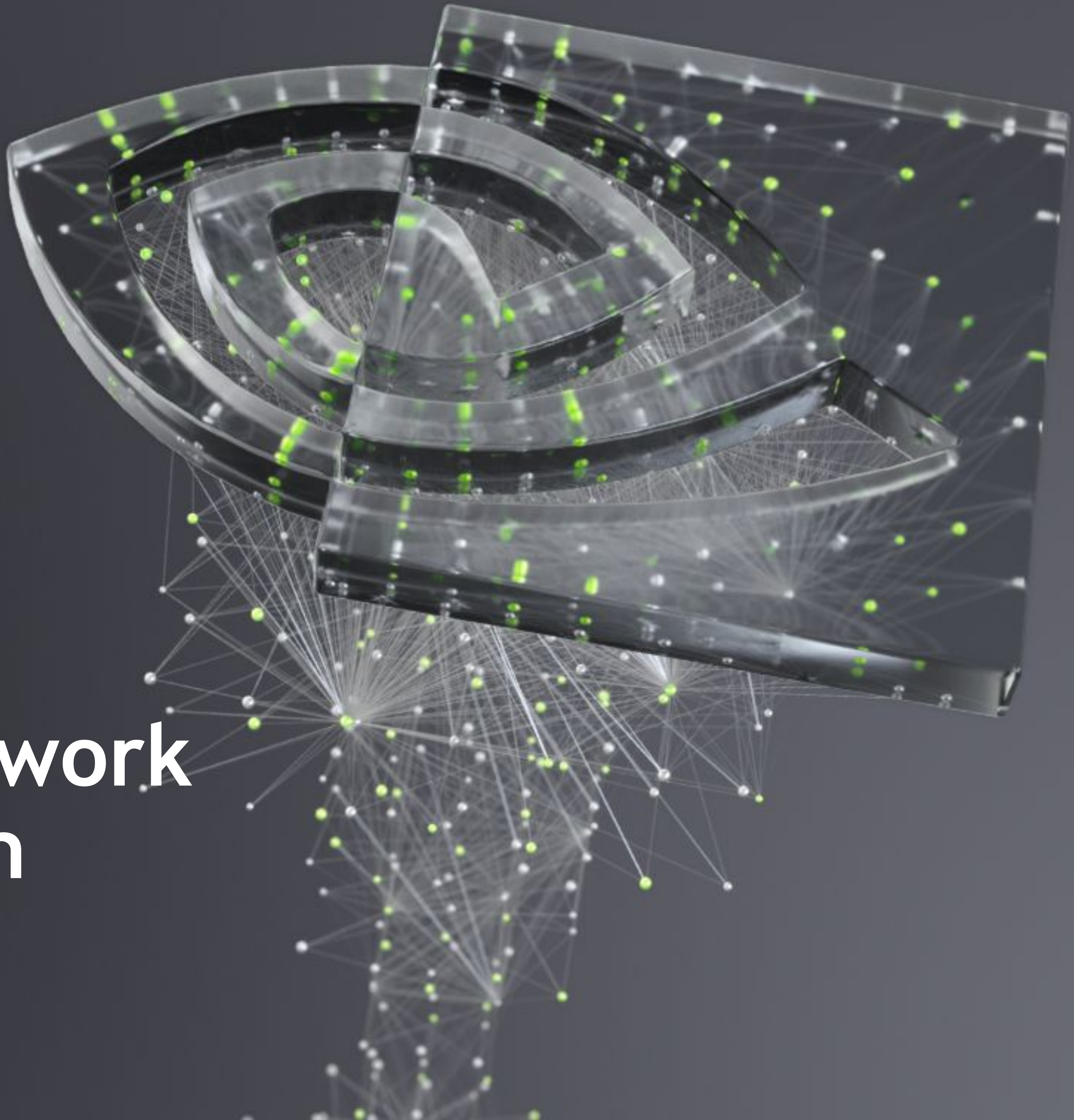
NVIDIA GPU and Network Operator News Flash

Kevin Jones



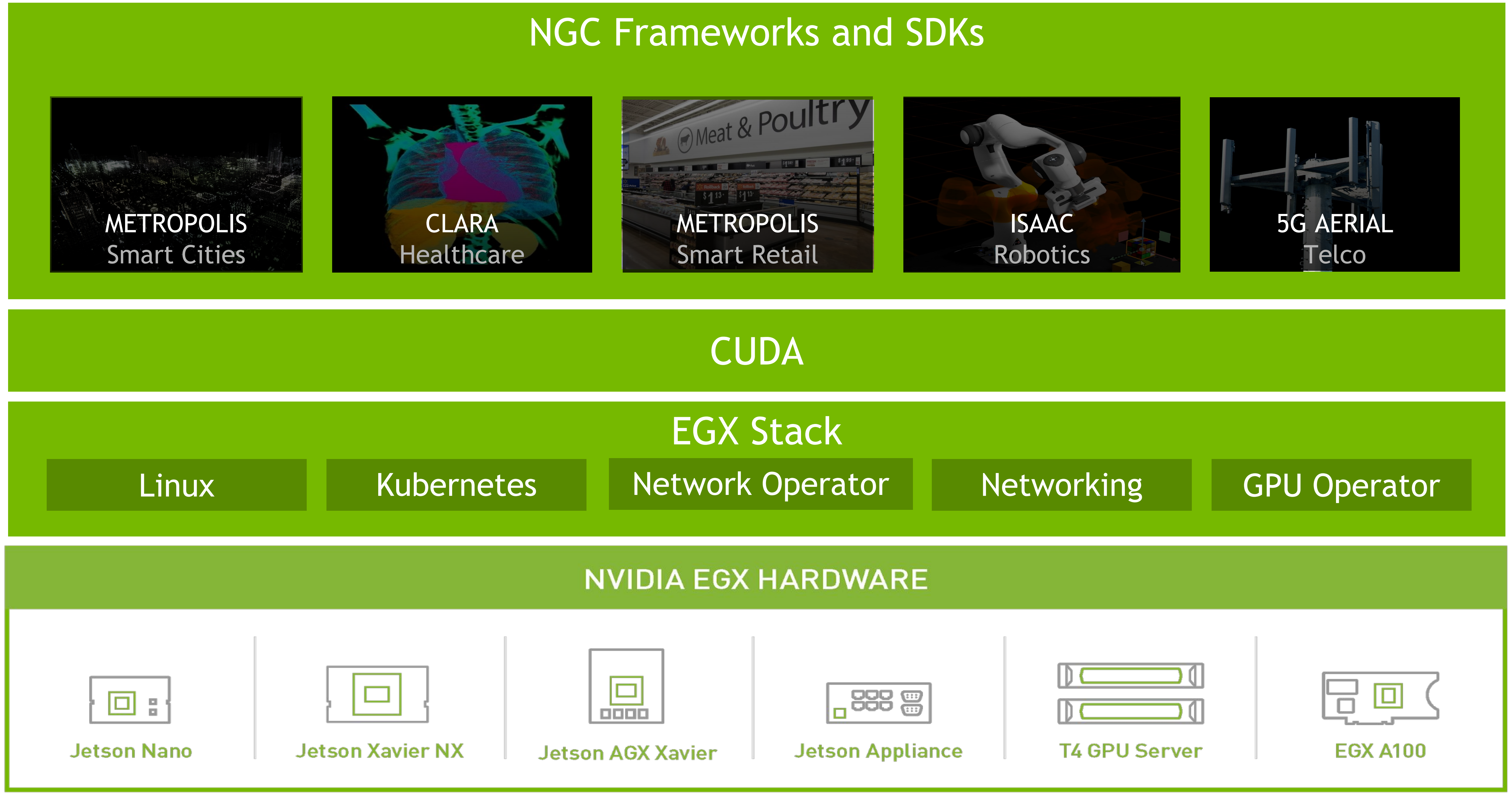
NVIDIA EGX PLATFORM

Cloud Native Platform for scale-out acceleration

Secure registry

Standard programming model

Certified systems



NGC Frameworks and SDKs

METROPOLIS
Smart Cities

CLARA
Healthcare

METROPOLIS
Smart Retail

ISAAC
Robotics

5G AERIAL
Telco

CUDA

EGX Stack

Linux

Kubernetes

Network Operator

Networking

GPU Operator

NVIDIA EGX HARDWARE

Jetson Nano

Jetson Xavier NX

Jetson AGX Xavier

Jetson Appliance

T4 GPU Server

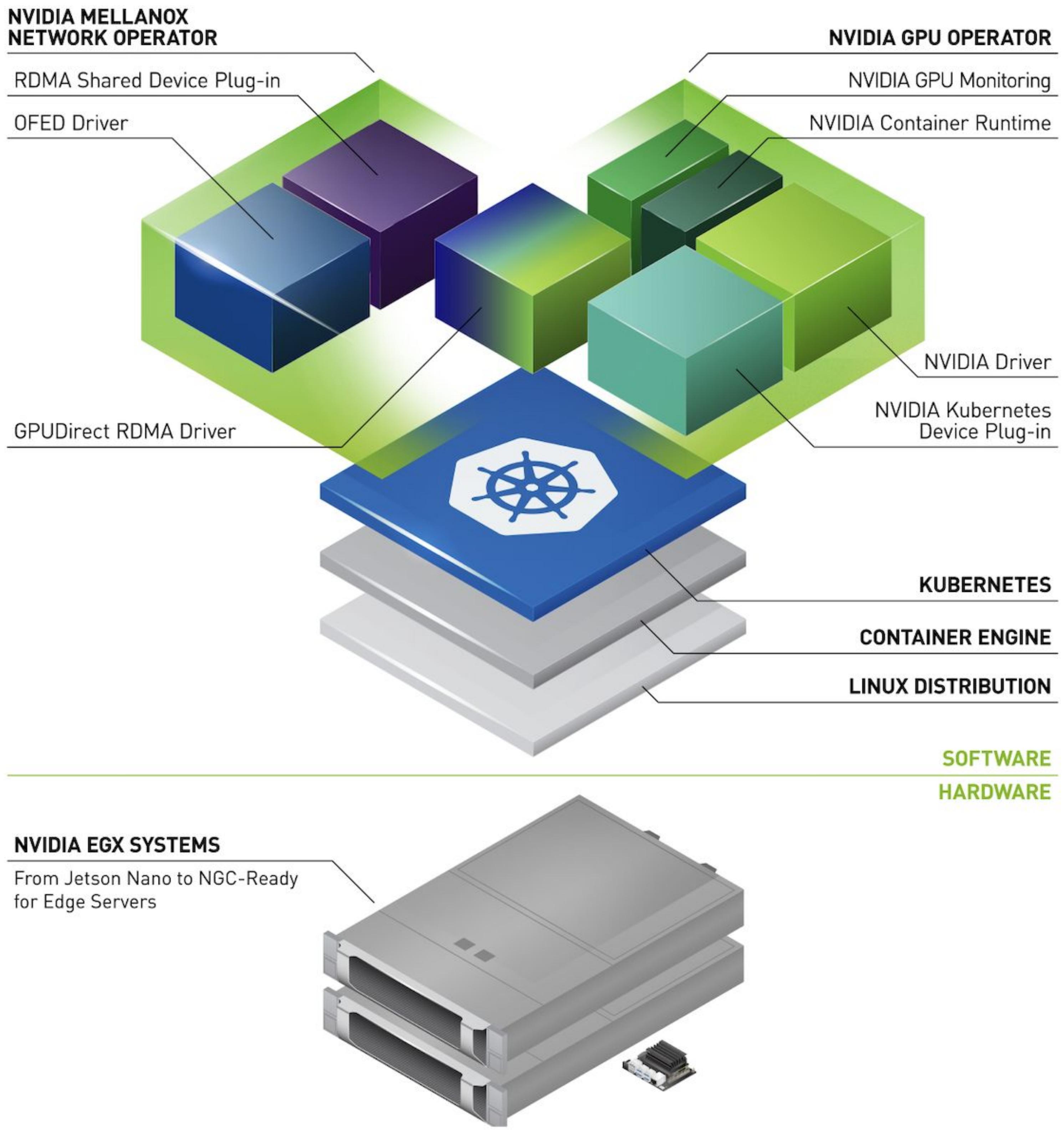
EGX A100

GPU-optimized containers

Cloud-native platform

NVIDIA EGX OPERATORS

Simplify GPU and SmartNIC Configuration on Kubernetes



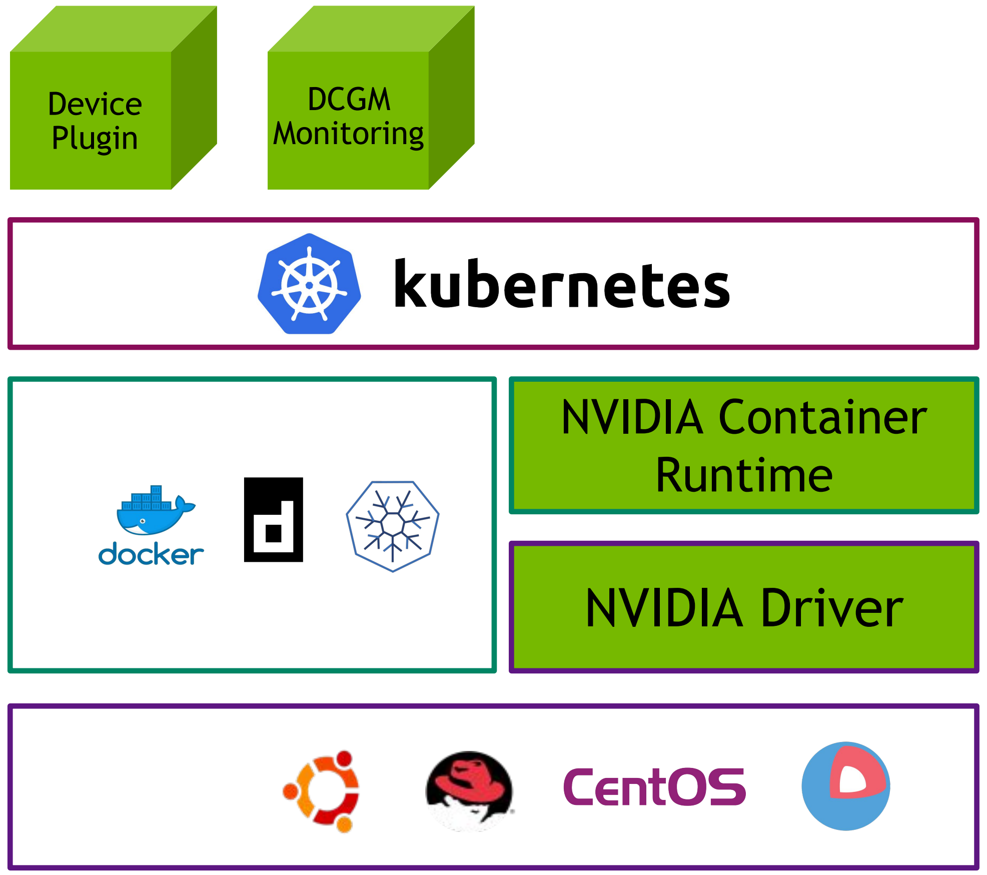
OPERATOR FRAMEWORK

- Operators are a pattern for developing Kubernetes-native applications
- Operators encode operational knowledge to automate administration tasks
- The NVIDIA EGX operators simplify GPU and SmartNIC configuration on Kubernetes
- When deployed together, they automatically enable GPUDirect RDMA
- NVIDIA EGX Operators are compatible with partner cloud platforms

GPU OPERATOR

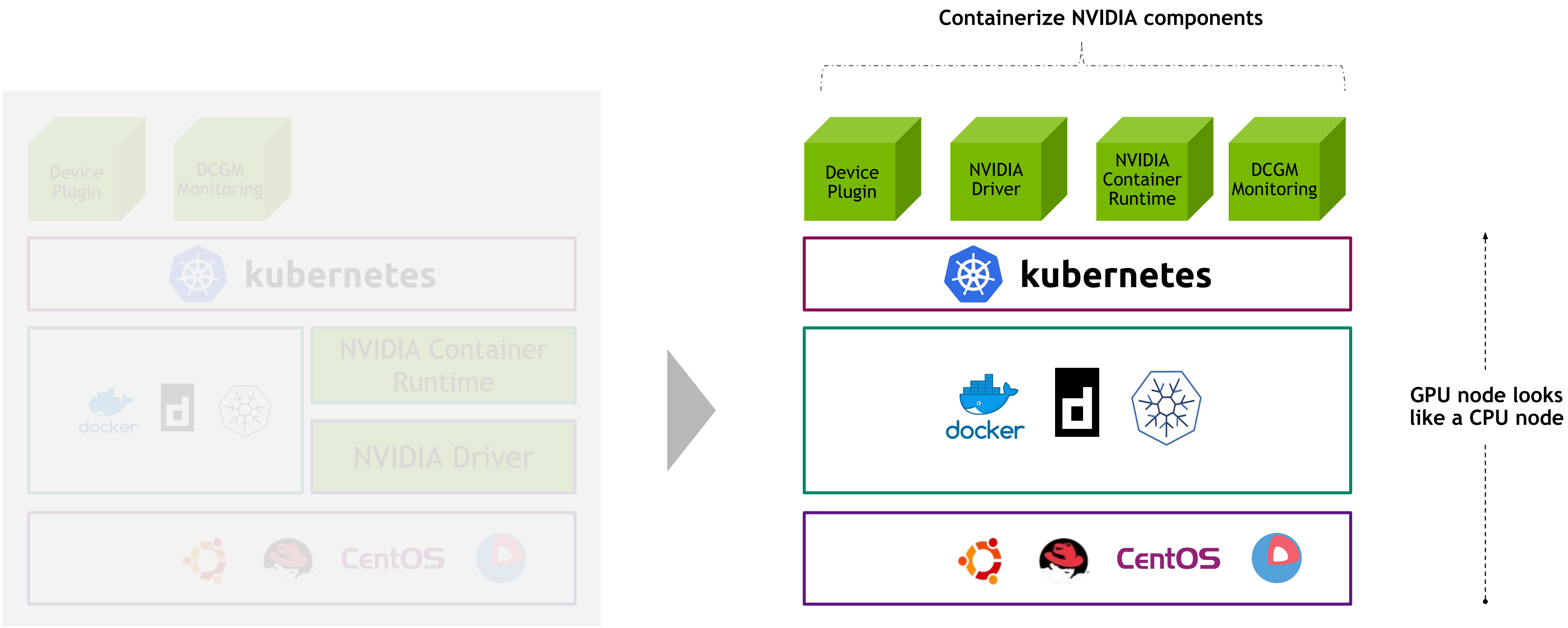
SIMPLIFYING GPU MANAGEMENT

Part I



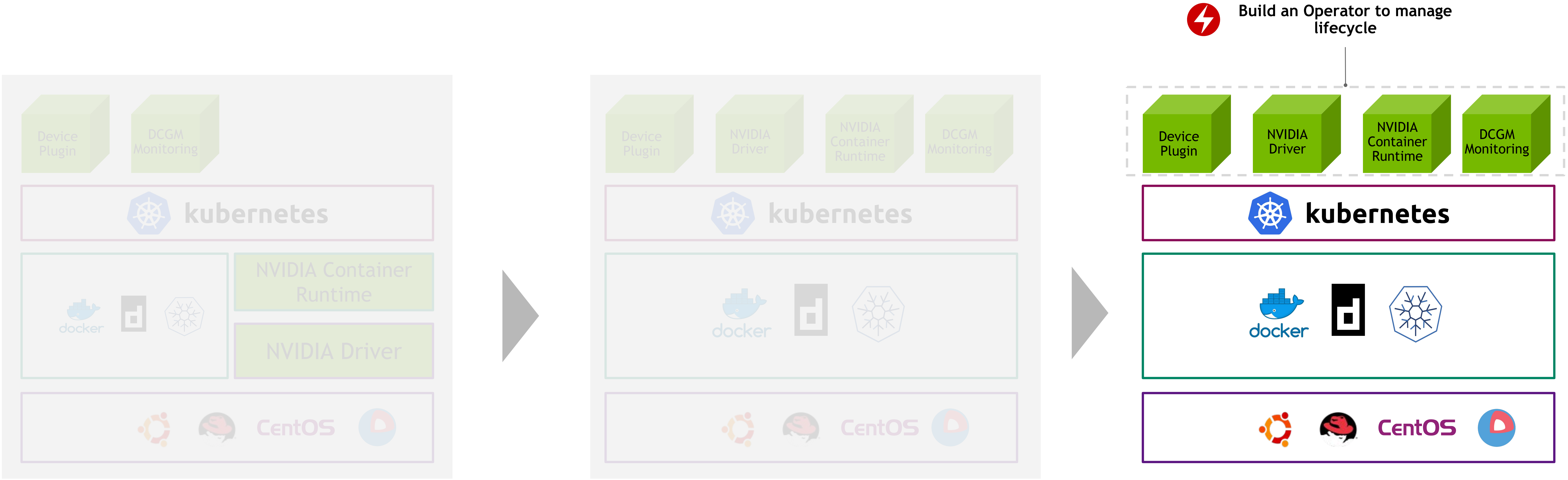
SIMPLIFYING GPU MANAGEMENT

Part I



SIMPLIFYING GPU MANAGEMENT

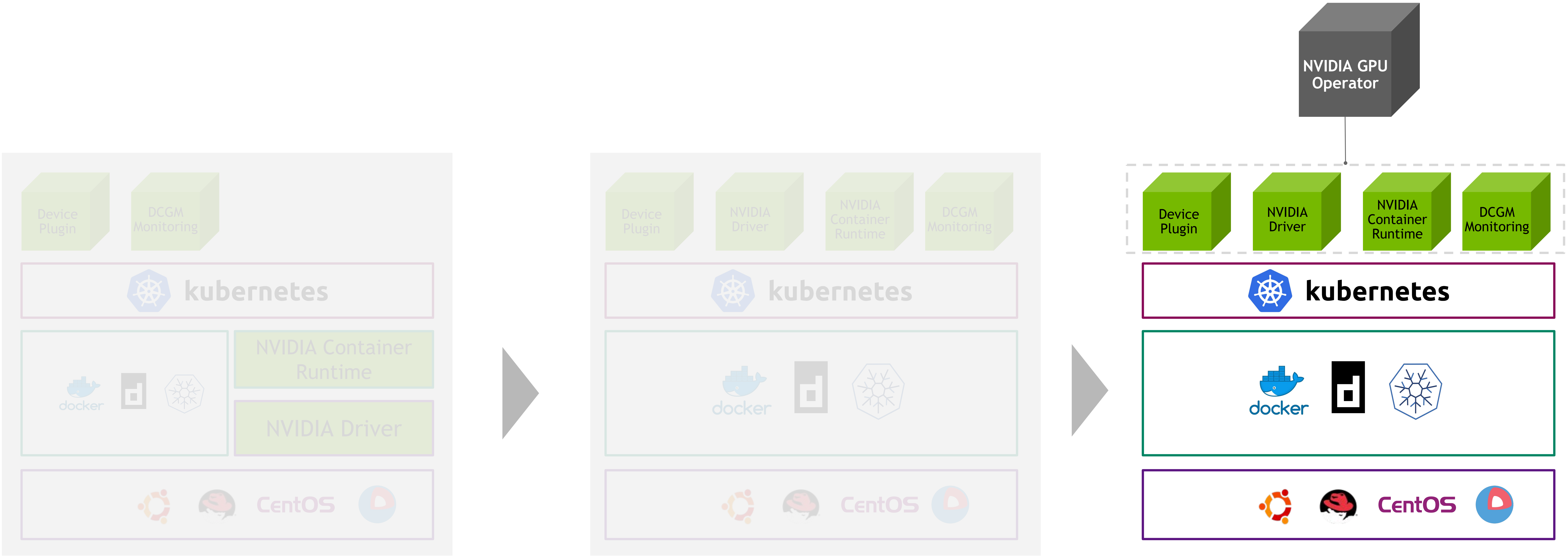
Part II



Build an Operator to manage lifecycle

NVIDIA GPU OPERATOR

https://github.com/NVIDIA/gpu-operator



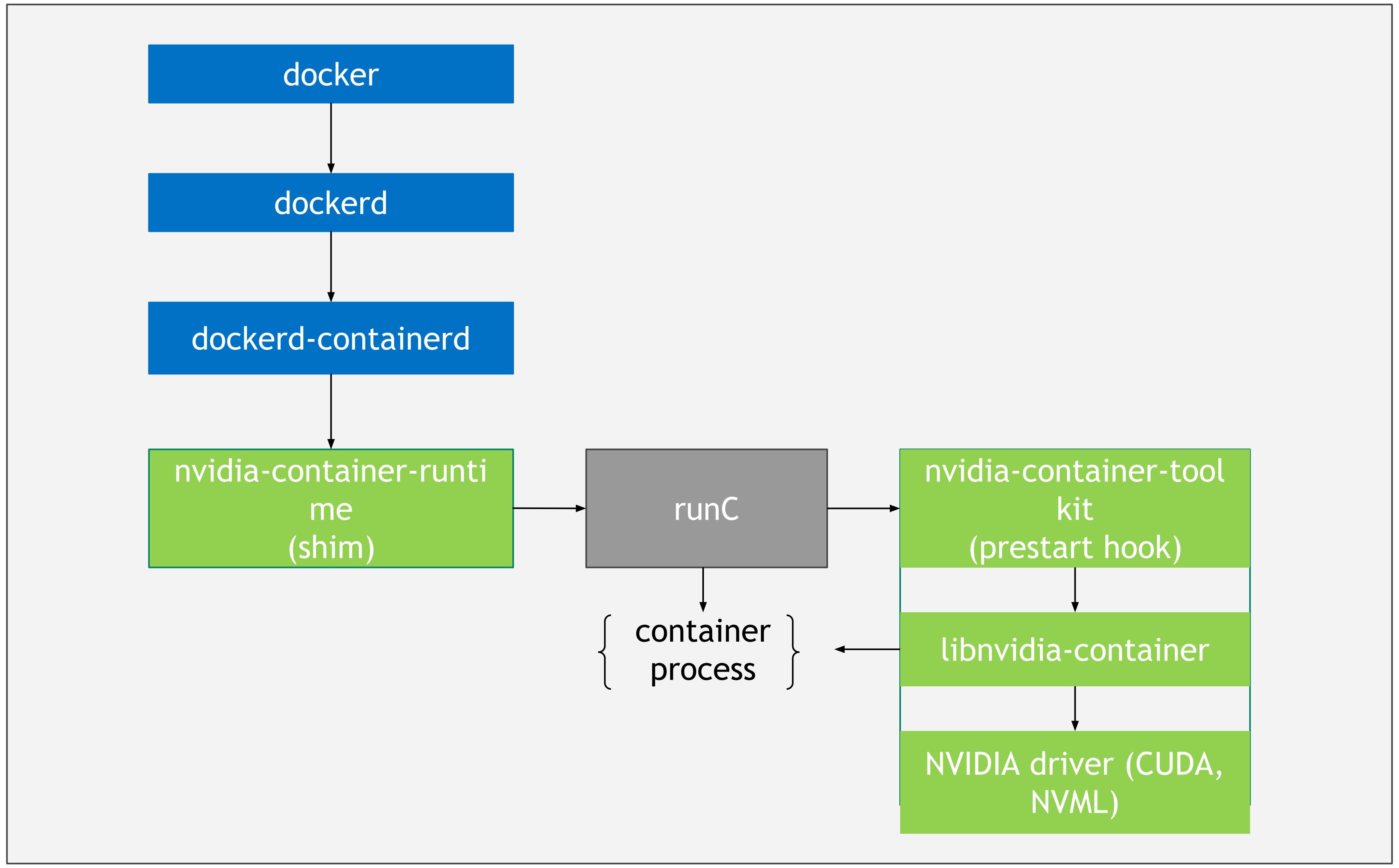
NVIDIA CONTAINER TOOLKIT

Enables GPU support in various container runtimes

- Run GPU containers with Docker, CRI-O, podman, LXC, Singularity etc.
- Integrates Linux container internals instead of wrapping specific runtimes (e.g. Docker)
- Exposes GPUs and drivers to the container

[Getting Started Documentation]

[Architecture and Packaging Overview]

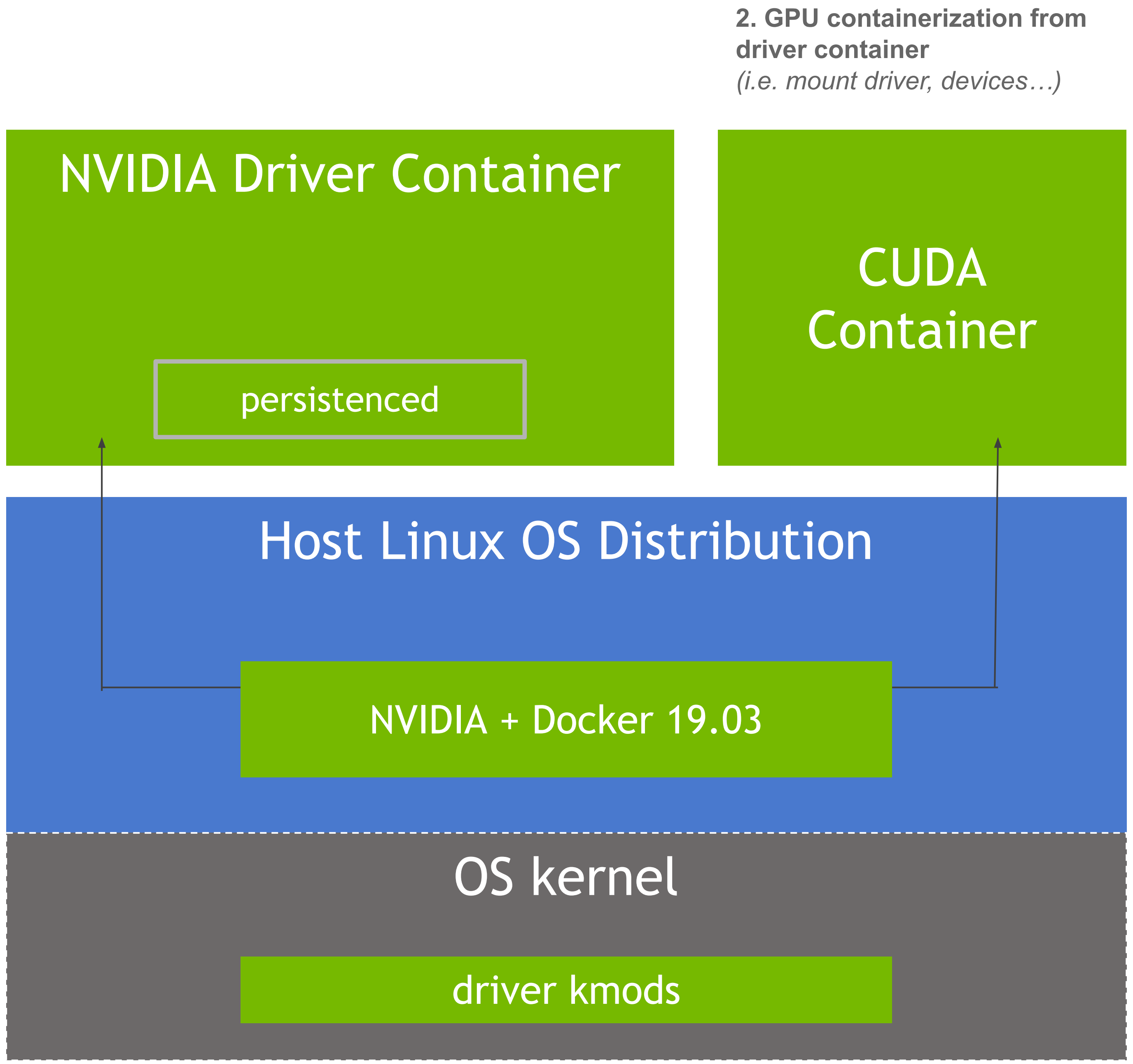

docker
dockerd
dockerd-containerd
nvidia-container-runtime (shim)
runC
nvidia-container-toolkit (prestart hook)
container process
libnvidia-container
NVIDIA driver (CUDA, NVML)

```
$ docker run -it --gpus all nvidia/cuda
```

CONTAINERIZED DRIVERS

NVIDIA drivers in containerized environments

- Goal is to simplify provisioning NVIDIA drivers (start/stop container)
- Other benefits
  - Speed
  - Use with container operating systems in the cloud (read-only rootfs, no tools)
  - Portable


2. GPU containerization from driver container
(i.e. mount driver, devices…)
NVIDIA Driver Container
persistenced
CUDA Container
Host Linux OS Distribution
NVIDIA + Docker 19.03
OS kernel
driver kmods

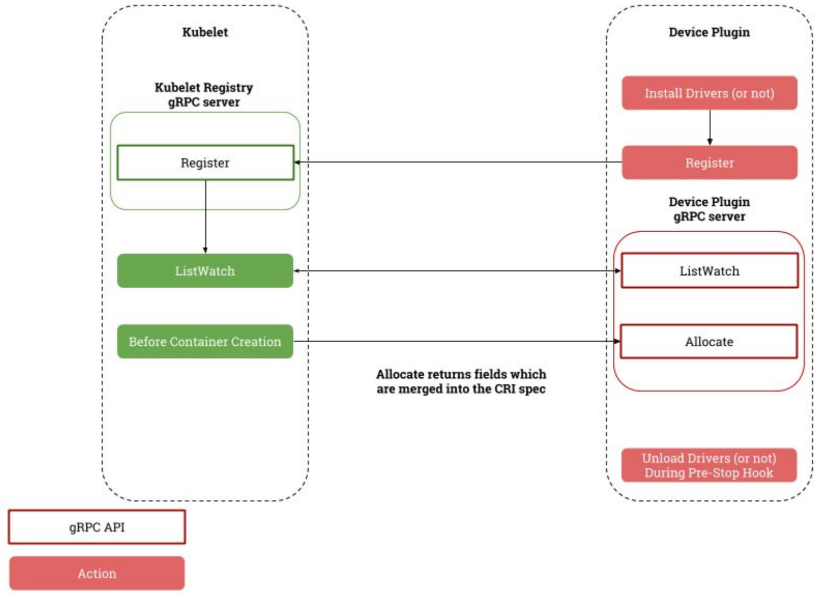
NVIDIA K8s DEVICE PLUGIN

Request GPUs as resources in podspecs

Kubernetes Device Plugins let pods access specialized hardware

The NVIDIA Kubernetes Device plugin:

- Enumerates the number of GPUs on each cluster node
- Keeps track of GPU health
- Runs GPU enabled pods

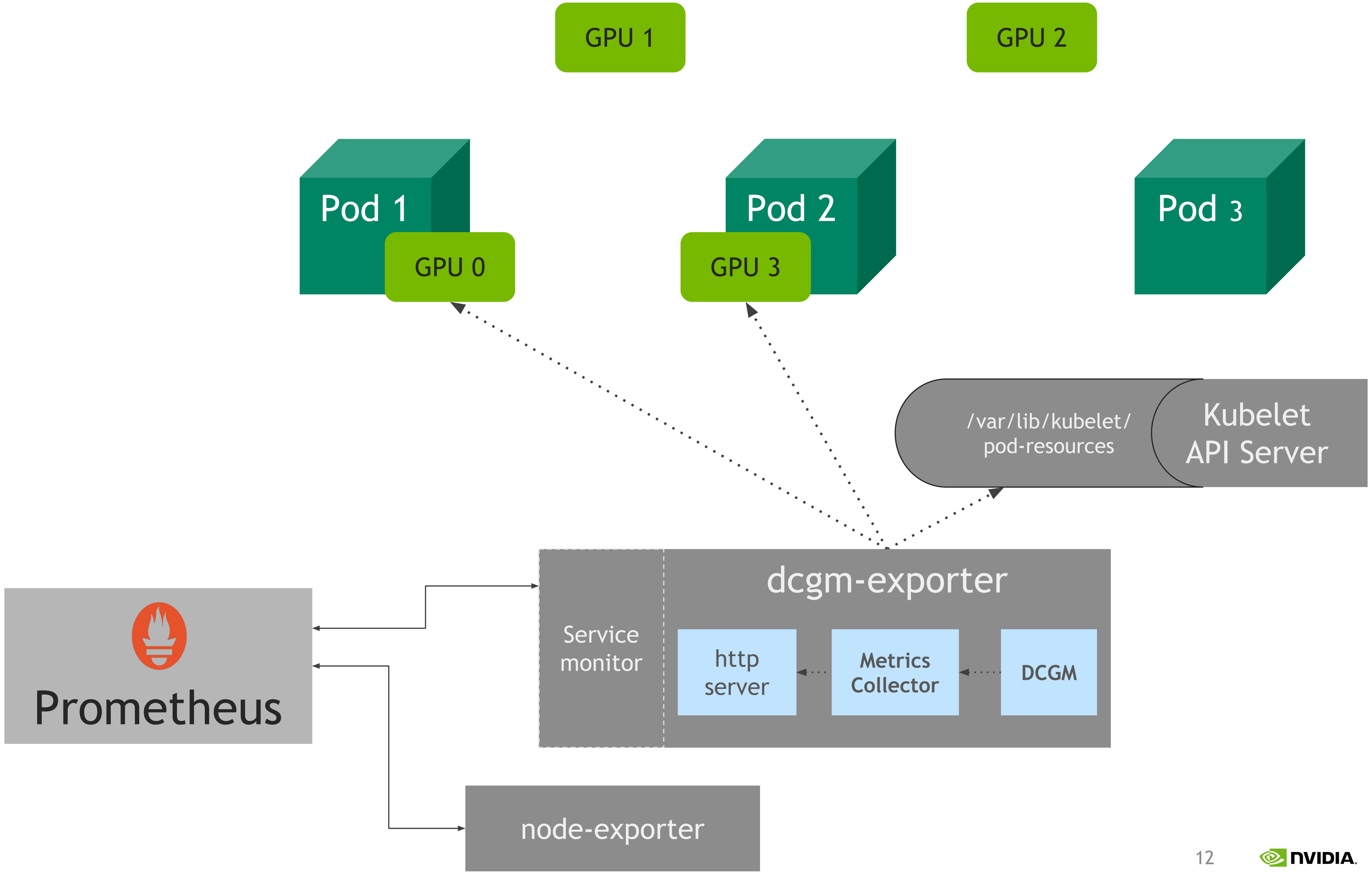

Kubelet
Kubelet Registry gRPC server
Register
ListWatch
Before Container Creation
Device Plugin
Install Drivers (or not)
Register
Device Plugin gRPC server
ListWatch
Allocate
Allocate returns fields which are merged into the CRI spec
Unload Drivers (or not) During Pre-Stop Hook
gRPC API
Action

```yaml
apiVersion: v1
kind: Pod
metadata:
  name: gpu-example
spec:
  containers:
    - name: gpu-example
      image: nvidia/cuda:11.0-base
      resources:
        limits:
          nvidia.com/gpu: 1
  nodeSelector:
    nvidia.com/gpu.product: A100-PCIE-40GB
    nvidia.com/cuda.runtime: 11.0
    nvidia.com/cuda.driver: 450.51.06
```

GPU TELEMETRY IN K8s USING DCGM

https://github.com/NVIDIA/gpu-monitoring-tools

- GPU device telemetry exposed via dcgm-exporter
- Pod level resource assignment is collected via the kubelet-pod resources (for pod-device mapping)
- Use node-exporter to expose node level information (including GPUs) to Prometheus

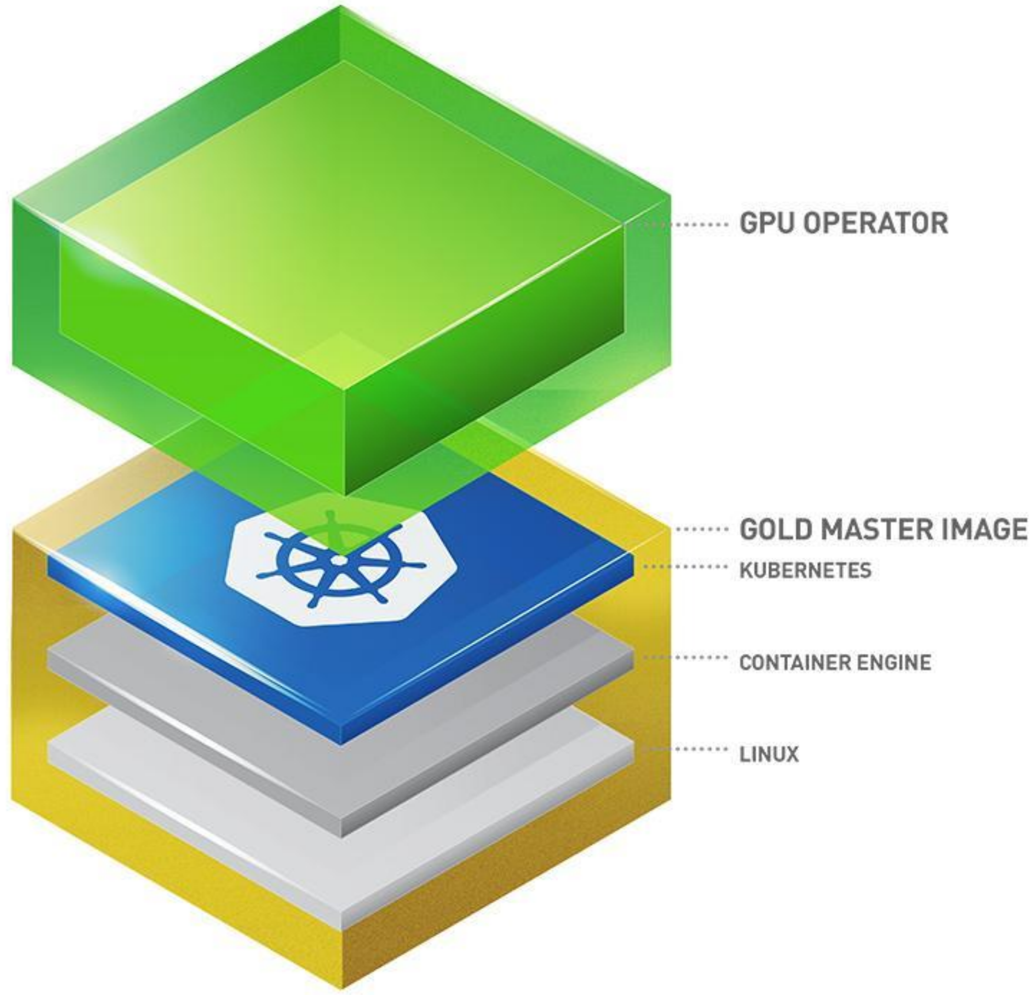


ROADMAP

- Upgrade management (handle driver and kernel updates, node reboots)
- Air-gapped installations (proxy, custom registries)
- Roles and bindings for RBAC
- MIG configurator in A100
- Integration with Kubernetes distributions

LINKS

- Open-source on GitHub: https://github.com/NVIDIA/gpu-operator
- Get started: https://docs.nvidia.com/datacenter/cloud-native/gpu-operator/overview.html
- Reach us: container-dev@ or #swgpu-linux-container on Slack



NETWORK OPERATOR

NVIDIA MELLANOX NETWORK OPERATOR

In A Nutshell

A Kubernetes Operator

Leverages Kubernetes custom resources and Operator framework

Defines a set of CRDs and acts on them reconcile the system

Configures fast networking, RDMA, and GPUDirect.

The Network Operator’s goal is to install the host networking components required to enable RDMA and GPUDirect in a Kubernetes cluster.

WHY DO WE NEED THIS?

Benefits of the Operator Approach

Deployment Experience

Simplify complex network deployment tasks

Portable across K8S platforms

Consistent deployment experience

Operational Efficiency

Manage network at cluster level

Put network administration on autopilot

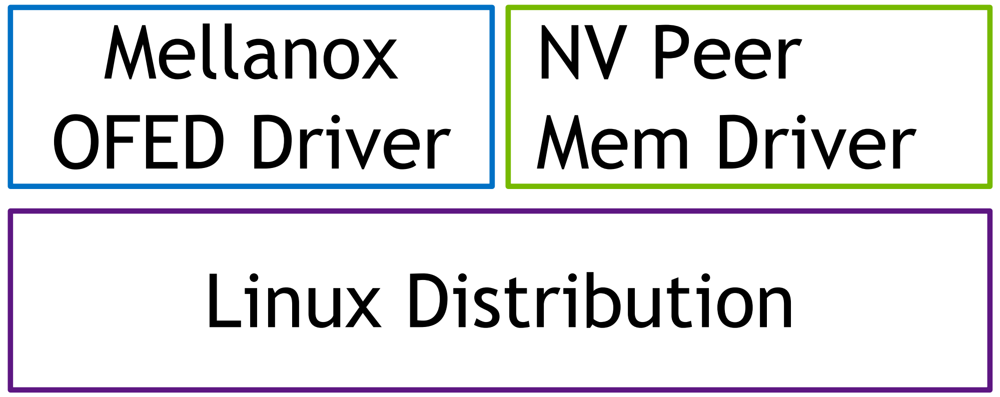
Architecture Aware

Enable GPUDirect RDMA automatically

Detect and expose SmartNIC capabilities

NVIDIA MELLANOX NETWORK OPERATOR

Enables RDMA and GPU Direct on Kubernetes



Mellanox OFED Driver

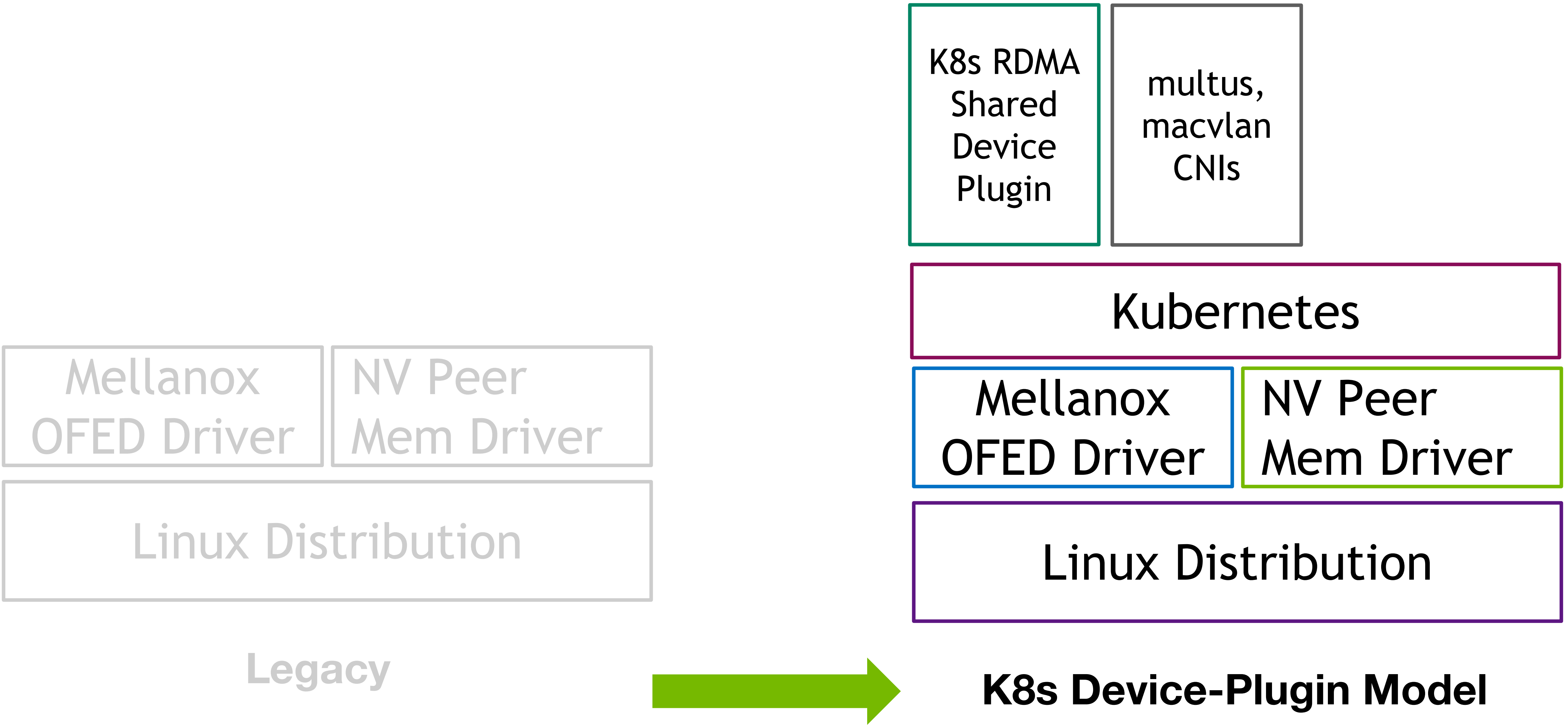
NV Peer Mem Driver

Linux Distribution

Legacy

NVIDIA MELLANOX NETWORK OPERATOR

Enables RDMA and GPU Direct on Kubernetes



Mellanox OFED Driver

NV Peer Mem Driver

Linux Distribution

Legacy

K8s RDMA Shared Device Plugin

multus, macvlan CNIs

Kubernetes

Mellanox OFED Driver

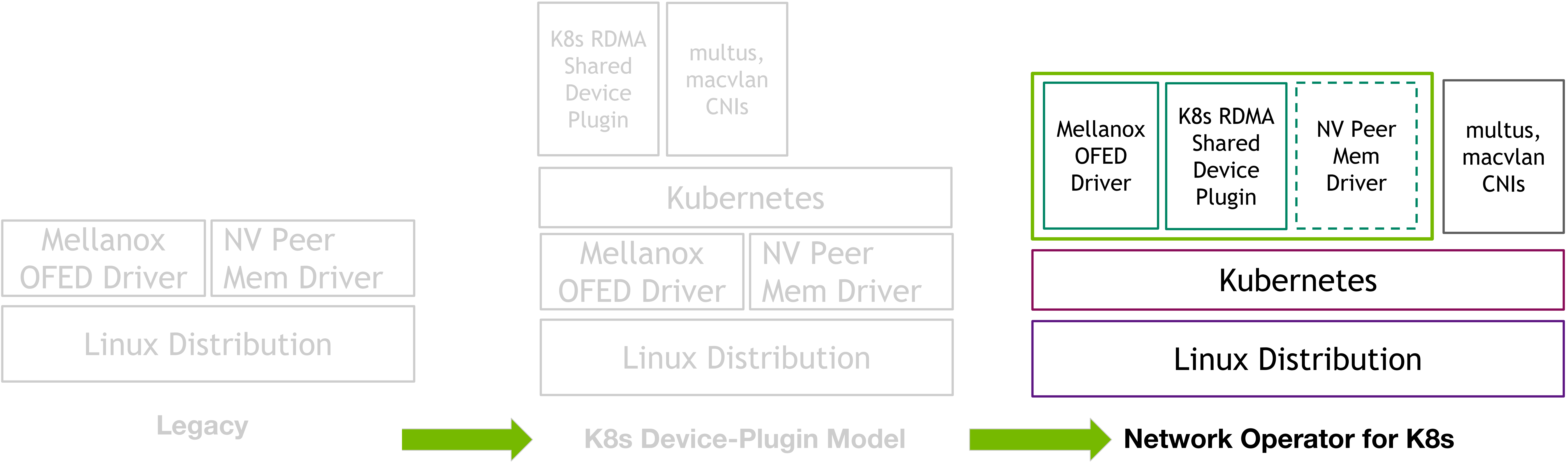
NV Peer Mem Driver

Linux Distribution

K8s Device-Plugin Model

NVIDIA MELLANOX NETWORK OPERATOR

Enables RDMA and GPU Direct on Kubernetes



Mellanox OFED Driver

NV Peer Mem Driver

Linux Distribution

Legacy

K8s RDMA Shared Device Plugin

multus, macvlan CNIs

Kubernetes

Mellanox OFED Driver

NV Peer Mem Driver

Linux Distribution

K8s Device-Plugin Model

Mellanox OFED Driver

K8s RDMA Shared Device Plugin

NV Peer Mem Driver

multus, macvlan CNIs

Kubernetes

Linux Distribution

Network Operator for K8s

MELLANOX OFED DRIVER CONTAINER

Loads Mellanox OFED driver into kernel

- Prebuilt for Distribution & Kernel
- Deployed onto nodes based on node labels
  - Arch, OS, Kernel, Mellanox PCI labels
- Exposes container rootfs to host to allow kernel module compilation against updated headers
- [Re]Loads kernel RDMA stack and Mellanox driver stack on container start
- Unloads kernel RDMA stack and Mellanox driver stack on container exit

K8S RDMA SHARED DEVICE PLUGIN

Run RDMA workloads in Kubernetes

```
apiVersion: v1
kind: Pod
metadata:
  name: rdma-pod
  annotations:
    k8s.v1.cni.cncf.io/networks:rdma-net-ipam
spec:
  containers:
  - image: mellanox/rping-test
    name: rdma-pod-ctr
    securityContext:
      capabilities:
        add: [ "IPC_LOCK" ]
    resources:
      limits:
        rdma/hca_shared_devices_a: 1
      requests:
        rdma/hca_shared_devices_a: 1
```

K8s RDMA Shared Device Plugins let pods perform RDMA by exposing RDMA device files to container in a shared manner.

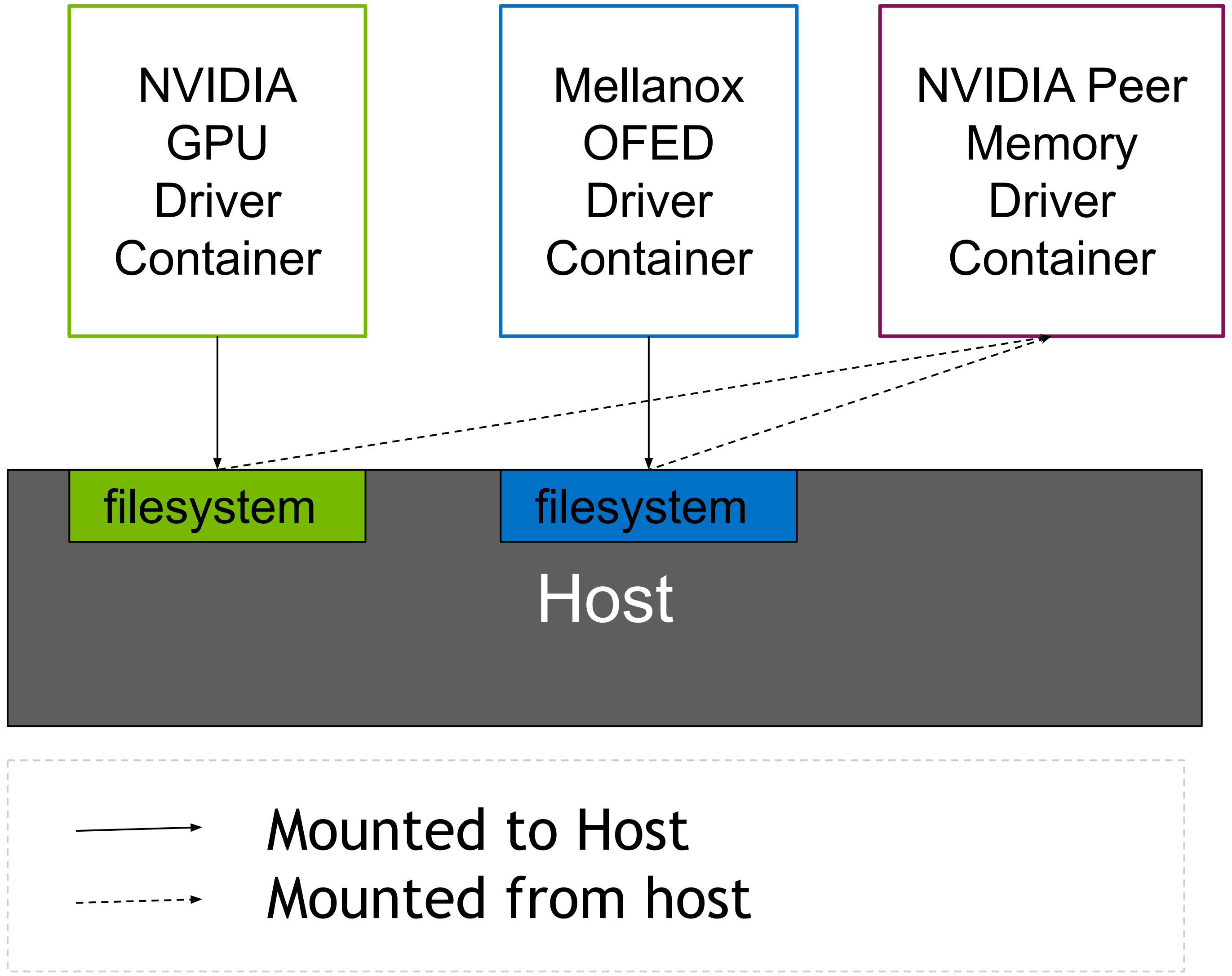
That is, all pods share the same RDMA device in a node.

K8s RDMA Shared Device plugin:

- Runs on RDMA enabled nodes
- Exposes “unlimited” number of RDMA resources
- Instructs kubelet to mount to pod relevant RDMA char devices under */dev/infiniband*

NVIDIA PEER MEMORY DRIVER CONTAINER

Compiles and Loads NV Peer Memory client driver into kernel

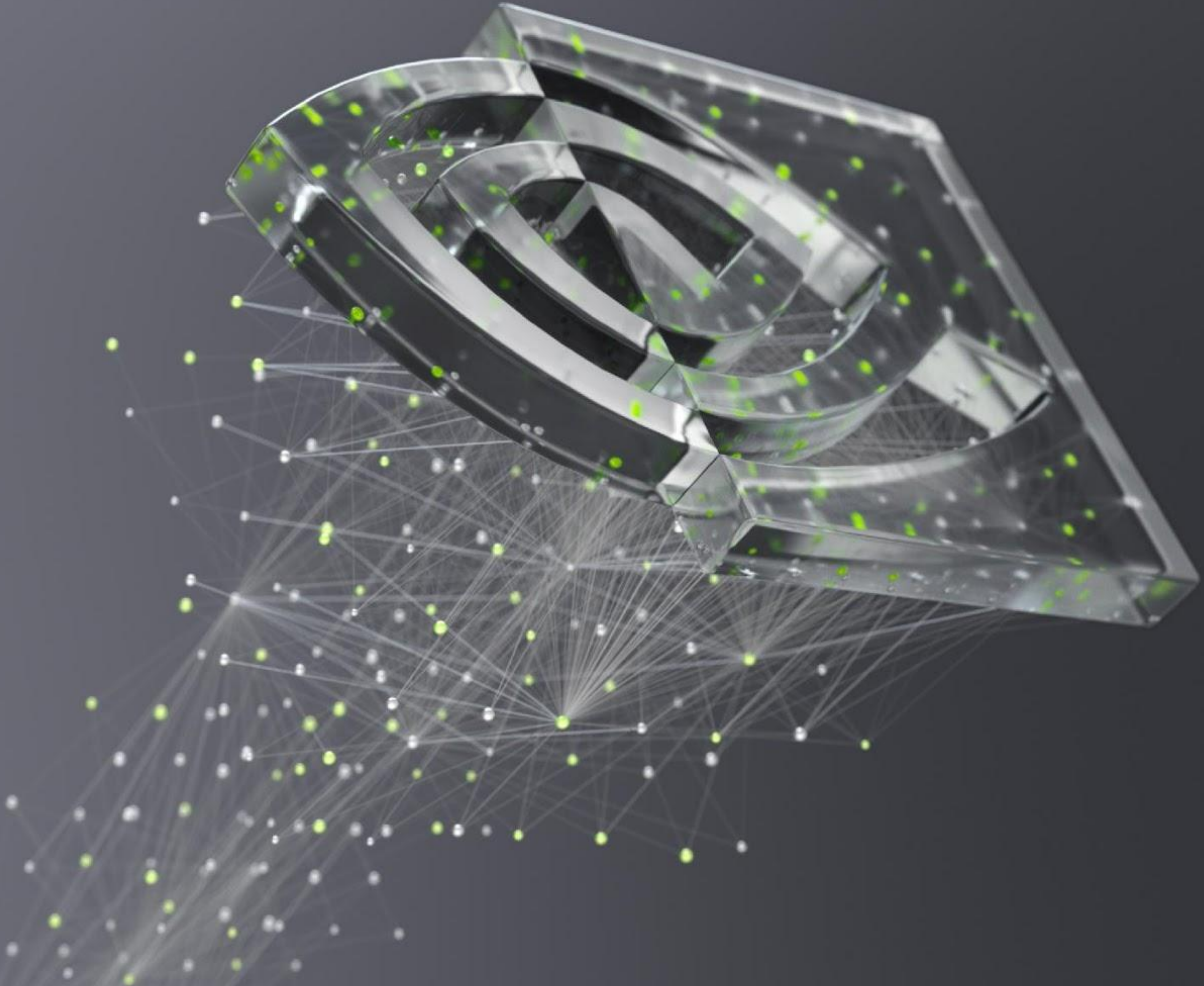

NVIDIA GPU Driver Container
Mellanox OFED Driver Container
NVIDIA Peer Memory Driver Container
filesystem
filesystem
Host
Mounted to Host
Mounted from host

- Deployed on GPU and RDMA enabled nodes
  - Based on Arch, OS, NIC and GPU labels
- Compiles against both Mellanox OFED and NVIDIA GPU Drivers.
- Loads nv_peer_mem module into kernel
- Unloads nv_peer_mem module on container exit

FUTURE WORK

Cloud Native GPU & RDMA accelerated Platform for Edge AI

- ✓ Helm Deployment - December
- ✓ Deploy NFD to label RDMA nodes - December
- ✓ Secondary network deployment (Macvlan shared mode) - December
- Kernel update support (Ubuntu only ATM)
- Openshift integration
- Secondary network deployment (SR-IOV)
- Network device Initialization (e.g udev)
- Network device configuration (e.g device up, MTU)
- ConnectX NIC as primary network
- Additional Platform support



NVIDIA