KubeCon | CloudNativeCon

North America 2025

Tutorial: Unlock the Future of Kubernetes and Accelerators with Dynamic Resource Allocation (DRA)

Rey Lejano, Red Hat

#KubeCon #CloudNativeCon

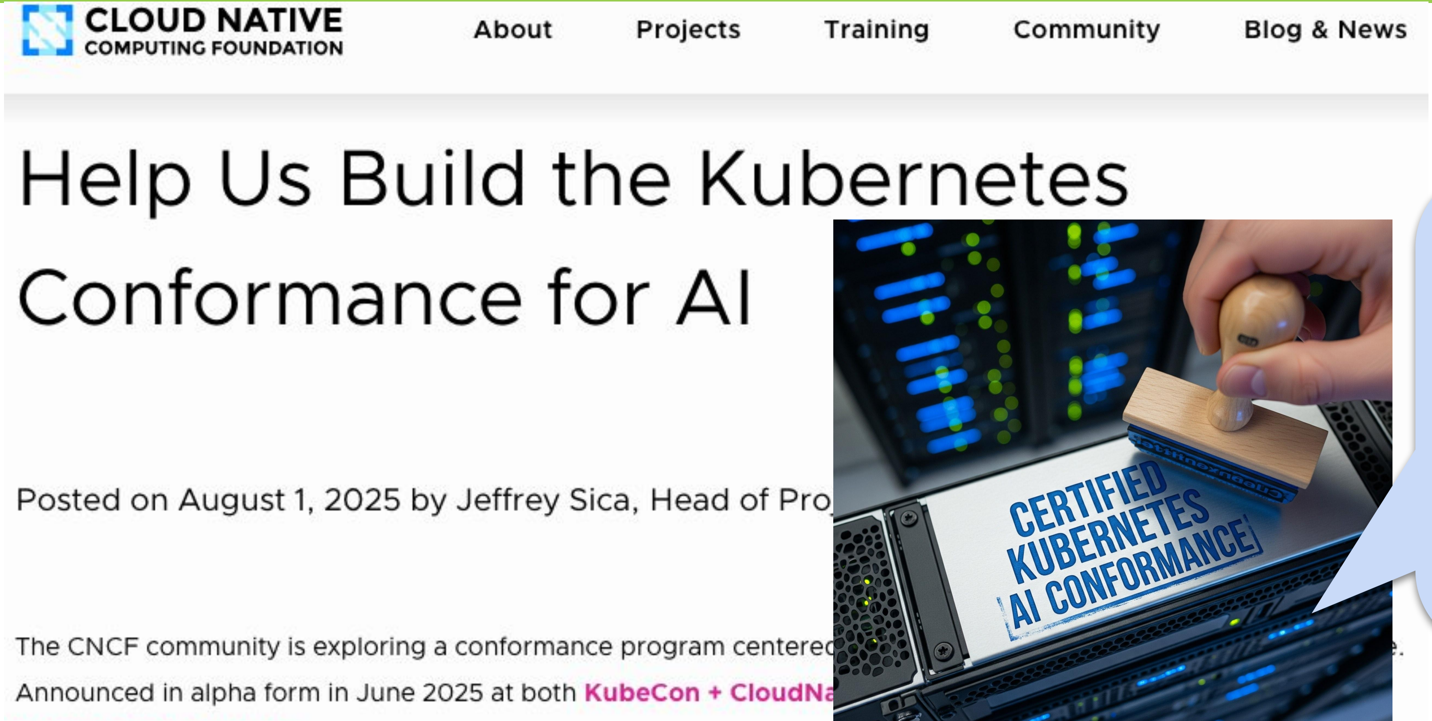
whoami

Rey Lejano

Red Hatter
Kubernetes maintainer
Kubernetes SIG Docs co-chair
Kubernetes SIG Security's Third-Party Audit subproject lead
Kubernetes v1.23 Release Lead
CNCF Ambassador
KCD San Francisco Bay Area organizer
Cloud Native San Francisco community group organizer

@reylejano
https://www.linkedin.com/in/rey-lejano/
@reylejano

Kubernetes AI Conformance


CLOUD NATIVE COMPUTING FOUNDATION

Help Us Build the Kubernetes Conformance for AI

Posted on August 1, 2025 by Jeffrey Sica, Head of Pro

The CNCF community is exploring a conformance program centered

Announced in alpha form in June 2025 at both KubeCon + CloudNa

*made by Gemini

MUST: Support Dynamic Resource Allocation (DRA) APIs to enable more flexible and fine-grained resource requests beyond simple counts

https://github.com/cncf/k8s-ai-conformance
https://www.cncf.io/blog/2025/08/01/help-us-build-the-kubernetes-conformance-for-ai/

Device Plugins (old way)

Device plugins let Pods access specialized hardware (e.g. GPUs) on nodes

- Request for a GPU in the Pod’s .spec.resource.containers.limits
    - the cluster will use limits == requests
- Node feature discovery is used to find PCI devices and label nodes

GPU Support in Kubernetes with Device Plugins

Host Components

- nvidia-container-toolkit
- nvidia-gpu-driver

Kubernetes Components

- k8s-device-plugin
- node-feature-discovery
- dcgm-exporter

```
apiVersion: v1
kind: Pod
metadata:
  name: gpu-example
spec:
  containers:
  - name: device-plugin-example
    image: nvidia/cuda
    resources:
      limits:
        nvidia.com/gpu: 1
  nodeSelector:
    nvidia.com/gpu.product: V100-PCIE-16GB
```

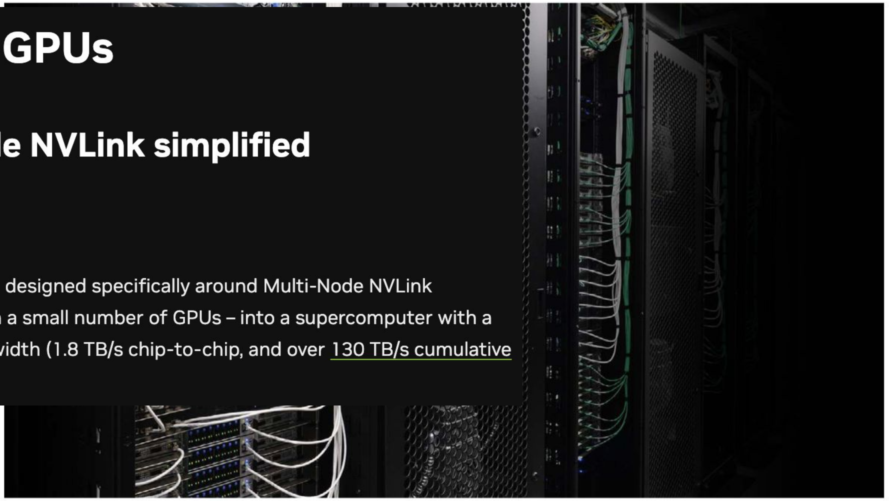
Dynamic Resource Allocation

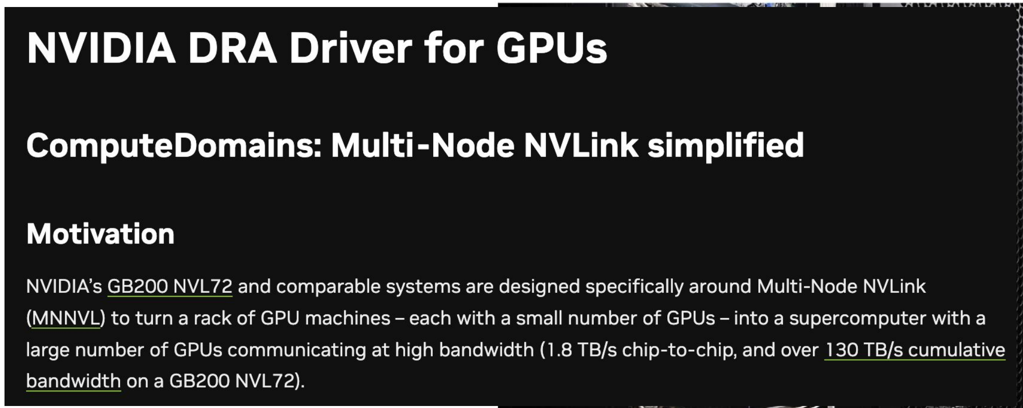
Why change?

- Enables hardware vendors to extend Kubernetes with DRA drivers
  - DRA drivers are responsible for the hardware
- Accelerators and other devices as well like NICs, FPGAs, network-attached devices
- Pass configuration parameters to devices
- Pod is scheduled when required resources are available
- Share resources (e.g. GPU) with multiple Pods
- Many more features to come

With DRA …

NVIDIA GB200 NVL72



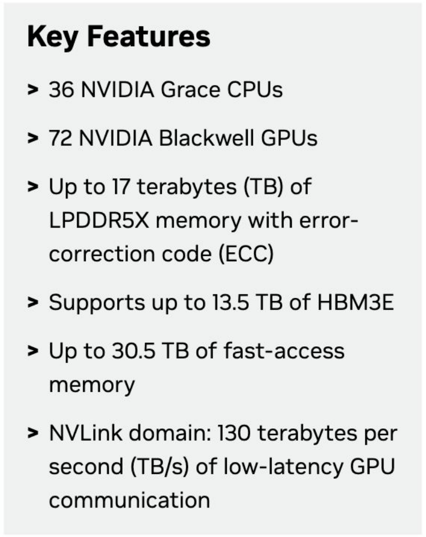


NVIDIA DRA Driver for GPUs

ComputeDomains: Multi-Node NVLink simplified

Motivation

NVIDIA's GB200 NVL72 and comparable systems are designed specifically around Multi-Node NVLink (MNNVL) to turn a rack of GPU machines – each with a small number of GPUs – into a supercomputer with a large number of GPUs communicating at high bandwidth (1.8 TB/s chip-to-chip, and over 130 TB/s cumulative bandwidth on a GB200 NVL72).



Key Features

> 36 NVIDIA Grace CPUs

> 72 NVIDIA Blackwell GPUs

> Up to 17 terabytes (TB) of LPDDR5X memory with error-correction code (ECC)

> Supports up to 13.5 TB of HBM3E

> Up to 30.5 TB of fast-access memory

> NVLink domain: 130 terabytes per second (TB/s) of low-latency GPU communication

Sources:
https://www.nvidia.com/en-us/data-center/gb200-nvl72/
https://docs.nvidia.com/datacenter/cloud-native/gpu-operator/latest/dra-cds.html

Device Plugings vs DRA

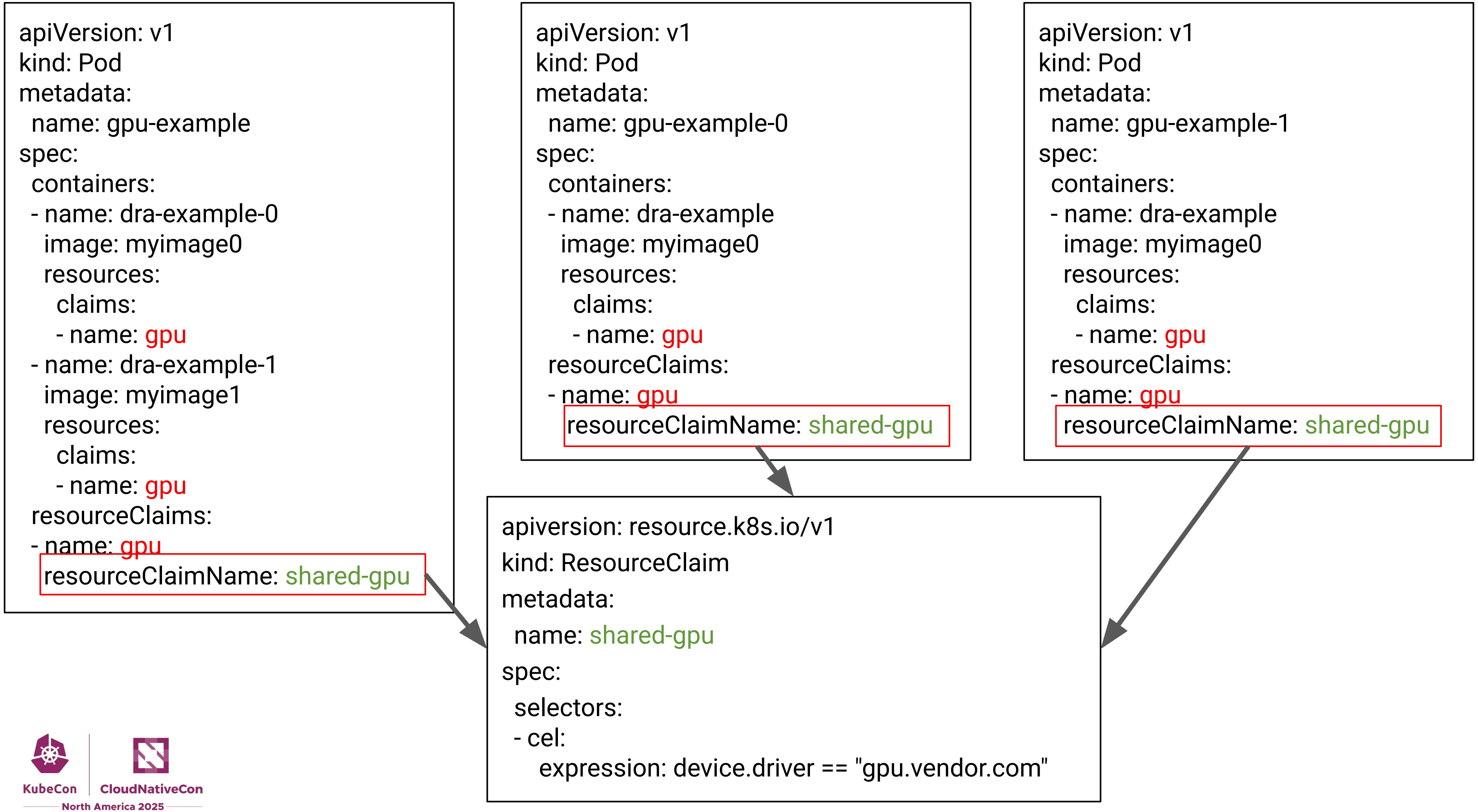
KubeCon CloudNativeCon North America 2025

Device Plugin

```
apiVersion: v1
kind: Pod
metadata:
  name: device-plugin-example
spec:
  containers:
  - name: device-plugin-ctr
    image: ai-workload:v2
    resources:
      limits:
        nvidia.com/gpu: 2
  nodeSelector:
    nvidia.com/gpu.product: V100-PCIE-16GB
```

DRA

```
apiVersion: v1
kind: Pod
metadata:
  name: dra-example
spec:
  containers:
  - name: dra-example-ctr
    image: ai-workload:v2
    resources:
      claims:
      - name: gpu0
      - name: gpu1
  resourceClaims:
  - name: gpu0
    resourceClaimName: shared-gpu
  - name: gpu1
    resourceClaimName: inference-gpu
```



```yaml
apiVersion: v1
kind: Pod
metadata:
  name: gpu-example
spec:
  containers:
  - name: dra-example-0
    image: myimage0
    resources:
      claims:
      - name: gpu
  - name: dra-example-1
    image: myimage1
    resources:
      claims:
      - name: gpu
  resourceClaims:
  - name: gpu
    resourceClaimName: shared-gpu
```

```yaml
apiVersion: v1
kind: Pod
metadata:
  name: gpu-example-0
spec:
  containers:
  - name: dra-example
    image: myimage0
    resources:
      claims:
      - name: gpu
  resourceClaims:
  - name: gpu
    resourceClaimName: shared-gpu
```

```yaml
apiVersion: v1
kind: Pod
metadata:
  name: gpu-example-1
spec:
  containers:
  - name: dra-example
    image: myimage0
    resources:
      claims:
      - name: gpu
  resourceClaims:
  - name: gpu
    resourceClaimName: shared-gpu
```

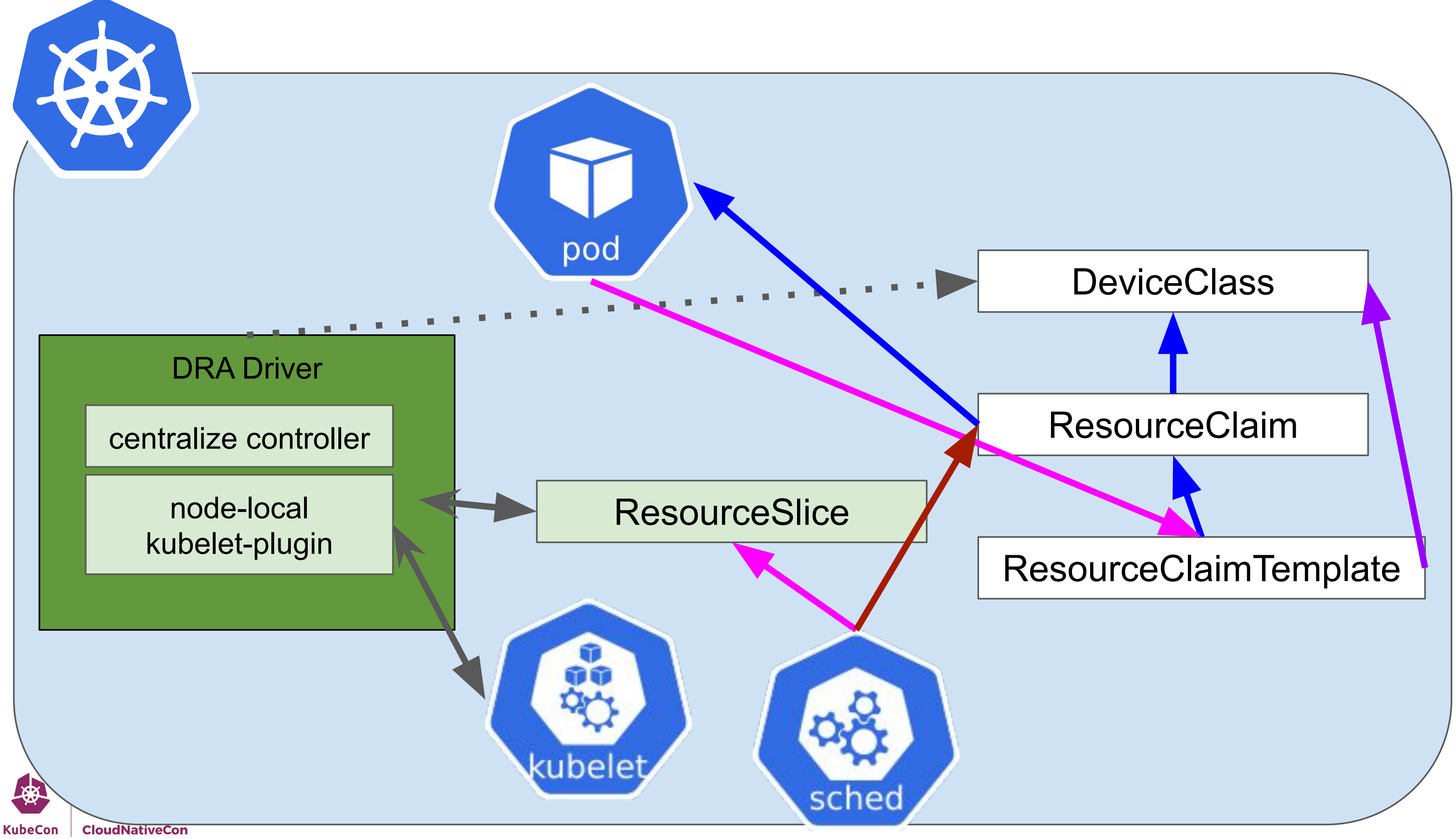
```yaml
apiversion: resource.k8s.io/v1
kind: ResourceClaim
metadata:
  name: shared-gpu
spec:
  selectors:
  - cel:
      expression: device.driver == "gpu.vendor.com"
```

DRA Resources

DRA Driver (from vendor)

- node-local, kubelet plugin (DaemonSet)
- controller (Deployment)

ResourceSlice (created by DRA driver) - represents devices in a pool e.g. NVIDIA A100 with 40Gi memory on a node and is bound to the node
DeviceClass - define a category of devices e.g. high performing gpu from gpu.vendor.com
ResourceClaimTemplate - template for ResourceClaims
ResourceClaims - request devices in a DeviceClass e.g. from DeviceClass gpu.vendor.com, request a GPU with 40Gi memory


pod
DeviceClass
DRA Driver
centralize controller
node-local kubelet-plugin
ResourceSlice
ResourceClaim
ResourceClaimTemplate
kubelet
sched

Detailed Workflow (notes from workflow)

1. Cluster admin: install DRA drivers
    Includes node-local kubelet plugin in a DaemonSet and controller in Deployment,
    ResourceSlice that represents devices on the node

DRA driver updates the cluster with what devices are available to Pods on the node by publishing info on the ResourceSlice

2. Cluster admin: create DeviceClass (if not created by DRA driver)
3. Cluster admin: if not created by the DRA driver helm chart or manifest, create a ServiceAccount, ClusterRole, ClusterRoleBinding for the DRA driver node-local kubelet plugin

The ClusterRole should be able to get ResourceClaims, get Nodes, full lifeycle for ResourceSlice
The ClusteRoleBinding binds the ServiceAccount and ClusterRole

4. Cluster admin: if not created by the DRA driver helm chart or manifest, creates PriorityClass for the DRA driver to prevent preemption of the DRA driver
5. Workload operators: Creates ResourceClaimTemplates or ResourceClaims that request specific device configurations from a specific DeviceClass
6. Workload operators: Create a Pod that references that ResourceClaimTemplate or ResourceClaim
7. Cluster checks for new workloads that refer to ResourceClaimTemplates or ResourceClaims

If a ResourceClaimTemplate is used then a controller generates a ResourceClaim per-Pod

8. For every Pod, the cluster checks all ResourceSlices to find a device that satisfies that the node can access the resource and is eligible to run the Pod and the ResourceSlice has unallocated resources that match the requirements of the ResourceClaim
9. After finding an eligible ResourceSlice for a Pod's ResourceClaim, the Scheduler updates the ResourceClaim with allocation details
10. Scheduler places the Pod on a node that can access the allocated resource.
11. The device driver and the kubelet configure the device and the Pod's access to the device

Lab Overview

Module 1 - Introduction to Dynamic Resource Allocation

- DRA Overview
- Cluster Setup (kind on RHEL)

Module 2 - DRA Under the Covers - DRA Driver

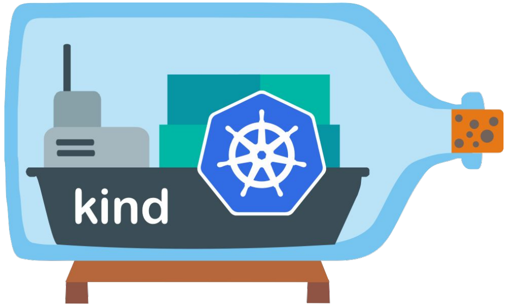
- DRA Resources
  - DRA Driver
  - ResourceSlice
  - DeviceClass
  - ResourceClaim
  - ResourceClaimTemplate

Module 3 - A Look into DRA Drivers

- NVIDIA DRA Driver
- Intel DRA Driver
- DRANET

Module 4 - Deploy a DRA Driver and Workloads

- Deploy a DeviceClass
- Create RBAC Authorization for the DRA Driver
- PriorityClass
- Deploy the DRA Driver
- ResourceSlice
- Ollama Pod Workloads
- Sharing a ResourceClaim
- Job with ResourceClaimTemplate



Thank you

Thank you to the following for their work on DRA:

- Patrick Ohly, Intel
- Kevin Klues, NVIDIA
- John Belamaric, Google
- et al.

Thank you to Red Hat for the VMs

Lab Access

Tutorial:
https://dra-tutorial.cloudnativeessentials.com
forwards to:
https://cloudnativeessentials.github.io/dra-tutorial/

VM Access:
https://dra-lab.cloudnativeessentials.com
forwards to:
https://catalog.demo.redhat.com/workshop/tetk9u

GitHub repo: https://github.com/cloudnativeessentials/dra-tutorial

Feedback

KubeCon | CloudNativeCon
North America 2025

Feedback QR Code

KubeCon | CloudNativeCon

North America 2025