KubeCon | CloudNativeCon

North America 2023

KubeCon | CloudNativeCon
North America 2023

Unlocking the Full Potential of GPUs for AI Workloads on Kubernetes

Kevin Klues, NVIDIA
kklues@nvidia.com

NVIDIA®

GPUs in Kubernetes Today

GPUs in Kubernetes Today

Enabling GPU support

Host-level Components

nvidia-container-toolkit

nvidia-gpu-driver

Kubernetes Components

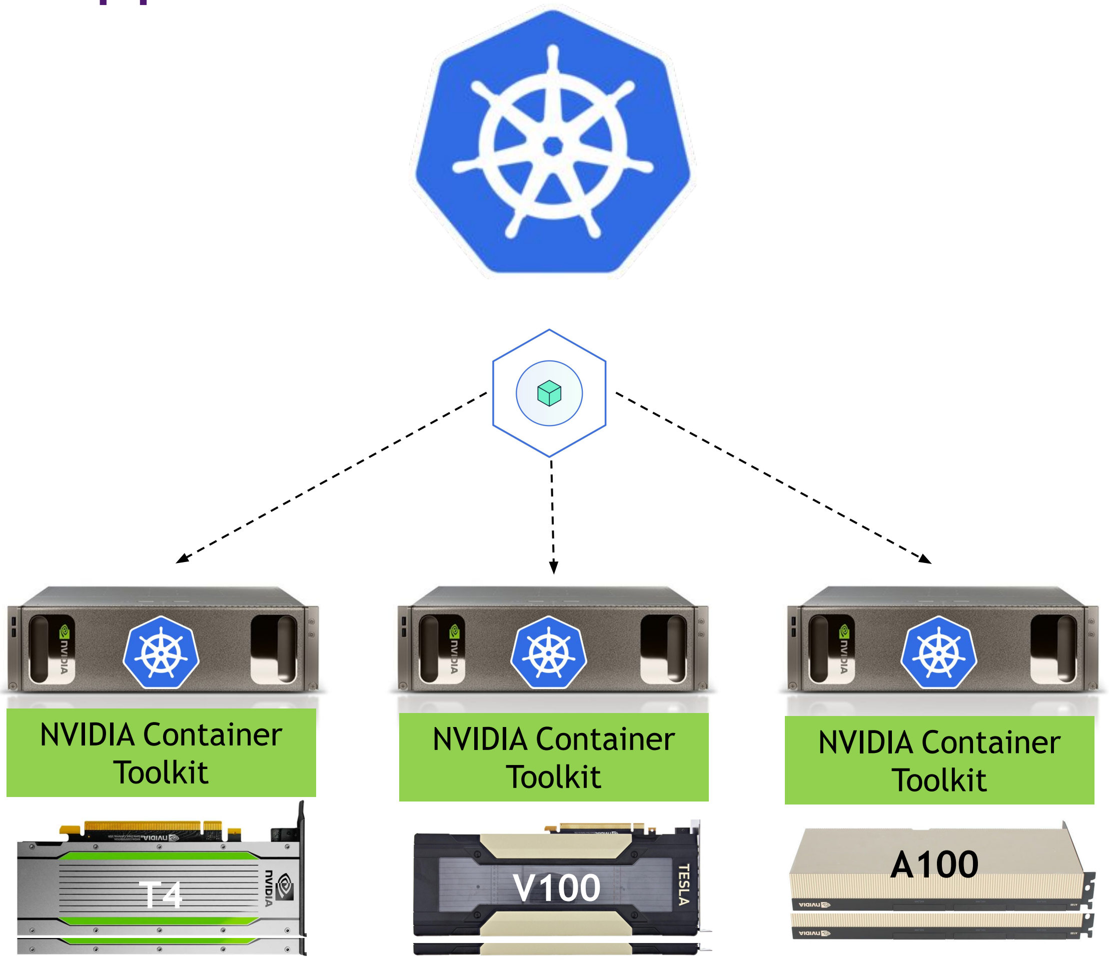
k8s-device-plugin

gpu-feature-discovery

nvidia-mig-manager

dcgm-exporter

...



GPUs in Kubernetes Today

Enabling GPU support

Host-level Components

nvidia-container-toolkit

nvidia-gpu-driver

Kubernetes Components

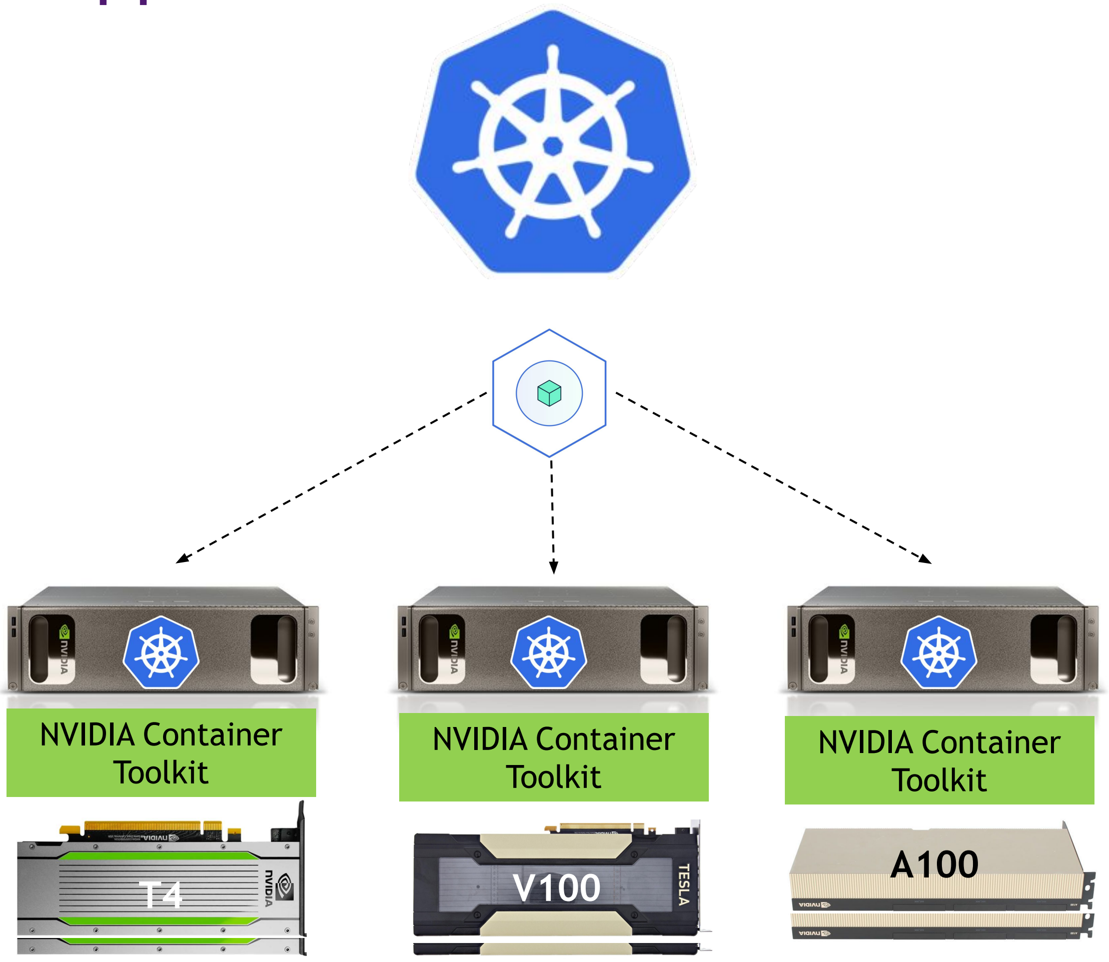
k8s-device-plugin

gpu-feature-discovery

nvidia-mig-manager

dcgm-exporter

...



GPUs in Kubernetes Today

Enabling GPU support

Host-level Components

nvidia-container-toolkit

nvidia-gpu-driver

Kubernetes Components

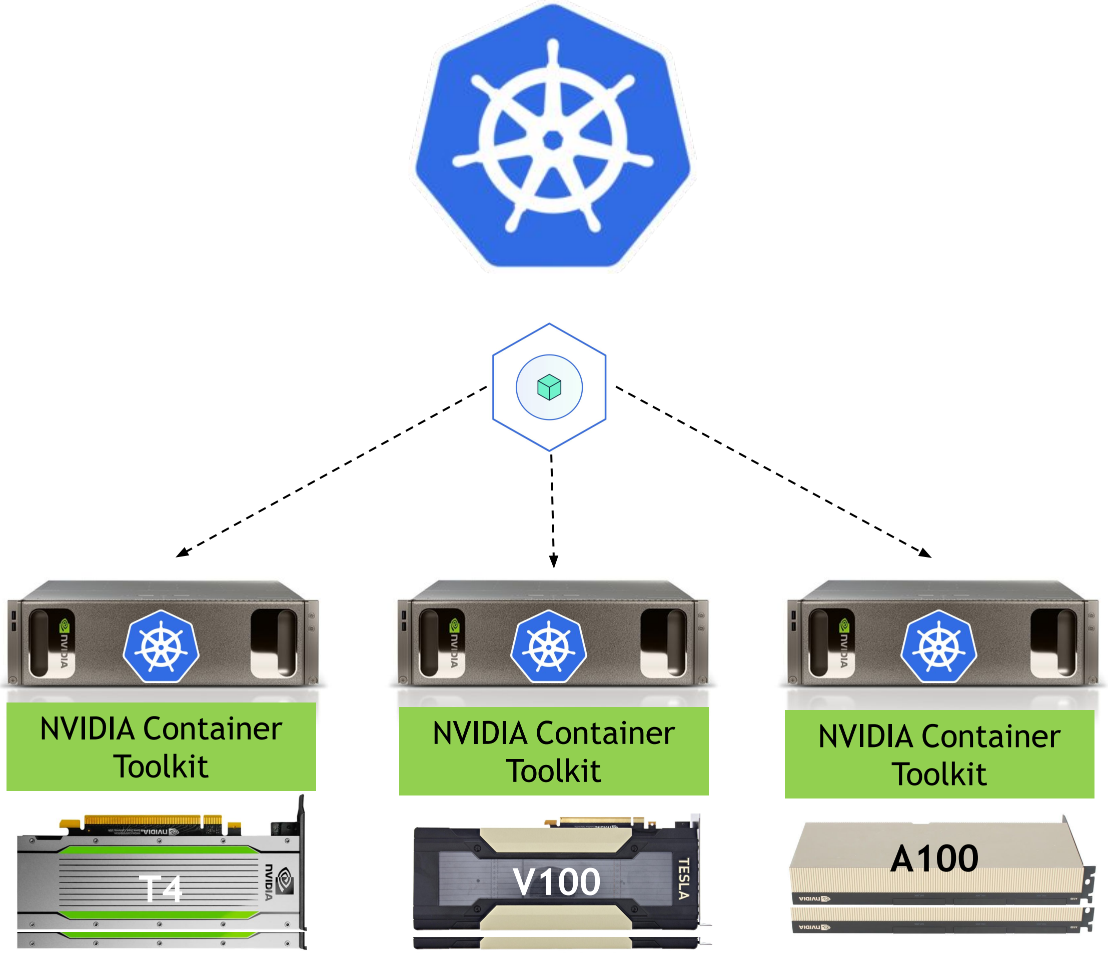
k8s-device-plugin

gpu-feature-discovery

nvidia-mig-manager

dcgm-exporter

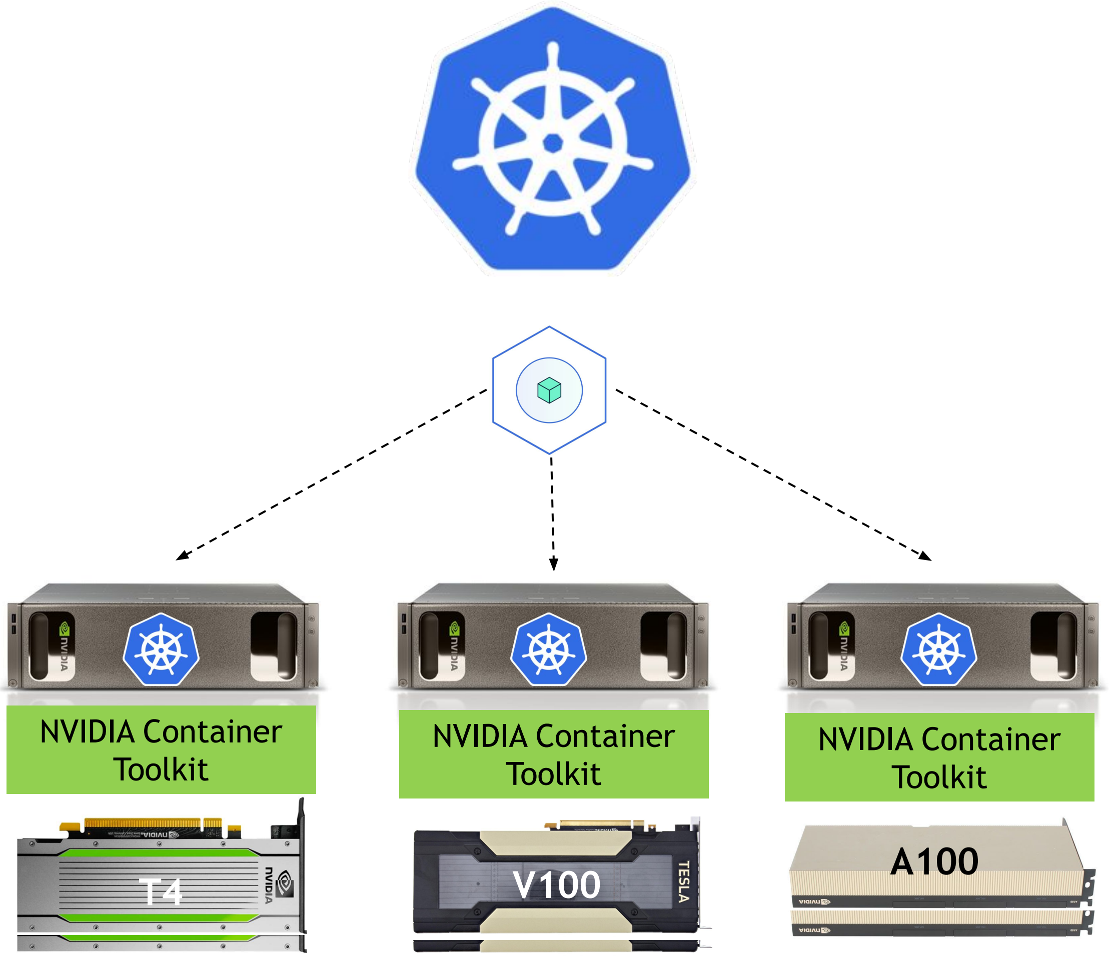
...



GPUs in Kubernetes Today

Requesting GPUs

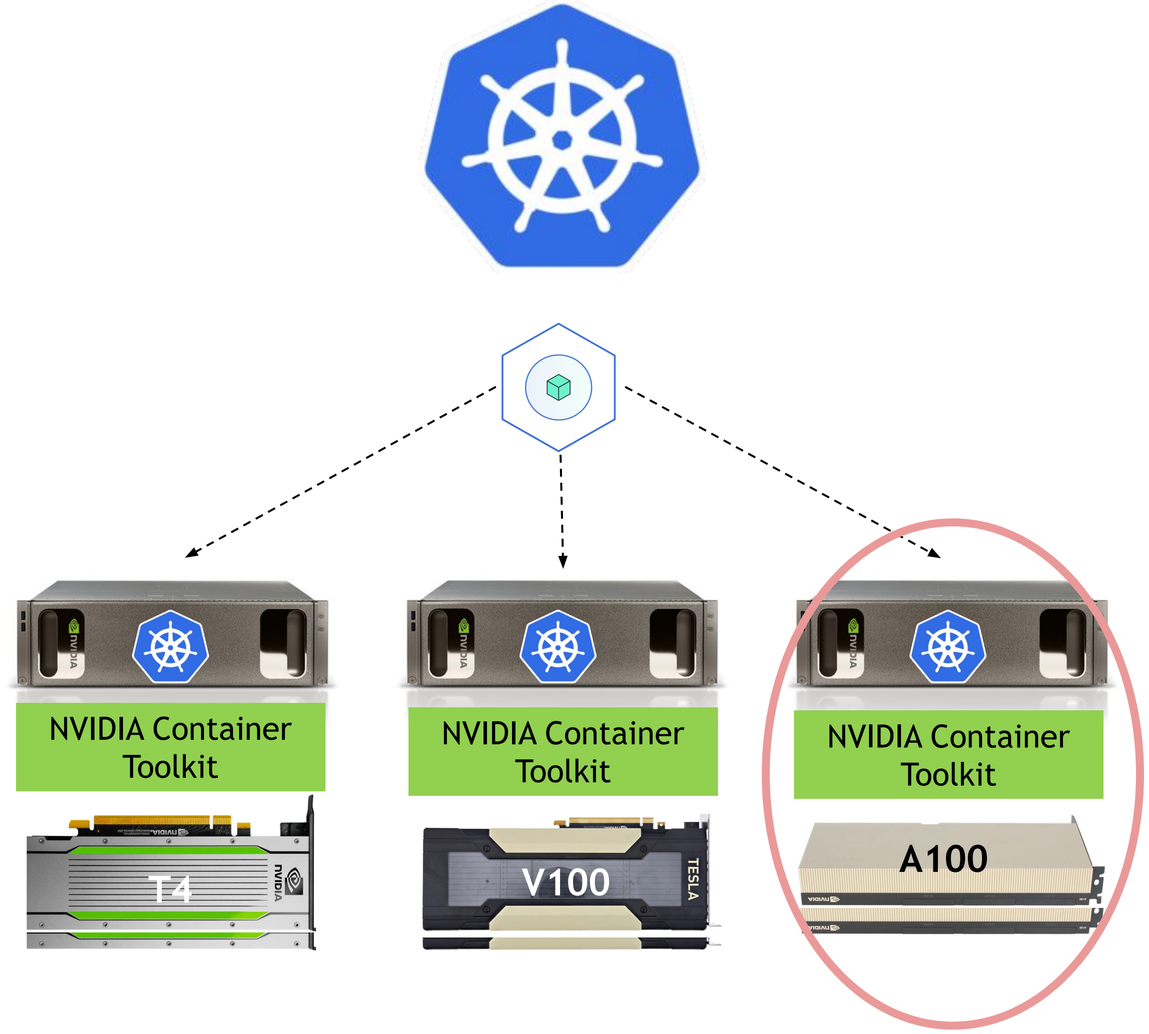
```yaml
apiVersion: v1
kind: Pod
metadata:
  name: gpu-example
spec:
  containers:
    - name: gpu-example
      image: nvidia/cuda
      resources:
        limits:
          nvidia.com/gpu: 2
```



GPUs in Kubernetes Today

Requesting GPUs on specific nodes

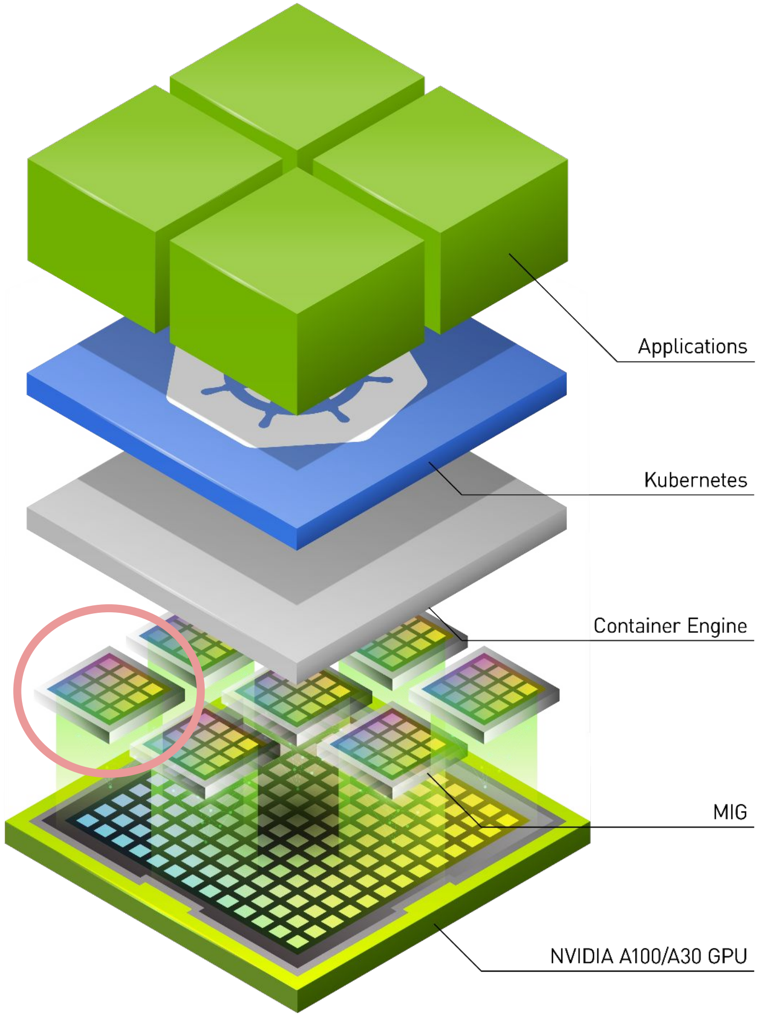
```
apiVersion: v1
kind: Pod
metadata:
  name: gpu-example
spec:
  containers:
    - name: gpu-example
      image: nvidia/cuda
      resources:
        limits:
          nvidia.com/gpu: 2
  nodeSelector:
    nvidia.com/gpu.product: A100-PCIE-40GB
    nvidia.com/cuda.runtime: 11.4
    nvidia.com/cuda.driver: 470.161.03
```



GPUs in Kubernetes Today

Requesting a fraction of a GPU

```
apiVersion: v1
kind: Pod
metadata:
  name: gpu-example
spec:
  containers:
    - name: gpu-example
      image: nvidia/cuda
      resources:
        limits:
          nvidia.com/mig-1g.5gb: 1
  nodeSelector:
    nvidia.com/gpu.product: A100-PCIE-40GB
    nvidia.com/cuda.runtime: 11.4
    nvidia.com/cuda.driver: 470.161.03
```



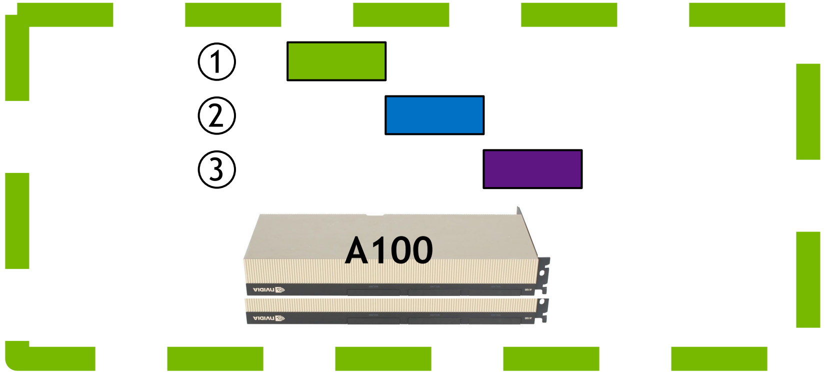
GPUs in Kubernetes Today

Requesting shared access to a GPU via time-slicing

```
apiVersion: v1
kind: Pod
metadata:
  name: gpu-example
spec:
  containers:
    - name: gpu-example
      image: nvidia/cuda
      resources:
        limits:
          nvidia.com/gpu.shared: 1
  nodeSelector:
    nvidia.com/gpu.product: A100-PCIE-40GB
    nvidia.com/cuda.runtime: 11.4
    nvidia.com/cuda.driver: 470.161.03
```

```
version: v1
sharing:
  timeSlicing:
    resources:
    - name: nvidia.com/gpu
      replicas: 10
    ...
```

k8s-device-plugin config file



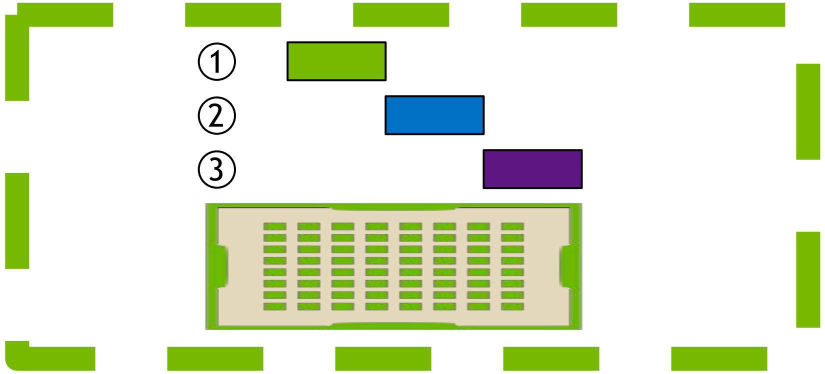
GPUs in Kubernetes Today

Requesting shared access to a fraction of a GPU via time-slicing

```
apiVersion: v1
kind: Pod
metadata:
  name: gpu-example
spec:
  containers:
    - name: gpu-example
      image: nvidia/cuda
      resources:
        limits:
          nvidia.com/mig-1g.5gb.shared: 1
  nodeSelector:
    nvidia.com/gpu.product: A100-PCIE-40GB
    nvidia.com/cuda.runtime: 11.4
    nvidia.com/cuda.driver: 470.161.03
```

k8s-device-plugin config file

```
version: v1
sharing:
  timeSlicing:
    resources:
    - name: nvidia.com/gpu
      replicas: 10
    - name: nvidia.com/mig-1g.5gb
      replicas: 10
    ...
```



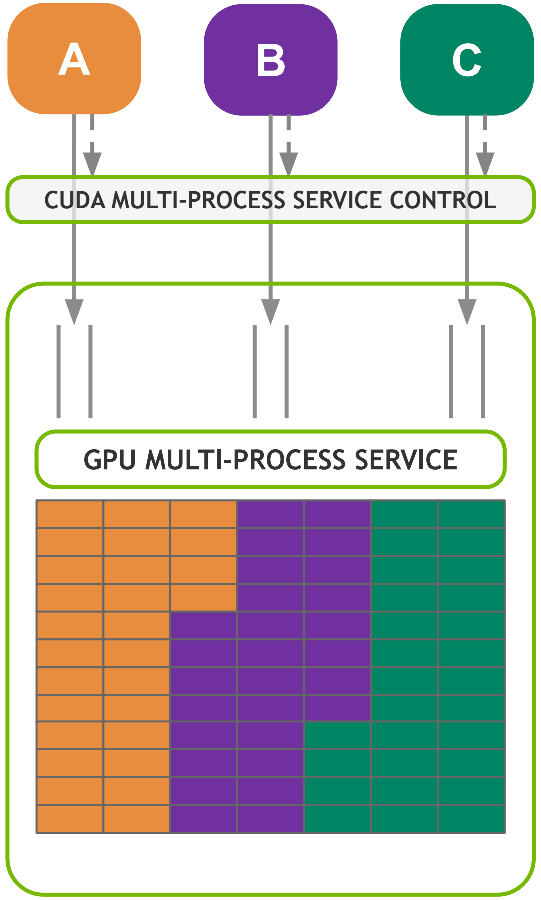
GPUs in Kubernetes Today

Requesting shared access to a GPU (or fraction of a GPU) via MPS

```
# Running directly on the host
$ nvidia-cuda-mps-control -d
```

```yaml
apiVersion: v1
kind: Pod
metadata:
  name: gpu-example
spec:
  hostIPC: true
  securityContext:
    runAsUser: 1000
  containers:
    - name: gpu-example
      image: nvidia/cuda
      resources:
        limits:
          nvidia.com/gpu: 1
```

```yaml
apiVersion: v1
kind: Pod
metadata:
  name: gpu-example
spec:
  hostIPC: true
  securityContext:
    runAsUser: 1000
  containers:
    - name: gpu-example
      image: nvidia/cuda
      resources:
        limits:
          nvidia.com/mig-1g.5gb: 1
```



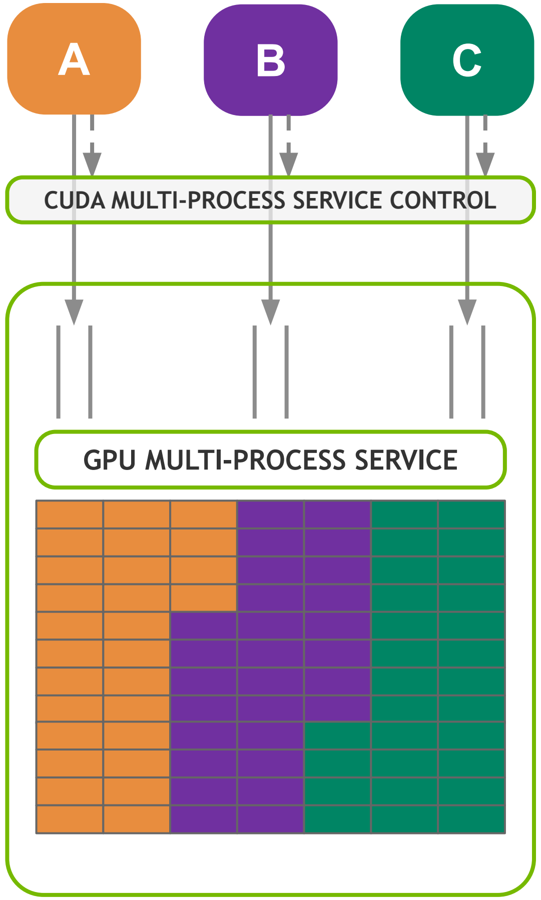
GPUs in Kubernetes Today

Requesting shared access to a GPU via MPS

```
# Running directly on the host
$ nvidia-cuda-mps-control -d
```

```
apiVersion: v1
kind: Pod
metadata:
  name: gpu-example
spec:
  hostIPC: true
  securityContext:
    runAsUser: 1000
  containers:
    - name: gpu-example
      image: nvidia/cuda
      resources:
        limits:
          nvidia.com/gpu: 1
```

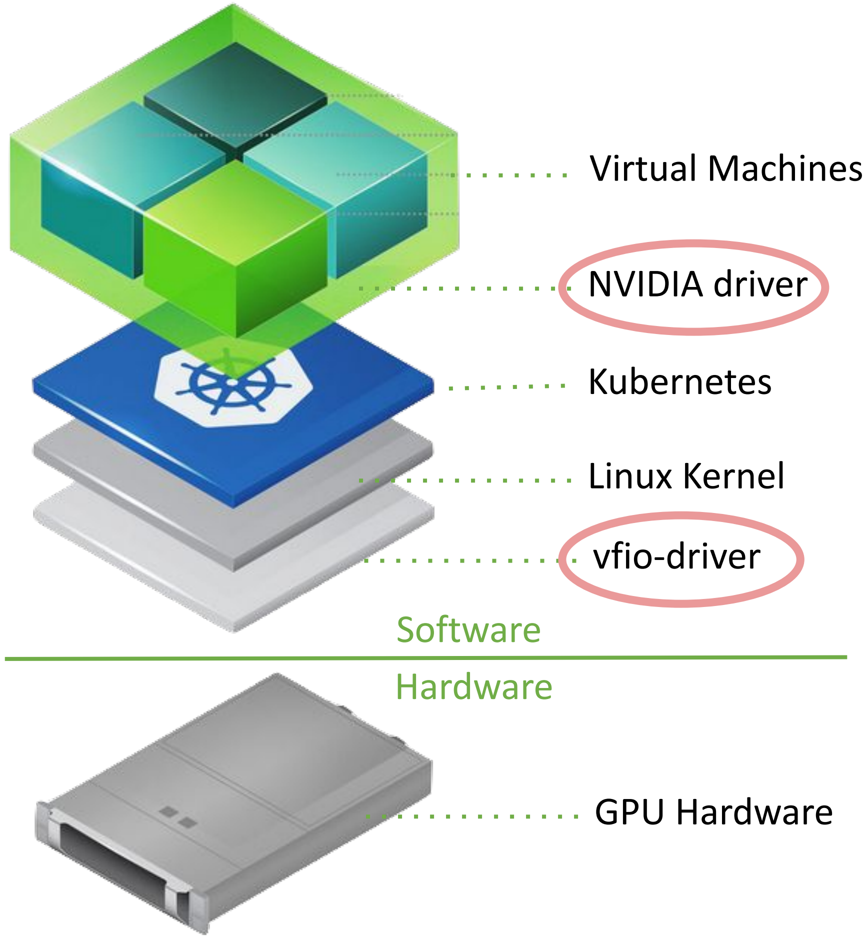
```
apiVersion: v1
kind: Pod
metadata:
  name: gpu-example
spec:
  hostIPC: true
  securityContext:
    runAsUser: 1000
  containers:
    - name: gpu-example
      image: nvidia/cuda
      resources:
        limits:
          nvidia.com/mig-1g.5gb: 1
```



GPUs in Kubernetes Today

Requesting access to a GPU for use in a VM

```
apiVersion: kubevirt.io/v1alpha3
kind: VirtualMachineInstance
metadata:
 name: vmi-gpu
spec:
 domain:
  devices:
   gpus:
   - deviceName: nvidia.com/GP102GL_Tesla_P40
     name: gpu1
```



Limitations

Limitations

- No support for having more than one GPU type per node

Limitations

- No support for having more than one GPU type per node
- No support for providing complex constraints when requesting a GPU

Limitations

- No support for having more than one GPU type per node
- No support for providing complex constraints when requesting a GPU
- No control over how oversubscribed GPUs are shared between jobs

Limitations

- No support for having more than one GPU type per node
- No support for providing complex constraints when requesting a GPU
- No control over how oversubscribed GPUs are shared between jobs
- Awkward, overly-burdensome support for MPS

Limitations

- No support for having more than one GPU type per node
- No support for providing complex constraints when requesting a GPU
- No control over how oversubscribed GPUs are shared between jobs
- Awkward, overly-burdensome support for MPS
- No ability to dynamic provision of MIG devices based on incoming requests

Limitations

- No support for having more than one GPU type per node
- No support for providing complex constraints when requesting a GPU
- No control over how oversubscribed GPUs are shared between jobs
- Awkward, overly-burdensome support for MPS
- No ability to dynamic provision of MIG devices based on incoming requests
- No ability to dynamically choose between NVIDIA and vfio drivers on a per-GPU basis

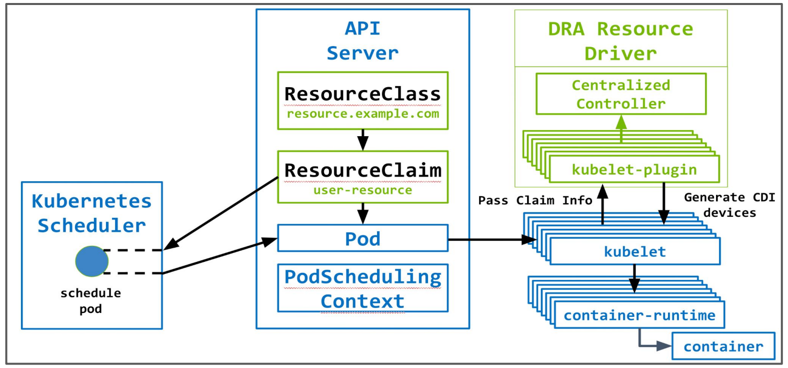
Limitations

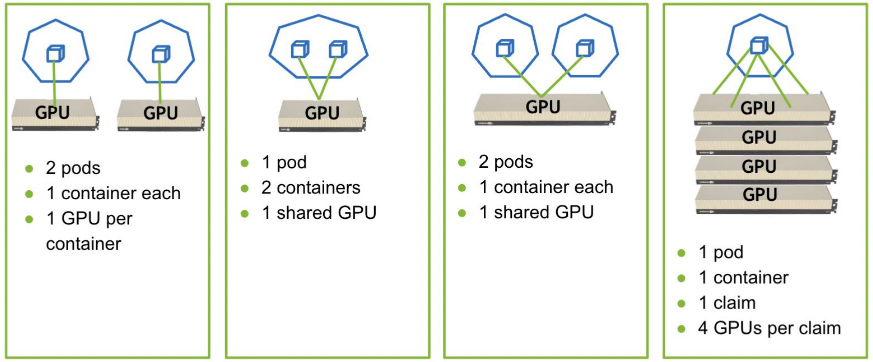
- No support for having more than one GPU type per node
- No support for providing complex constraints when requesting a GPU
- No control over how oversubscribed GPUs are shared between jobs
- Awkward, overly-burdensome support for MPS
- No ability to dynamic provision of MIG devices based on incoming requests
- No ability to dynamically choose between NVIDIA and vfio drivers on a per-GPU basis
- … the list goes on …

Limitations

- No support for having more than one GPU type per node

Dynamic Resource Allocation (DRA)





- 2 pods
- 1 container each
- 1 GPU per container

- 1 pod
- 2 containers
- 1 shared GPU

- 2 pods
- 1 container each
- 1 shared GPU

- 1 pod
- 1 container
- 1 claim
- 4 GPUs per claim

- … the list goes on …

Outline

- Overview of DRA
- Details of NVIDIA’s DRA resource driver for GPUs
- DEMO: Dynamic MIG with Time-slicing and MPS in Kind
- DEMO: Specifying complex constraints on GKE
- DEMO: Triton Inference server on GKE

Dynamic Resource Allocation

- New way of requesting resources available (as an ***alpha*** feature) in Kubernetes 1.26+
- Provides an ***alternative*** to the “count-based” interface of e.g. `nvidia.com/gpu:2`
- Puts full control of the API to request resources in the hands of 3rd-party developers
- Key concepts:

```
    ResourceClass (in-tree API) → ClassParameters (vendor-specific API)
    ResourceClaim (in-tree API) → ClaimParameters (vendor-specific API)
```

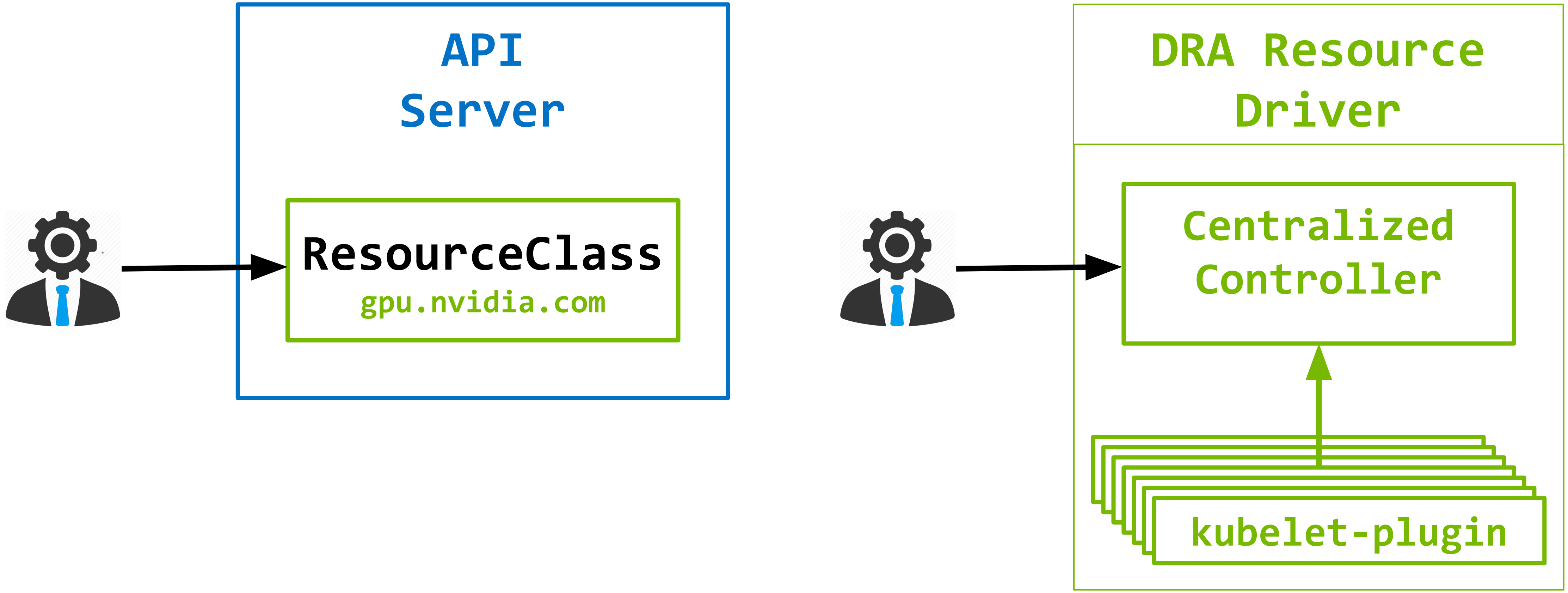
Dynamic Resource Allocation

- New way of requesting resources available (as an ***alpha*** feature) in Kubernetes 1.26+
- Provides an ***alternative*** to the “count-based” interface of e.g. `nvidia.com/gpu:2`
- Puts full control of the API to request resources in the hands of 3rd-party developers
- Key concepts:

```
ResourceClass (in-tree API) → ClassParameters (vendor-specific API)
ResourceClaim (in-tree API) → ClaimParameters (vendor-specific API)
```

- Kubecon EU 2023:
  Device Plugins 2.0: How to Build a Driver for Dynamic Resource Allocation

DRA Resource Driver for GPUs



DRA Resource Driver for GPUs

```yaml
apiVersion: v1
kind: Pod
metadata:
  name: gpu-example
spec:
  containers:
  - name: ctr
    image: nvidia/cuda
    command: ["nvidia-smi", "-L"]
    resources:
      limits:
        nvidia.com/gpu: 2
```

DRA Resource Driver for GPUs

```yaml
apiVersion: v1
kind: Pod
metadata:
  name: gpu-example
spec:
  containers:
  - name: ctr
    image: nvidia/cuda
    command: ["nvidia-smi", "-L"]
    resources:
      limits:
        nvidia.com/gpu: 2
```

```yaml
---
apiVersion: resource.k8s.io/v1alpha2
kind: ResourceClaimTemplate
metadata:
  name: unique-gpu
spec:
  spec:
    resourceClassName: gpu.nvidia.com

---
apiVersion: v1
kind: Pod
metadata:
  name: gpu-example
spec:
  containers:
  - name: ctr
    image: nvidia/cuda
    command: ["nvidia-smi" "-L"]
    resources:
      claims:
      - name: gpu0
      - name: gpu1
  resourceClaims:
  - name: gpu0
    source:
      resourceClaimTemplateName: unique-gpu
  - name: gpu1
    source:
      resourceClaimTemplateName: unique-gpu
```

DRA Resource Driver for GPUs

```
apiVersion: v1
kind: Pod
metadata:
  name: gpu-example
spec:
  containers:
  - name: ctr
    image: nvidia/cuda
    command: ["nvidia-smi", "-L"]
    resources:
      limits:
        nvidia.com/gpu: 2
```

```
---
apiVersion: resource.k8s.io/v1alpha2
kind: ResourceClaimTemplate
metadata:
  name: unique-gpu
spec:
  spec:
    resourceClassName: gpu.nvidia.com

---
apiVersion: v1
kind: Pod
metadata:
  name: gpu-example
spec:
  containers:
  - name: ctr
    image: nvidia/cuda
    command: ["nvidia-smi" "-L"]
    resources:
      claims:
      - name: gpu0
      - name: gpu1
  resourceClaims:
  - name: gpu0
    source:
      resourceClaimTemplateName: unique-gpu
  - name: gpu1
    source:
      resourceClaimTemplateName: unique-gpu
```

Associated with the DRA Driver and installed by the cluster admin

DRA Resource Driver for GPUs

```yaml
apiVersion: v1
kind: Pod
metadata:
  name: gpu-example
spec:
  containers:
  - name: ctr
    image: nvidia/cuda
    command: ["nvidia-smi", "-L"]
    resources:
      limits:
        nvidia.com/gpu: 2
```

```yaml
---
apiVersion: resource.k8s.io/v1alpha2
kind: ResourceClaimTemplate
metadata:
  name: unique-gpu
spec:
  spec:
    resourceClassName: gpu.nvidia.com

---
apiVersion: v1
kind: Pod
metadata:
  name: gpu-example
spec:
  containers:
  - name: ctr
    image: nvidia/cuda
    command: ["nvidia-smi" "-L"]
    resources:
      claims:
      - name: gpu0
      - name: gpu1
  resourceClaims:
  - name: gpu0
    source:
      resourceClaimTemplateName: unique-gpu
  - name: gpu1
    source:
      resourceClaimTemplateName: unique-gpu
```

DRA Resource Driver for GPUs

```
apiVersion: v1
kind: Pod
metadata:
  name: gpu-example
spec:
  containers:
  - name: ctr
    image: nvidia/cuda
    command: ["nvidia-smi", "-L"]
    resources:
      limits:
        nvidia.com/gpu: 2
```

```
---
apiVersion: resource.k8s.io/v1alpha2
kind: ResourceClaimTemplate
metadata:
  name: unique-gpu
spec:
  spec:
    resourceClassName: gpu.nvidia.com

---
apiVersion: v1
kind: Pod
metadata:
  name: gpu-example
spec:
  containers:
  - name: ctr
    image: nvidia/cuda
    command: ["nvidia-smi" "-L"]
    resources:
      claims:
      - name: gpu0
      - name: gpu1
  resourceClaims:
  - name: gpu0
    source:
      resourceClaimTemplateName: unique-gpu
  - name: gpu1
    source:
      resourceClaimTemplateName: unique-gpu
```

DRA Resource Driver for GPUs

```yaml
apiVersion: v1
kind: Pod
metadata:
  name: gpu-example
spec:
  containers:
  - name: ctr
    image: nvidia/cuda
    command: ["nvidia-smi", "-L"]
    resources:
      limits:
        nvidia.com/gpu: 2
```

```yaml
---
apiVersion: resource.k8s.io/v1alpha2
kind: ResourceClaimTemplate
metadata:
  name: unique-gpu
spec:
  spec:
    resourceClassName: gpu.nvidia.com

---
apiVersion: v1
kind: Pod
metadata:
  name: gpu-example
spec:
  containers:
  - name: ctr
    image: nvidia/cuda
    command: ["nvidia-smi" "-L"]
    resources:
      claims:
      - name: gpu0
      - name: gpu1
  resourceClaims:
  - name: gpu0
    source:
      resourceClaimTemplateName: unique-gpu
  - name: gpu1
    source:
      resourceClaimTemplateName: unique-gpu
```

DRA Resource Driver for GPUs

DRA Resource Driver for GPUs

```
---
apiVersion: resource.k8s.io/v1alpha2
kind: ResourceClaimTemplate
metadata:
  name: unique-gpu
spec:
  spec:
    resourceClassName: gpu.nvidia.com
```

```
apiVersion: v1
kind: Pod
metadata:
  name: gpu-example
spec:
  containers:
    - name: ctr0
      resources:
        claims:
        - name: gpu
    - name: ctr1
      resources:
        claims:
        - name: gpu
resourceClaims:
  - name: gpu
    source:
      resourceClaimName: unique-gpu
```

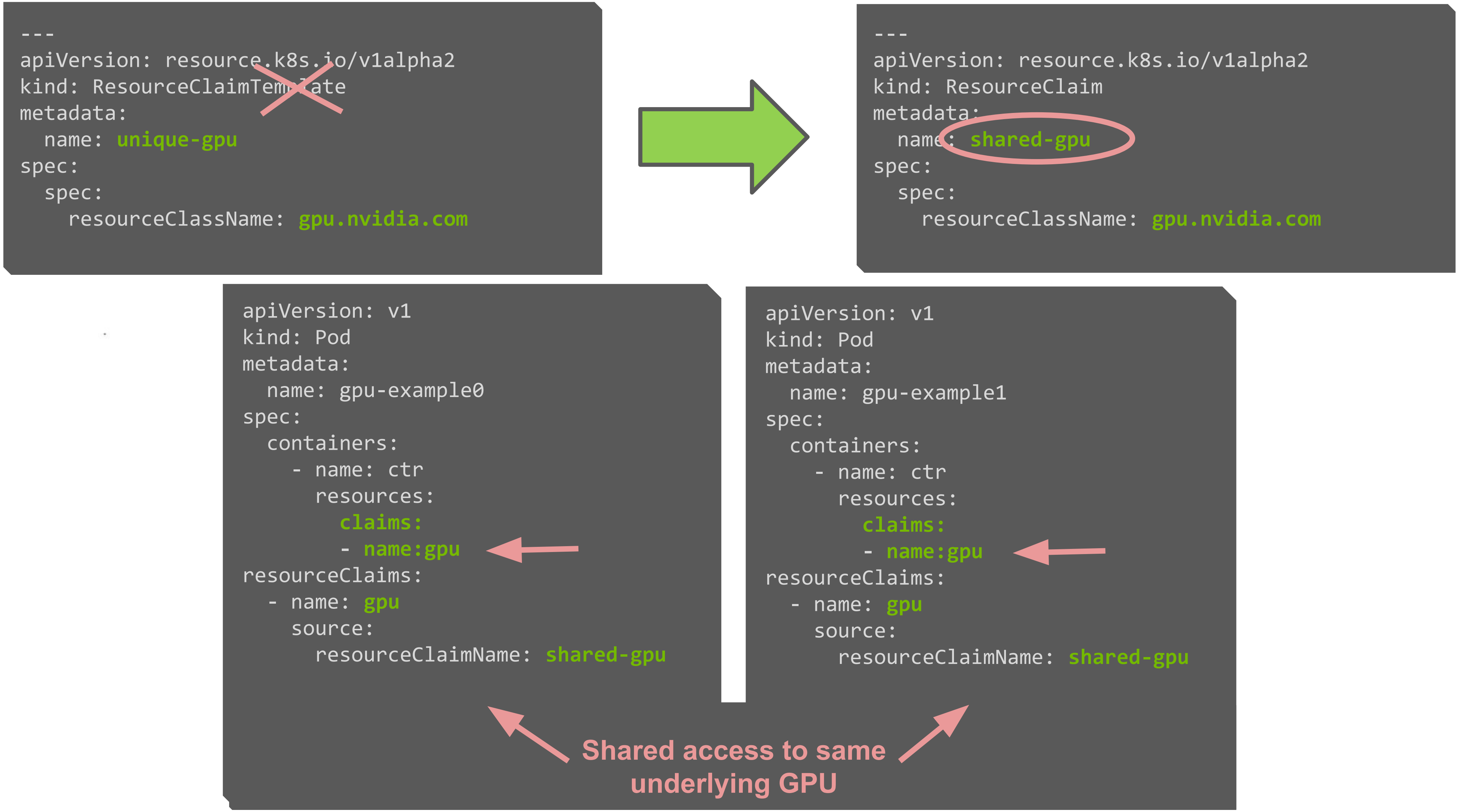
Shared access
to same
underlying GPU

DRA Resource Driver for GPUs

```
---
apiVersion: resource.k8s.io/v1alpha2
kind: ResourceClaim~~Template~~
metadata:
  name: unique-gpu
spec:
  spec:
    resourceClassName: gpu.nvidia.com
```

```
---
apiVersion: resource.k8s.io/v1alpha2
kind: ResourceClaim
metadata:
  name: shared-gpu
spec:
  spec:
    resourceClassName: gpu.nvidia.com
```

DRA Resource Driver for GPUs



```
---
apiVersion: resource.k8s.io/v1alpha2
kind: ResourceClaim~~Template~~
metadata:
  name: unique-gpu
spec:
  spec:
    resourceClassName: gpu.nvidia.com
```

```
---
apiVersion: resource.k8s.io/v1alpha2
kind: ResourceClaim
metadata:
  name: shared-gpu
spec:
  spec:
    resourceClassName: gpu.nvidia.com
```

```
apiVersion: v1
kind: Pod
metadata:
  name: gpu-example0
spec:
  containers:
    - name: ctr
      resources:
        claims:
        - name:gpu
resourceClaims:
  - name: gpu
    source:
      resourceClaimName: shared-gpu
```

```
apiVersion: v1
kind: Pod
metadata:
  name: gpu-example1
spec:
  containers:
    - name: ctr
      resources:
        claims:
        - name:gpu
resourceClaims:
  - name: gpu
    source:
      resourceClaimName: shared-gpu
```

Shared access to same underlying GPU

DRA Resource Driver for GPUs

- No support for having more than one GPU type per node
- No support for providing complex constraints when requesting a GPU
- **No control over how oversubscribed GPUs are shared between jobs**
- Awkward, overly-burdensome support for MPS
- No ability to dynamic provision of MIG devices based on incoming requests
- No ability to dynamically choose between NVIDIA and vfio drivers on a per-GPU basis
- … the list goes on …

ClassParameters and ClaimParameters

`ResourceClass (in-tree API) → ClassParameters (vendor-specific API)`

`ResourceClaim (in-tree API) → ClaimParameters (vendor-specific API)`

ClassParameters and ClaimParameters

`ResourceClass (in-tree API) → ClassParameters (vendor-specific API)`

`ResourceClaim (in-tree API) → ClaimParameters (vendor-specific API)`

ClassParameters and ClaimParameters

ClassParameters and ClaimParameters

```
apiVersion: resource.k8s.io/v1alpha2
kind: ResourceClaimTemplate
metadata:
  name: unique-gpu
spec:
  resourceClassName: gpu.nvidia.com
```

```
apiVersion: resource.k8s.io/v1alpha2
kind: ResourceClaim
metadata:
  name: shared-gpu
spec:
  resourceClassName: gpu.nvidia.com
```

ClassParameters and ClaimParameters

```
apiVersion: resource.k8s.io/v1alpha2
kind: ResourceClaimTemplate
metadata:
  name: unique-gpu
spec:
  resourceClassName: gpu.nvidia.com
  parametersRef:
    apiGroup: <api-group>
    kind: <claim-parameters-kind>
    name: <claim-parameters-name>
```

```
apiVersion: resource.k8s.io/v1alpha2
kind: ResourceClaim
metadata:
  name: shared-gpu
spec:
  resourceClassName: gpu.nvidia.com
  parametersRef:
    apiGroup: <api-group>
    kind: <claim-parameters-kind>
    name: <claim-parameters-name>
```

ClassParameters and ClaimParameters

```
apiVersion: resource.k8s.io/v1alpha2
kind: ResourceClaimTemplate
metadata:
  name: unique-gpu
spec:
  resourceClassName: gpu.nvidia.com
  parametersRef:
    apiGroup: gpu.resource.nvidia.com
    kind: GpuClaimParameters
    name: single-gpu
```

```
apiVersion: resource.k8s.io/v1alpha2
kind: ResourceClaim
metadata:
  name: shared-gpu
spec:
  resourceClassName: gpu.nvidia.com
  parametersRef:
    apiGroup: gpu.resource.nvidia.com
    kind: GpuClaimParameters
    name: single-gpu
```

```
apiVersion: gpu.resource.nvidia.com/v1alpha1
kind: GpuClaimParameters
metadata:
  name: single-gpu
spec:
  count: 1
```

ClassParameters and ClaimParameters

```yaml
---
apiVersion: v1
kind: Pod
metadata:
  name: gpu-example
spec:
  containers:
  - name: ctr
    image: nvidia/cuda
    command: ["nvidia-smi" "-L"]
    resources:
      claims:
      - name: gpus
  resourceClaims:
  - name: gpus
    source:
      resourceClaimTemplateName: two-unique-gpu
```

```yaml
---
apiVersion: resource.k8s.io/v1alpha2
kind: ResourceClaimTemplate
metadata:
  name: two-unique-gpus
spec:
  spec:
    resourceClassName: gpu.nvidia.com
    parametersRef:
      apiGroup: gpu.resource.nvidia.com
      kind: GpuClaimParameters
      name: two-gpus

---
apiVersion: gpu.resource.nvidia.com/v1alpha1
kind: GpuClaimParameters
metadata:
  name: two-gpus
spec:
  count: 2
```

ClassParameters and ClaimParameters

- No support for having more than one GPU type per node
- No support for providing complex constraints when requesting a GPU
- ~~No control over how oversubscribed GPUs are shared between jobs~~
- Awkward, overly-burdensome support for MPS
- No ability to dynamic provision of MIG devices based on incoming requests
- No ability to dynamically choose between NVIDIA and vfio drivers on a per-GPU basis

```yaml
apiVersion: gpu.resource.nvidia.com/v1alpha1
kind: GpuClaimParameters
metadata:
  name: single-gpu
spec:
  count: 1
```

ClassParameters and ClaimParameters

- No support for having more than one GPU type per node
- No support for providing complex constraints when requesting a GPU
- ~~No control over how oversubscribed GPUs are shared between jobs~~
- Awkward, overly-burdensome support for MPS
- No ability to dynamic provision of MIG devices based on incoming requests
- No ability to dynamically choose between NVIDIA and vfio drivers on a per-GPU basis

```yaml
apiVersion: gpu.resource.nvidia.com/v1alpha1
kind: GpuClaimParameters
metadata:
  name: single-gpu
spec:
  count: 1
```

ClassParameters and ClaimParameters

- No support for having more than one GPU type per node
- No support for providing complex constraints when requesting a GPU
- ~~No control over how oversubscribed GPUs are shared between jobs~~
- Awkward, overly-burdensome support for MPS
- No ability to dynamic provision of MIG devices based on incoming requests
- No ability to dynamically choose between NVIDIA and vfio drivers on a per-GPU basis

```
apiVersion: gpu.resource.nvidia.com/v1alpha1
kind: GpuClaimParameters
metadata:
  name: single-gpu
spec:
  count: 1
  sharing:
    strategy: MPS
```

ClassParameters and ClaimParameters

- No support for having more than one GPU type per node
- No support for providing complex constraints when requesting a GPU
- ~~No control over how oversubscribed GPUs are shared between jobs~~
- Awkward, overly-burdensome support for MPS
- No ability to dynamic provision of MIG devices based on incoming requests
- No ability to dynamically choose between NVIDIA and vfio drivers on a per-GPU basis

```yaml
apiVersion: gpu.resource.nvidia.com/v1alpha1
kind: GpuClaimParameters
metadata:
  name: single-gpu
spec:
  count: 1
  sharing:
    strategy: MPS
    mpsConfig:
      maxConnections: <int>
      activeThreadPercentage: <int>
      pinnedDeviceMemoryLimit: <quantity>
```

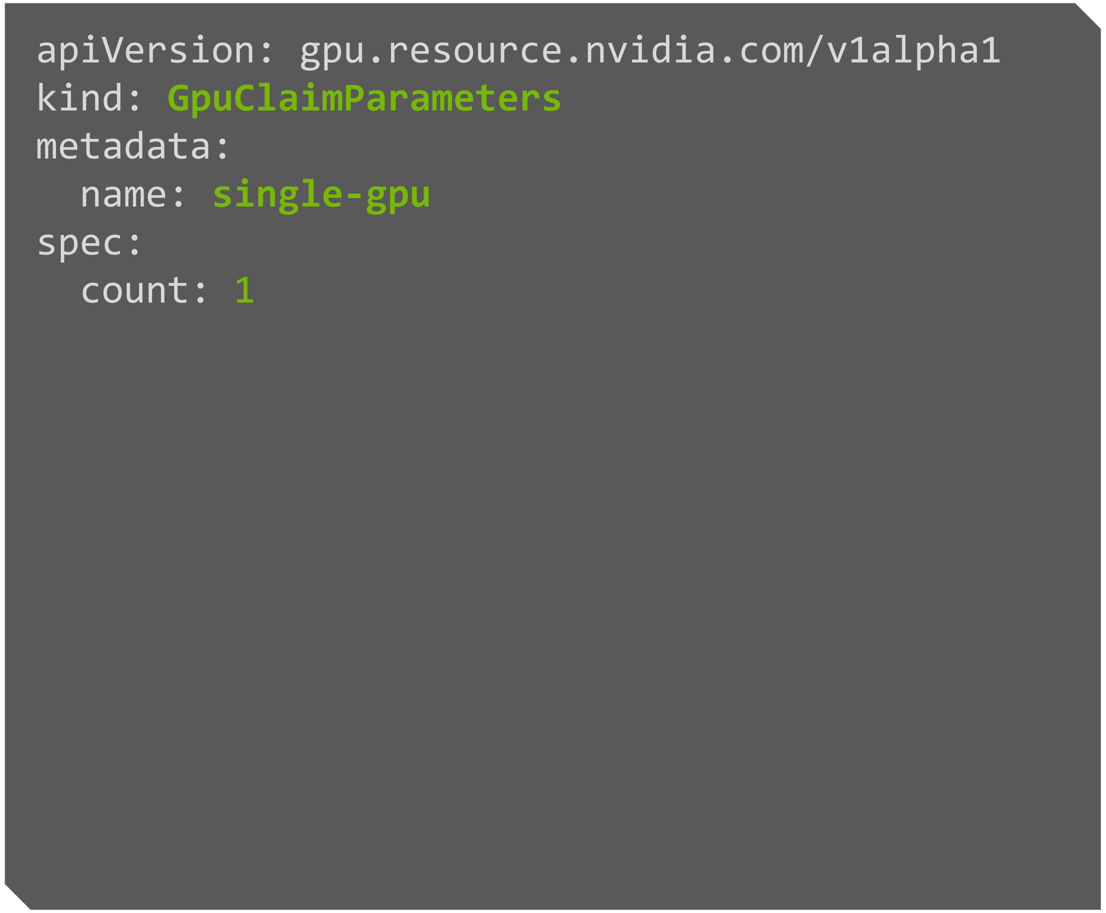
ClassParameters and ClaimParameters

- No support for having more than one GPU type per node
- No support for providing complex constraints when requesting a GPU
- ~~No control over how oversubscribed GPUs are shared between jobs~~
- Awkward, overly-burdensome support for MPS
- No ability to dynamic provision of MIG devices based on incoming requests
- No ability to dynamically choose between NVIDIA and vfio drivers on a per-GPU basis

```yaml
apiVersion: gpu.resource.nvidia.com/v1alpha1
kind: GpuClaimParameters
metadata:
  name: single-gpu
spec:
  count: 1
  sharing:
    strategy: TimeSlicing
    timeSlicingConfig:
      timeSlice: <Default|Short|Medium|Long>
```

ClassParameters and ClaimParameters

- No support for having more than one GPU type per node
- No support for providing complex constraints when requesting a GPU
- ~~No control over how oversubscribed GPUs are shared between jobs~~
- ~~Awkward, overly-burdensome support for MPS~~
- No ability to dynamic provision of MIG devices based on incoming requests
- No ability to dynamically choose between NVIDIA and vfio drivers on a per-GPU basis



```yaml
apiVersion: gpu.resource.nvidia.com/v1alpha1
kind: GpuClaimParameters
metadata:
  name: single-gpu
spec:
  count: 1
```

ClassParameters and ClaimParameters

- No support for having more than one GPU type per node
- No support for providing complex constraints when requesting a GPU
- ~~No control over how oversubscribed GPUs are shared between jobs~~
- ~~Awkward, overly-burdensome support for MPS~~
- No ability to dynamic provision of MIG devices based on incoming requests
- No ability to dynamically choose between NVIDIA and vfio drivers on a per-GPU basis

```
apiVersion: gpu.resource.nvidia.com/v1alpha1
kind: GpuClaimParameters
metadata:
  name: single-gpu
spec:
  count: 1
```

```
type Selector[T any] struct {
    Properties    *T
    AndExpression []Selector[T]
    OrExpression  []Selector[T]
}
```

ClassParameters and ClaimParameters

- No support for having more than one GPU type per node
- No support for providing complex constraints when requesting a GPU
- ~~No control over how oversubscribed GPUs are shared between jobs~~
- ~~Awkward, overly-burdensome support for MPS~~
- No ability to dynamic provision of MIG devices based on incoming requests
- No ability to dynamically choose between NVIDIA and vfio drivers on a per-GPU basis

```
apiVersion: gpu.resource.nvidia.com/v1alpha1
kind: GpuClaimParameters
metadata:
  name: single-gpu
spec:
  count: 1
  selector:
    orExpression:
    - productName: "*t4*"
    - andExpression:
      - productName: "*v100*"
      - memory:
          value: 16G
          operator: LessThanOrEqualTo
```

ClassParameters and ClaimParameters

- ~~No support for having more than one GPU type per node~~
- ~~No support for providing complex constraints when requesting a GPU~~
- ~~No control over how oversubscribed GPUs are shared between jobs~~
- ~~Awkward, overly burdensome support for MPS~~
- No ability to dynamic provision of MIG devices based on incoming requests
- No ability to dynamically choose between NVIDIA and vfio drivers on a per-GPU basis

```yaml
apiVersion: gpu.resource.nvidia.com/v1alpha1
kind: GpuClaimParameters
metadata:
  name: single-gpu
spec:
  count: 1
```

ClassParameters and ClaimParameters

- ~~No support for having more than one GPU type per node~~
- ~~No support for providing complex constraints when requesting a GPU~~
- ~~No control over how oversubscribed GPUs are shared between jobs~~
- ~~Awkward, overly burdensome support for MPS~~
- No ability to dynamic provision of MIG devices based on incoming requests
- No ability to dynamically choose between NVIDIA and vfio drivers on a per-GPU basis

```yaml
apiVersion: gpu.resource.nvidia.com/v1alpha1
kind: MigClaimParameters
metadata:
  name: mig-1g.5gb
spec:
  profile: 1g.5gb
```

ClassParameters and ClaimParameters

- ~~No support for having more than one GPU type per node~~
- ~~No support for providing complex constraints when requesting a GPU~~
- ~~No control over how oversubscribed GPUs are shared between jobs~~
- ~~Awkward, overly burdensome support for MPS~~
- No ability to dynamic provision of MIG devices based on incoming requests
- No ability to dynamically choose between NVIDIA and vfio drivers on a per-GPU basis

```yaml
apiVersion: gpu.resource.nvidia.com/v1alpha1
kind: MigClaimParameters
metadata:
  name: mig-1g.5gb
spec:
  profile: 1g.5gb
```

```yaml
apiVersion: gpu.resource.nvidia.com/v1alpha1
kind: VfioGpuClaimParameters
metadata:
  name: vm-gpu
spec:
  selector: …
```

ClassParameters and ClaimParameters

- ~~No support for having more than one GPU type per node~~

- ~~No support for providing complex constraints when requesting a GPU~~

- ~~No control over how oversubscribed GPUs are shared between jobs~~

- ~~Awkward, overly burdensome support for MPS~~

- No ability to dynamic provision of MIG devices based on incoming requests

- No ability to dynamically choose between NVIDIA and vfio drivers on a per-GPU basis

```yaml
apiVersion: gpu.resource.nvidia.com/v1alpha1
kind: MigClaimParameters
metadata:
  name: mig-1g.5gb
spec:
  profile: 1g.5gb
```

```yaml
apiVersion: gpu.resource.nvidia.com/v1alpha1
kind: VfioGpuClaimParameters
metadata:
  name: vm-gpu
spec:
  selector: …
```

ClassParameters and ClaimParameters

- ~~No support for having more than one GPU type per node~~
- ~~No support for providing complex constraints when requesting a GPU~~
- ~~No control over how oversubscribed GPUs are shared between jobs~~
- ~~Awkward, overly burdensome support for MPS~~
- No ability to dynamic provision of MIG devices based on incoming requests
- No ability to dynamically choose between NVIDIA and vfio drivers on a per-GPU basis

```
apiVersion: gpu.resource.nvidia.com/v1alpha1
kind: MigClaimParameters
metadata:
  name: mig-1g.5gb
spec:
  profile: 1g.5gb
```

```
apiVersion: gpu.resource.nvidia.com/v1alpha1
kind: VfioGpuClaimParameters
metadata:
  name: vm-gpu
spec:
  selector: …
```

ClassParameters and ClaimParameters

- ~~No support for having more than one GPU type per node~~
- ~~No support for providing complex constraints when requesting a GPU~~
- ~~No control over how oversubscribed GPUs are shared between jobs~~
- ~~Awkward, overly-burdensome support for MPS~~
- ~~No ability to dynamic provision of MIG devices based on incoming requests~~
- ~~No ability to dynamically choose between NVIDIA and vfio drivers on a per-GPU basis~~

```
apiVersion: gpu.resource.nvidia.com/v1alpha1
kind: MigClaimParameters
metadata:
  name: mig-1g.5gb
spec:
  profile: 1g.5gb
```

```
apiVersion: gpu.resource.nvidia.com/v1alpha1
kind: VfioGpuClaimParameters
metadata:
  name: vm-gpu
spec:
  selector: …
```

ClassParameters and ClaimParameters

- ~~No support for having more than one GPU type per node~~
- ~~No support for providing complex constraints when requesting a GPU~~
- ~~No control over how oversubscribed GPUs are shared between jobs~~
- ~~Awkward, overly-burdensome support for MPS~~
- ~~No ability to dynamic provision of MIG devices based on incoming requests~~
- No ability to dynamically choose between NVIDIA and vfio drivers on a per-GPU basis

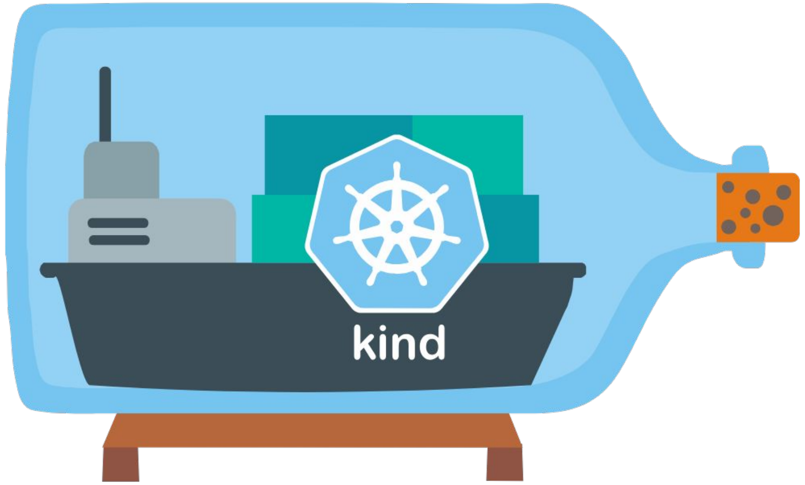
```
apiVersion: gpu.resource.nvidia.com/v1alpha1
kind: MigClaimParameters
metadata:
  name: mig-1g.5gb
spec:
  profile: 1g.5gb
```

```
apiVersion: gpu.resource.nvidia.com/v1alpha1
kind: VfioGpuClaimParameters
metadata:
  name: vm-gpu
spec:
  selector: ...
```

Not yet available

Resources

- DRA resource driver for GPUs
  - https://github.com/NVIDIA/k8s-dra-driver



DEMO Resources

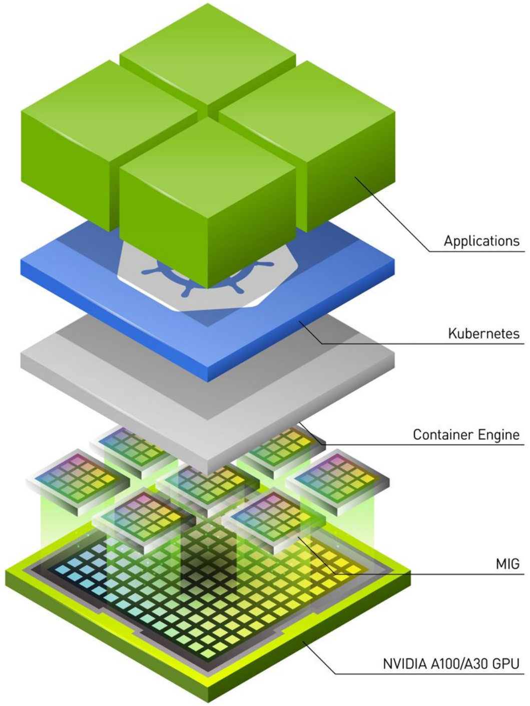
- DEMO: Dynamic MIG with Time-slicing and MPS in Kind
  - https://github.com/NVIDIA/k8s-dra-driver/demo/sharing
- DEMO: GPU selectors on GKE
  - https://github.com/NVIDIA/k8s-dra-driver/demo/gke
- DEMO: Triton Inference server on GKE
  - https://github.com/NVIDIA/k8s-dra-driver/demo/gke/tms

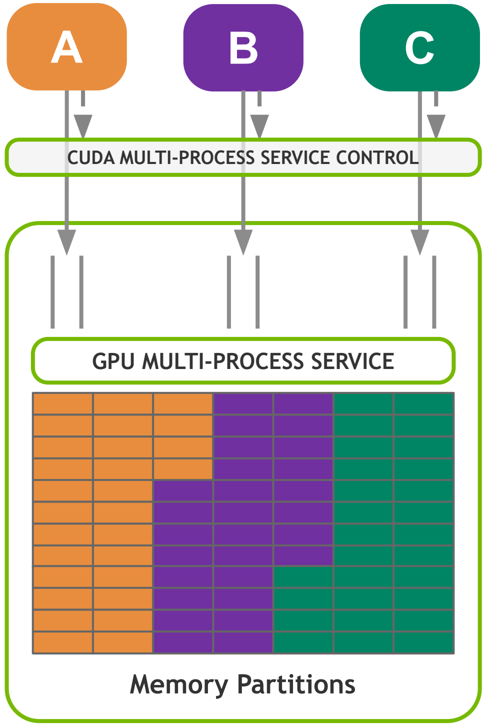
Dynamic MIG with Time-slicing and MPS

Physical Partitioning + Logical Partitioning

Dynamically partition a GPU into smaller GPUs (i.e. MIG Devices)

Provide shared access to a MIG Device (with additional memory partitioning) via MPS



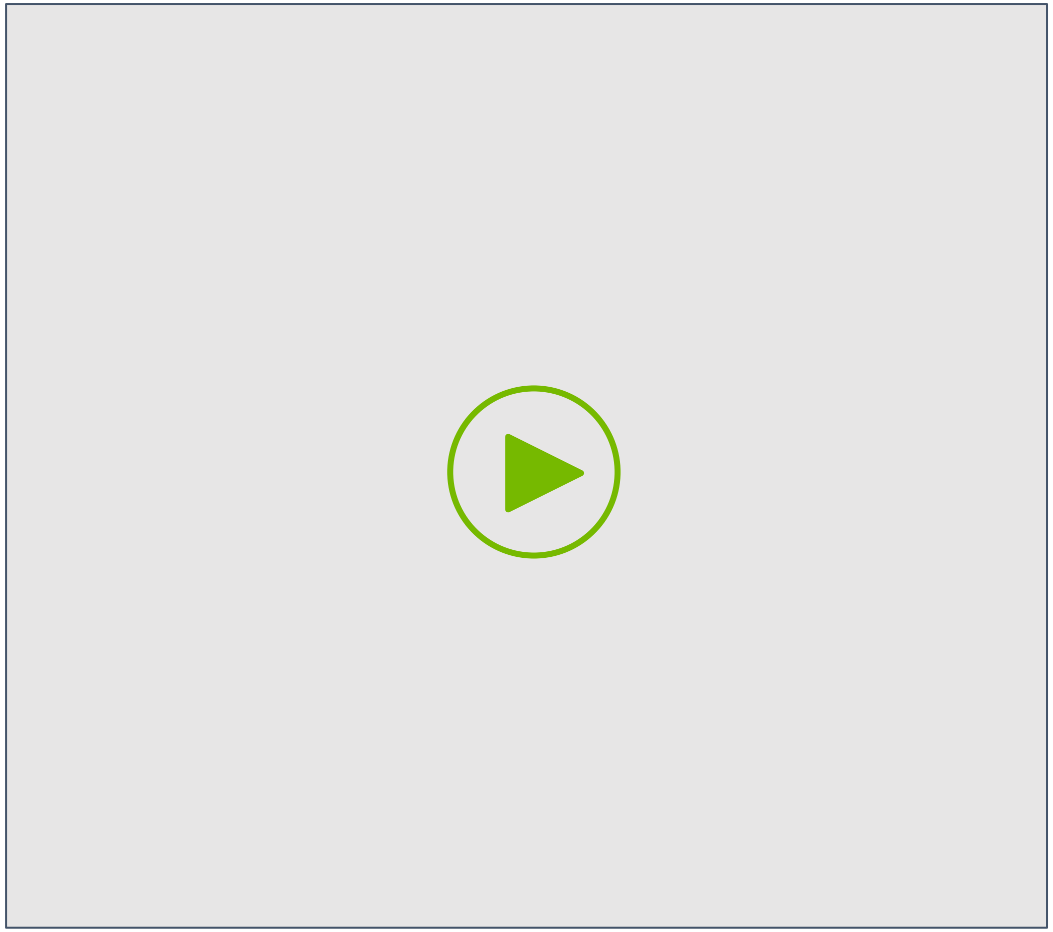


A B C

CUDA MULTI-PROCESS SERVICE CONTROL

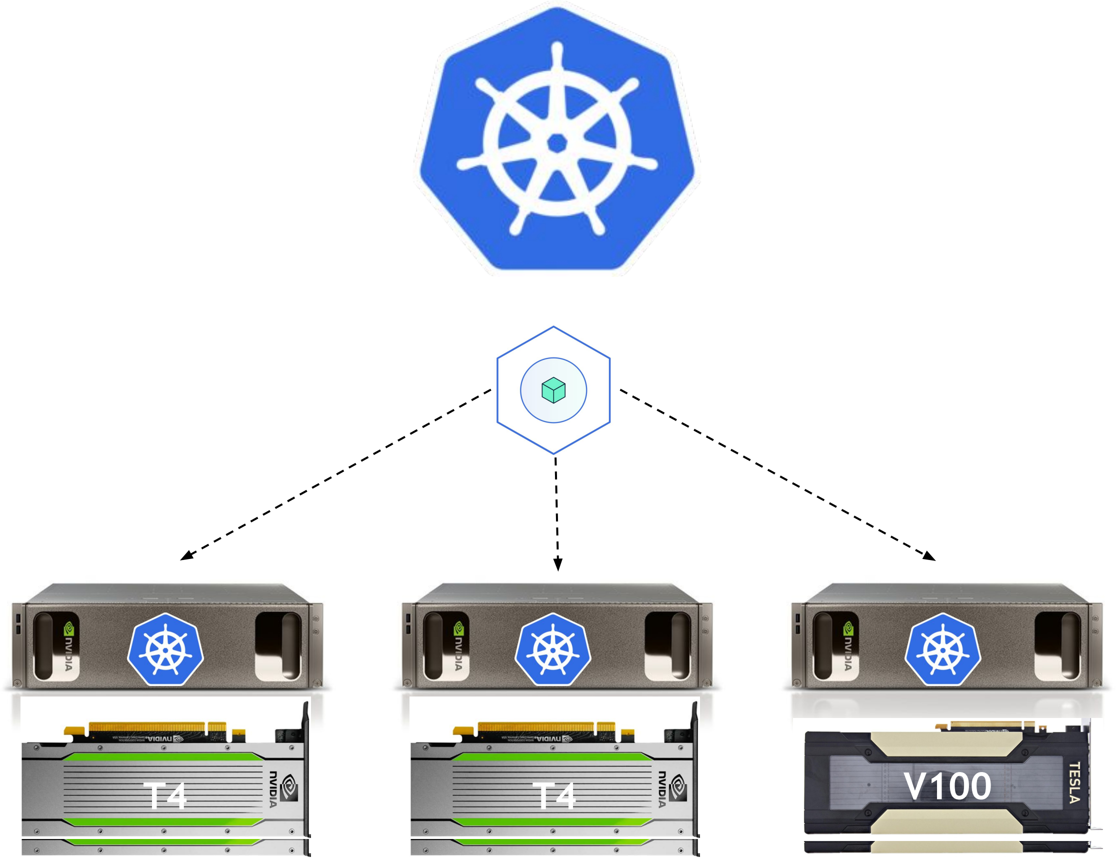
GPU MULTI-PROCESS SERVICE

Memory Partitions



GPU selectors on GKE

```
apiVersion: gpu.resource.nvidia.com/v1alpha1
kind: GpuClaimParameters
metadata:
  namespace: kubecon-demo
  name: inference-gpu
spec:
  selector:
    andExpression:
    - memory:
        value: 16G
        operator: LessThanOrEqualTo
    - cudaComputeCapability:
        value: 7.5
        operator: GreaterThanOrEqualTo
```

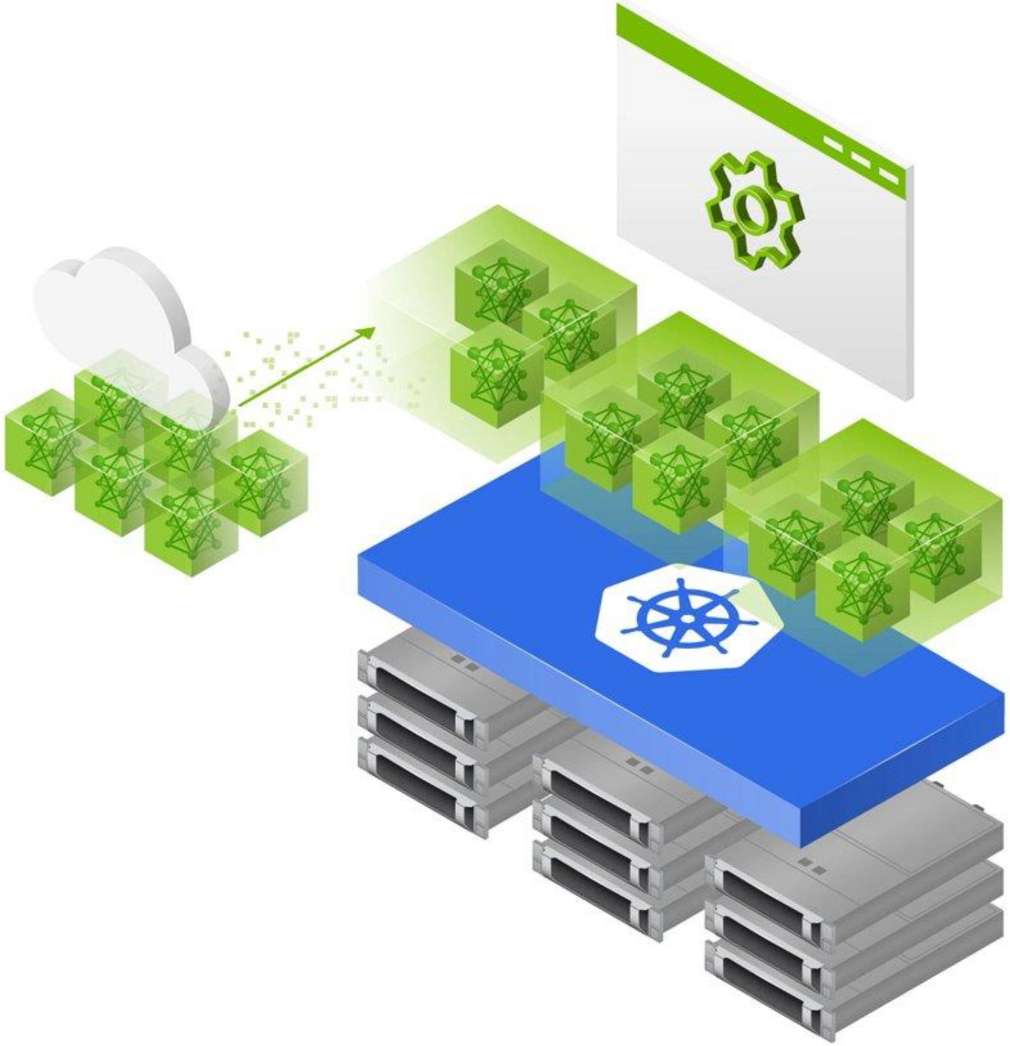


Triton Management Service (TMS)

- Automates the deployment of multiple Triton Inference Servers, each serving models with different GPU requirements

- At present, there is no good way to pick and choose which GPU a given server is going to be given access to

- With DRA, TMS is able to “right-size” the GPU given to a server by using selectors provided in the `GpuClaimParamaters` objects



Wrapping Up

Work email:
kklues@nvidia.com

@klueska everywhere else