
KubeCon
CloudNativeCon
North America 2022
BUILDING FOR THE ROAD AHEAD
DETROIT 2022
TiKV
TiKV
HELM

BUILDING FOR THE ROAD AHEAD

DETROIT 2022

Improving User Experience for Device Consumption in Kubernetes

Kate Goldenring, Alexander Kanevskiy, Patrick Ohly

Improving User Experience for Device Consumption in Kubernetes

Kate Goldenring
Senior Software Engineer
Fermyon

KubeCon | CloudNativeCon
North America 2022

BUILDING FOR THE ROAD AHEAD

DETROIT 2022

KubeCon | CloudNativeCon
North America 2022

BUILDING FOR THE ROAD AHEAD

DETROIT 2022

October 24-28, 2021

Alexander Kanevskiy
Principal Engineer
Intel

Patrick Ohly
Senior Software Engineer
Intel

Kubernetes' birthplace

Uniformity of type of resources in data centers

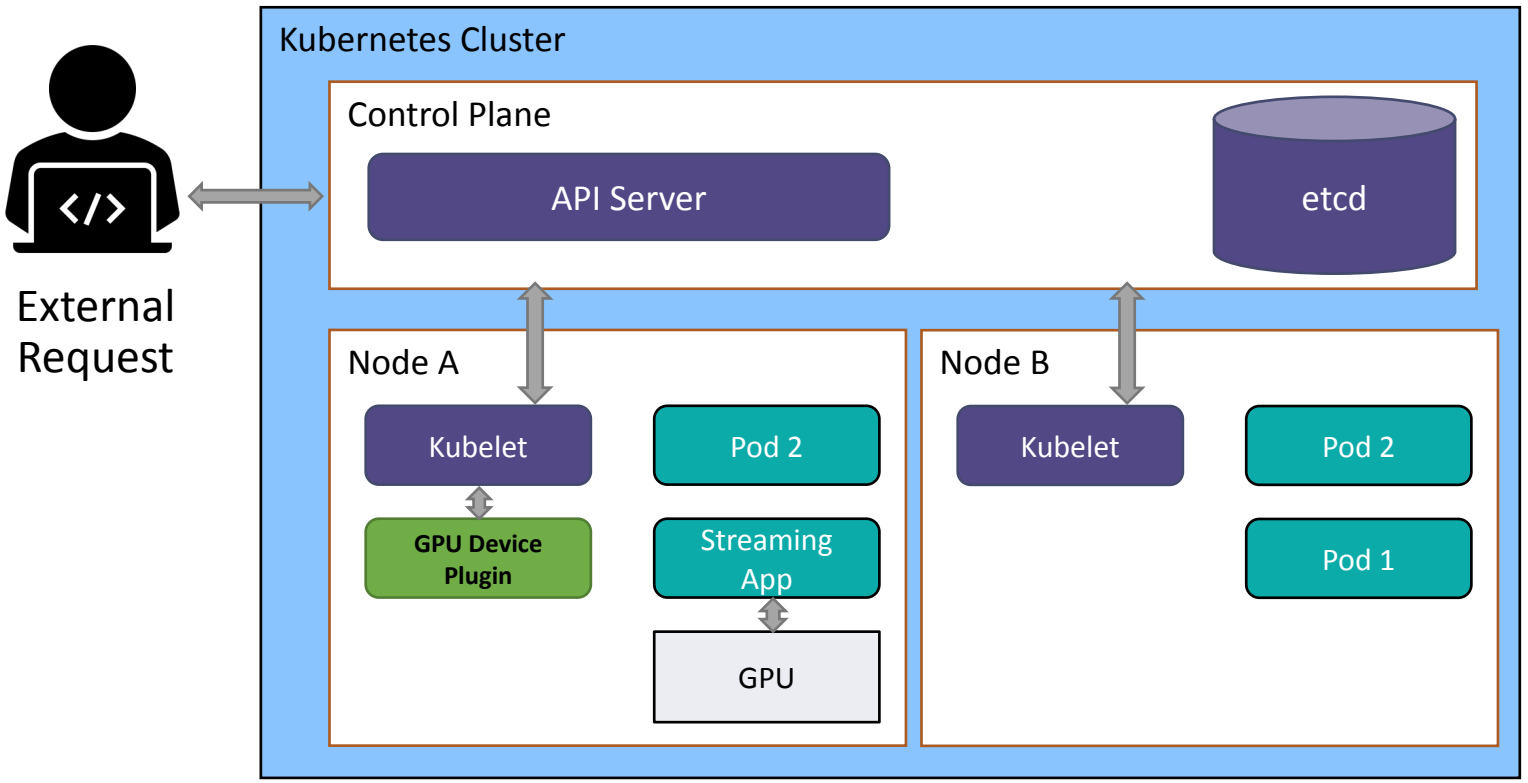
PodSpec: A Kubernetes Application UX


```
kind: Pod
metadata:
  name: streaming-app
spec:
  containers:
  - name: streaming-app
    image: streaming-app:1.0
    resources:
      requests:
        memory: 100Mi
        cpu: 10m
      limits:
        memory: 200Mi
        cpu: 50m
```

- Kubernetes allows you to *declare* what resources must be available for a Pod to be scheduled
- Pod will only be scheduled on Node with adequate resources
- The kubelet reserves the *request*ed amount of resource for Container
- The kubelet enforces resource *limit*
- Kubernetes resource types: CPU, memory (RAM), and hugepages

Servers started to vary, some having more specialized hardware

Kubernetes Device Plugin Interface

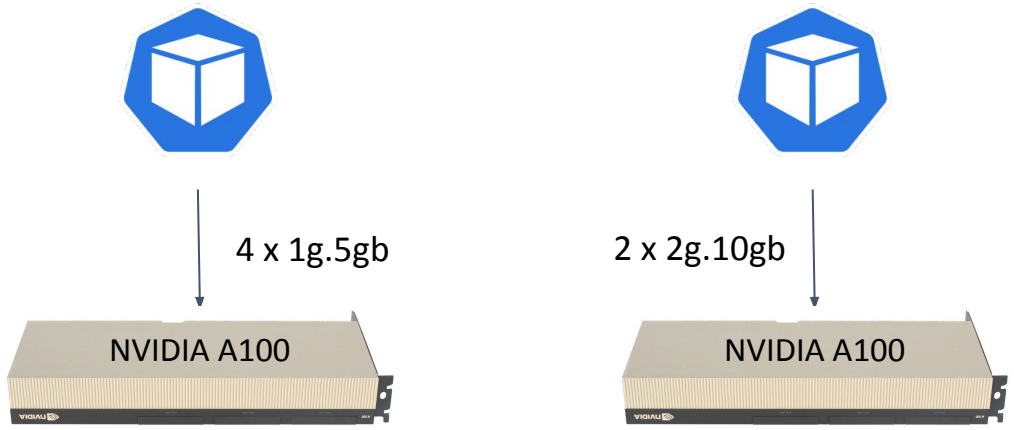



```
kind: Pod
metadata:
  name: streaming-app
spec:
  containers:
  - name: streaming-app
    image: streaming-app:1.0
    resources:
      requests:
        memory: 100Mi
        cpu: 10m
        vendor.com/gpu: 1
      limits:
        memory: 200Mi
        cpu: 50m
        vendor.com/gpu: 1
```

Device Plugins
Is it enough for modern world?

Devices UX: Device “size”?

- Many modern accelerators have internal resources
- Example of NVIDIA MIG:
  - GPU RAM
  - GPU compute units

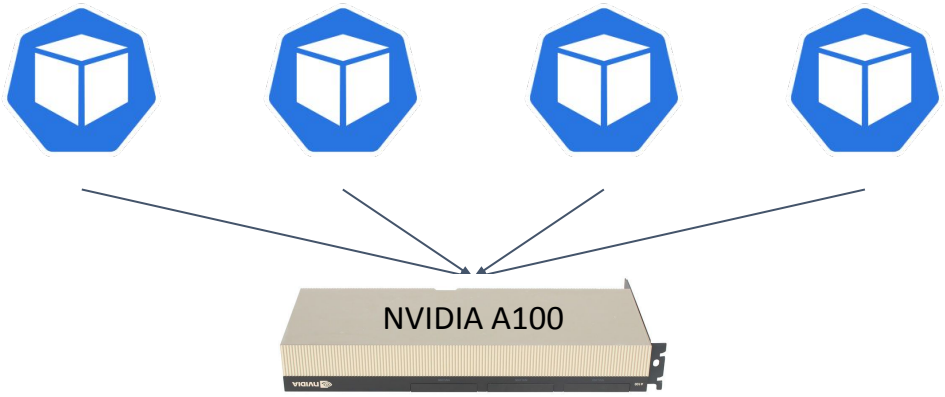



```
nodeSelector:
  csp.com/gpu-partition-size: 1g.5gb
...
containers:
...
- name: container-1
  image: registry.user.io/app:1.0
  resources:
    limits:
      nvidia.com/gpu: 1
```

Reference: Improving GPU Utilization using Kubernetes - Maulin Patel, Google

Devices UX: Shared device?

- Accelerators are not 100% utilized all the time
- Time-sharing between workloads is often needed



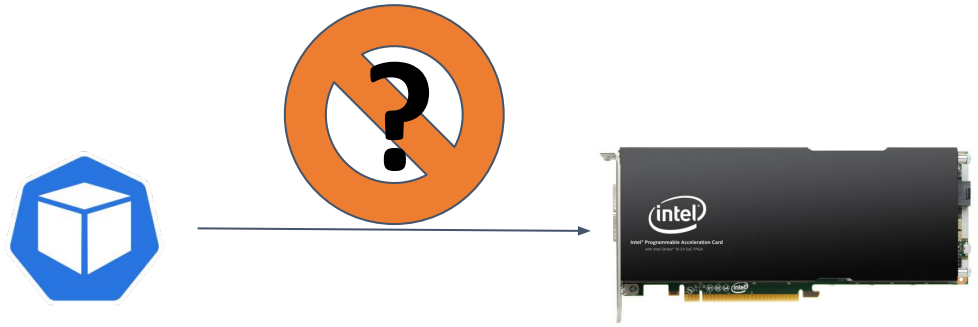

```
nodeSelector:
  csp.com/gpu-sharing-strategy: time-sharing
  csp.com/max-time-shared-clients-per-gpu: 10
...
containers:
...
- name: container-1
  image: registry.user.io/app:1.0
  resources:
    limits:
      nvidia.com/gpu: 1
```

Reference: Improving GPU Utilization using Kubernetes - Maulin Patel, Google

Devices UX: Optional accelerator?

- Some workloads can benefit from optional accelerator devices
  - e.g. crypto for service mesh components
- Can fallback to CPU-only

- Not possible to express in current device model




```
containers:
...
- name: container-1
  image: registry.user.io/app:1.0
  resources:
    requests:
      vendor.com/accelerator: 0
    limits:
      vendor.com/accelerator: 1
```

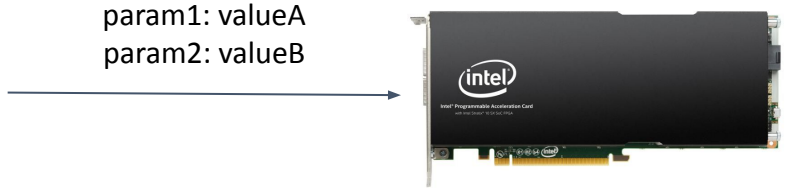
Devices UX: Device parameters?


```
resources:
  limits:
    fpga.intel.com/d5005-compress: 1
```

```
apiVersion: fpga.intel.com/v2
kind: FpgaRegion
metadata:
  name: d5005
spec:
  interfaceId: bfac4d851ee856
```

```
apiVersion: fpga.intel.com/v2
kind: AcceleratorFunction
metadata:
  name: d5005-compress
spec:
  afuId: d8424dc4a4a3c413f89e433683f9040b
  interfaceId: bfac4d851ee856
  mode: region
```

param1: valueA
param2: valueB

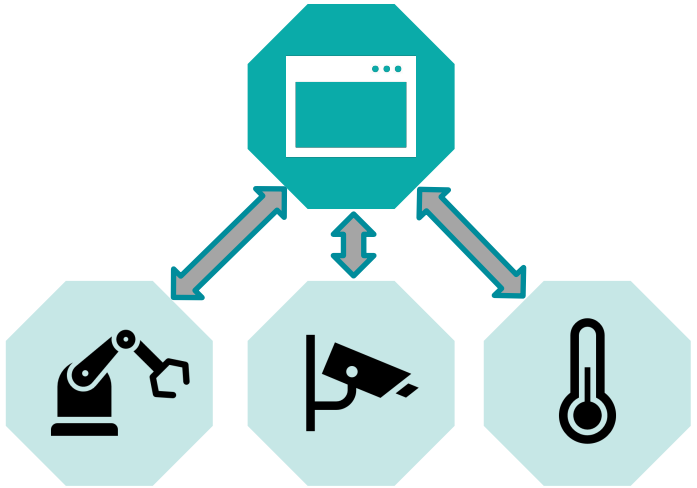



```
containers:
...
- name: container-1
  resources:
    limits:
      fpga.intel.com/region-bfac4d851ee856: 1
  env:
    - name: fpga_afuId
      value: d8424dc4a4a3c413f89e433683f9040b
```


Apps started leveraging resources external to the servers

Expanding K8s Device UX beyond Static Hardware

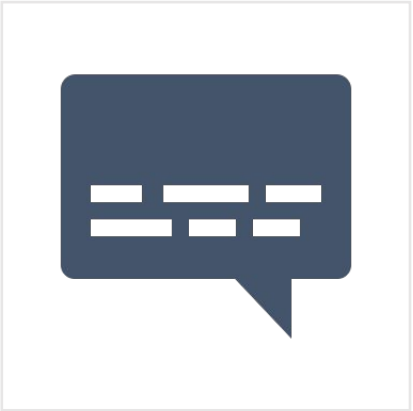

```
kind: Pod
metadata:
  name: factory-app
spec:
  containers:
  - name: factory-app
    image: factory-app:1.0
    resources:
      requests:
        memory: 100Mi
        cpu: 10m
        vendor.com/thermometer: 1
        vendor.com/ip-camera: 2
        vendor.com/robot: 1
```



How to expand Kubernetes functionality

Operator

CustomResourceDefinition (CRD) + Controller



Kubernetes core

Kubernetes Enhancement Proposal (KEP) + implementation

Akri: A Kubernetes Resource Interface

- Discovers devices. Handles dynamic appearance and disappearance of IoT devices
- Creates a device plugin for each discovered device, representing them as a native Kubernetes resources
- Kubernetes Operator
- Open-source CNCF Sandbox Project

Using other devices with Akri

1. Declare what to add as an extended Kubernetes resource

```
kind: Configuration
metadata:
  name: ip-camera
spec:
  discoveryHandler:
    name: onvif
     discoveryDetails:
       ipAddresses:
          action: Exclude
          items:
          - 10.0.0.1
          - 10.0.0.2
        macAddresses:
          action: Exclude
          items: []
        scopes:
          action: Include
          items:
          - onvif://..GreatONVIFCamera
          - onvif://..AwesomeONVIFCamera
```

2. Akri finds it and advertises it to kubelet

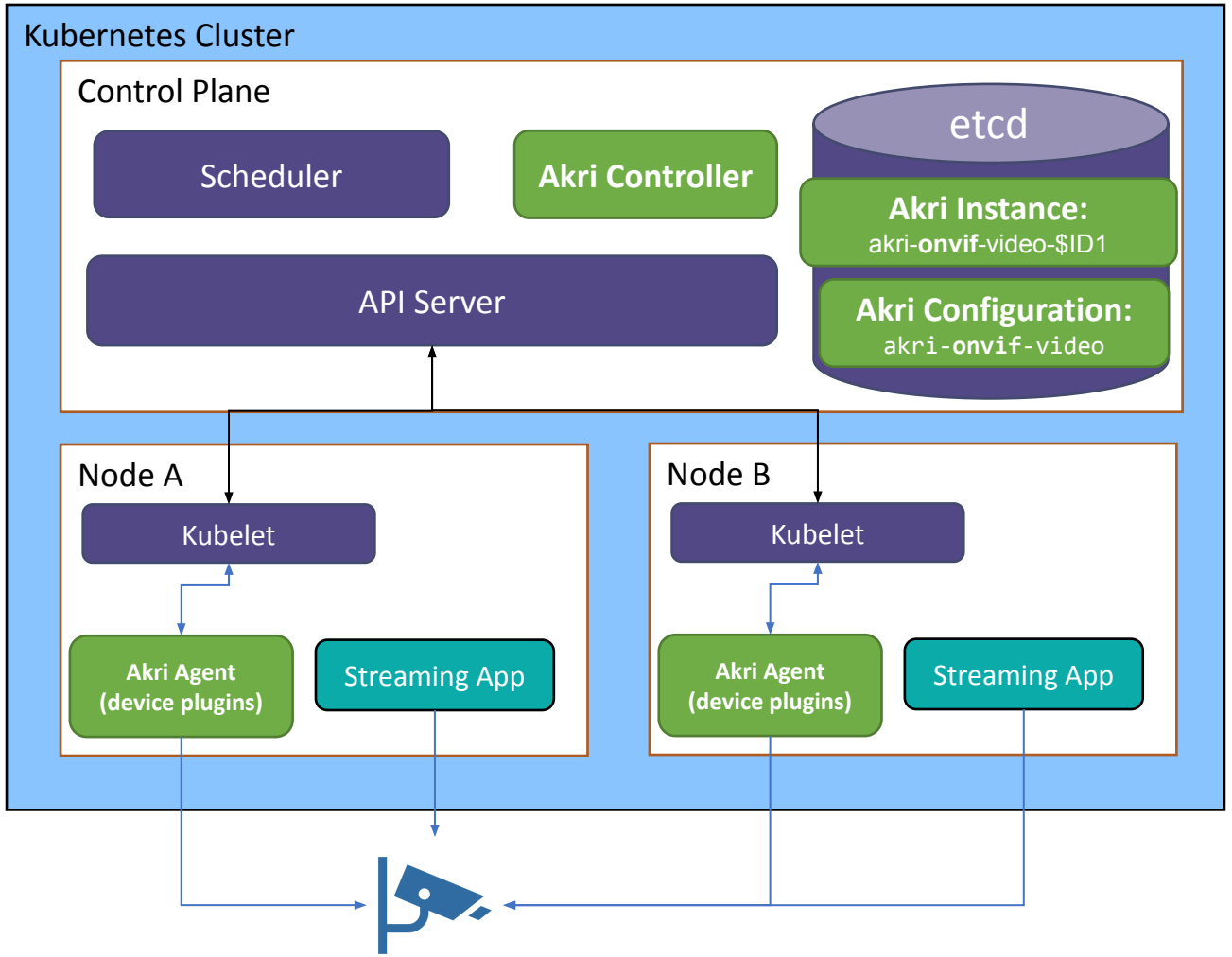
3. Request it in your Pod spec just like any other resource

```
kind: Pod
metadata:
  name: streaming-app
spec:
  containers:
  - name: streaming-app
    image: streaming-app:1.0
    resources:
      requests:
        akri.sh/akri-onvif-video-$ID1: 1
```

Akri Operator Components


```
kind: Configuration
metadata:
  name: akri-onvif-video
spec:
  discovery-handler:
    name: onvif
  brokerSpec:
    spec:
     containers:
     - name: streaming-app
       image: streaming-app:1.0
```

```
kind: Pod
metadata:
  name: streaming-app
spec:
  containers:
  - name: streaming-app
    image: streaming-app:1.0
    resources:
      requests:
        akri.sh/akri-onvif-video-$ID1: 1
```



Akri Instance CRD: Handling resource sharing

- Akri Instance regulates device usage across nodes
- Akri Agents update the instance when they discover the device (`nodes`)
- Akri Agents check and reserve the device availability (`deviceUsage`) before allowing a Pod requesting the device to be run.

```
kind: Instance
metadata:
  name: akri-onvif-video-ID1
  namespace: default
  ownerReferences:
  - apiVersion: akri.sh/v0
    kind: Configuration
    name: akri-onvif-video
spec:
  brokerProperties:
    ONVIF_DEVICE_IP_ADDRESS: 10.0.0.4
    ONVIF_DEVICE_MAC_ADDRESS: 48:0f:cf:7e:1a:5e
    ONVIF_DEVICE_SERVICE_URL: http://10.0.0.4:1000/onvif/device_service
  configurationName: akri-onvif
  deviceUsage:
    akri-onvif-video-ID1-0: node-a
    akri-onvif-video-ID1-1: node-b
    akri-onvif-video-ID1-2: ""
  nodes:
  - node-a
  - node-b
  shared: true
```

Improving Akri UX be better?

Dream UX

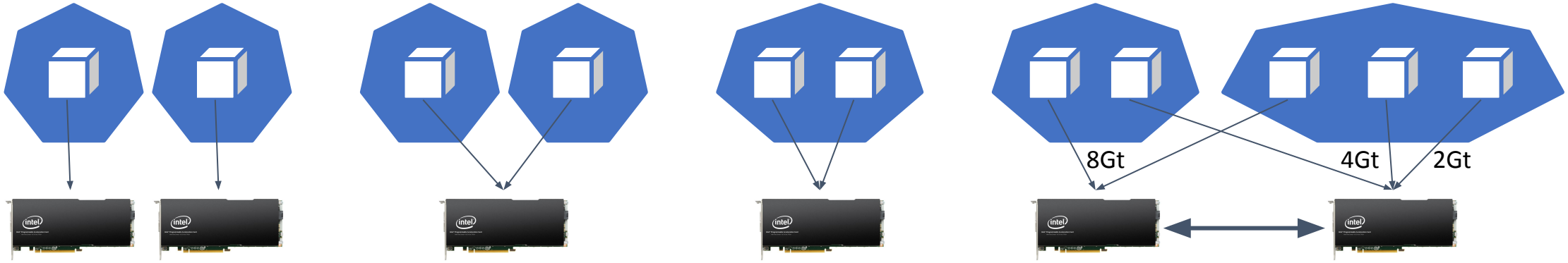
```
kind: Pod
metadata:
  name: streaming-app
spec:
  containers:
  - name: streaming-app
    image: streaming-app:1.0
    resources:
      requests:
        memory: 100Mi
        cpu: 10m
        akri.sh/usb-thermometer: 1
        akri.sh/ip-camera: 2
        akri.sh/opcua-robot:
                fast: 1
                precise: 1
```

Today

```
kind: Pod
metadata:
  name: streaming-app
spec:
  containers:
  - name: streaming-app
    image: streaming-app:1.0
    resources:
      requests:
        memory: 100Mi
        cpu: 10m
        akri.sh/usb-thermometer-$ID1: 1
        akri.sh/ip-camera-$ID1: 1
        akri.sh/ip-camera-$ID2: 1
        akri.sh/opcua-fast-robot-$ID1: 1
        akri.sh/opcua-precise-robot-$ID1: 1
```

Devices UX
What do we want?

Devices usage scenarios



- Concept
  - Separation of “Claim” with parameters and the use of “Allocated Instance”

- Example Usages
  - 2 Pods, 2 Devices
  - 2 Pods sharing same Device
  - 2 Containers within Pod to share Device
  - Multiple Containers in multiple Pods to share dynamically partitioned interconnected Devices (e.g. different device RAM slots)

Container Orchestrated Devices WG

- Idea and early discussions
  - early in 2019 - NVIDIA and Intel
- Informal group forming – KubeCon NA’19
  - NVIDIA, Intel, IBM, RedHat
- Formal COD WG era under CNCF TAG-Runtime - summer 2020
  - NVIDIA: Renaud Gaubert, Evan Lezar, Kevin Klues, Zvonko Kaiser
  - Intel: Alexander Kanevskiy, Ed Bartosh, Krisztian Litkey, Patrick Ohly
  - RedHat: Mrunal Patel, Urvashi Mohnani
  - IBM: Mike Brown

Container Orchestrated Devices WG

- CDI: Container Device Interface for runtimes
  - https://github.com/container-orchestrated-devices/container-device-interface
    - CRI-O 1.23.2+
    - containerd 1.7 (unreleased)
- Dynamic Resource Allocation - KEP-3063

Dynamic Resource Allocation

DRA Design Decisions

- Some aspects inspired by Container Storage Interface (CSI) volumes:
  - ResourceClaim object represents requirements.
  - Can be specified via template inside PodSpec.
  - Pods specify claims, containers reference them.
  - Allocation can be immediate or wait for usage by a Pod.

- Differences:
  - ResourceClaim status represents the full state (allocated, reserved, …).
  - Parameters are separate objects and opaque to Kubernetes, with validation through CRD.
  - Communication between scheduler and resource driver through apiserver.

DRA for Users: Inline Claim


```yaml
apiVersion: v1
kind: ResourceClass
metadata:
  name: example
driverName:
  test-driver.cdi.k8s.io
```

```yaml
apiVersion: v1
kind: ConfigMap
metadata:
  name: inline-claim-parameters
  namespace: default
data:
  a: b
```

```yaml
apiVersion: v1
kind: Pod
metadata:
  name: test-inline-claim
spec:
  containers:
  - name: with-resource
    image: registry.k8s.io/e2e-test-images/dra-test:1.0
    resources:
      claims:
      - my-resource
  - name: without-resource
    image: registry.k8s.io/e2e-test-images/dra-test:1.0
  resourceClaims:
  - name: my-resource
    claim:
      template:
        metadata:
          labels:
            app: inline-resource
        spec:
          resourceClassName: example
          parameters:
            kind: ConfigMap
            name: inline-claim-parameters
```

DRA for Users: Separate Claim


```yaml
apiVersion: v1
kind: ResourceClaim
metadata:
  name: external-claim
spec:
  resourceClassName: example
  parameters:
    kind: ConfigMap
    name: external-claim-parameters
```

```yaml
apiVersion: v1
kind: ResourceClass
metadata:
  name: example
driverName:
  test-driver.cdi.k8s.io
```

```yaml
apiVersion: v1
kind: ConfigMap
metadata:
  name: inline-claim-parameters
  namespace: default
data:
  a: b
```

```yaml
apiVersion: v1
kind: Pod
metadata:
  name: test-external-claim
spec:
  containers:
  - name: with-resource
    image: registry.k8s.io/e2e-test-images/dra-test:1.0
    resources:
      claims:
      - my-resource
  - name: without-resource
    image: registry.k8s.io/e2e-test-images/dra-test:1.0
  resourceClaims:
  - name: my-resource
    claim:
      resourceClaimName: external-claim
```

DRA for Kubernetes

- KEP accepted for 1.25, updated for 1.26
- Code to be merged into 1.26, pending in PR #111023
- Modified components:
  - kube-apiserver: new types in core API group
  - kube-controller-manager:
    - create claims from inline templates
    - remove completed pods from claim status
  - kube-scheduler: filter nodes, communication with resource drivers
  - kubelet: prepare resources, inject into containers

DRA for Vendors

- Must implement controller and kubelet plugin.
- Controller must handle allocation and resource tracking:
  - could be done through CRDs (similar to how Akri does it)
- Document deployment and usage for customers.

- Generic helper code to be published in k8s.io/dynamic-resource-allocation Go module.
- Open source drivers:
  - Reference resource driver in k/k/test/e2e/dra/test-driver
  - NVIDIA: https://gitlab.com/nvidia/cloud-native/k8s-dra-driver

Contact

- Container Orchestrated Devices WG
  - Every second Tuesday
  - 7:00-8:00 PST
  - Kubernetes Slack: #sig-node
  - CNCF Slack: #tag-runtime
- docs.akri.sh
  - Every first Tuesday
  - 9:00-10:00 PST
  - Kubernetes Slack: #akri

- Kate Goldenring
  - Twitter: @KateGoldenring
  - Slack: @Kate Goldenring
- Alexander Kanevskiy
  - alexander.kanevskiy@intel.com
  - Slack: @akanevskiy
- Patrick Ohly
  - patrick.ohly@intel.com
  - Slack: @pohly

BUILDING FOR THE ROAD AHEAD

DETROIT 2022

Q & A

KubeCon | CloudNativeCon
North America 2022

BUILDING FOR THE ROAD AHEAD

DETROIT 2022

Please scan the QR Code above to leave feedback on this session