



KubeCon



CloudNativeCon

Europe 2026



KubeCon



CloudNativeCon

Europe 2026

#KubeCon #CloudNativeCon

Slinky Expanded: Slurm, Kubernetes, and DRA

Marlow Warnicke (NVIDIA)

Praveen Krishna (Google)



What is Slurm?



KubeCon



CloudNativeCon

Europe 2026

- Leading HPC Workload Manager
 - Workload Manager = Scheduler + Resource Manager
 - Roughly equivalent to "Orchestrator"
 - Scheduler:
 - Prioritize and decide which jobs to run on which parts of the cluster
 - Resource Manager:
 - Track node state and resources
 - Launch jobs
- Manages the majority of the TOP500 supercomputers
 - Also manages most AI/ML training workloads
 - Scales beyond 15,000 nodes in the cluster
- Open-Source
 - GPL-v2+



What is Slinky?



KubeCon



CloudNativeCon

Europe 2026



Slurm-Bridge

Integration with Kubernetes
scheduling API

Slurm Operator

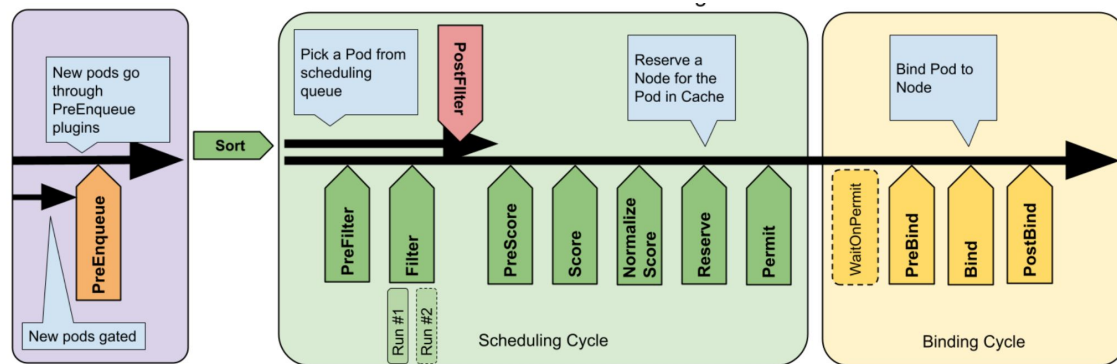
Managing Slurm running within
Kubernetes

Slurm Bridge - Slurm as a Kubernetes Scheduler

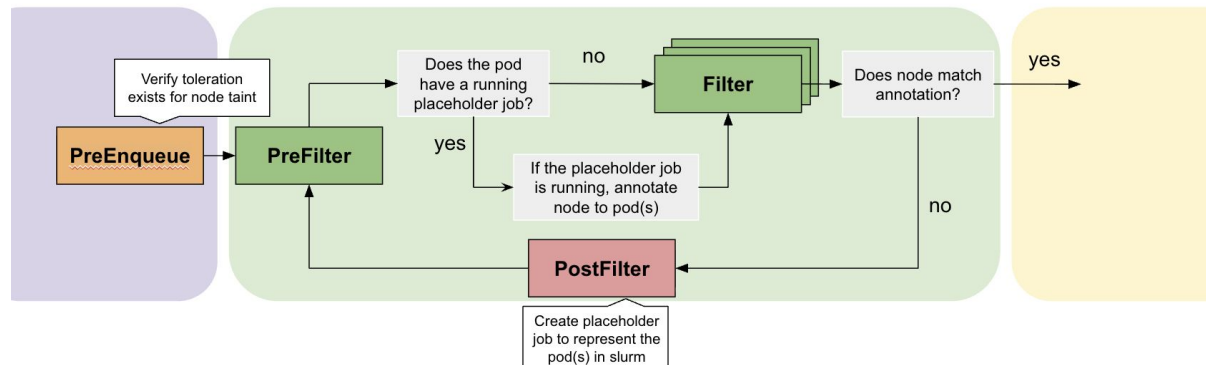
- Slurm as a Kubernetes scheduler using the Scheduling Framework
 - Uses PreEnqueue, PreFilter, Filter, and PostFilter for placement decision
 - Uses PreBind to generate DRA ResourceClaims
- Translate pod resources into a Slurm placeholder job
 - Placeholder job in Slurm will determine when and where pod(s) run
 - Placeholder job is an “external job” in Slurm
 - Will leverage new Workload resource coming in 1.36
- Scheduled by Slurm, launched by kubelet
 - Slurm schedules to nodes running slurmd or “external nodes” without slurmd

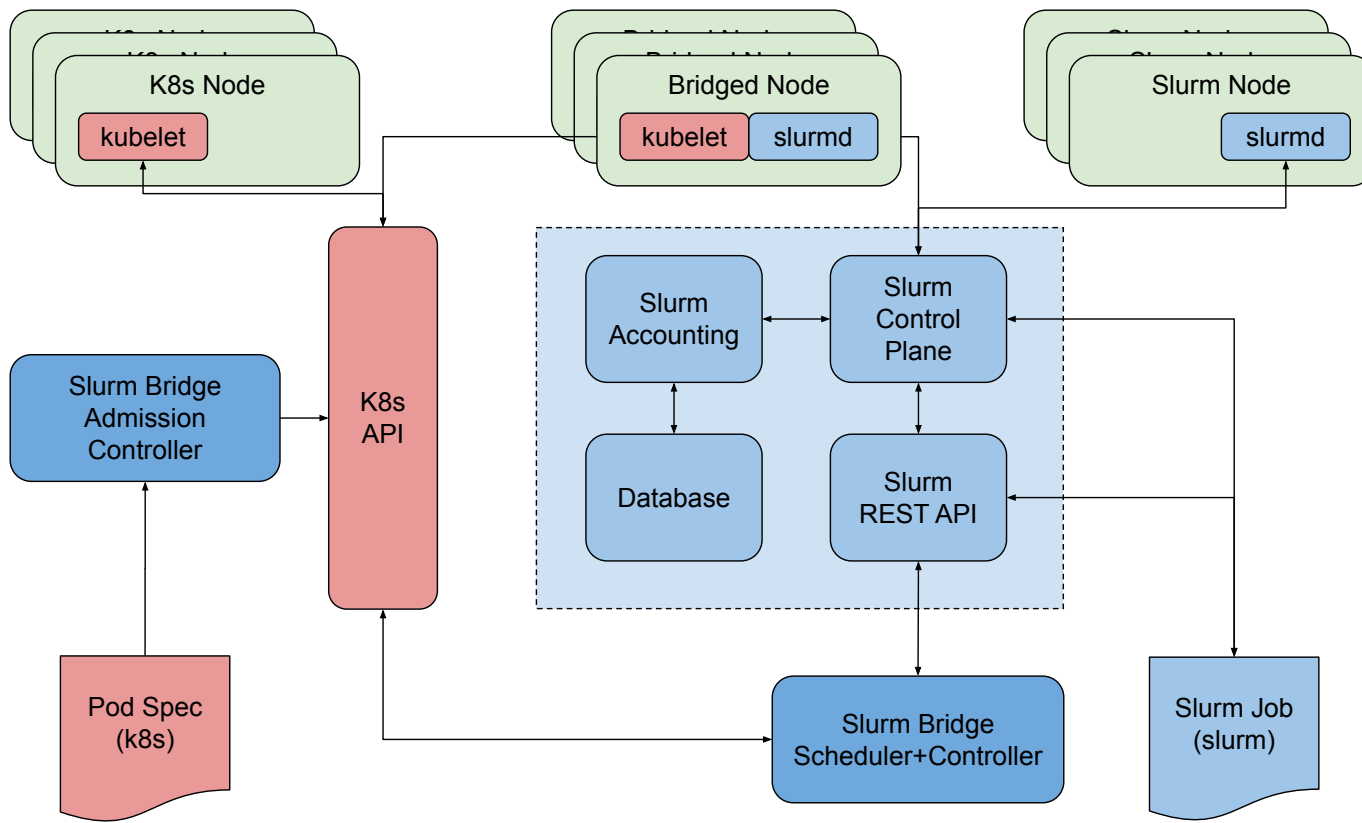
Slurm Bridge - Slurm as a Kubernetes Scheduler

Kubernetes Scheduler Framework



Slurm-Bridge Mappings







KubeCon



CloudNativeCon

Europe 2026

CPU DRA Driver



What is DRA ?



KubeCon



CloudNativeCon

Europe 2026

Dynamic Resource Allocation (DRA) provides a flexible, built-in Kubernetes API for requesting, allocating, and sharing hardware resources

Why we built the CPU DRA Driver



KubeCon



CloudNativeCon

Europe 2026

The Problem: Kubelet **Static CPU Policy** for guaranteed CPU allocation is insufficient. Workloads cannot dictate specific CPU assignments, and hardware heterogeneity is ignored.

★ align by socket not work when pod using multiple gpu card #122

Open

leemingeer opened on Dec 13, 2023 · edited by leemingeer

What happened?

I am running an AI application and expect to achieve the highest performance by using the Topology Manager to establish NUMA affinity binding between CPU and GPU.

so I restart kubelet with such options

```
kubeReserved:
  cpu: "1"
  memory: "2Gi"
topologyManagerPolicy: restricted
topologyManagerPolicyOptions:
  prefer-closest-numa-nodes: "true"
cpuManagerPolicy: static
cpuManagerPolicyOptions:
  align-by-socket: "true"
  distribute-cpus-across-numa: "true"
featureGates:
  CPUManagerPolicyAlphaOptions: true
  TopologyManagerPolicyOptions: true
  TopologyManagerPolicyAlphaOptions: true
```

then I use multiple GPU cards,
if the number of GPU cards are more than one socket, the allocated cpu and gpu are not aligned by socket.

What did you expect to happen?

The CPU cores requested by the pod should be from the NUMA node where the GPUs are located.

Example of HPC application expectations for CPU placement options:

```
--cores-per-socket --cpus-per-gpu --cpus-per-task
--extra-node-info=<sockets>[:cores[:threads]] --gpus-per-node --gpus-per-socket
--gpus-per-task --mincpus --ntasks --ntasks-per-core --ntasks-per-gpu
--ntasks-per-node --ntasks-per-socket --sockets-per-node --threads-per-core
```

Even finer-grained layout management for separate processes:

```
--distribution={*|block|cyclic|arbitrary|plane=<size>}[:{*|block|cyclic|fcyclic}[:{*|block|cyclic|fcyclic}]][, {Pack|NoPack}]
```

Why we built the CPU DRA Driver



KubeCon



CloudNativeCon

Europe 2026

The Problem: Kubelet **Static CPU Policy** for guaranteed CPU allocation is insufficient. Workloads cannot dictate specific CPU assignments, and hardware heterogeneity is ignored.

The Solution: Dynamic Resource Allocation (DRA)



Expressive

Use CEL expressions to make precise requests, such as specific NUMA domains or performance vs. efficiency cores.



Extensible

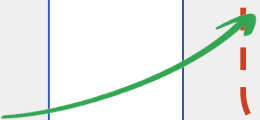
Expose new hardware properties or intra node topology without touching the core kubelet or scheduler logic

Request Guaranteed CPUs through **DRA ResourceClaims** just like GPUs and NICs

```
apiVersion: v1
kind: Pod
metadata:
  name: my-training-job
spec:
  resources:
    requests:
      cpu: "5"
      memory: "10Gi"
    limits:
      cpu: "5"
      memory: "10Gi"
  containers:
  - name: app
    image: myapp:0.1
    resources:
      claims:
        - name: "gpus-nic-cpu-request"
      resourceClaims:
        - name: "gpus-nic-cpu-request"
          resourceClaimName: gpu-nic-cpu-claim
```



```
apiVersion: resource.k8s.io/v1
kind: ResourceClaim
metadata:
  name: gpu-nic-cpu-claim
spec:
  devices:
    requests:
      - name: gpu
        deviceClassName: gpu.nvidia.com
      - name: nic
        deviceClassName: dranet
      - name: cpu
        deviceClassName: dracpu
        count: 5
        selectors:
          - cel:
              expression: device.attributes["dra.cpu"].coreType == "p-core"
```



CPU DRA Driver



KubeCon



CloudNativeCon

Europe 2026

- Installed as a daemonset on the cluster.
- Publishes CPUs through **ResourceSlice**.
- Workloads use **ResourceClaims** for requesting **Guaranteed/Pinned CPUs**.
- Disable Kubelet **Static CPU Policy**. Ensures the DRA driver has exclusive control over CPU pinning without Kubelet interference.



kubernetes-sigs / dra-driver-cpu

Resource Representation: Individual Mode



KubeCon



CloudNativeCon

Europe 2026

- Each CPU is enumerated as a device in the `ResourceSlice` along with all its properties.
- Used by **External Schedulers like SLURM** to inject specific CPU requests when we translate from Generic Resource (GRES) to `ResourceClaim`

```
apiVersion: resource.k8s.io/v1
kind: ResourceSlice
spec:
  driver: dra.cpu
  nodeName: gke-dratest-a2-dratest-pkrishn-a2-8gp-d9fcca48-pcds
  devices:
    - name: "cpudev0"
      attributes:
        "dra.cpu/cpuID": 0
        "dra.cpu/coreID": 0
        "dra.cpu/coreType": "p-core"
        "dra.cpu/l3CacheID": 0
        "dra.cpu/socketID": 0
        "dra.cpu/numaNode": 0
    - name: "cpudev48"
      attributes:
        "dra.cpu/cpuID": 48
        "dra.cpu/coreID": 24
        "dra.cpu/coreType": "p-core"
        "dra.cpu/l3CacheID": 11
        "dra.cpu/socketID": 1
        "dra.cpu/numaNode": 1
```

Detailed topology
information using
device attributes

Resource Representation: Grouped Mode



KubeCon



CloudNativeCon

Europe 2026

- Uses DRA Consumable Capacity to **pool CPUs into logical blocks** based on node topology or properties (like core-type).
- Advanced CPU Allocation strategies like packing or spreading cores based on the grouping.

```
apiVersion: resource.k8s.io/v1
kind: ResourceSlice
spec:
  driver: dra.cpu
  nodeName: gke-dratest-a2-dratest-pkrishn-a2-8gp-d9fcc48-pcds1
  devices:
    - name: cpudevsocket0
      attributes:
        dra.cpu/socketID: 0
        dra.cpu/numaNodeID: 0
        dra.cpu/smtEnabled: true
      capacity:
        dra.cpu/cpu:
          value: "32"
      allowMultipleAllocations: true
    - name: cpudevsocket1
      attributes:
        dra.cpu/socketID: 1
        dra.cpu/numaNodeID: 1
        dra.cpu/smtEnabled: true
      capacity:
        dra.cpu/cpu:
          value: "32"
      allowMultipleAllocations: true
```

One device
per CPU
socket

Resource Alignment



KubeCon



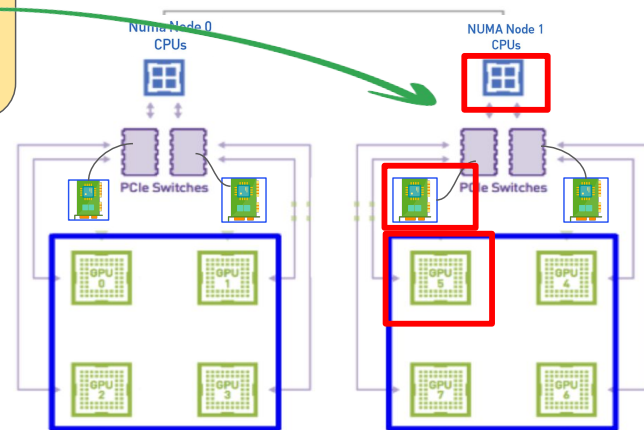
CloudNativeCon

Europe 2026

Better Performance through Resource Alignment across CPUs/Accelerators/NICs

```
apiVersion: resource.k8s.io/v1
kind: ResourceClaim
metadata:
  name: gpu-nic-cpu-request
spec:
  devices:
    requests:
      - name: gpu
        deviceClassName: gpu.nvidia.com
      - name: nic
        deviceClassName: dronet
      - name: cpu
        deviceClassName: dracpu
        count: 5
  constraints:
    - matchAttribute: 'resource.kubernetes.io/pcieRoot'
      requests: ["gpu", "nic"]
    - matchAttribute: 'dra.net/numaNode'
      requests: ["cpu", "nic"]
```

CPU, GPU and
NIC from same
NUMA domain



Architecture



KubeCon



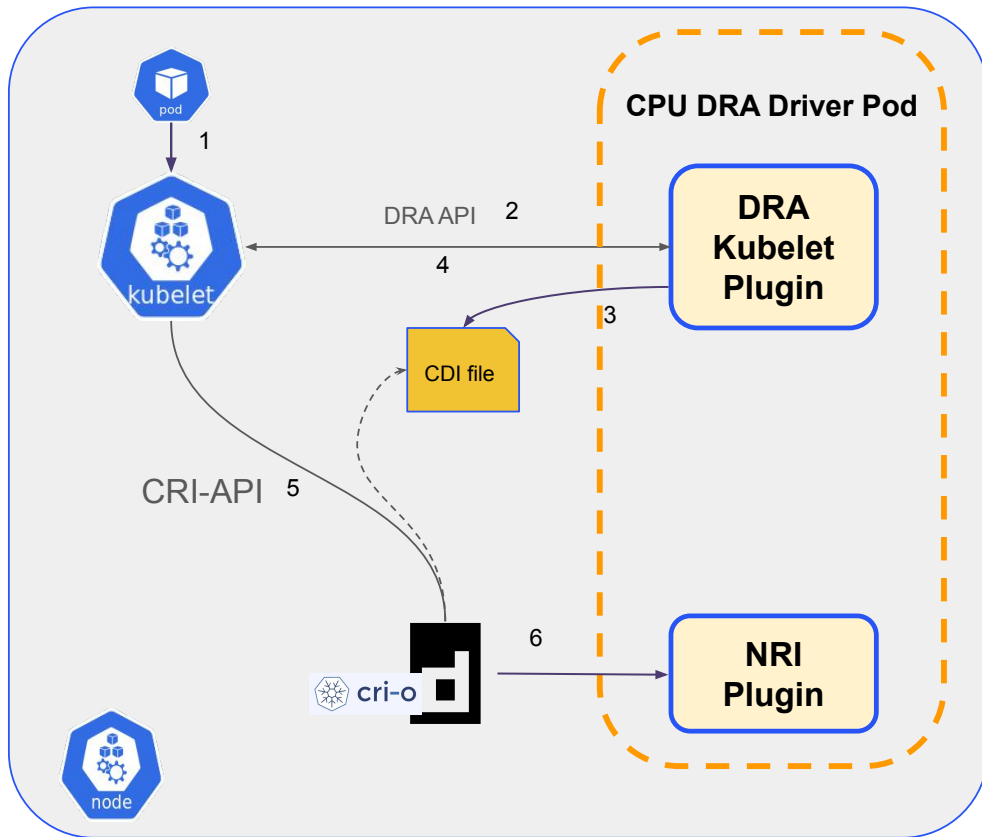
CloudNativeCon

Europe 2026

Two Core Components

1. DRA Kubelet Plugin

- Topology Discovery
- Claim Processing
- Runtime Communication through CDI



Architecture



KubeCon



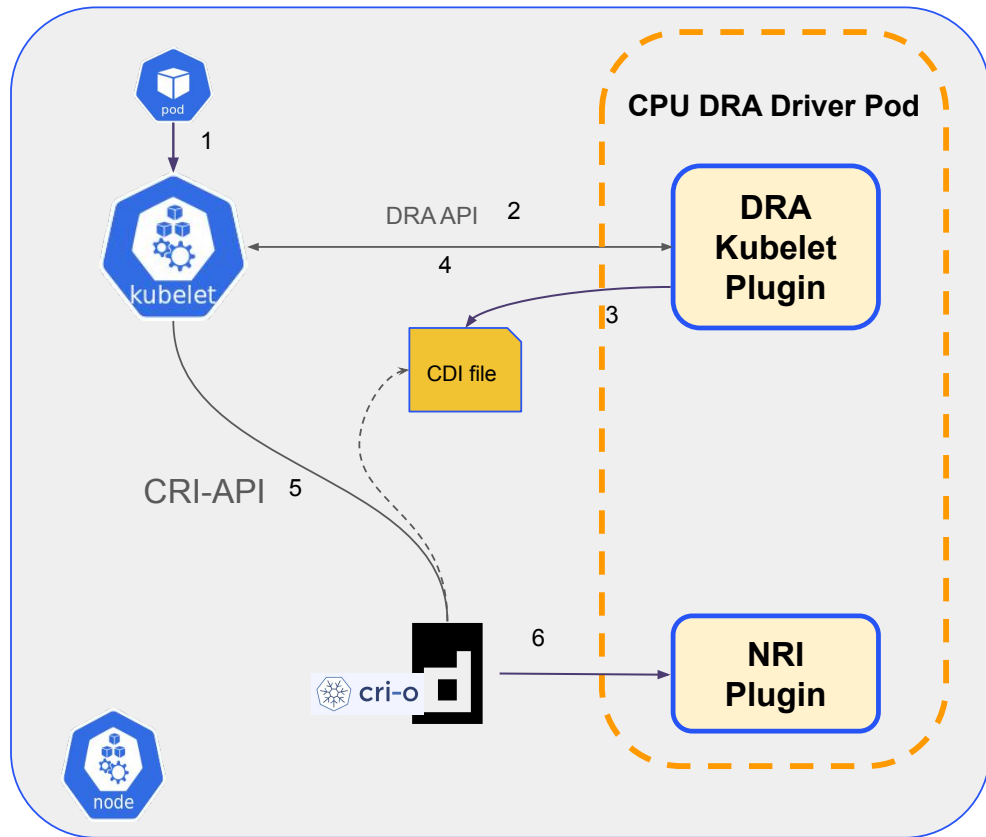
CloudNativeCon

Europe 2026

Two Core Components

2. Node Resource Interface (NRI) Plugin

- Intercept container lifecycle events and modify resource configuration.
- Enforces exclusive CPU pinning based directly on the ResourceClaim allocation.
- Dynamically updates the shared CPU pool for other containers across all pods





KubeCon



CloudNativeCon

Europe 2026

Slurm-Bridge with DRA

What Changed with Slurm Bridge



KubeCon



CloudNativeCon

Europe 2026

- External node support was added to Slurm, so the controller can schedule nodes it doesn't manage (ie, no slurmd)
- DRA drivers have become available (such as CPU DRA driver) to allow us to propagate the chosen fine-grained assignment specified in the controller to the node
- **Result:** We no longer need to have slurmd on every node. Combined with DRA, we get with all the nice features of Slurm scheduling down to the resource!

With DRA changes Slurm-Bridge has ...



KubeCon



CloudNativeCon

Europe 2026

- Each node can run both **Slurm** and **Kubernetes** workloads, ~~not both concurrently~~
 - The kubelet will manage *on-node* resource assignment for all workloads
- **Node controller can now add and remove external nodes**
- **Can now handle multiple pods per node (alpha at this moment)**
- Device plugins ~~are will be supported, but not Dynamic Resource Allocation (DRA)~~
 - ~~○ DRA is still in alpha and has a volatile API, currently~~
- **DRA is supported**
 - CPU DRA Driver
 - GPU Example Driver
 - NVIDIA GPU Driver (alpha)



KubeCon



CloudNativeCon

Europe 2026

Demo

What's Next

1. Extend this effort to include memory.
2. Enable Native understanding of DRA managed CPU/Memory in Scheduler, Kubelet - [KEP-5517](#)
3. Better CPU/Memory modelling with newer DRA features like Partitionable Devices ([KEP-4815](#)), List Type Attributes ([KEP-5698](#)) etc. to better represent node micro-topology.



KubeCon



CloudNativeCon

Europe 2026



Slinky



CPU DRA Driver



KubeCon



CloudNativeCon

Europe 2026

Questions?



KubeCon



CloudNativeCon

Europe 2026





KubeCon



CloudNativeCon

Europe 2026

Demo Slides

Demo



CloudNativeCon

Europe 2026

Job	State	Age	Completed	Job	State	Age	Completed
job-sleep-dra-wk885	0/1	Completed	74s	job-sleep-dra-wk885	0/174s		
job-sleep-dra-jmjc8	0/1	Pending	0	0s			
job-sleep-dra-jmjc8	0/1	Pending	0	0s			
job-sleep-dra-jmjc8	0/1	Pending	0	job-sleep-dra-jmjc8	0/1	Pending	0
0/1	Pending	0	0s				
job-sleep-dra-jmjc8	0/1	Pending	0	0s			
job-sleep-dra-jmjc8	0/1	Pending	0	0s			
job-sleep-dra-jmjc8	0/1	Pending	0	0s			
job-sleep-dra-jmjc8	0/1	Pending	0	0s			
job-sleep-dra-jmjc8	0/1	Pending	0	0s			
job-sleep-dra-jmjc8	0/1	Pending	0	0s			
job-sleep-dra-jmjc8	0/1	Pending	0	5s			
job-sleep-dra-jmjc8	0/1	Pending	0	5s			
job-sleep-dra-jmjc8	0/1	Pending	0/1	0/1	Pending	Pending	0
0	5s						
5s							
job-sleep-dra-jmjc8	0/1	Pending	0	5s			
job-sleep-dra-jmjc8	0/1	Pending	0	5s			
job-sleep-dra-jmjc8	0/1	Pending	0	5s			
job-sleep-dra-jmjc8	0/1	Pending	0	5s			
job-sleep-dra-jmjc8	0/1	ContainerCreating	0	5s			
job-sleep-dra-jmjc8	0/1	ContainerCreating	0	5s			
job-sleep-dra-jmjc8	1/1	Running	0	6s			
job-sleep-dra-jmjc8	1/1	Running	0	6s			

```

[2] /bin/bash 86x21
Every 1.0s: echo 'SLURM PODS'; kubec... notaunicorn: Mon Mar 23 11:17:02 2026

SLURM PODS
No resources found in slurm namespace.

SLURM BRIDGE PODS (1 total, 0 pending)
NAME          READY    STATUS    RESTARTS   AGE   IP          NODE
NOMINATED     NODE    READINESS GATES
job-sleep-dra-jmjc8 1/1      Running   0           51s   10.244.9.7  slurm-bridge-dem
o-worker8      <none>   <none>

PODGROUP STATUS
No resources found in slurm-bridge namespace.

JOB STATUS / JOBSET STATUS
No resources found in slurm-bridge namespace.
NAME          STATUS    COMPLETIONS   DURATION   AGE
job-sleep-dra Running   0/1           51s        51s

SINFO
PARTITION    AVAIL    TIMELIMIT    NODES    STATE    NODELIST

```

```

$ kubectl delete -f hack/examples/dra/gpu-example/
job.yaml
job.batch "job-sleep-dra" deleted from slurm-bridge namespace
$ kubectl delete -f hack/examples/dra/gpu-example/
ob.yaml
job.batch/job-sleep-dra created
$ kubectl get resourceclaim -n slurm-bridge
o v1de
job-sleep-dra-jnjc8-extended-resources-6nflx STATE allocated,reserved AGE 1s
$ kubectl describe -n slurm-bridge resourceclaim job-sleep-dra-jnjc8-extended-resources-6nflx
Name: job-sleep-dra-jnjc8-extended-resources-6nflx
Namespace: slurm-bridge
Labels: <none>
Annotations: resource.kubernetes.io/extended-resource-claim: true
API Version: resource.k8s.io/v1
Kind: ResourceClaim
Metadata:
  Creation Timestamp: 2026-03-23T10:16:16Z
  Generate Name: job-sleep-dra-jnjc8-extended-resources-
  Owner References:
    API Version: v1
    Block Owner Deletion: true
    Controller: true
    Kind: Pod
    Name: job-sleep-dra-jnjc8
    UID: a6a652ac-7c01-4b14-b7d8-690b96f0f8f
  Resource Version: 7300
  UID: 4151c802-5656-41d5-90fa-92ebfad52737
Spec:
  Devices:
    Requests:
      Exactly:
        Allocation Mode: ExactCount
        Count: 16
        Device Class Name: dra.cpu
        Selectors:
          Cel:
            Expression: device.attributes['dra.cpu'].cpuID in [0,1,2,3,4,5,6,7,8,9,10,11,12,13,14,15]
      Name:
    Exactly:
        Allocation Mode: ExactCount
        Count: 9
        Device Class Name: gpu.example.com
        Selectors:
          Cel:
            Expression: device.attributes['gpu.example.com'].index in [0,1,2,3,4,5,6,7,8]
      Name:
    Allocation:
        Allocation Timestamp: 2026-03-23T10:16:16Z
        Devices:
        Results:

```

Demo



CloudNativeCon

Europe 2026

```
/bin/bash 108x26
```

job-sleep-dra-wk8850	0/1	Completed		job-sleep-dra-wk8850	0/174s
Completed	0	74s			
job-sleep-dra-jmjc8	0/1	Pending	0	0s	
job-sleep-dra-jmjc8	0/1	Pending	0	0s	
job-sleep-dra-jmjc8	0/1	Pending	0	job-sleep-dra-jmjc80s	
0/1 Pending	0	0s			
job-sleep-dra-jmjc8	0/1	Pending	0	0s	
job-sleep-dra-jmjc8	0/1	Pending	0	0s	
job-sleep-dra-jmjc8	0/1	Pending	0	0s	
job-sleep-dra-jmjc8	0/1	Pending	0	0s	
job-sleep-dra-jmjc8	0/1	Pending	0	0s	
job-sleep-dra-jmjc8	0/1	Pending	0	5s	
job-sleep-dra-jmjc8	0/1	Pending	0	5s	
job-sleep-dra-jmjc8	0/1	Pending	0	0/1Pending	Pending
0/1 Pending	0	0s			0
job-sleep-dra-jmjc8	0/1	Pending	0	5s	
job-sleep-dra-jmjc8	0/1	Pending	0	5s	
job-sleep-dra-jmjc8	0/1	Pending	0	5s	
job-sleep-dra-jmjc8	0/1	Pending	0	5s	
job-sleep-dra-jmjc8	0/1	ContainerCreating	0	5s	
job-sleep-dra-jmjc8	0/1	ContainerCreating	0	5s	
job-sleep-dra-jmjc8	1/1	Running	0	6s	
job-sleep-dra-jmjc8	1/1	Running	0	6s	

```
/bin/bash 86x21
```

```
Every 1.0s:      echo 'SLURM PODS';      kubec...  notaunicorn: Mon Mar 23 11:17:23 2026
```

SLURM PODS

```
SLURM BRIDGE PODS (1 total, 0 pending)
```

NAME	READY	STATUS	RESTARTS	AGE	IP	NODE
NOMINATED	NODE	READINESS	GATES			
job-sleep-dra-jnjc8	1/1	Running	0	72s	10.244.9.7	slurm-bridge-dem
o-worker8	<none>	<none>				

PODGROUP STATUS

```
No resources found in slurm-bridge namespace.
```

JOB STATUS / JOBSET STATUS

```
No resources found in slurm-bridge namespace.
```

SINFO

PARTITION	AVAIL	TIMELIMIT	NODES	STATE	NODELIST
-----------	-------	-----------	-------	-------	----------

```

Name:          gpu

```

```

Status:
Allocation:
Allocation Timestamp: 2026-03-23T10:16:16Z
Devices:
Results:
Device: cpudev00
Driver: dra.cpu
Pool: slurm-bridge-demo-worker8
Request: cpu
Device: cpudev001
Driver: dra.cpu
Pool: slurm-bridge-demo-worker8
Request: cpu
Device: cpudev002
Driver: dra.cpu
Pool: slurm-bridge-demo-worker8
Request: cpu
Device: cpudev003
Driver: dra.cpu
Pool: slurm-bridge-demo-worker8
Request: cpu
Device: cpudev004
Driver: dra.cpu
Pool: slurm-bridge-demo-worker8
Request: cpu
Device: cpudev005
Driver: dra.cpu
Pool: slurm-bridge-demo-worker8
Request: cpu
Device: cpudev006
Driver: dra.cpu
Pool: slurm-bridge-demo-worker8
Request: cpu
Device: cpudev007
Driver: dra.cpu
Pool: slurm-bridge-demo-worker8
Request: cpu
Device: cpudev008
Driver: dra.cpu
Pool: slurm-bridge-demo-worker8
Request: cpu
Device: cpudev009
Driver: dra.cpu
Pool: slurm-bridge-demo-worker8
Request: cpu
Device: cpudev010
Driver: dra.cpu
Pool: slurm-bridge-demo-worker8
Request: cpu
Device: cpudev011
Driver: dra.cpu
Pool: slurm-bridge-demo-worker8
Request: cpu

```

Demo



Europe 2026

IP	/bin/bash 108x26				
job-sleep-dra-wk885	0/1	Completed	job-sleep-dra-wk885@	0/174s	
Completed	0	74s			
job-sleep-dra-jnjc8	0/1	Pending	0	0s	
job-sleep-dra-jnjc8	0/1	Pending	0	0s	
job-sleep-dra-jnjc8	0/1	Pending	0	job-sleep-dra-jnjc8@	
0/1	Pending	0	0s		
job-sleep-dra-jnjc8	0/1	Pending	0	0s	
job-sleep-dra-jnjc8	0/1	Pending	0	0s	
job-sleep-dra-jnjc8	0/1	Pending	0	0s	
job-sleep-dra-jnjc8	0/1	Pending	0	0s	
job-sleep-dra-jnjc8	0/1	Pending	0	0s	
job-sleep-dra-jnjc8	0/1	Pending	0	0s	
job-sleep-dra-jnjc8	0/1	Pending	0	5s	
job-sleep-dra-jnjc8	0/1	Pending	0	5s	
job-sleep-dra-jnjc8	0/1	Pending	0/1	Pending	Pending 0
0	5s				
5s					
job-sleep-dra-jnjc8	0/1	Pending	0	5s	
job-sleep-dra-jnjc8	0/1	Pending	0	5s	
job-sleep-dra-jnjc8	0/1	Pending	0	5s	
job-sleep-dra-jnjc8	0/1	Pending	0	5s	
job-sleep-dra-jnjc8	0/1	ContainerCreating	0	5s	
job-sleep-dra-jnjc8	0/1	ContainerCreating	0	5s	
job-sleep-dra-jnjc8	1/1	Running	0	6s	
job-sleep-dra-jnjc8	1/1	Running	0	6s	

```

/bin/bash 86x21
Every 1.0s: echo 'SLURM PODS'; kubectl exec -n slurm-bridge pod/slurm-bridge-dem -- sh -c 'cat /etc/passwd'
SLURM PODS
No resources found in slurm namespace.

SLURM BRIDGE PODS (1 total, 0 pending)
NAME                                READY   STATUS    RESTARTS   AGE   IP              NODE
job-sleep-dra-jmjc8                1/1    Running   0           87s   10.244.9.7      slurm-bridge-dem
o-worker8                           <none>   <none>    <none>      <none>

PODGROUP STATUS
No resources found in slurm-bridge namespace.

JOB STATUS / JOBSET STATUS
No resources found in slurm-bridge namespace.
NAME                                STATUS    COMPLETIONS   DURATION   AGE
job-sleep-dra-jmjc8                Running   0/1           88s        88s

SINFO
PARTITION  AVAIL  TIMELIMIT  NODES  STATE MODELIST

```

```

/bin/bash 110x53
Driver: dra.cpu
Pool: slurm-bridge-demo-worker8
Request: cpu
Device: gpu-0
Driver: gpu.example.com
Pool: slurm-bridge-demo-worker8
Request: gpu
Device: gpu-1
Driver: gpu.example.com
Pool: slurm-bridge-demo-worker8
Request: gpu
Device: gpu-2
Driver: gpu.example.com
Pool: slurm-bridge-demo-worker8
Request: gpu
Device: gpu-3
Driver: gpu.example.com
Pool: slurm-bridge-demo-worker8
Request: gpu
Device: gpu-4
Driver: gpu.example.com
Pool: slurm-bridge-demo-worker8
Request: gpu
Device: gpu-5
Driver: gpu.example.com
Pool: slurm-bridge-demo-worker8
Request: gpu
Device: gpu-6
Driver: gpu.example.com
Pool: slurm-bridge-demo-worker8
Request: gpu
Device: gpu-7
Driver: gpu.example.com
Pool: slurm-bridge-demo-worker8
Request: gpu
Device: gpu-8
Driver: gpu.example.com
Pool: slurm-bridge-demo-worker8
Request: gpu

Node Selector:
Node Selector Terms:
Match Fields:
Key: metadata.name
Operator: In
Values:
slurm-bridge-demo-worker8

Reserved For:
Name: job-sleep-dra-jnjc8
Resource: pods
UID: a6a652ac-7c01-4b14-b7d8-6909b96f0f8f

Events:
<none>

~ mallow/fix-demo [2] U:1 T:1 ~/repos/slurm-bridge kubectrl get resourceclaim -n slurm-bridge
-o wide
```



KubeCon



CloudNativeCon

Europe 2026

Extra

Slurm <-> K8S Resources Translation



KubeCon



CloudNativeCon

Europe 2026

- Slurm tracks hardware using the Generic Resource (GRES) model
- The Slurm-Bridge handles Kubernetes Pod scheduling, allows Slurm to make the complex topological decisions using GRES, and then dynamically injects those decisions back into Kubernetes using DRA **ResourceClaims**.
 - Translates resource request in the pod spec into a Slurm placeholder job.
 - Map slurm's allocation into Kubernetes using Dynamic Resource Allocation (DRA) **ResourceClaims**
 - Uses Common Expression Language (CEL) selectors in **ResourceClaims** to request the exact resources that slurm choose.
 - On the node, the GPU and the CPU DRA driver simply honors that claim.

DRA CPU Driver - Architecture

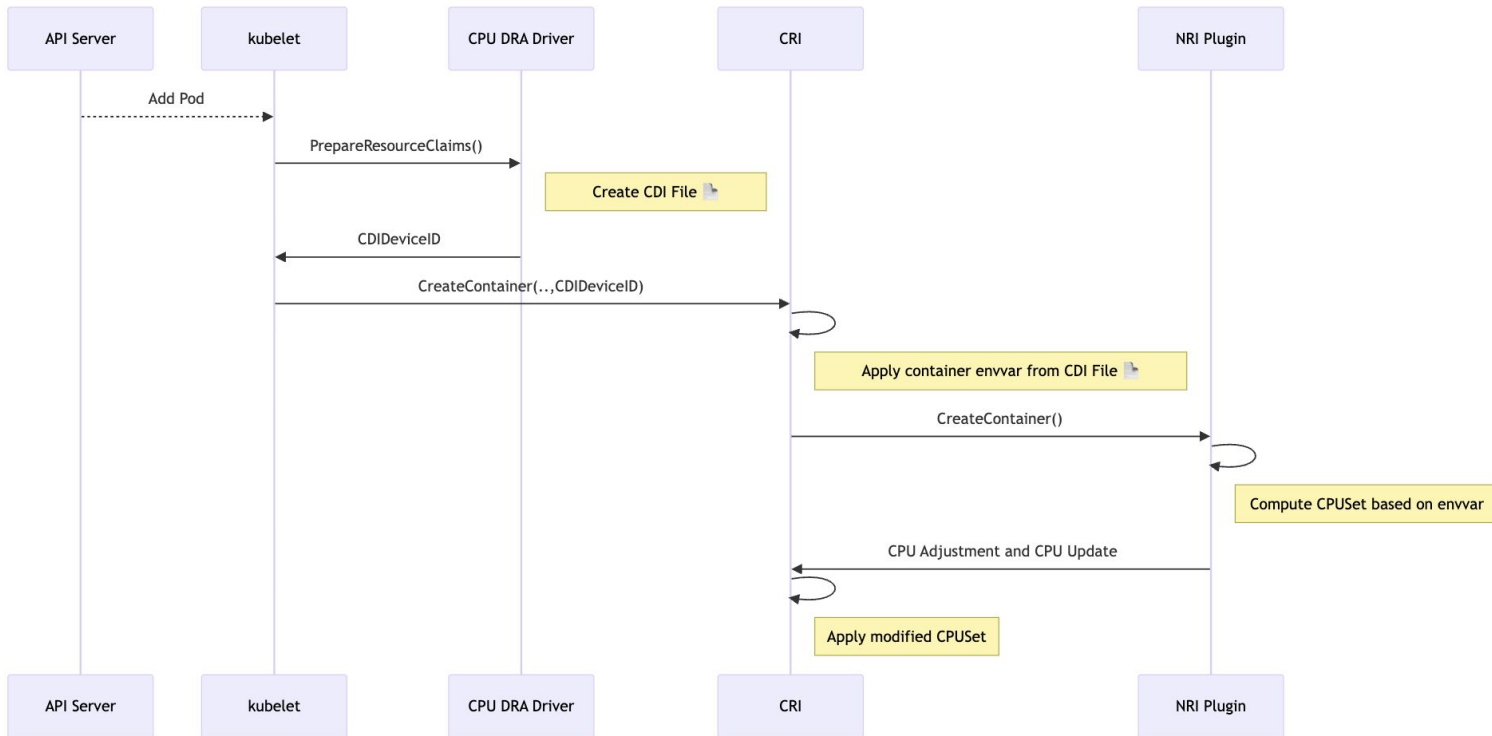


KubeCon



CloudNativeCon

Europe 2026



Workload Configuration Requirements



KubeCon



CloudNativeCon

Europe 2026

- Option 1

```
# Example: Pod Level Resources
spec:
  resources: # Pod Level Resources
    requests:
      cpu: "16" # 10 (exclusive cpu's for claim1) + 4 (exclusive cpu's for claim2) + 2 (sha
    limits:
      cpu: "16"
  containers:
    - name: main-app
      image: ...
      resources:
        # Omit CPU requests/limits, or set both to 10
        claims:
          - name: claim1
    - name: worker
      image: ...
      resources:
        # Omit CPU requests/limits, or set both to 4
        claims:
          - name: claim2
    - name: sidecar1
      image: ...
      # Omit CPU resources, or ensure the combined requests/limits for sidecar1 and sidecar
    - name: sidecar2
      image: ...
      # Omit CPU resources, or ensure the combined requests/limits for sidecar1 and sidecar
  resourceClaims:
    - name: claim1
      resourceClaimName: cpu-claim-10 # Requests 10 CPUs
    - name: claim2
      resourceClaimName: cpu-claim-4 # Requests 4 CPUs
```

- Option 2

```
# Example: Container Level Mirroring
spec:
  containers:
    - name: my-container
      image: ...
      resources:
        requests:
          cpu: "10" # Must match the CPU count in "claim1"
        limits:
          cpu: "10" # Must match the CPU count in "claim1"
        claims:
          - name: claim1
      resourceClaims:
        - name: claim1
          resourceClaimName: cpu-claim-10 # Requests 10 CPUs
```