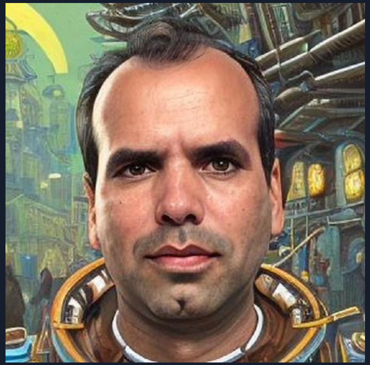
Explain
How Kubernetes Works with GPUs
Like I’m 5

Carlos Santana

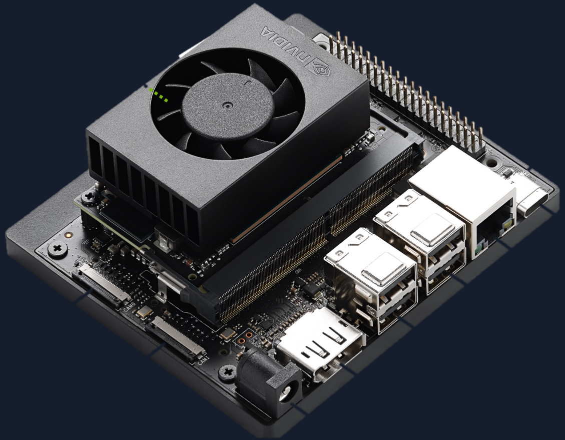
Sr. WW Spec. SA, Containers at AWS
CNCF Ambassador

Learning at home

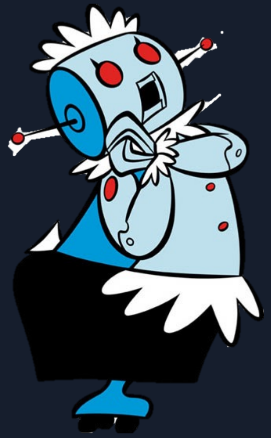
kubernetes + =



k8s
experience



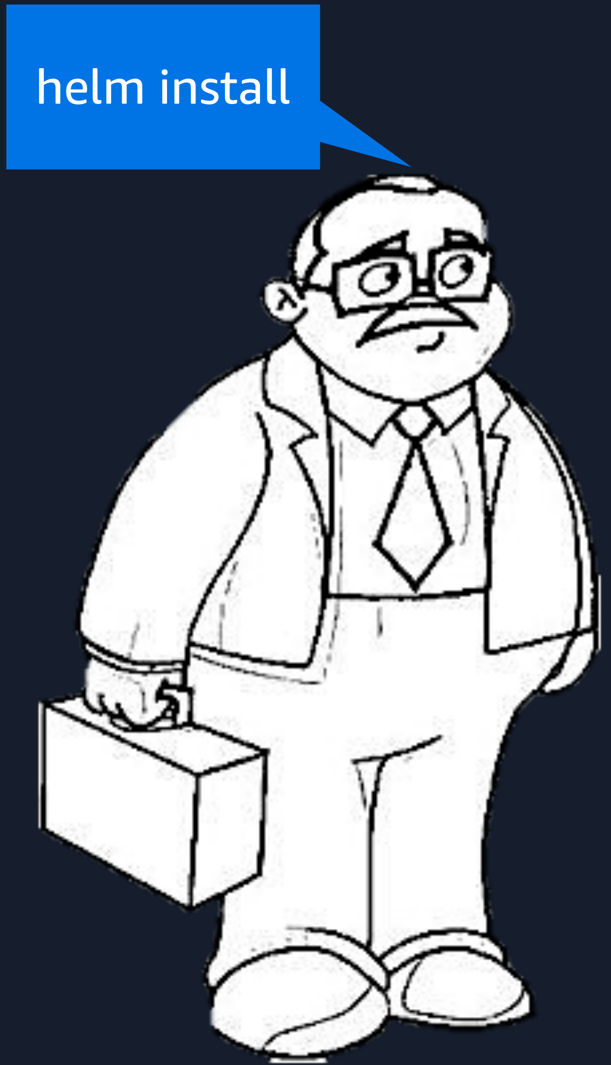
NVIDIA
Jetson GPU



Generative
AI & ML

Images:
https://warnerbros.fandom.com/wiki/Rosey
https://developer.nvidia.com/embedded/learn/get-started-jetson-orin-nano-devkit

Learn and Be Curious



CUDA

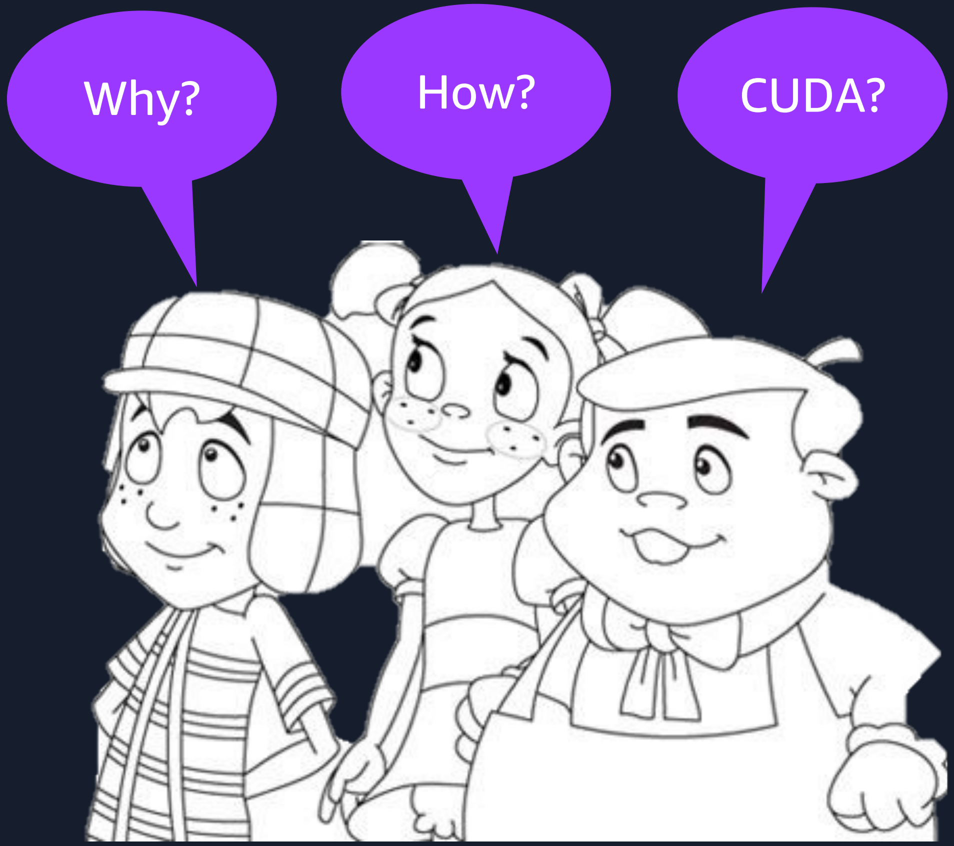
Container Toolkit

Device Driver

Device Plugin

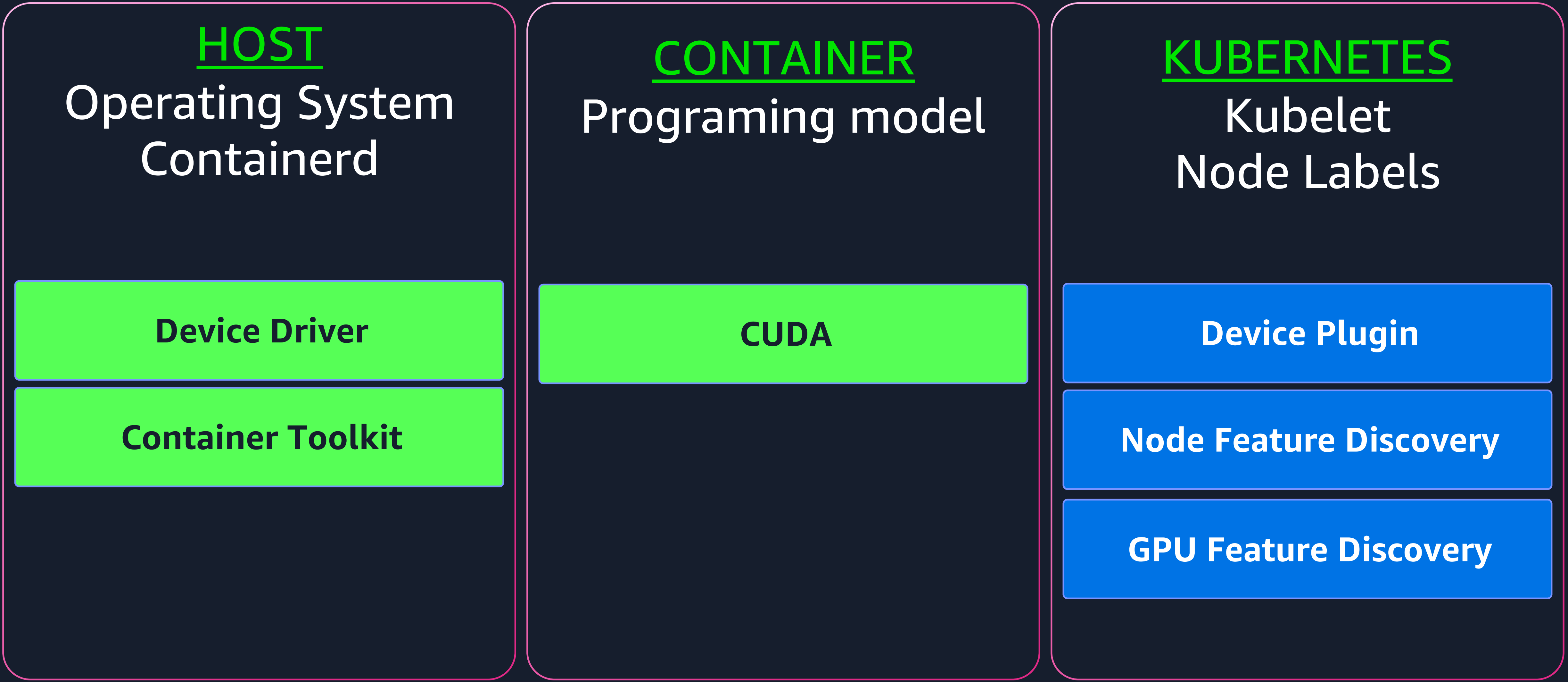
GPU Feature Discovery

Node Feature Discovery



Images:
https://www.pinterest.com/pin/879116789737240075/
https://www.pinterest.com/pin/433401164162584656/

Main areas



HOST

Operating System
Containerd

- Device Driver
- Container Toolkit

CONTAINER

Programing model

- CUDA

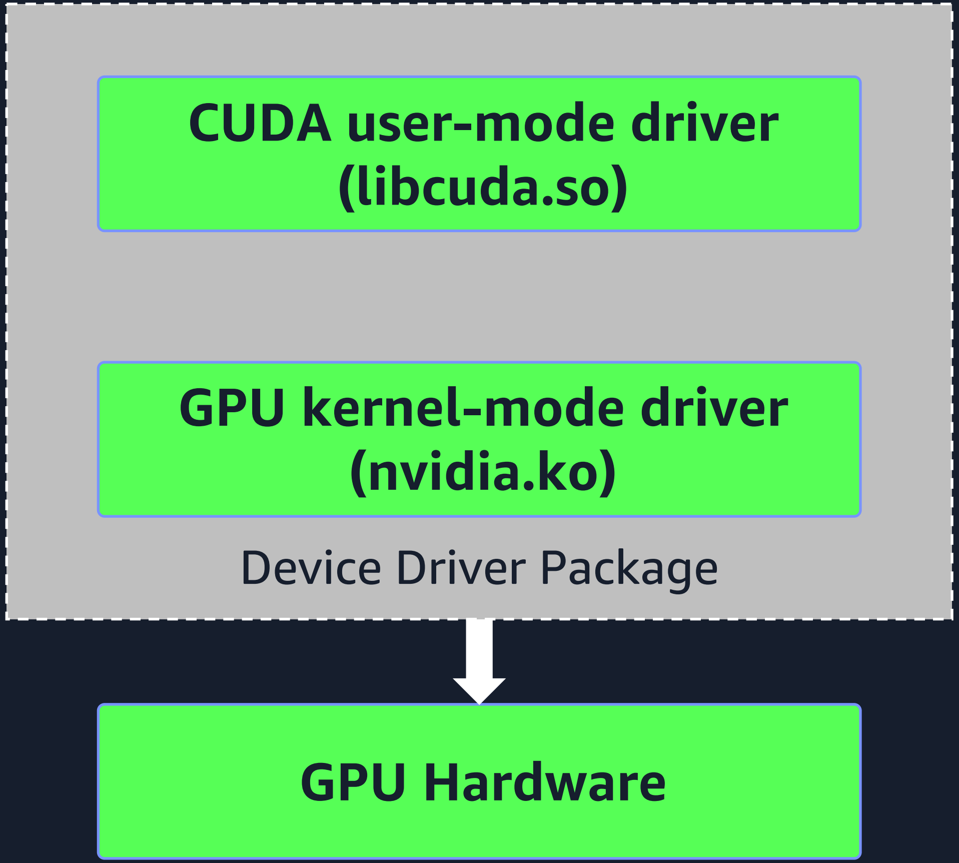
KUBERNETES

Kubelet
Node Labels

- Device Plugin
- Node Feature Discovery
- GPU Feature Discovery

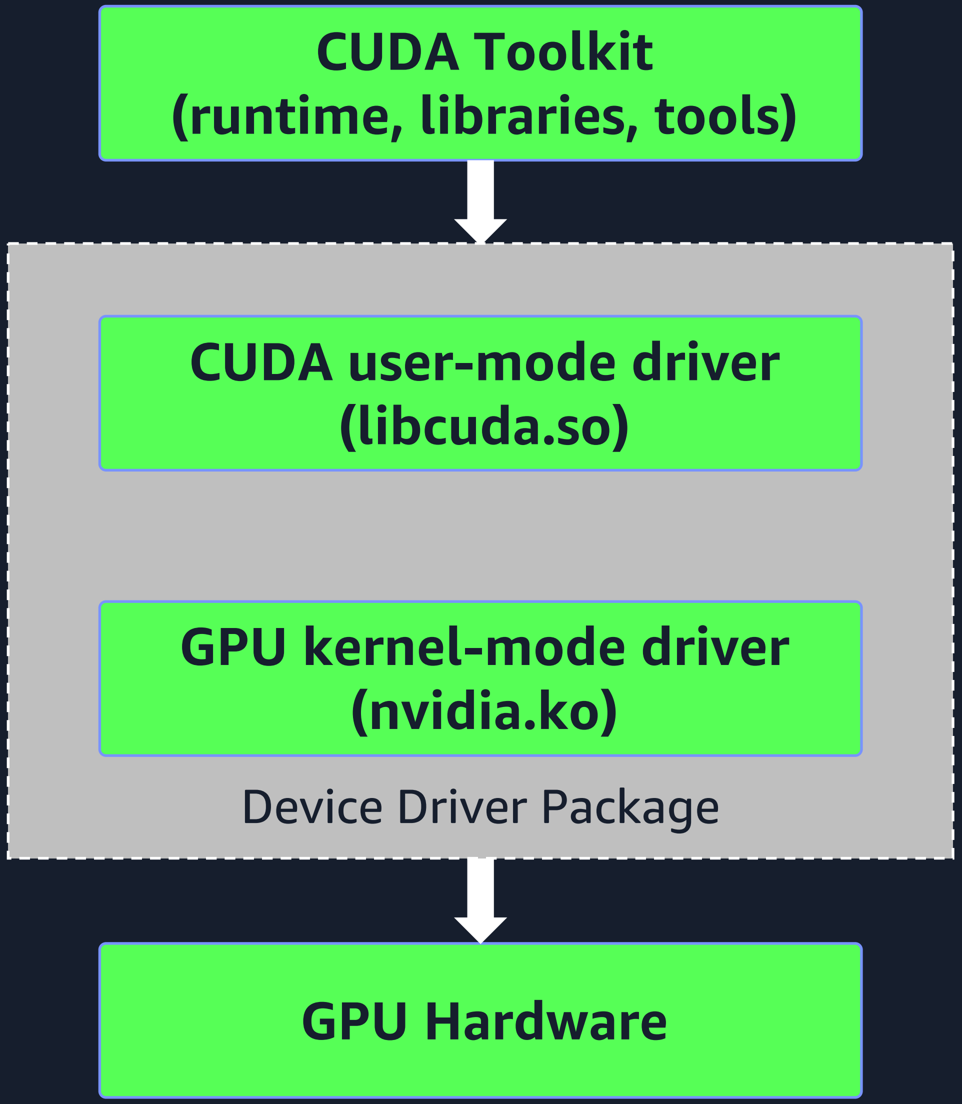
Device Driver

- Implements the CUDA Driver API
- Use Vendor pre-installed drivers
- NVIDIA Jetson JetPack OS
- Cloud Providers include driver in OS images
  - Example:
    - AWS EKS AIM EKS AL23, BR, Auto Mode
- Driver package include CUDA user mode driver
- Do not use GPU Operator unless you know why


CUDA user-mode driver (libcuda.so)
GPU kernel-mode driver (nvidia.ko)
Device Driver Package
GPU Hardware

CUDA Toolkit

- Toolkit at build time necessary for linking
- Compatibility between CUDA and Driver
- Container Image usually comes with CUDA runtime
- Do not mount toolkit runtime from host path, unless you know why



Driver and CUDA versions

nvidia-smi for PCI GPUs
On Jetson dGPU:
/proc/driver/nvidia/version

```
$ cat /proc/driver/nvidia/version
VRM version: NVIDIA UNIX Open Kernel
Module for aarch64  540.3.0

$ head -n 1 /etc/nv_tegra_release
# R36 (release), REVISION: 3.0 (aka 36.3)
```

nvcc (CUDA compiler)
ldconfig

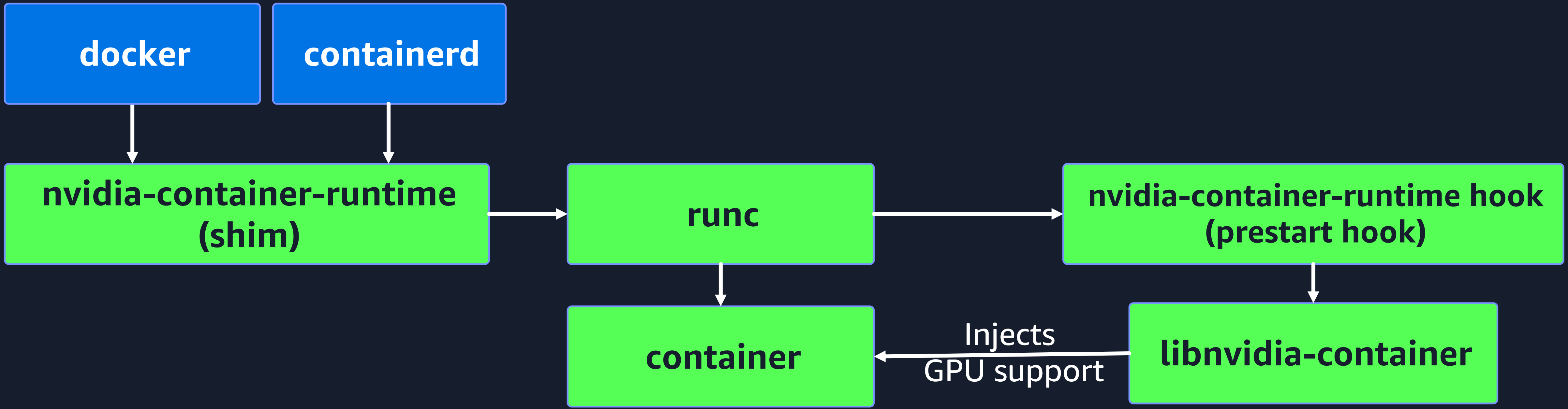
```
$ nvcc --version
Cuda compilation tools, release 12.2, V12.2.140

$ ldconfig –p | grep libcudart
libcudart.so.12 (libc6,AArch64)
```

CUDA Toolkit	Minimum Required Driver Version
CUDA 12.x	>=525.60.13
CUDA 11.x	>=450.80.02

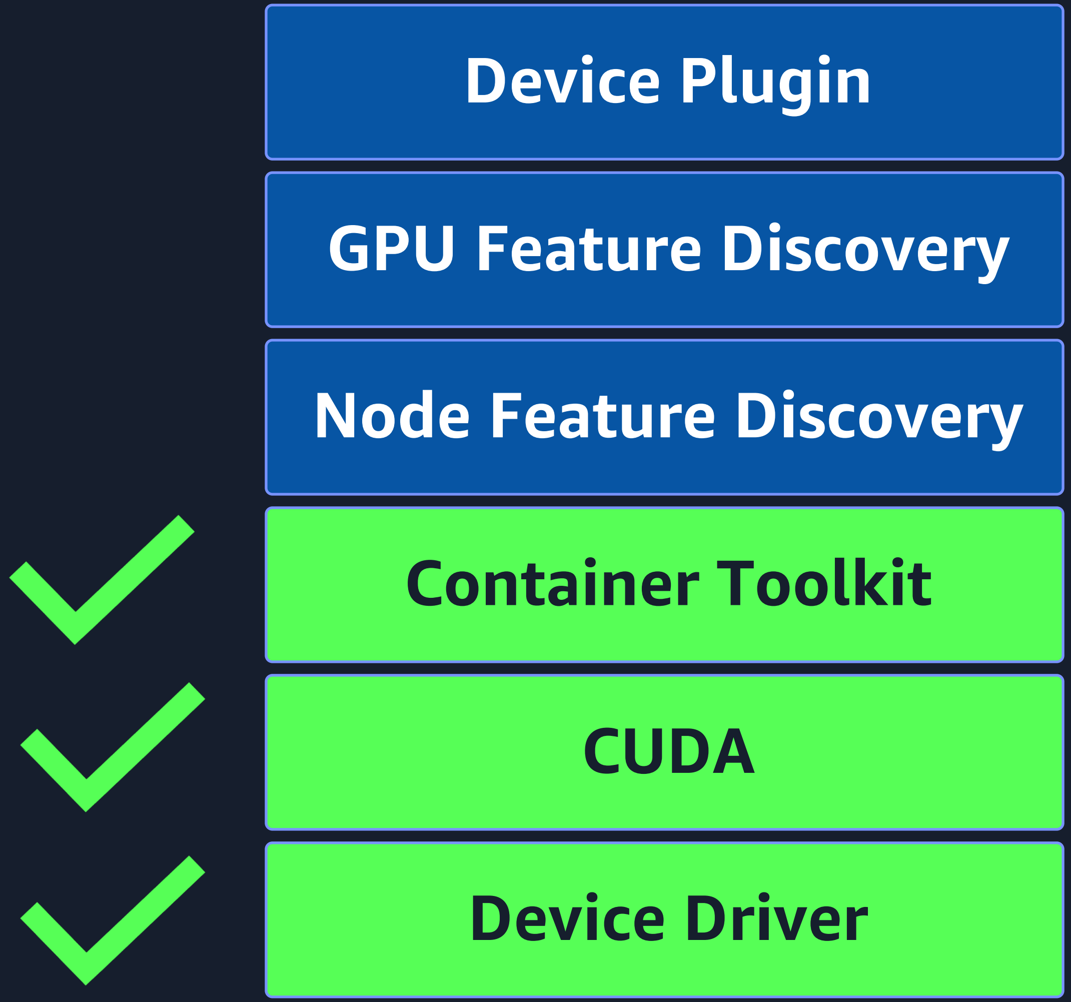
Container Toolkit

- Run GPU-accelerated containers
- Cloud Providers includes this
- For Jetson you need to install docker engine (containerd) and install the toolkit

```
$ nvidia-ctk runtime configure  --runtime=containerd
$ cat /etc/containerd/config.toml
default_runtime_name = "nvidia"
[plugins."io.containerd.grpc.v1.cri".containerd.runtimes.nvidia.options]
  BinaryName = "/usr/bin/nvidia-container-runtime"
```


docker
containerd
nvidia-container-runtime (shim)
runc
nvidia-container-runtime hook (prestart hook)
container
Injects GPU support
libnvidia-container

GPU Containers



Device Plugin

GPU Feature Discovery

Node Feature Discovery

✓ Container Toolkit

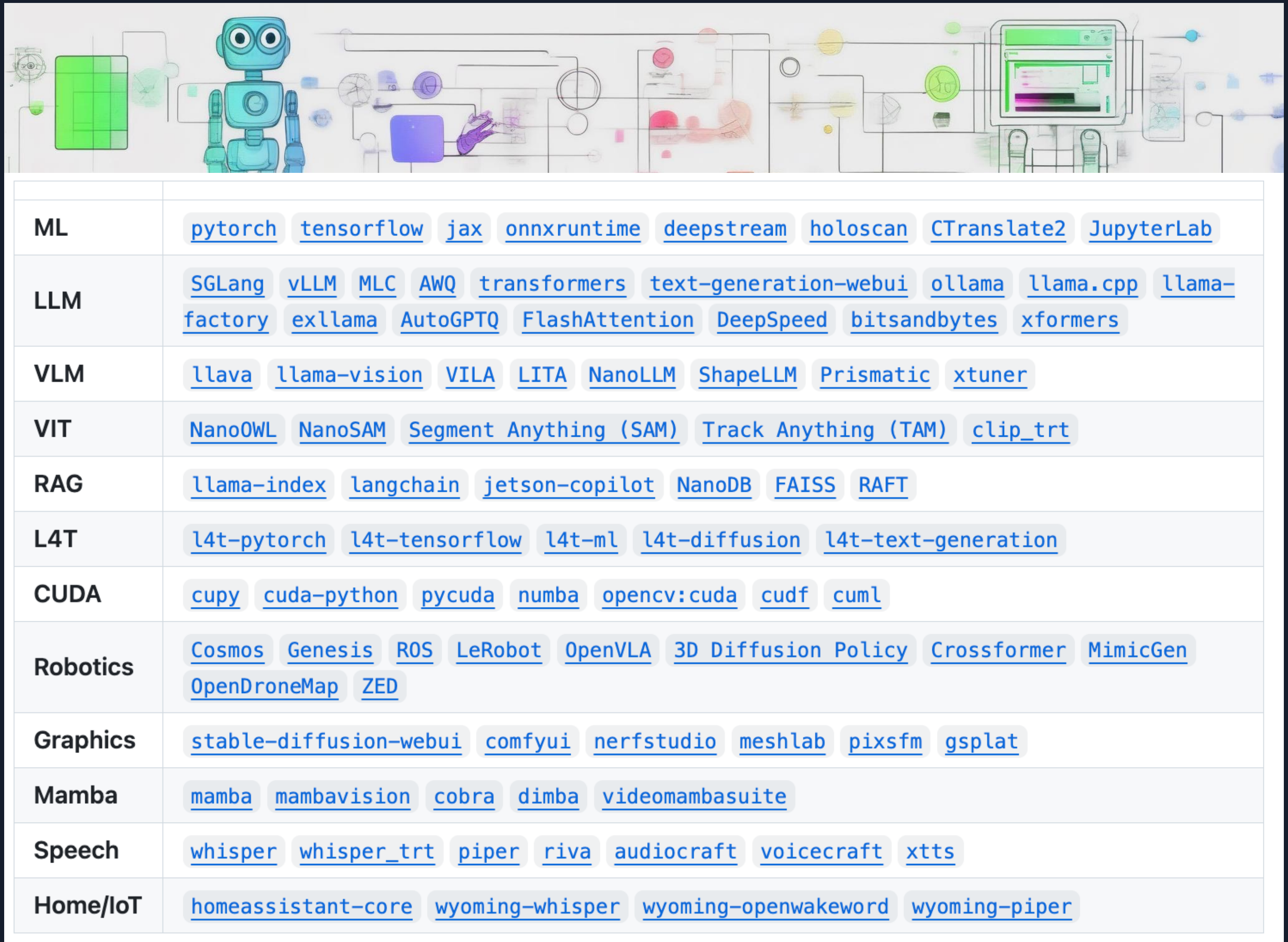
✓ CUDA

✓ Device Driver

```
$ docker run --runtime nvidia --gpus all
```

Jetson Containers

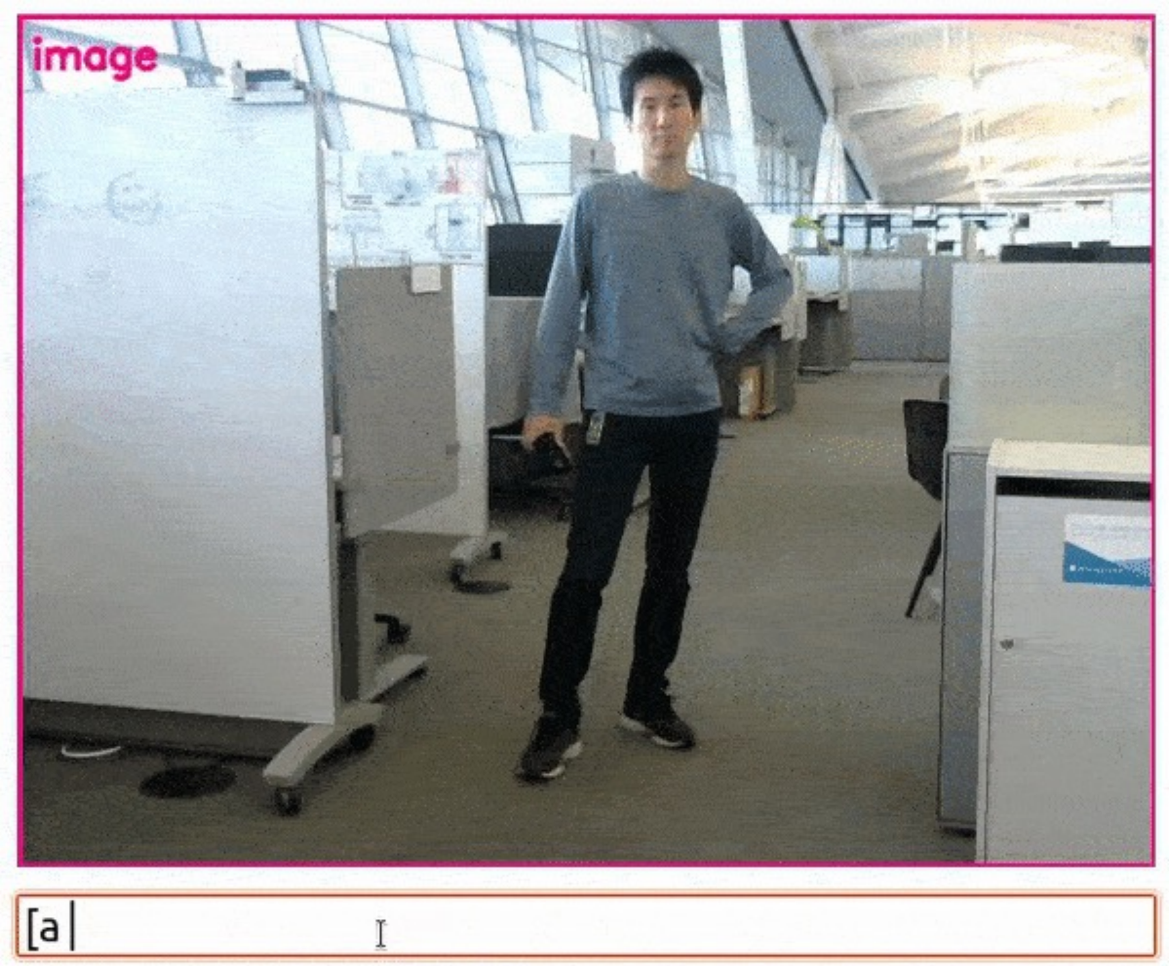
https://github.com/dusty-nv/jetson-containers



ML	pytorch tensorflow jax onnxruntime deepstream holoscan CTranslate2 JupyterLab
LLM	SGLang vLLM MLC AWQ transformers text-generation-webui ollama llama.cpp llama-factory exllama AutoGPTQ FlashAttention DeepSpeed bitsandbytes xformers
VLM	llava llama-vision VILA LITA NanoLLM ShapeLLM Prismatic xtuner
VIT	NanoOWL NanoSAM Segment Anything (SAM) Track Anything (TAM) clip_trt
RAG	llama-index langchain jetson-copilot NanoDB FAISS RAFT
L4T	l4t-pytorch l4t-tensorflow l4t-ml l4t-diffusion l4t-text-generation
CUDA	cupy cuda-python pycuda numba opencv:cuda cudf cuml
Robotics	Cosmos Genesis ROS LeRobot OpenVLA 3D Diffusion Policy Crossformer MimicGen OpenDroneMap ZED
Graphics	stable-diffusion-webui comfyui nerfstudio meshlab pixsfm gsplat
Mamba	mamba mambavision cobra dimba videomambasuite
Speech	whisper whisper_trt piper riva audiocraft voicecraft xtts
Home/IoT	homeassistant-core wyoming-whisper wyoming-openwakeword wyoming-piper

Jetson Containers

- Modular container build system
- Latest AI/ML packages
- Wraps docker run --runtime=nvidia
- Discord and community meetings



Jetson Containers

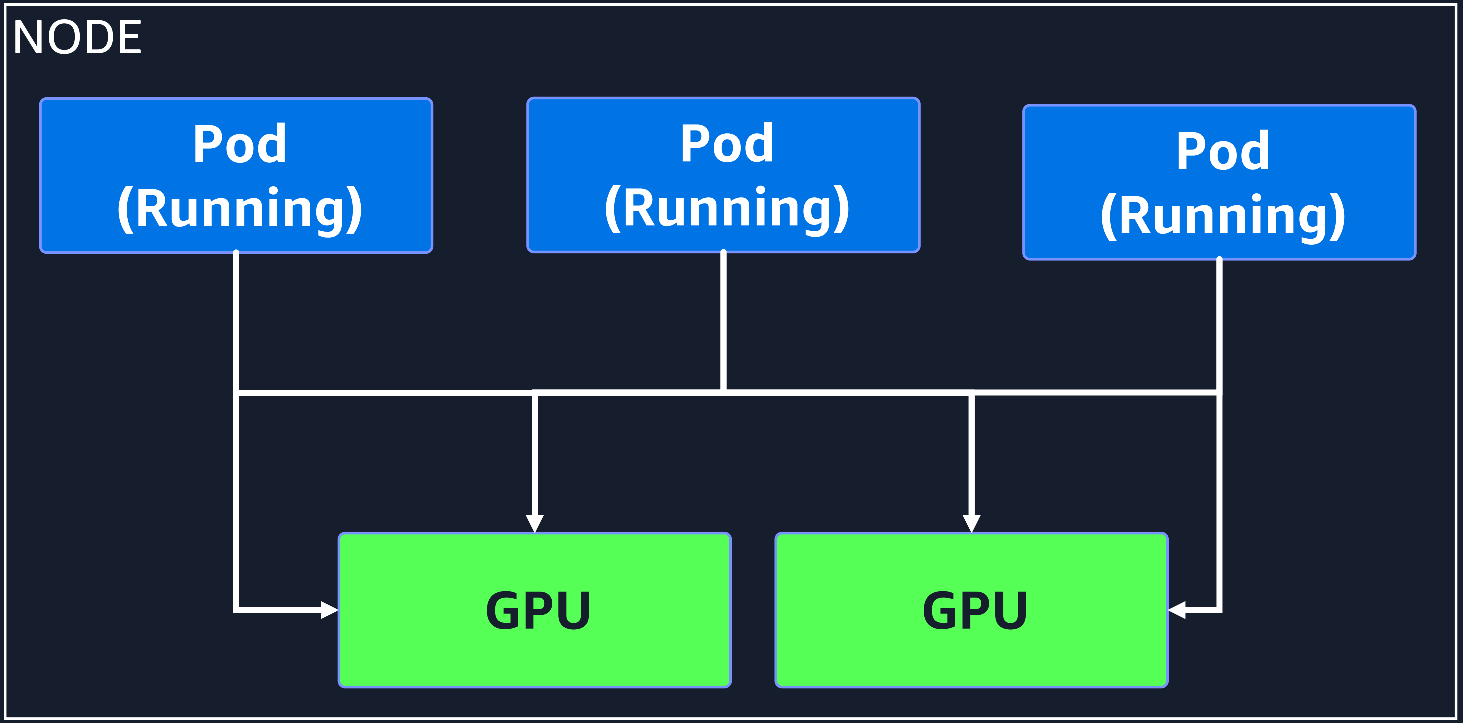
```
$ jetson-containers run --name ollama $(autotag ollama)

>>> How many kubernetes administrators takes to deploy a
cluster?


In a typical scenario, a small team of 2-3 administrators may be
sufficient for deploying and managing a Kubernetes cluster in a
simple environment.
However, larger organizations or complex deployments could
require teams of 5-10 or more administrators to ensure proper
management and scalability.
```

Kubernetes

- Device Plugin required?
  - No scheduling
  - There is no allocations
  - All Pods access all GPUs

```
$ kubectl run myllm --image myllmImage
```

Without allocatable nvidia.com/gpu



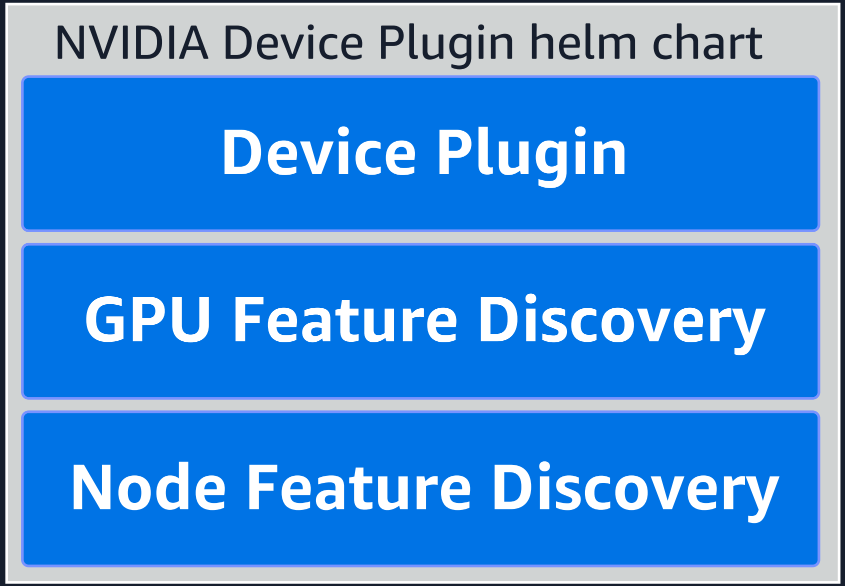
✅ Container Toolkit

✅ CUDA

✅ Device Driver

Kubernetes + GPU Device Plugin

- NVIDIA Device Plugin
  - Helm Chart
- Schedule Pods
- Allocate GPUs
- Share a GPU
  - Time-slicing
  - MPS
  - MIG


NVIDIA Device Plugin helm chart

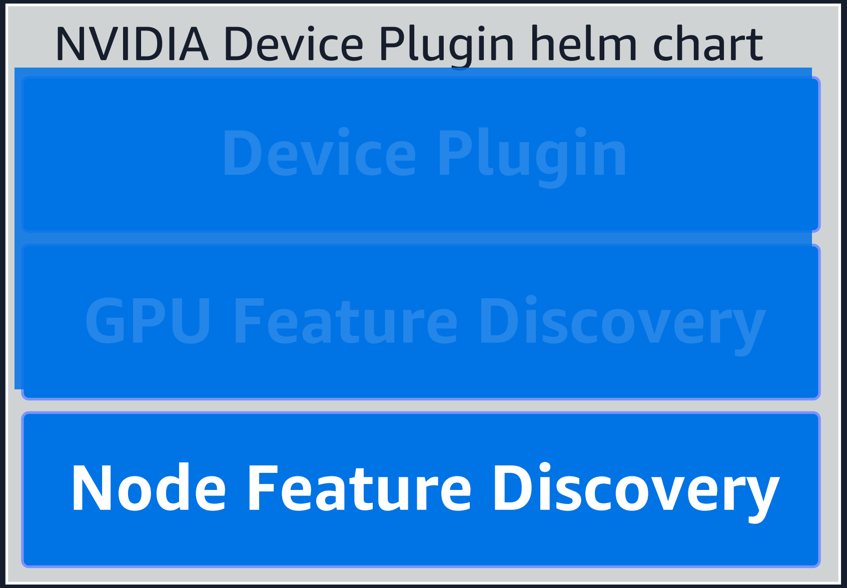
Device Plugin

GPU Feature Discovery

Node Feature Discovery

Node Feature Discovery (NFD)

- Advertise features using labels
- Detects hardware on node
- CPU, Kernel, OS, PCI
- PCI VENDOR NVIDIA (10ed)

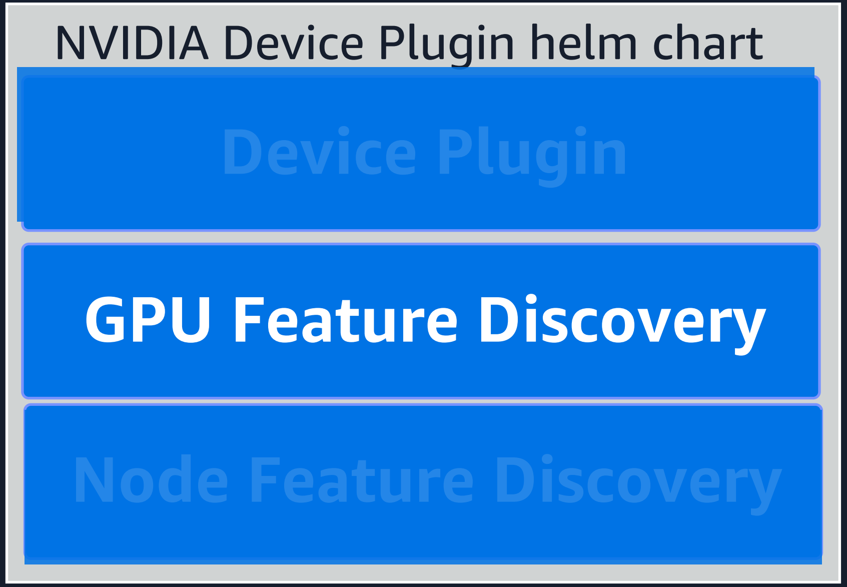


```
$ kubectl get no jetson -o json

  "cpu-cpuid.USCAT": "true",
  "cpu-model.vendor_id": "ARM",
  "kernel-config.PREEMPT": "true",
  "kernel-version.full": "5.15.136-tegra",
  "storage-nonrotationaldisk": "true",
  "system-os_release.ID": "ubuntu",
  "system-os_release.VERSION_ID": "22.04",

  ”pci-10ed.present": "true",
```

Enables the GPU Feature Discovery →

GPU Feature Discovery (GFD)

- Advertise features using labels
- Detects NVIDIA GPU
- Labels with nvidia.com/*



```
$ kubectl get no jetson -o json

"nvidia.com/cuda.driver-version.full": ”540.3.0",
"nvidia.com/cuda.runtime-version.full": "12.2",
"nvidia.com/gpu.count": "1",
"nvidia.com/gpu.replicas": "4",
"nvidia.com/gpu.product": "Orin-SHARED",
"nvidia.com/gpu.sharing-strategy": "time-slicing",
"nvidia.com/mig.capable": "false",
"nvidia.com/mps.capable": "false",
"nvidia.com/vgpu.present": "false"

"nvidia.com/gpu.present": "true",
```

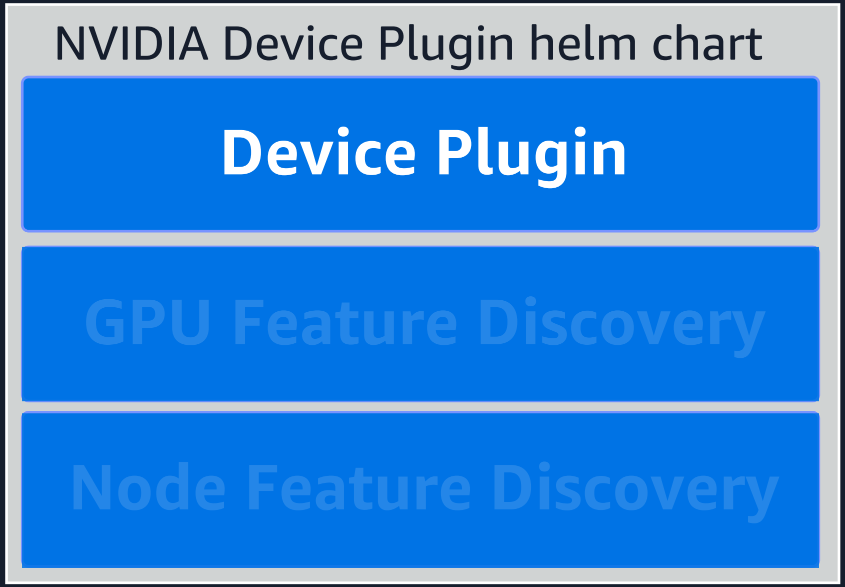
Enables the Device Plugin deamon → **"nvidia.com/gpu.present": "true",**

Device Plugin

- Kubelet Device Plugin API
- Expose the number of GPUs
- Tracks health of the GPUs

Kubelet patches this status

```
k get nodes jetson -o json | jq .status.allocatable
{
  "cpu": "8",
  ”memory": "16032384Ki",
  "nvidia.com/gpu": "4",
  "pods": "110"
}
```



NVIDIA Device Plugin helm chart

Device Plugin

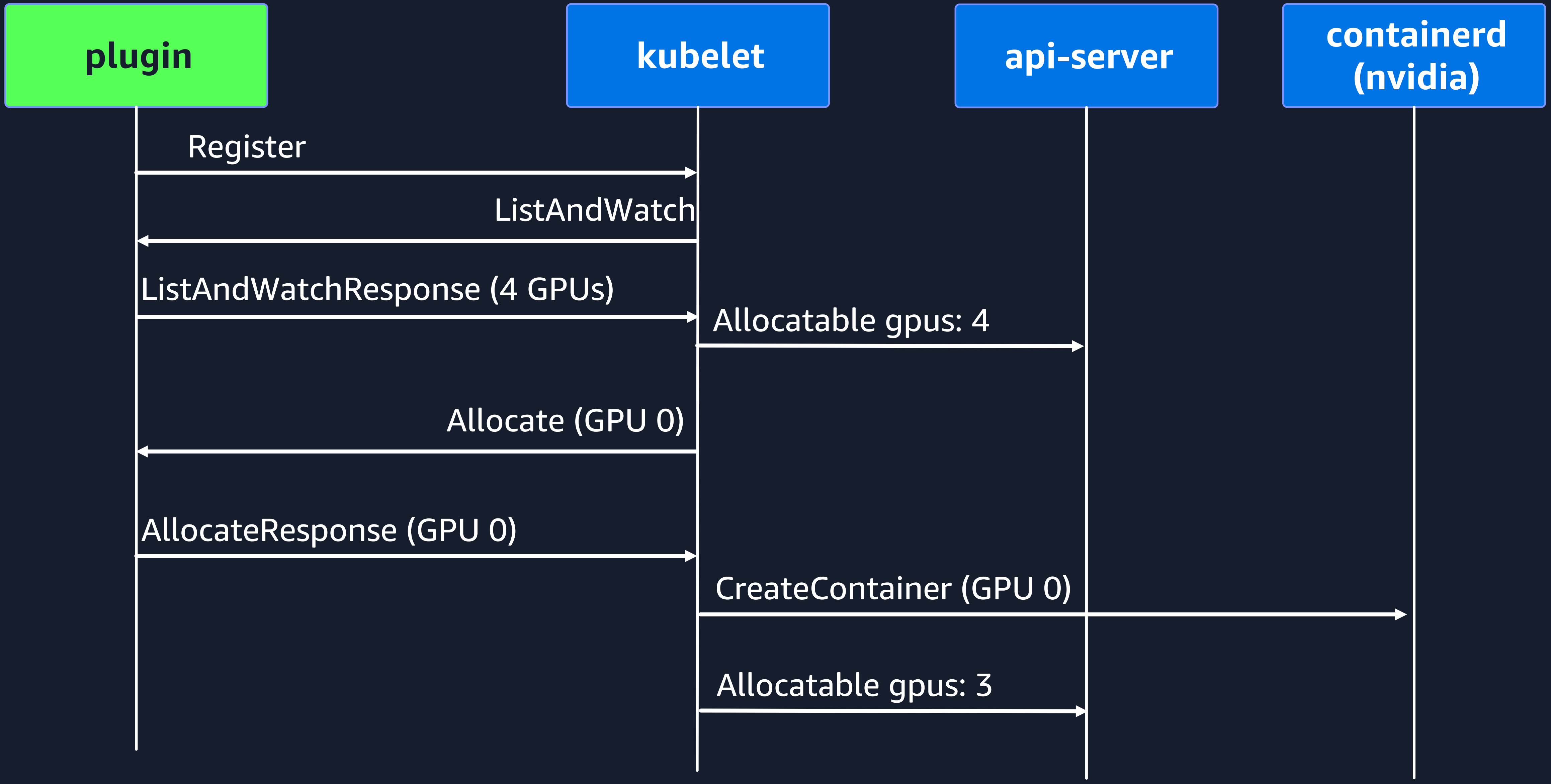
GPU Feature Discovery

Node Feature Discovery

`limit` instead of `request`

```
containers:
- name: cuda-container
  image: nvcr.io/nvidia/k8s/cuda-sample
  resources:
    limits:
      nvidia.com/gpu: 1 # requesting 1 GPU
```

Device Plugin



plugin

kubelet

api-server

containerd (nvidia)

Register

ListAndWatch

ListAndWatchResponse (4 GPUs)

Allocatable gpus: 4

Allocate (GPU 0)

AllocateResponse (GPU 0)

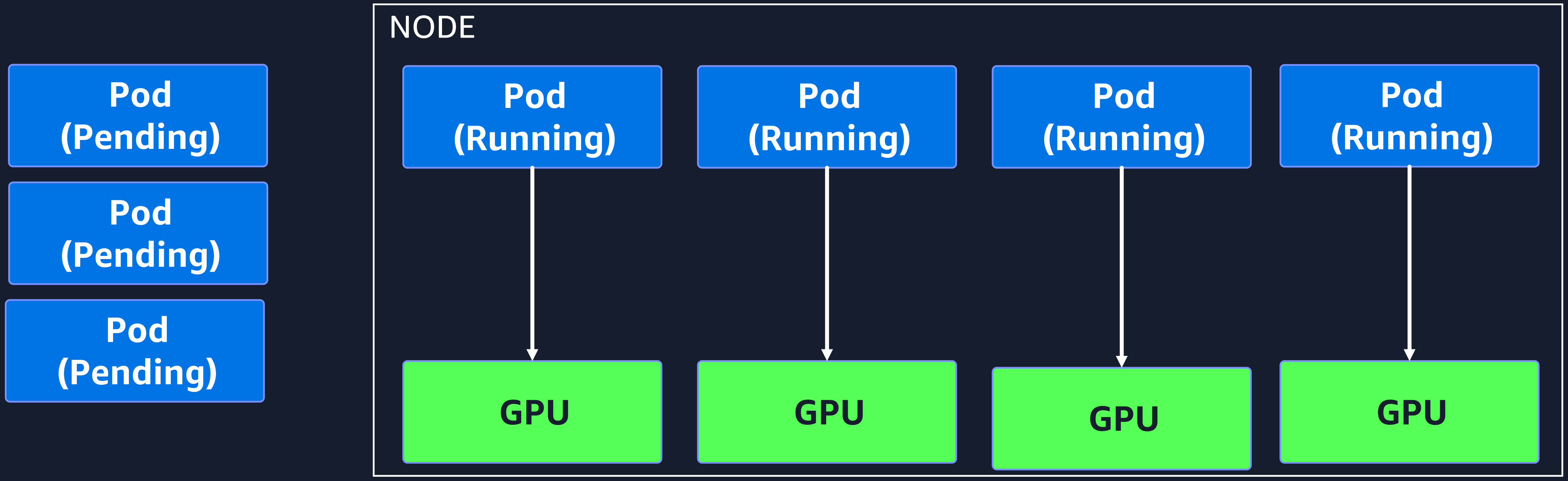
CreateContainer (GPU 0)

Allocatable gpus: 3

Scheduling Pods

- Allocatable status on Node
- Pods states how much GPU they need
- As Pod wait for GPU, they are pending

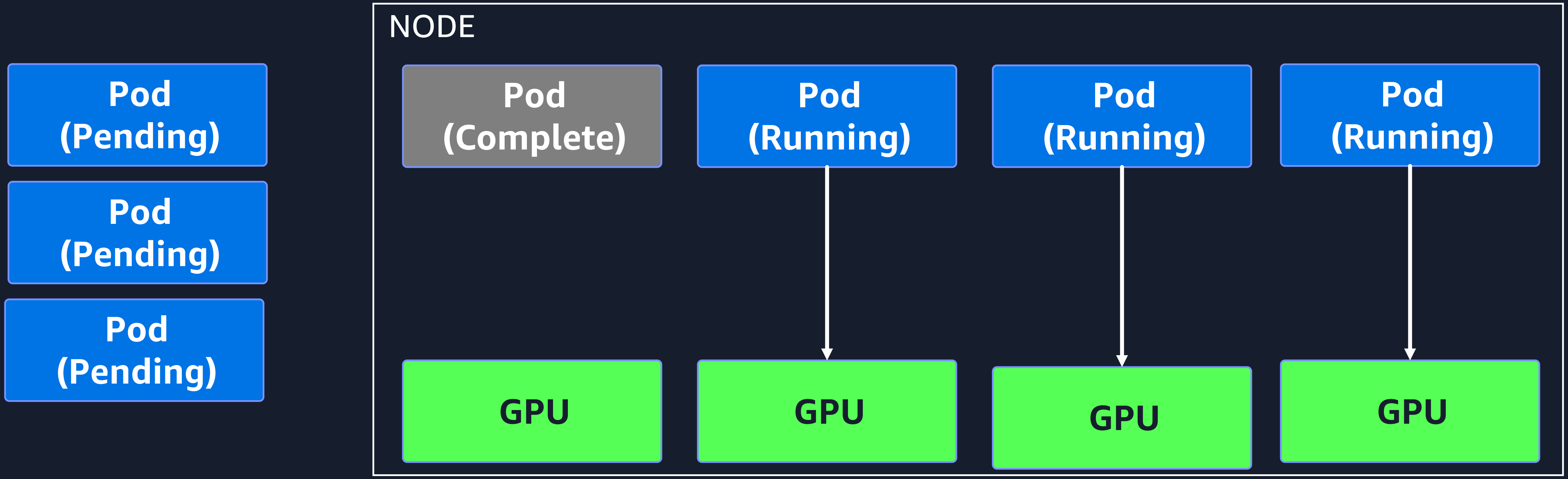
```
containers:
- name: cuda-container
  resources:
    limits:
      nvidia.com/gpu: 1 # requesting 1 GPU
```



Scheduling Pods

- Allocatable status on Node
- Pods states how much GPU they need
- As Pod wait for GPU, they are pending

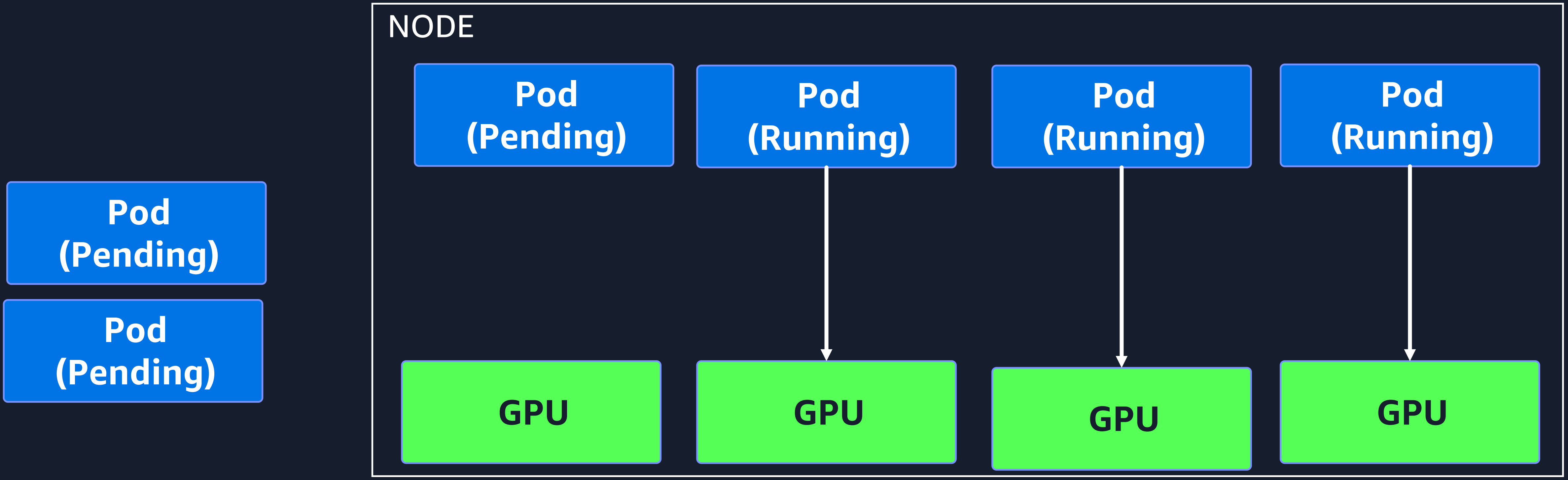
```
containers:
- name: cuda-container
  resources:
    limits:
      nvidia.com/gpu: 1 # requesting 1 GPU
```



Scheduling Pods

- Allocatable status on Node
- Pods states how much GPU they need
- As Pod wait for GPU, they are pending

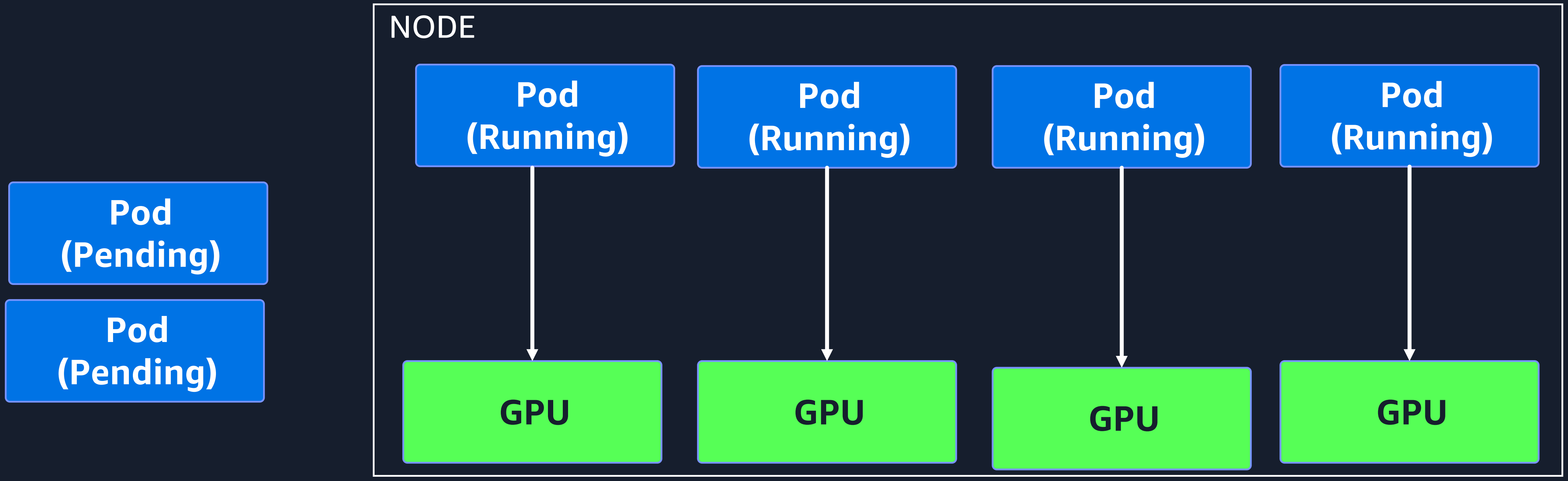
```
containers:
- name: cuda-container
  resources:
    limits:
      nvidia.com/gpu: 1 # requesting 1 GPU
```



Scheduling Pods

- Allocatable status on Node
- Pods states how much GPU they need
- As Pod wait for GPU, they are pending

```
containers:
- name: cuda-container
  resources:
    limits:
      nvidia.com/gpu: 1 # requesting 1 GPU
```



Why and How

Run Containers

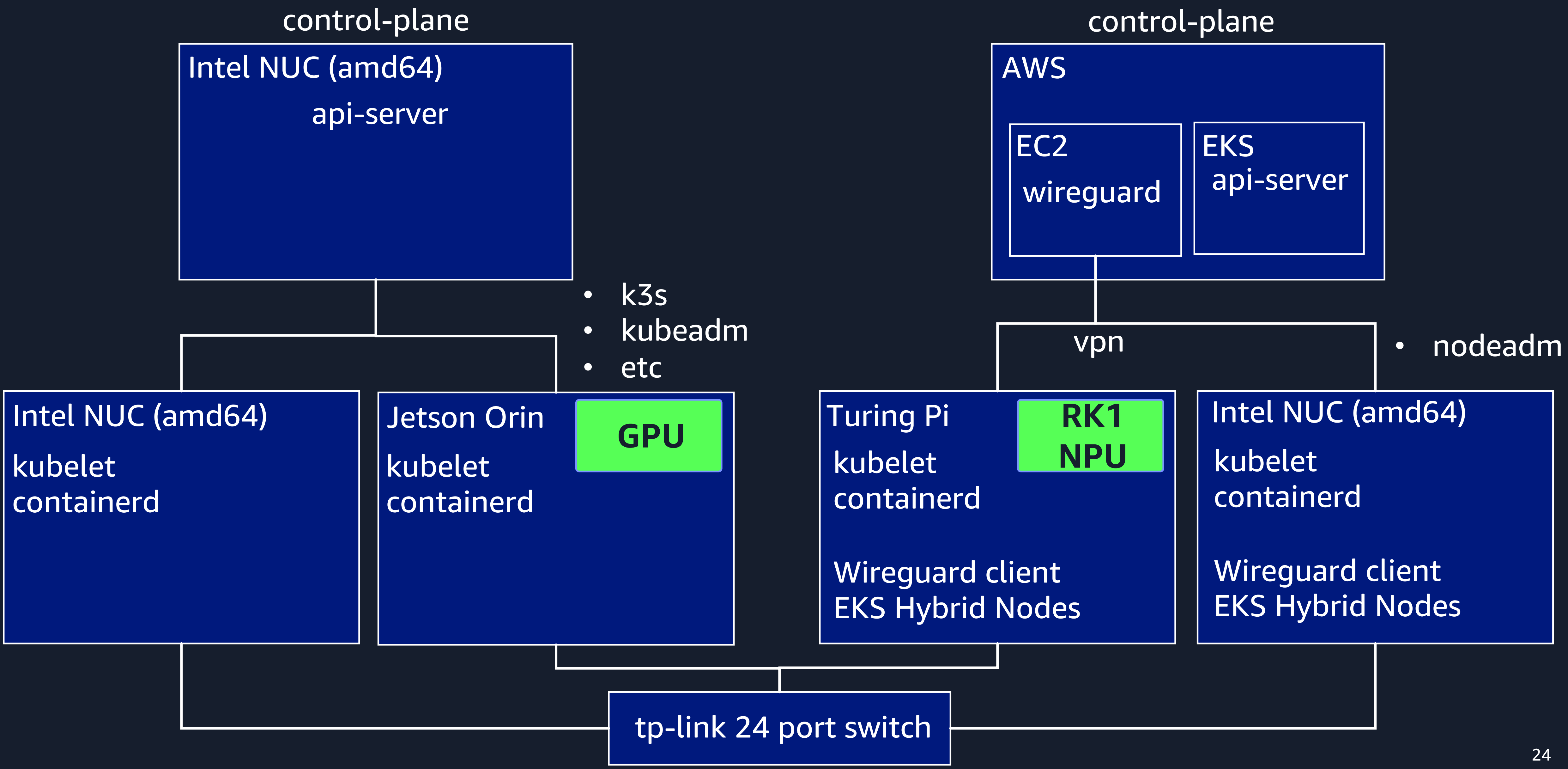
- Device Driver
- Container Toolkit
- CUDA

Orchestrate Containers

kubernetes

- Device Plugin
- Node Feature Discovery
- GPU Feature Discovery

Kubernetes Home lab



control-plane

Intel NUC (amd64)
api-server

- k3s
- kubeadm
- etc

Intel NUC (amd64)
kubelet
containerd

Jetson Orin
GPU
kubelet
containerd

control-plane

AWS

EC2
wireguard

EKS
api-server

vpn

- nodeadm

Turing Pi
RK1 NPU
kubelet
containerd

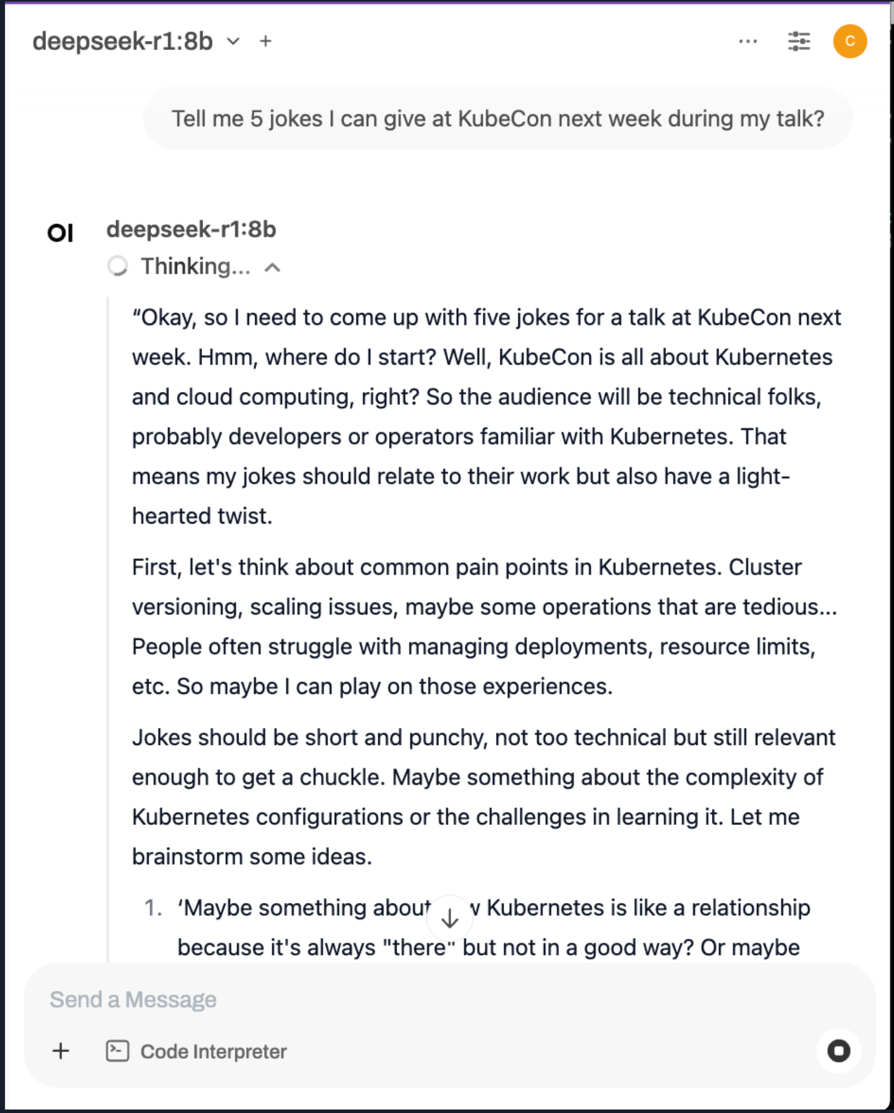
Wireguard client
EKS Hybrid Nodes

Intel NUC (amd64)
kubelet
containerd

Wireguard client
EKS Hybrid Nodes

tp-link 24 port switch

Deep Seek R1 on Jetson kubernetes



Tell me 5 jokes I can give at KubeCon next week during my talk?

deepseek-r1:8b

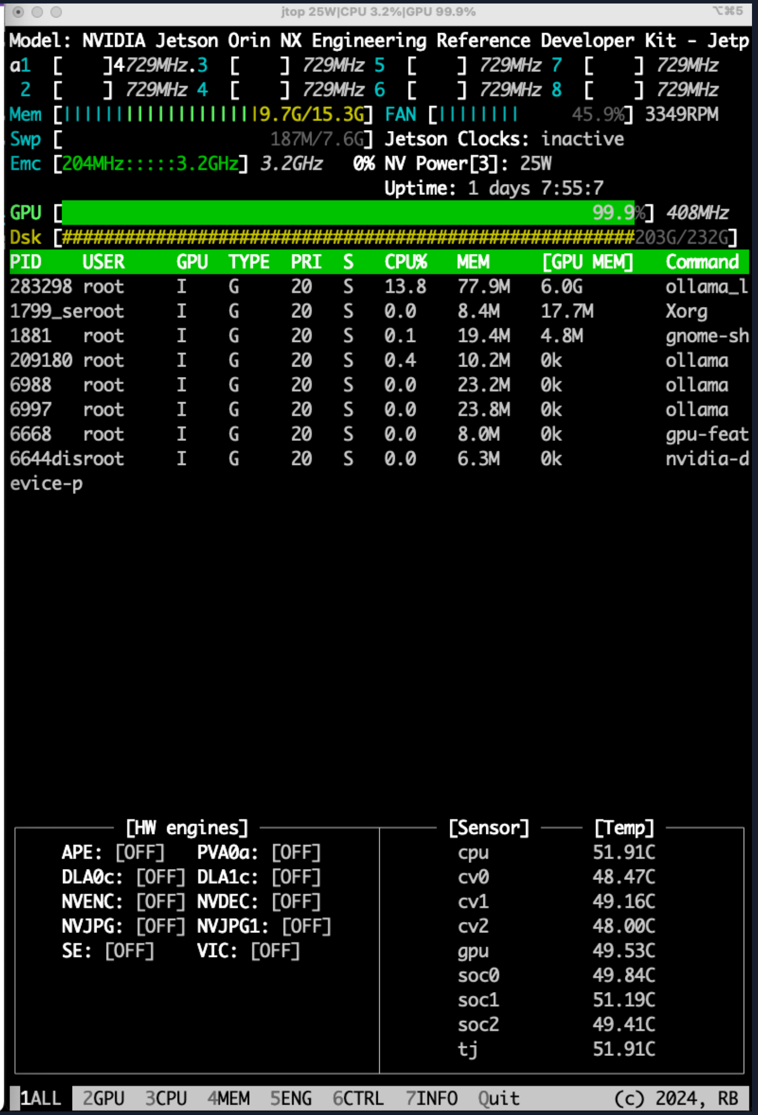
Thinking...

"Okay, so I need to come up with five jokes for a talk at KubeCon next week. Hmm, where do I start? Well, KubeCon is all about Kubernetes and cloud computing, right? So the audience will be technical folks, probably developers or operators familiar with Kubernetes. That means my jokes should relate to their work but also have a light-hearted twist.

First, let's think about common pain points in Kubernetes. Cluster versioning, scaling issues, maybe some operations that are tedious... People often struggle with managing deployments, resource limits, etc. So maybe I can play on those experiences.

Jokes should be short and punchy, not too technical but still relevant enough to get a chuckle. Maybe something about the complexity of Kubernetes configurations or the challenges in learning it. Let me brainstorm some ideas.

1. 'Maybe something about v Kubernetes is like a relationship because it's always "there" but not in a good way? Or maybe



```
jtop 25W|CPU 3.2%|GPU 99.9%                                        ⌥⌘5
Model: NVIDIA Jetson Orin NX Engineering Reference Developer Kit - Jetp
a1  [    ]4729MHz.3  [    ] 729MHz 5  [    ] 729MHz 7  [    ] 729MHz
 2  [    ] 729MHz 4  [    ] 729MHz 6  [    ] 729MHz 8  [    ] 729MHz
Mem [||||||||||||||||||9.7G/15.3G] FAN [|||||||||      45.9%] 3349RPM
Swp [                   187M/7.6G] Jetson Clocks: inactive
Emc [204MHz::::3.2GHz] 3.2GHz   0% NV Power[3]: 25W
                                   Uptime: 1 days 7:55:7
GPU [                                               99.9%] 408MHz
Dsk [##################################################203G/232G]
PID    USER      GPU  TYPE  PRI  S   CPU%   MEM    [GPU MEM]   Command
283298 root      I    G     20   S   13.8   77.9M  6.0G        ollama_l
1799_seroot      I    G     20   S   0.0    8.4M   17.7M       Xorg
1881   root      I    G     20   S   0.1    19.4M  4.8M        gnome-sh
209180 root      I    G     20   S   0.4    10.2M  0k          ollama
6988   root      I    G     20   S   0.0    23.2M  0k          ollama
6997   root      I    G     20   S   0.0    23.8M  0k          ollama
6668   root      I    G     20   S   0.0    8.0M   0k          gpu-feat
6644disroot      I    G     20   S   0.0    6.3M   0k          nvidia-d
evice-p

 ──────────── [HW engines] ─────────────────── [Sensor] ──── [Temp] ────
    APE: [OFF]    PVA0a: [OFF]               cpu          51.91C
    DLA0c: [OFF]  DLA1c: [OFF]               cv0          48.47C
    NVENC: [OFF]  NVDEC: [OFF]               cv1          49.16C
    NVJPG: [OFF]  NVJPG1: [OFF]              cv2          48.00C
    SE: [OFF]     VIC: [OFF]                 gpu          49.53C
                                             soc0         49.84C
                                             soc1         51.19C
                                             soc2         49.41C
                                             tj           51.91C
 1ALL  2GPU  3CPU  4MEM  5ENG  6CTRL  7INFO  Quit          (c) 2024, RB
```

Resources

- https://github.com/csantanapr/k8s-nvidia

AWS Booth
S300

Thank you!

Carlos Santana

csantanapr

santana.dev

Complete the Survey