THE LINUX FOUNDATION
OPEN SOURCE SUMMIT
NORTH AMERICA

Embedded Linux Conference

Cloud Native AI:

From Conformance to Production

Jake Pineda
CNCF Membership Development

In 2026, the real breakthrough in AI won't come from billion-dollar training runs. It will come from running hundreds of smaller, specialized models reliably and securely at scale. Inference is where AI gets real.

That's a fundamentally cloud native challenge — and it's platform engineers who build the open source infrastructure that makes enterprise AI practical and scalable.

Jonathan Bryce

Executive Director, CNCF

Operational Issues in Cloud-Native AI Stacks

Why production AI breaks in ways that standard Kubernetes monitoring doesn't catch.

GPU Driver Mismatch

Driver version inconsistency across nodes produces
silent inference corruption. No error. Wrong output.

Gang Scheduling Deadlocks

Distributed training jobs partially acquire GPUs and
block. Cluster appears busy. No job makes progress.

Bursty Resource Instability

AI workloads spike 10-50x during inference bursts.
Standard HPA on CPU/RAM lags too slow to react.

Observability Blind Spots

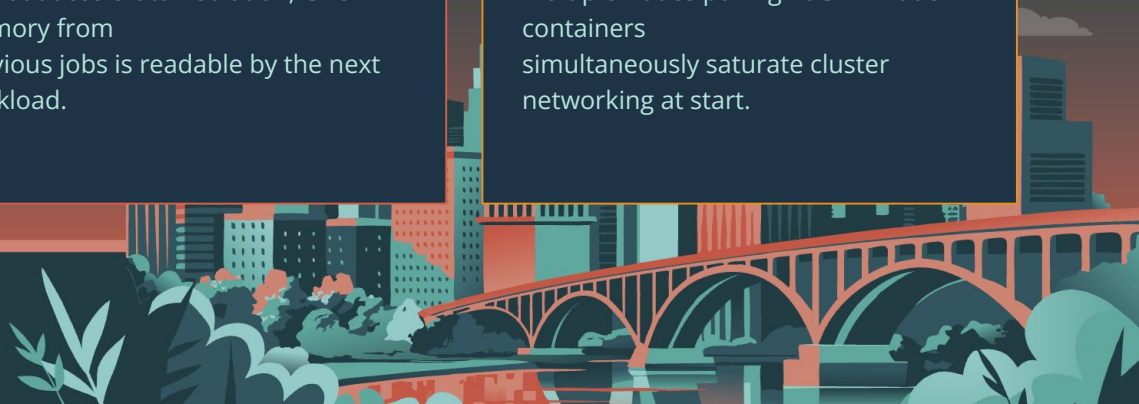
GPU utilization, temperature, and memory pressure
invisible to standard Prometheus without DCGM.

Multi-Tenant GPU Leakage

Without accelerator isolation, GPU memory from
previous jobs is readable by the next workload.

Image Pull Congestion

Multiple nodes pulling 10GB+ model containers
simultaneously saturate cluster networking at start.



Security Failure Modes in AI Infrastructure

Threat vectors specific to AI workloads that general K8s hardening doesn't address.

Threat Vectors

⚠ **Accelerator Privilege Escalation** Unrestricted GPU device access enables container escapes via driver exploits.

⚠ **Cross-Tenant Memory Exposure** GPU VRAM not zeroed between jobs leaks model weights or input data to adjacent tenants.

⚠ **Supply Chain via Model Containers** 10GB+ model images from unverified registries with no signature validation at pull.

⚠ **Inference Traffic Interception** No mTLS on inference endpoints in default K8s Gateway configurations.

⚠ **Unaudited Accelerator Access Logs** GPU access typically invisible to SIEM/audit pipelines; no K8s audit event emitted.

Conformance Mitigations

✓ **Isolated Accelerator Access** K8s resource management mediates all GPU access. No direct device exposure.

✓ **VRAM Zeroing via DRA Policy** Driver-level memory clearing enforced at resource allocation boundaries.

✓ **Signed Image Verification** Cosign + OPA policy enforces signature check before any model container starts.

✓ **Gateway API mTLS Enforcement** Conformance requires Gateway API; mTLS profiles enforced at ingress layer.

✓ **Audit-Ready Accelerator Events** DCGM Exporter exposes GPU access events in Prometheus format; SIEM-ingestible.

Community Insights: State of Adoption

Source: CNCF Annual Cloud Native Survey, Jan 2026 — 1,000+ organizations surveyed.

98%

Cloud native adoption across surveyed orgs

82%

Container users deploy Kubernetes in production

66%

Use K8s to host generative AI workloads

7%

Deploy AI models daily vs. 47% occasionally

The gap that matters: 47% deploy AI occasionally. Only 7% run it daily in production. The tooling exists.

Community Insights: What Teams Actually Report

Top operational pain points from CNCF member organizations running AI in production.

63%	report GPU utilization below 40% due to scheduling inefficiency	**Volcano gang scheduling + HAMi vGPU**
58%	experienced silent inference failure caused by driver inconsistency	**Conformance driver/runtime verification**
51%	lack standardized observability across AI inference services	**OpenTelemetry GenAI Semantic Conventions**
44%	report vendor lock-in as top blocker for multi-cloud AI deployments	**Kubernetes AI Conformance portability**
39%	have no formal process for validating AI platform security posture	**Conformance security and robustness baseline**

What Is AI Conformance?

A community-built solution to a community-identified problem.

Open Community Standard

Defined by contributors, governed in the open. Any vendor can participate. No single company controls the outcome.

Clear Expectations for AI Clusters

Specifies what a Kubernetes platform must expose to be considered production AI-ready. Testable and verifiable.

Portable, Reproducible AI Workloads

Deploy once on any conformant platform. Migrate without rewriting infrastructure. Portability becomes contractual.

What the Standard Covers

Four domains. Community-defined. Testable. Publicly auditable.

Scheduling & Accelerators

- Dynamic Resource Allocation (DRA) API
- Driver/runtime verification on nodes
- GPU sharing: partitioning and time-slicing
- Gang scheduling for distributed jobs
- Cluster autoscaling by accelerator demand
- HPA with accelerator-aware custom metrics

Metrics & Telemetry

- Standardized accelerator performance metrics
- AI service/workload observability layer
- Prometheus-compatible metric exposure
- GenAI Semantic Convention support (OTel)
- Utilization, memory, temperature visibility

Security & Robustness

- Isolated, mediated accelerator access
- Robust AI operator with CRD support
- Infrastructure/driver failure mitigation
- Device degradation detection paths
- Container crash recovery handling

Networking

- Gateway API with advanced traffic routing
- Weighted splits for A/B inference testing
- Header-based routing for model versions
- AI inference traffic management baseline
- mTLS profile compatibility

Security Implementation: What to Actually Configure

Conformance security domain — practical implementation checklist.

Accelerator Isolation

`resourceClaim.spec.devices.requests[].adminAccess: false`

Deny adminAccess in all DRA ResourceClaims unless explicitly required.

VRAM Clearing Policy

`nvidia-device-plugin: gpuMemoryClearing: enabled`

Enforce GPU memory zeroing at deallocation via device plugin flag.

Namespace GPU Quotas

`ResourceQuota: requests.nvidia.com/gpu: 4`

Prevent single namespace from starving cluster. Apply per team/project.

Audit Logging via DCGM

`dcgm-exporter: --collectors=gpm,nvlink,xid`

XID errors expose hardware faults and potential isolation breaches.

Model Image Signing

`cosign verify --key cosign.pub $IMAGE`

Enforce via OPA Gatekeeper policy. Block unsigned model containers at admission.

Gateway mTLS Profile

`Gateway.spec.infrastructure.parametersRef: mtls-strict`

Require mutual TLS on all inference service endpoints via Gateway API.

NetworkPolicy for GPU Nodes

`podSelector: nvidia.com/gpu: 'true'`

Isolate GPU node traffic. Deny east-west by default; allow only approved workloads.

Seccomp on AI Workloads

`securityContext.seccompProfile.type: RuntimeDefault`

Restrict syscalls available to inference containers. Kills most driver exploit paths.

Benchmarking Framework: Key Metrics

How to measure a Cloud Native AI environment against conformance standards.

Accelerator Efficiency

GPU Utilization *DCGM: DCGM_FI_DEV_GPU_UTIL*	**≥80% under sustained load**
GPU Memory Efficiency *DCGM: DCGM_FI_DEV_MEM_COPY_UTIL*	**<10% idle memory at scale**

Inference Performance

Time to First Token (TTFT) *OTel: gen_ai.server.ttft (P50/P95/P99)*	*[INSERT YOUR BASELINE]*
Tokens per Second (TPS) *vLLM: vllm:avg_generation_throughput*	*[INSERT YOUR BASELINE]*

Scheduling & Scaling

Gang Scheduling Success Rate *Volcano: volcano.sh/job-succeeded / total*	**≥99% for conformant platforms**
HPA Scale-Out Latency *Time: metric threshold breach → pod Ready*	**<90s for inference workloads**

Security & Reliability

XID Error Rate *DCGM: DCGM_FI_DEV_XID_ERRORS_COUNT*	**0 unresolved XID in production**
Workload Recovery Time *Time: pod failure → workload rescheduled*	*[INSERT YOUR BASELINE]*

Reference Stacks: Four Certified Platform Approaches

Real tooling choices from actual PRODUCT.yaml submissions — same conformance, different architectures.

CoreWeave CKS

v1.34

GPU-native neocloud — hyperscale AI training

SCHEDULER SUNK (Slurm-on-K8s) for large distributed jobs + Kueue gang scheduling
OPERATORS KubeRay for distributed PyTorch/JAX training across 1,000+ GPU nodes
NETWORKING RDMA/InfiniBand fabric + Gateway API for inference traffic routing
OBSERVABILITY DCGM Exporter + Prometheus — GPU metrics at cluster scale
HARDWARE Plugin-based GPU isolation (DRA migration in progress)

AKS + KAITO

v1.34

Managed cloud — LLM inference-as-a-service

SCHEDULER Kueue for gang scheduling + HPA with custom DCGM metrics
OPERATORS KAITO (CNCF sandbox) — AI Toolchain Operator auto-deploys vLLM/NIM
NETWORKING Istio service mesh + Gateway API for weighted model version routing
OBSERVABILITY DCGM Exporter — per-accelerator metrics fed to Azure Monitor
HARDWARE DRA (GA in K8s 1.34) — NVIDIA Device Plugin + GPU Operator

Kubermatic KKP

v1.34

European sovereign cloud — on-prem + hybrid

SCHEDULER Kueue gang scheduling + Cluster Autoscaler for GPU node pools
OPERATORS KubeRay for distributed training; full CRD lifecycle management
NETWORKING KubeLB 1.2 AI Gateway (kgateway) — optimized inference routing
OBSERVABILITY DCGM Exporter — Prometheus stack with GPU utilization HPA
HARDWARE DRA stable APIs — flexible GPU claims by attribute/topology

Red Hat OpenShift 4.20

v1.34

Enterprise hybrid cloud — multi-vendor GPU support

SCHEDULER Kueue for gang scheduling — distributed training job coordination
OPERATORS Kubeflow Trainer (TrainJob/TrainingRuntime CRDs) for ML pipelines
NETWORKING OpenShift Service Mesh (Istio) + Gateway API inference routing
OBSERVABILITY DCGM for NVIDIA + AMD GPU Operator for AMD accelerator metrics
HARDWARE DRA + NVIDIA/AMD Device Plugins — multi-vendor GPU support

Practical Implementation: Platform Foundation

Kubernetes v1.34+

1. Enable DRA: --feature-gates=DynamicResourceAllocation=true

2. Install nvidia-k8s-device-plugin with DRA mode enabled

3. Set ResourceQuota for GPU namespaces before AI team onboarding

Why: DRA replaces integer GPU requests with attribute-aware allocation. Required for conformance.

KEDA v2.13+ (AI-Aware Autoscaling)

1. Deploy with Prometheus scaler enabled

2. Create ScaledObject targeting vLLM queue_size or DCGM_FI_DEV_GPU_UTIL

3. Set cooldownPeriod: 60 to avoid GPU thrash on bursty inference

Why: Standard HPA on CPU/RAM misses inference load. KEDA scales on actual GPU pressure.

OpenTelemetry (GenAI Tracing)

1. Deploy OTel Collector with batch processor + Prometheus exporter

2. Enable GenAI Semantic Conventions in your inference SDK

3. Configure gen_ai.server.ttft and gen_ai.usage.input_tokens metrics

Why: Observability required by conformance standard. The only way to measure TTFT at P99.

Practical Implementation: GPU Scheduling & Efficiency

Stop leaving GPU utilization on the table.

Volcano (Gang Scheduling)

1. Replace default K8s scheduler with Volcano in kube-system

2. Set job.spec.minAvailable = total GPU count for distributed jobs

3. Use Queue resource to enforce fair-share across teams

Why: Prevents partial GPU acquisition deadlocks. Required for multi-node training at scale.

HAMi (GPU Virtualization)

1. Deploy HAMi DaemonSet on all GPU nodes

2. Set resources.limits: nvidia.com/gpumem: 8192 per container

3. Monitor vGPU utilization via HAMi's built-in Prometheus exporter

Why: DaoCloud achieved 80%+ GPU utilization (vs. ~40% industry avg) using HAMi vGPU partitioning.

Fluid (Dataset Caching)

1. Create Dataset CRD pointing to your model artifact store (S3/OSS)

2. Set dataset.spec.mounts[].options: readOnly=true for inference

3. Pre-warm cache before job submission with fluid.io/dataset annotation

Why: Eliminates storage bottleneck that stalls GPU jobs during data loading. 2-5x job start improvement.

Practical Implementation: Model Serving & Delivery

KServe (Inference API)

1. Deploy InferenceService CRD with transformer + predictor

2. Set autoscaling.knative.dev/target: 1 (RPS per replica)

3. Use canary.trafficPercent for zero-downtime model rollouts

Why: Standardized API wrapper. Plugs directly into Gateway API for weighted inference splits.

Kubeflow (MLOps)

1. Deploy Kubeflow Pipelines for reproducible training DAGs

2. Use KFServing for model registry and versioning

3. Integrate with OTel for pipeline step-level latency tracing

Why: IBM cut ML training time from 90s → 35s with Knative Eventing + Kubeflow. Measured.

Dragonfly (Image Delivery)

1. Deploy Dragonfly Manager + Scheduler + Seed Peer

2. Configure containerd to use Dragonfly as proxy registry

3. Set peerTaskConfig.minSeedPeerRatio: 0.3 for large model pulls

Why: P2P distribution: 100+ nodes pull 10GB+ model containers simultaneously without network saturation.

Certified Kubernetes — AI Platform

Program grew from 18 → 31 certified platforms since November 2025 launch.
v1.35 KARs now include agentic workload validation and in-place pod resizing.

GKE (Google)	AKS (Azure)	Oracle OKE	Amazon EKS	CoreWeave CKS
Red Hat OpenShift	ROSA (AWS)	Kubermatic KKP	SUSE RKE2	Gardener
Giant Swarm	Baidu CCE	DaoCloud	VMware vSphere	Akamai LKE
OVHcloud MKS	SpectroCloud	JD Cloud JCK	Unicom Cloud	Rafay MKS

+ 11 More

Real-World Stacks: What Certification Looks Like in Practice

CoreWeave

v1.33 + v1.34

GPU-native neocloud built on Kubernetes from day one

- SUNK scheduler (Slurm on K8s) for hyperscale distributed training
- Gang scheduling via Kueue — all-or-nothing GPU job acquisition
- Plugin-based GPU isolation — DRA migration underway as vendor support matures
- KubeRay managing RayClusters across 1,000+ GPU node jobs

Kueue + KubeRay + DCGM + plugin GPU isolation

AKS + KAITO

v1.34

Managed LLM inference via CNCF sandbox KAITO operator

- KAITO auto-provisions vLLM/NVIDIA NIM inference pods from model presets
- Kueue gang scheduling + HPA on custom DCGM accelerator metrics
- Istio + Gateway API — weighted splits for A/B model version testing
- DRA (GA in K8s 1.34) replaces integer nvidia.com/gpu resource requests

Kueue + Istio + KAITO/vLLM + DRA + DCGM

Kubermatic KKP

v1.34

European digital sovereignty — on-prem AI with full portability

- DRA stable APIs — GPU claims by attribute/topology, not integer counts
- KubeLB 1.2 AI Gateway (kgateway) — optimized inference load balancing
- Kueue for gang scheduling + Cluster Autoscaler for GPU node pools
- DCGM + Prometheus HPA — scale on GPU utilization, not just CPU

DRA + Kueue + kgateway + DCGM + Prometheus

Red Hat OpenShift

v1.34

Enterprise hybrid cloud with AMD + NVIDIA multi-vendor GPU support

- Kubeflow Trainer operator with TrainJob/TrainingRuntime CRDs
- Kueue gang scheduling — distributed ML job coordination at enterprise scale
- Dedicated AMD GPU Operator alongside NVIDIA — multi-vendor conformance
- OpenShift Service Mesh (Istio) + Gateway API inference routing

Kueue + Kubeflow Trainer + NVIDIA + AMD GPU support

Why Cloud Native AI Wins

01

Portability

Run AI workloads across cloud, hybrid, and on-prem without rewriting infrastructure. Conformance makes portability contractual, not aspirational.

02

Efficiency

HAMi vGPU + DRA + KEDA push GPU utilization from the industry's ~40% average toward 80%+. Open source tooling. No vendor markup.

03

Velocity

Volcano + Kubeflow + standardized tooling cuts experiment cycles from months to days. IBM measured it. OpenAI measured it. The numbers are real.

How the Standard Gets Built

Open governance. Community-defined. Vendor-neutral.

01

Community Proposal

Working group members identify gaps and propose capability requirements via GitHub issues and KEPs.

02

Public Review

Open comment period. Any contributor can challenge, modify, or strengthen the requirement.

03

Test Suite Development

Conformance tests written openly in the k8s-ai-conformance repo. Platforms self-certify by running them.

04

Public Certification

Certified platforms listed at landscape.cncf.io. Any buyer can verify claims independently.

github.com/cncf/k8s-ai-conformance | contribute.cncf.io | ask.cncf.io

Key Takeaways

01 **The operational problems are specific and solvable.**

Driver mismatch, gang scheduling deadlocks, GPU underutilization, silent security failures. Each has a named open source fix. None requires vendor lock-in.

02 **Conformance gives you a production baseline, not a ceiling.**

It defines the minimum viable AI-capable cluster. Your production stack builds on top of it. The benchmark metrics tell you when you're there.

03 **The open ecosystem built this. You own it.**

275,000 contributors across 190 countries defined these standards. No single vendor controls the roadmap. That's what makes it durable.

Get Involved

The standard is built in the open. So is the community.

AI Conformance Repo

github.com/cncf/k8s-ai-conformance

Explore the tests. Run them against your platform. Submit results.

Contribute to CNCF

contribute.cncf.io

Find a working group, SIG, or project. All work is public and reviewed openly.

Ask CNCF

ask.cncf.io

Community support, questions, implementation guidance from practitioners.

CNCF Sandbox

github.com/cncf/sandbox

Early-stage cloud native AI projects looking for contributors and adopters.

Thank You

Jake Pineda

CNCF Membership Development

cncf.io • #CloudNativeAI