KubeCon CloudNativeCon India 2026

#KubeCon #CloudNativeCon

Plug In and Scale: Serving LLM Models on Kubernetes Made Simple

Shrinidhi Venkataraman & Nithin Rajan
AstraZeneca

AstraZeneca

A global, science-led, patient-focused pharmaceutical business — pioneering in Oncology, BioPharma, and Rare disease.

96,100

Employees worldwide

FY2025

$58.7 bn

Total revenue

FY2025

86%

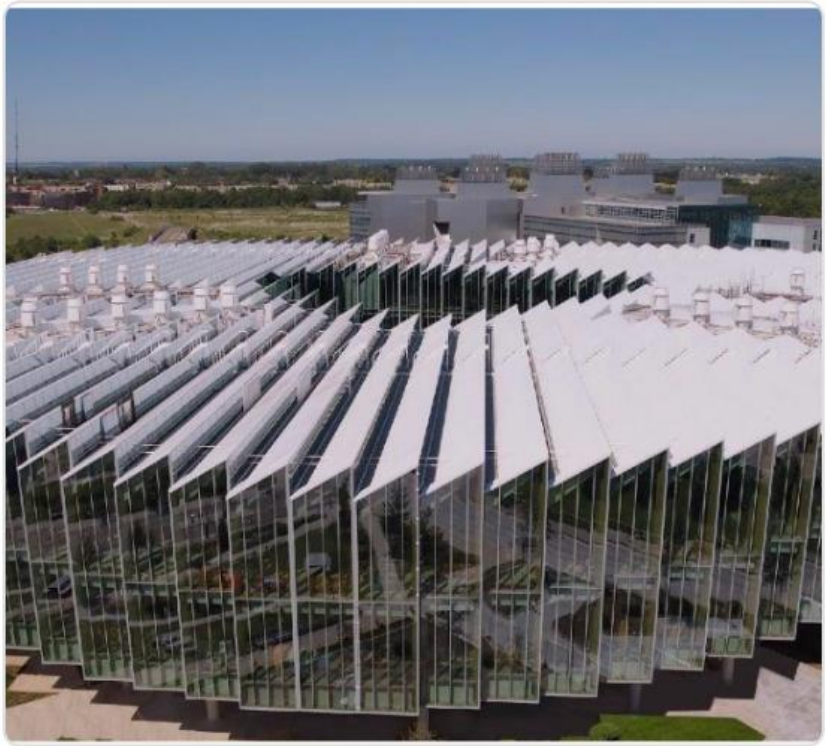
Say it's a great place to work

Employee sentiment

88.1%

Scope 1 & 2 GHG reduction

vs. 2015 baseline



Inference vs Serving

Two different jobs — one runs the model, one runs the model in production.

THE ENGINE

LLM Inference

Raw model computation

Goal	Generate output tokens from input
Metric	Speed — tokens per second (TPS)
Function	Run the trained model on new inputs
Focus	Fast execution for a single request

THE INFRASTRUCTURE

LLM Serving

Production wrapper around the engine

Goal	Manage requests, scaling, efficiency
Metric	Throughput, reliability, cost per token
Function	Queuing, scaling, hardware utilization
Focus	High volume across many users, reliably

Inference Runtimes Compared

KubeCon CloudNativeCon India 2026

vLLM

Primary Focus
- High-throughput text completion and broad model support.

Core Innovation
- Paged Attention
- VRAM fragmentation management

Workflow
- Standard prompt-in, token-out APIs

Best for
- Bulk processing, high-traffic generic APIs, and diverse model needs.

SG Lang

Primary Focus
- Complex workflows, agents, and structured generation

Core Innovation
- Radix Attention
- Chunked prefill refix-sharing across requests

Workflow
- Specialized language for controlling agent execution and JSON output.

Best for
- Multi-turn chat
- RAG and AI agents with repeated context.

Triton

Primary Focus
- Multi-model and multi-framework enterprise orchestration

Core innovation
- Dynamic batching and multi-framework integration

Workflow
- Integrates via custom C++/Python backends or specialized wrappers

Best for
- Deploying varied AI workloads (like CV, NLP, and audio) in one unified infrastructure.

What is an Inference stack ?

The full environment required to serve a model to **real users at scale** — it connects requests to the engine and keeps the infrastructure from buckling under heavy traffic.

STACKS WE'LL LOOK AT

managed

Baseten

Hosted serving platform for fast, managed model deployment.

open source

vLLM Production Stack

Cloud-native router, autoscaling, KV-cache & LoRA management.

nvidia

NVIDIA Dynamo

Disaggregated serving across vLLM, TensorRT-LLM, and SGLang.

Aggregated vs Disaggregated Serving

Where prefill and decode run — together, or on separate GPUs.

AGGREGATED

Prefill and decode share the same GPUs

Prefill

Favors a low degree of parallelism

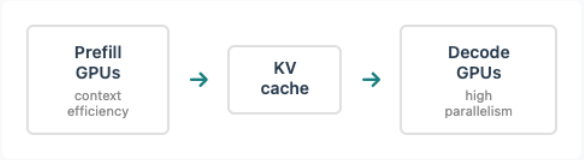
Decode

Favors a high degree of parallelism

One pool of GPUs does both — simplest to run, but the two phases compete for the same hardware.

DISAGGREGATED

Dedicated GPUs per phase, KV cache passed between



Prefill

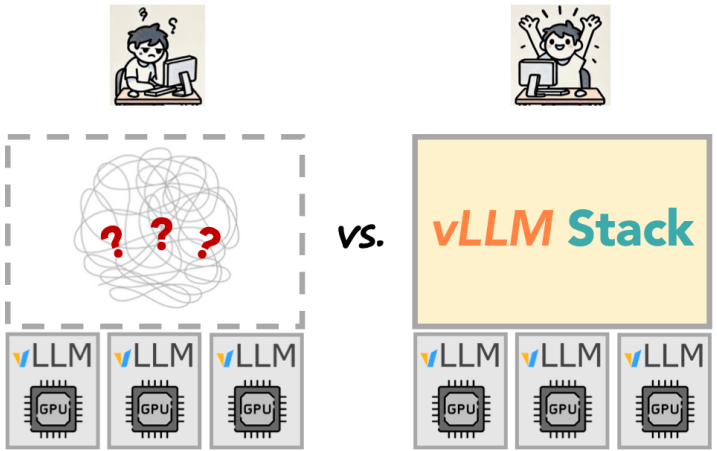
Dedicated GPUs improve context efficiency

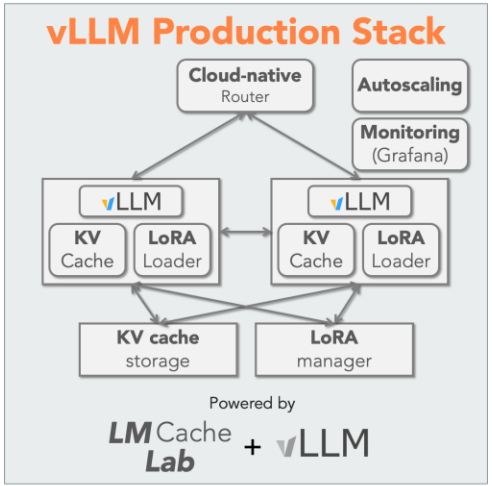
Decode

Scaled independently to optimize generation

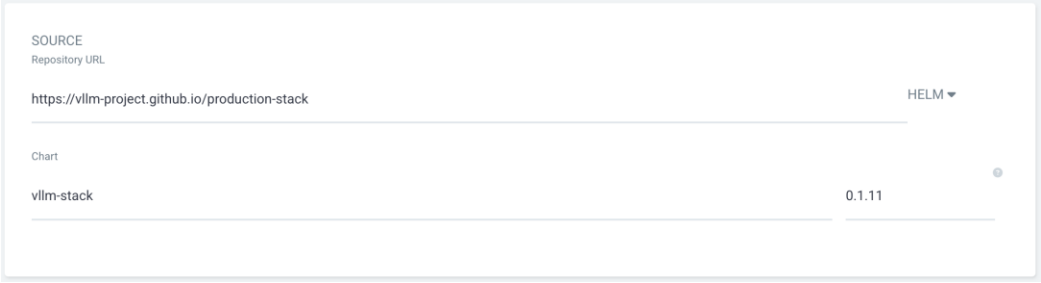
Each phase scales on its own — more moving parts, far better GPU utilization at scale.

vLLM Production Stack





Deploying the Stack



```yaml
servingEngineSpec:
  runtimeClassName: ""
  modelSpec:
  - name: "opt125m"
    repository: "vllm/vllm-openai"
    tag: "latest"
    modelURL: "facebook/opt-125m"

    replicaCount: 1

    requestCPU: 6
    requestMemory: "16Gi"
    requestGPU: 1
    # Optional resource limits - if not specified, only GPU will have a limit
    # limitCPU: "8"
    # limitMemory: "32Gi"
```

Sleep and Wakeup Mode

Code Blame Raw

```yaml
servingEngineSpec:
  runtimeClassName: ""
  modelSpec:
  - name: "granite"
    repository: "vllm/vllm-openai"
    tag: "latest"
    modelURL: "ibm-granite/granite-3.0-3b-a800m-instruct"
    replicaCount: 1
    requestCPU: 6
    requestMemory: "16Gi"
    requestGPU: 1
    pvcStorage: "50Gi"
    vllmConfig:
      extraArgs: ["--enable-sleep-mode"]
    env:
      - name: VLLM_SERVER_DEV_MODE
        value: "1"
```

Put the engine to sleep and check its sleeping state:

```
curl -X POST http://localhost:30080/sleep?id=b36921ab-6611-58c0-a941-16c51296446b | jq
```

```
] Sleep mode freed 39.26 GiB memory, 1.20 GiB memory is still in use.
:210] It took 5.749613 seconds to fall asleep.
/sleep HTTP/1.1" 200 OK
```

NVIDIA Dynamo

The same disaggregated topology runs on whichever engine fits the workload

vLLM

```
image: …/vllm-runtime
command: dynamo.vllm
args:
  - --model
  - Qwen/Qwen3-0.6B
  - --disaggregation-mode
  - prefill # / decode
```

Default — broad model support.

TensorRT-LLM

```
image: …/tensorrtllm-runtime
command: dynamo.trtllm
args:
  - --model-path
  - Qwen/Qwen3-0.6B
  - --extra-engine-args
  - configs/prefill.yaml
```

Peak throughput on NVIDIA GPUs.

SGLang

```
image: …/sglang-runtime
command: dynamo.sglang
args:
  - --model-path
  - Qwen/Qwen3-0.6B
  - --disaggregation-mode
  - prefill # / decode
```

Agents & structured gen · NIXL transfer.

Scale to Zero

KubeCon CloudNativeCon India 2026

KEDA drops idle deployments to zero replicas and wakes them on the first request.

scaledobject.yaml · KEDA

```yaml
apiVersion: keda.sh/v1alpha1
kind: ScaledObject
spec:
  minReplicaCount: 0   # scale-to-zero
  maxReplicaCount: 5
  pollingInterval: 10   # seconds
  cooldownPeriod: 360 # 6 min
```

01 **Scale up on queue depth**

`dynamo_frontend_queued_requests > 5`

02 **Hold ≥ 1 replica while work is pending**

A guard metric prevents scaling to zero mid-flight.

03 **Wake from zero on new traffic**

nginx_ingress_keepalive fires on the first hit.

Benchmarking with AI Perf

KubeCon CloudNativeCon India 2026

NVIDIA AIPerf drives load against the live endpoint and reports what users actually feel.

Higher Load Benchmark (100 concurrency, 10000 requests)

```
aiperf profile \
  --tokenizer "${TOKENIZER}" \
  --api-key "${AZSERVE_TOKEN}" \
  --model "${MODEL_ID}" \
  --endpoint-type chat \
  --url "${PUBLIC_BASE_URL}" \
  --concurrency 100 \
  --request-count 10000 \
  --benchmark-duration 300.0 \
  --benchmark-grace-period 30.0 \
  --streaming
```

RESULT · 100 CONCURRENCY · 10,000 REQUESTS

70,516 tps Output tokens / sec	**661** /s Requests / sec
12.03 ms Time to first token (avg)	**0.0** % Errors over 60,230 reqs

AI Perf – Live Run

KubeCon CloudNativeCon India 2026

To measures the performance of generative AI models served by preferred inference solution.

Profiling NVIDIA AIPerf 98%

Profile Progress

Profiling	98%
Records	60%
Status:	Profiling
Progress:	98,478 / 100,000 requests (98.5%)
Errors:	0 / 60,230 (0.0%)
Request Rate:	674.4 requests/s
Processing Rate:	413.6 records/s
Elapsed:	2m 26s
ETA:	2.3s

Real-Time Metrics

Metric	avg	min	max	p99	p90	p50	std
TTFT (ms)	12.03	5.31	338.93	72.16	17.86	7.71	18.11
TTST (ms)	7.14	0.00	313.06	20.07	9.07	6.26	11.76
Req Latency (ms)	1,419.66	1,297.56	1,896.25	1,671.23	1,529.52	1,395.90	77.33
ITL (ms)	13.33	10.20	18.17	15.57	14.27	13.13	0.66
Output TPS/User	75.18	55.03	98.01	79.80	78.09	76.19	3.46
OSL (tokens)	106.59	104.00	112.00	112.00	112.00	105.00	2.87
ISL (tokens)	550.00	550.00	550.00	550.00	550.00	550.00	0.00
Output TPS	70,516.46	N/A	N/A	N/A	N/A	N/A	N/A
Req/sec	661.57	N/A	N/A	N/A	N/A	N/A	N/A
Requests	60,230.00	N/A	N/A	N/A	N/A	N/A	N/A

Application Logs

```
/home/anthony/nvidia/projects/aiperf/ajc/aiohttp-trace/artifacts/openai_gpt-oss-120b-openai-chat-concurrency1000/inputs.json
15:48:11.911 INFO dataset_manager_cc96d08f inputs.json file generated in 0.00 seconds
15:48:11.911 INFO dataset_manager_cc96d08f Dataset configured in 0.31 seconds
15:48:11.911 INFO system_controller All services configured in 1.31 seconds
15:48:11.911 INFO system_controller AIPerf System is CONFIGURED
15:48:11.912 INFO timing_manager_3b70e706 Credit issuing strategy for Request_Rate started
15:48:11.915 INFO system_controller All services started profiling successfully
15:48:11.915 INFO system_controller AIPerf System is PROFILING
```

1 Overview 2 Progress 3 Metrics 4 Workers 5 Logs esc Restore View

Observability

Every replica reports to Prometheus. Grafana shows the whole fleet in real time. Requests, latency, and GPU utilization all fall away together as the deployment drains and scales to zero exactly the behavior we want.

What's Next?

KubeCon CloudNativeCon India 2026

NVIDIA Enterprise AI Factory

16 Racks
18 nodes per rack
72 Blackwell & 36 Grace per rack
Water cooled

2 ARM64 / Grace CPU's per node
4 Blackwell GPU's per node
6-10x H200 performance each
Unified memory architecture

NVIDIA Run:ai

- Deploy any Inference workload from UI
- Fractional GPU, Distributed Inference & Monitoring