KubeCon CloudNativeCon North America 2026

#KubeCon #CloudNativeCon

Composing Inference on Kubernetes

Gateway, Schedule, Serve, Observe

Karena Angell (Red Hat) · Alex Chircop (Google)

November 11, 2026 · KubeCon + CloudNativeCon North America 2026



Thesis: composability across six explicit lenses

> **“Composability in AI is the ability to assemble capabilities into larger systems while keeping the parts understandable, reusable, and changeable.”**

1. WORKLOAD LENS

Workload Composability

What gets combined: nesting inference calls, RAG pipelines, and agentic loops inside one application.

2. CONTROL-FLOW LENS

Control-Flow Composability

Who decides what runs next: static pipelines vs. dynamic routing, delegation, and live endpoint picking.

3. RESOURCE LENS

Resource Composability

How hardware is allocated: GPUs, TPUs, and DPUs scheduled and shared per stage (DRA, Kueue, HAMi).

4. INFORMATION LENS

Information Composability

What survives the hop: W3C trace propagation, OTel GenAI semantics, and end-to-end cost attribution.

5. HUMAN-AGENT LENS

Human-Agent Composability

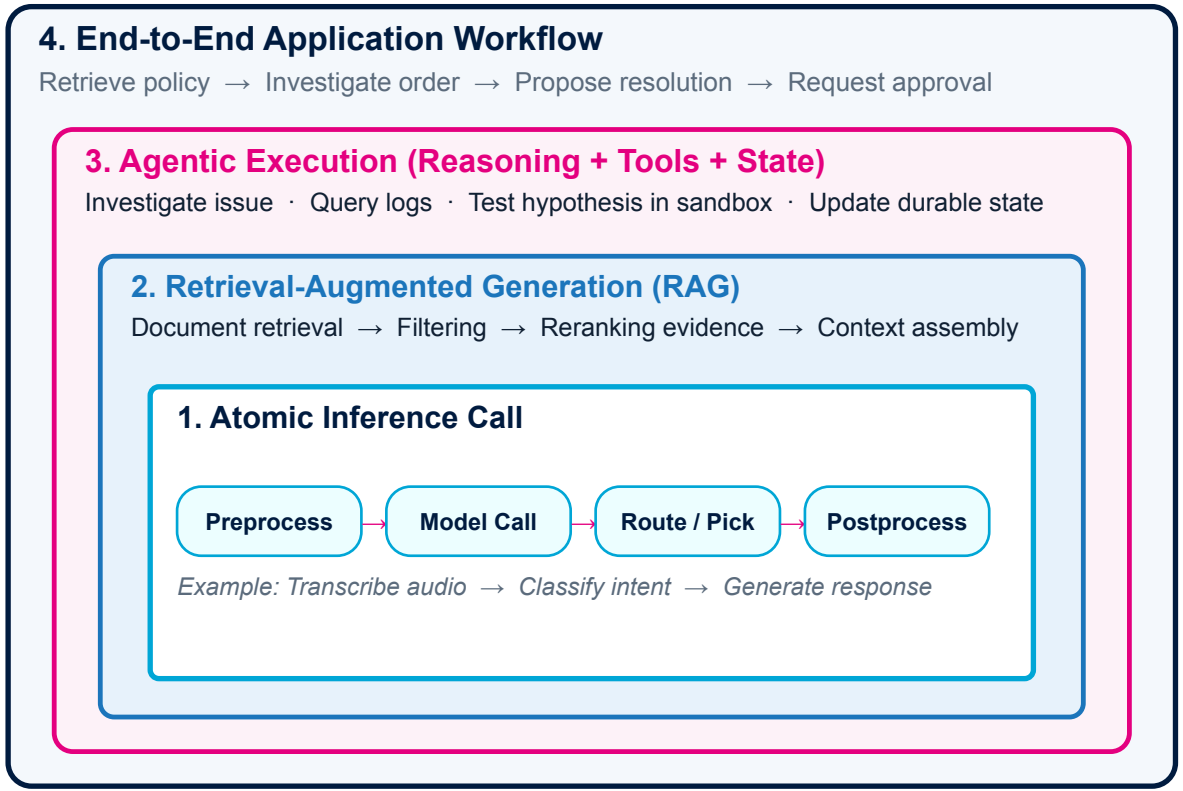
Who owns decisions & boundaries: authority handoffs, security isolation, state management, and runtime efficiency.

6. ARCHITECTURAL LENS

Architectural Composability

What can change alone: swapping gateways, schedulers, serving engines, or telemetry without replatforming.

Workload composability: everything nests



KEY INSIGHT

Why Nesting Matters to Platform Teams

- **Not three separate platforms:** Inference, RAG, and Agents are nested layers of the same workload.
- **Recursive calls:** An agent invokes a RAG pipeline as a tool; that RAG pipeline fans out to multiple model calls.
- **Hybrid control:** A fixed workflow can invoke a single bounded agent for one ambiguous step.
- **The trap:** Building separate silos for 'Chat', 'RAG', and 'Agents' duplicates routing, scheduling, and observability.

When a pilot becomes shared infrastructure

KubeCon CloudNativeCon North America 2026

What breaks when nested workload composition meets a shared multi-team platform

01

One model endpoint becomes everyone's dependency

A single team's pilot endpoint gets adopted across products without traffic isolation, SLO tiers, or backpressure controls.

02

Gateway policy & accelerator allocation live outside platform controls

Routing rules live in app code while GPU/TPU capacity is handed out via static side agreements rather than declarative scheduling.

03

Latency, queueing, and cost cannot be correlated

When tail latency spikes, signals are split across gateway proxies, scheduler queues, model runtimes, and device utilization.

04

A serving-engine swap means replatforming the application path

Client code, auth, and telemetry are tightly coupled to one runtime image—preventing upgrades from TGI to vLLM or SGLang.

The composable architecture

Control-Flow Lens

GATEWAY

- Edge & cluster routing
- Endpoint Picker (EPP)
- Auth, retry & backpressure
- Survives backend swaps

Resource Lens

SCHEDULE

- Workload admission & quota
- Multi-team fairness
- Topology-aware placement
- Batch & online arbitration

Architectural & Agent Lens

SERVE

- Pluggable model runtimes
- Prefill / decode pools
- Sandboxed agent workers
- Stable client contract

Information Lens

OBSERVE

- W3C trace propagation
- OTel GenAI conventions
- Cross-hop latency correlation
- Token & GPU cost attribution

ACCELERATE
Hardware Seam

Heterogeneous Accelerators: GPUs, TPUs, and DPUs on one control plane · Device Plugins · Dynamic Resource Allocation (DRA ResourceClaim) · Fractional GPU/VRAM sharing (HAMi)

KUBERNETES CONTROL PLANE · Declarative APIs · Pod & Job Lifecycle · Container & Sandbox Runtime · Multi-Cluster Fleet

Each piece is independently useful. The system works when the seams are explicit: routing, placement, serving, policy, and telemetry. Every box here is also a lens; which one is under pressure depends on what's failing.


KubeCon
CloudNativeCon
North America 2026
Control-flow composability
Who decides what runs next?

Common ways work can connect

1. Sequence

A → B → C

One step produces input for the next (e.g., preprocess → model call → postprocess).

2. Routing

Intent → [Model A | Model B | Tool]

A runtime decision selects the right capability, model tier, or cluster path.

3. Parallel Work

Fan-out → [T1 // T2 // T3] → Join

Independent retrievals, guardrails, or model calls run concurrently and combine.

4. Delegation

Coordinator → [Specialist Agents]

An orchestrator decomposes a goal and assigns subtasks to specialist workers.

5. Feedback Loops

Generator ⇄ Evaluator / Critic

Output is scored, verified, and revised iteratively until criteria are met.

6. Event-Driven

State Change / Event → Trigger Action

External signals, webhooks, or queue events asynchronously wake the next step.

Static vs. Dynamic Control Flow: Composition decided in advance (fixed DAG) gives predictable cost and latency. Composition decided during execution (autonomous agent) buys flexibility—at the cost of predictable spend, termination, and behavior.

Many workload paths across a fleet of clusters

CONTROL-FLOW: SEQUENCE

Latency-sensitive chat

Gateway → Endpoint Picker → Model Runtime

Routes directly to the cluster and replica with low-latency capacity and warm KV prefix cache.

CONTROL-FLOW: SEQUENCE + ROUTING

RAG near data

Gateway → Retrieval → Model Runtime → Policy

Routes to the cluster authorized to reach private data, combining vector search, reranking, and generation.

CONTROL-FLOW: ROUTING + FEEDBACK

Batch + fine-tuning + RL

Queue → Scheduler → Distributed Job

Admitted onto clusters with fair-share, reserved, or off-peak GPU/TPU capacity for throughput and post-training.

CONTROL-FLOW: DELEGATION + EVENT-DRIVEN

Agentic workloads

Agent Runtime → Sandbox → Virtual Network → Inference Gateway

Correlated telemetry across multi-turn tool loops—requiring security isolation for untrusted execution, durable state management across turns/waits, and infrastructure efficiency via fast suspend/resume.

Watch it: five real workload shapes, one architecture

- **LIVE LAB RECORDING (5 Real Traffic Shapes on One Stack)**

▶ **Lab Execution Reel**

1. Multi-Turn Chat
Prefix-cache affinity

Reuses warm KV cache on the same replica via llm-d EPP.

2. Tool-Calling Burst
Short + long interleaving

Agent interleaves rapid tool calls with deep context turns.

3. Multi-Tenant Traffic
Gateway fairness

Concurrent tenants contend without starving interactive SLOs.

4. Burst After Idle
Zero-to-peak spike

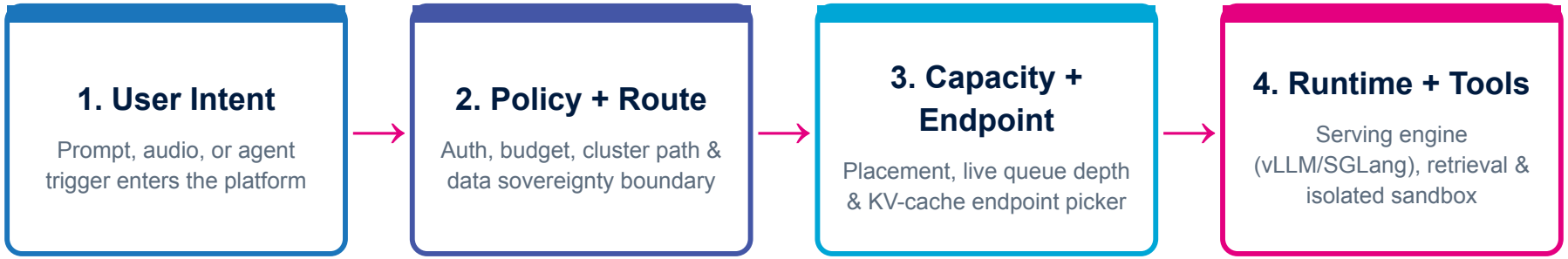
Sudden request surge triggers reactive scaling & backpressure.

5. Batch via Kueue
Quota & admission

Distributed batch job admitted fairly alongside live serving.

Same six architecture components (Kubernetes, Accelerate, Schedule, Gateway, Serve, Observe) - five traffic shapes, zero replatforming.

A composable request is a chain of choices



The Request Composes First — Composable Infrastructure Is the Platform's Response

Each arrow above is an explicit control-flow decision across platform seams:

- **Cluster & Gateway Seam:** Which fleet location, ingress policy, and model route handles this intent?
- **Scheduling & Endpoint Seam:** Which accelerator pool has quota, and which replica has warm KV cache?
- **Serving & Sandbox Seam:** Which runtime engine executes generation, and which sandbox isolates tool calls?

Schedule: admission, placement, and resource composability

KubeCon CloudNativeCon North America 2026

ADMISSION & PLACEMENT

Questions the Platform Must Answer

- **Placement & Topology:** Which cluster, node, and NVLink/ICI topology domain should this run on?
- **Heterogeneous Capacity:** What GPU, TPU, or DPU capacity is actually allocatable right now?
- **Multi-Team Fairness:** What is fair across teams, priorities, and preemptible batch workloads?
- **Backpressure Strategy:** What waits in queue, degrades to a smaller model, or routes elsewhere?

RESOURCE COMPOSABILITY

Project examples across components

Kueue	llm-d	KubeRay
HAMi	Kthena	KEDA
Volcano	KAI Scheduler	DRA

Heterogeneous Accelerators (GPUs, TPUs, DPUs):

Can GPUs, TPUs, and DPUs be advertised via Device Plugins or Dynamic Resource Allocation (ResourceClaim) and allocated per stage (or shared via HAMi)—not reserved whole for every request?

Gateway: where control-flow decisions meet

KubeCon CloudNativeCon North America 2026

Client / App / Agent
Standard OpenAI / GenAI API request

→

Inference Gateway + Picker (EPP)
Auth, policy, Envoy ext_proc gRPC routing

→

Model Runtime Pool(s)
vLLM / SGLang replicas + KV-cache state

LIVE SERVING SIGNALS

Routing signals come from serving

The Endpoint Picker (EPP) reads live queue depth, LoRA adapter state, and KV-cache locality directly from model servers over gRPC external processing—while workload admission stays with the scheduler.

- **Evaluated per request:** Avoids overloaded replicas and maximizes prefix-cache hits.

TWO CONTROL LOOPS

Admission ≠ Routing

The admission scheduler decides whether a pod can exist (quota, fairness, placement). The endpoint picker decides which existing pod serves this request: two distinct control loops.

- **Swappable picker seam:** GIE reference picker, llm-d EPP, or Kthena router.

The gateway boundary should survive backend change. The picker itself is a swappable interface.

Evaluated across: Gateway API · K8s Inference Gateway (GIE) · llm-d · Envoy AI Gateway · agentgateway · kgateway · Istio · Praxis

Observe: information composability

KubeCon CloudNativeCon North America 2026

What Has to Survive the Hop	Propagated Signal (W3C + OTel GenAI)	Why It Matters to Operators
Knowledge: retrieved documents, search chunks & evidence	W3C traceparent across gateway, retrieval & rerank child spans	**Which cluster, gateway, data store, and backend handled the request?**
Context: instructions, routing policy & queue admission wait	Envoy ext_proc picker span + scheduler queue & TTFT attributes	**Was tail latency caused by admission queueing, routing, or model runtime?**
State: multi-turn progress, KV-cache hits & active tool calls	Session correlation + gen_ai.* engine & sandbox execution spans	**Which model replica, engine path, or agent tool step is slow or looping?**
Provenance: caller identity, tenant budget & source citations	Baggage / tenant attributes + token usage & GPU energy (Kepler)	**Who or what workload owns the spend, and where did a claim come from?**

The Mechanism: W3C trace-context propagation carries correlation across gateway, Envoy ext_proc endpoint picker, serve, and model; OpenTelemetry and GenAI semantic conventions give an already-connected trace a shared vocabulary.

Subtler Failure Mode: Syntactic compatibility without semantic compatibility—where two components exchange valid JSON and intact spans, but disagree on what “confidence”, “cached”, or “completed” means.

Serve: keep runtimes behind a stable contract

KubeCon CloudNativeCon North America 2026

PLUGGABLE ENGINES

Engines Evolve Faster Than Platforms

- **Pluggable Serving Stack:** KServe patterns, vLLM, SGLang, Triton, Ray Serve, Ollama, and future runtimes must sit behind one stable contract.
- **Real-World Proof (TGI):** Hugging Face moved Text Generation Inference (TGI) into maintenance mode in favor of vLLM and SGLang—proving why engines must be swappable.
- **Preserve Upstream Seams:** Changing the runtime for continuous batching, quantization, or prefill/decode disaggregation must not break clients or gateways.

BOUNDARY TEST

The Serving Boundary Test

If swapping the serving engine means rewriting routing, authentication, metrics, or cost attribution, the serving component is doing too much platform work.

- **Pass criteria:** Swap vLLM ↔ SGLang with zero changes to Gateway API, EPP, or OTel dashboards.

Quiet Lens: Capability Composability

Right-size the building block: too small and every app rebuilds serving plumbing; too large and it locks in routing and policy.

Human-agent composability: dividing responsibility

1. Human Defines Goal
Sets intent, policy budget, and scoped authority boundary

Delegates Task →

2. Agent Executes in Sandbox
Gathers evidence, runs tools & models inside isolated stateful runtime

Handoff Packet →

3. Human Owns Decision
Reviews evidence, assumptions & open questions; authorizes action

Delegating a task ≠ Granting authority ≠ Assigning accountability | Why new CNCF agentic projects exist: Security Isolation, Infrastructure Efficiency & State Management

EFFICIENCY + STATE MANAGEMENT

Agent Substrate

Actor/worker runtime with sub-second RAM suspend/resume.

- **Infrastructure Efficiency:** Packs thousands of bursty agents without holding idle pods during LLM/human waits.
- **State Management:** Checkpoints and restores durable multi-turn state across tool calls and handoffs.

POLICY SECURITY ISOLATION

OpenShell

Policy-enforced filesystem, network, and MCP boundaries.

- **Security Isolation (Policy):** Intercepts shell commands, file reads, credentials, and network egress per agent task.
- **Audit & Governance:** Emits OCSF audit trails so human owners can verify what an agent touched.

KERNEL SECURITY ISOLATION

gVisor

User-space kernel sandbox for untrusted code execution.

- **Security Isolation (Kernel):** Isolates syscalls in a memory-safe user-space kernel (used by K8s agent-sandbox).
- **Infra Efficiency vs. VMs:** Delivers strong multi-tenant isolation with fast container-native startup.

Diagnose by boundary first

Operator Symptom (What Hurts)	Platform Boundary to Tackle First	Composability Lens
Capacity exists on the cluster and stays unused while requests queue	Gateway + Endpoint Picker (EPP) + Schedule	**Control-flow**
One team or batch job starves another during traffic bursts	Admission + Quota Fairness (Kueue / DRA / HAMi)	**Resource**
Serving runtime swap (TGI → vLLM / SGLang) breaks client apps	Gateway / Serve API Contract	**Architectural**
Latency spike or GPU/token cost cannot be explained end to end	Observe (W3C traceparent + OTel GenAI spans)	**Information**
Agent reaches somewhere it shouldn't, idles on costly compute, loses state, or lacks decision owner	Sandbox Isolation + Stateful Runtime + Accountability (gVisor, OpenShell, Agent Substrate)	**Human-agent**

Open ecosystem map: interfaces before projects

Gateway + routing

- Gateway API
- Inference GW
- Envoy AI GW
- llm-d
- agentgateway
- kgateway
- Istio
- Praxis

North-south & east-west AI routing, InferencePool, and swappable Endpoint Picker (EPP).

Schedule + capacity

- Kueue
- Volcano
- KAI Scheduler
- Kthena
- KubeRay
- HAMi
- KEDA
- llm-d
- DRA

Fair admission, topology placement, and heterogeneous GPU, TPU & DPU allocation/sharing.

Serve + engines

- KServe
- vLLM
- SGLang
- Triton
- Ollama
- Ray Serve
- TGI (maint.)

Pluggable model runtimes, prefill/decode disaggregation, and stable serving contracts.

Observe + agentic runtime

- OpenTelemetry
- Prometheus
- Kepler
- gVisor
- Agent Substrate
- OpenShell
- OpenRL

End-to-end traces & cost (OTel/Kepler) + agent security isolation (gVisor/OpenShell), efficiency & state management (Agent Substrate).

Architectural composability: what can change independently?

KubeCon CloudNativeCon North America 2026

SWAP 1: GATEWAY & PICKER

Envoy AI GW ⇄ agentgateway ⇄ Istio / kgateway / Praxis
llm-d EPP ⇄ GIE Reference Picker

Unchanged: Scheduler, InferencePool, Serving Engines & OTel Traces

SWAP 2: SERVING ENGINE

vLLM ⇄ SGLang ⇄ Triton ⇄ Ray Serve
(or migrating off legacy TGI)

Unchanged: Gateway API Contract, Endpoint Picker, Auth & Cost Attribution

SWAP 3: OBSERVABILITY BACKEND

Jaeger / Tempo ⇄ Cloud Trace ⇄ Prometheus / VictoriaMetrics

Unchanged: W3C traceparent headers, OTel GenAI spans & upstream services

Further Left on the Stack: Schedule & Accelerate Are Independently Swappable Too

Swap Admission Scheduler (Kueue ⇄ Volcano ⇄ KAI Scheduler) or Swap Accelerators (GPU ⇄ TPU ⇄ DPU via DRA ResourceClaim) — and Gateway, Serve, and Observe don't notice.

Gateway, Serve, and Observe are today's examples, not the only ones: Kubernetes, Accelerate, and Schedule are independently swappable too. Logical composition depends on explicit seams—not microservice count.

Watch it: the architecture survives real swaps

- **LIVE LAB RECORDING (3 Live Component Swaps Back-to-Back)**

▶ **Live Swap Verification Reel**

Swap 1: Three Gateways at Once

agentgateway + Istio + Envoy AI Gateway

All three installed concurrently on the same cluster, routing live requests to the exact same InferencePool without conflict.

Swap 2: Live Endpoint Picker Swap

llm-d EPP ⇄ GIE Reference Picker

Swapped live against the active InferencePool—same Envoy ext_proc contract, zero client or model server restarts.

Swap 3: Three Admission Loops

Kueue ⇄ Volcano ⇄ KAI Scheduler

Three independent scheduling and admission controllers verified against the same inference and batch workloads.

None of this is simulated - this is what architectural composability means when you actually test the boundaries.

The composability test: explicit choices over vertical lock-in

Composability isn't one property. It's workload, control-flow, resource, information, human-agent, and architectural choices, made explicit.

Architectural	Change the serving engine (TGI → vLLM → SGLang) or gateway while preserving the client and picker contract.
Control-flow	Route by live capacity and KV-cache state (EPP), not just a static service endpoint.
Information	Attribute latency and token/energy cost through an OpenTelemetry trace that actually connects end to end.
Human-agent	Enforce security isolation, infrastructure efficiency, and state management while knowing who owns every decision.
Resource & workload	Absorb new accelerators (GPUs, TPUs, DPUs via DRA / HAMi) and nested workload types without forking the platform.



KubeCon CloudNativeCon North America 2026

#KubeCon #CloudNativeCon

Thank you

Gateway, Schedule, Serve, Observe

Karena Angell (Red Hat) · Alex Chircop (Google)

KubeCon + CloudNativeCon North America 2026 · Salt Lake City

