MCP Dev Summit 2026 · New York

Kubernetes-Native Agent Discovery: A Unified Registry for MCP Servers and Skills

How Kubernetes CRDs, existing projects, and kro composition create a discoverable agentic infrastructure

Carlos Santana
Sr. Specialist Solutions Architect, Kubernetes — AWS
@csantanapr

THE STARTING POINT

Organizations already run Kubernetes

- Production workloads at scale
- Platform teams building internal developer platforms
- Kubernetes as a control plane for platform engineering
- Infrastructure abstracted via CRDs

The Kubernetes API gives you

- Schema validation (spec + status)
- Admission & mutation webhooks
- Eventual consistency & drift detection
- RBAC, audit logging, GitOps
- Years of operational experience

AI Agents are just another workload that can be modeled with the Kubernetes API.

THE ECOSYSTEM

Kubernetes Projects for AI Agents

kagent

Solo.io → CNCF Sandbox

- Agent, ModelConfig, RemoteMCPServer CRDs
- MCPServer via KMCP controller
- A2A protocol for agent communication
- Web UI + CLI
- Skills from OCI images or Git

CNCF | MCP Native

K8s Agent Sandbox

kubernetes-sigs

- Secure sandboxed execution for agents
- gVisor / Kata runtime isolation
- Warm pool for fast pod startup
- RBAC-scoped per agent run
- Network policy enforcement

K8s SIG | Security

Sympozium

OpenClaw Project

- SympoziumInstance, AgentRun, MCPServer CRDs
- SkillPack & PersonaPack bundles
- Policy enforcement via webhooks
- Channels: Slack, Telegram, Discord
- Persistent memory per agent

Open Source | MCP Native

COMMON PATTERN

These projects provide CRDs for MCP Servers

kagent MCPServer (KMCP)

```yaml
apiVersion: kagent.dev/v1alpha1
kind: MCPServer
metadata:
  name: everything-mcp-server
  namespace: kagent
spec:
  transportType: stdio
  deployment:
    cmd: npx
    args: ["-y", "@modelcontextprotocol/server-everything"]
    port: 3000
# status: no url field ⚠
```

Sympozium MCPServer

```yaml
apiVersion: sympozium.ai/v1alpha1
kind: MCPServer
metadata:
  name: everything
  namespace: sympozium-system
spec:
  transportType: stdio
  deployment:
    image: node:22-slim
    cmd: npx
    args: ["-y", "@modelcontextprotocol/server-everything"]
status:
  url: http://everything.sympozium-system.svc:8080 ✓
```

Similar spec, different status contracts. kagent doesn't expose `status.url` — the Service is created by convention with the same name.

THE CHALLENGE

No consistent API across MCP Server registries

- Each project defines its own MCPServer CRD
- Status fields differ — some have `url`, some don't
- Agents must be pre-configured with endpoints
- No unified discovery across the cluster
- Brittle in dynamic environments

What if you could

- Wrap any MCPServer CRD into a unified API
- Compose MCP servers from Deployments + Services
- Reference cloud-hosted MCP gateways
- Expose a consistent `status.url` for all
- Build a discoverable agentic registry

THE SOLUTION

kro — Kubernetes Resource Orchestrator

What kro does

- Create new CRDs declaratively via ResourceGraphDefinitions
- Compose resources or reference existing ones
- CEL expressions for dynamic wiring
- Dependency ordering, conditionals, readiness gates
- RGDs can compose other RGDs

K8s SIG Cloud Provider EKS Capability

For MCP registries, kro lets you

1. Compose — Build MCPServer from Deployment + Service

2. Wrap — Reference existing kagent/Sympozium MCPServers via externalRef

3. Unify — Expose a consistent `status.url` regardless of source

4. Extend — Model cloud MCP gateways as K8s resources

PATTERN 1: COMPOSE

kro MCPServer — Deployment + Service

RGD creates Deployment + Service

```yaml
apiVersion: kro.run/v1alpha1
kind: ResourceGraphDefinition
metadata:
  name: mcpserver
spec:
  schema:
    kind: MCPServer
    spec:
      name: string | required=true
      image: string | required=true
      port: integer | default=8080
    status:
      url: "http://${service.metadata.name}
        .${service.metadata.namespace}
        :${string(service.spec.ports[0].port)}"
  resources:
    - id: deployment  # apps/v1 Deployment
    - id: service     # v1 Service
```

Instance + Status

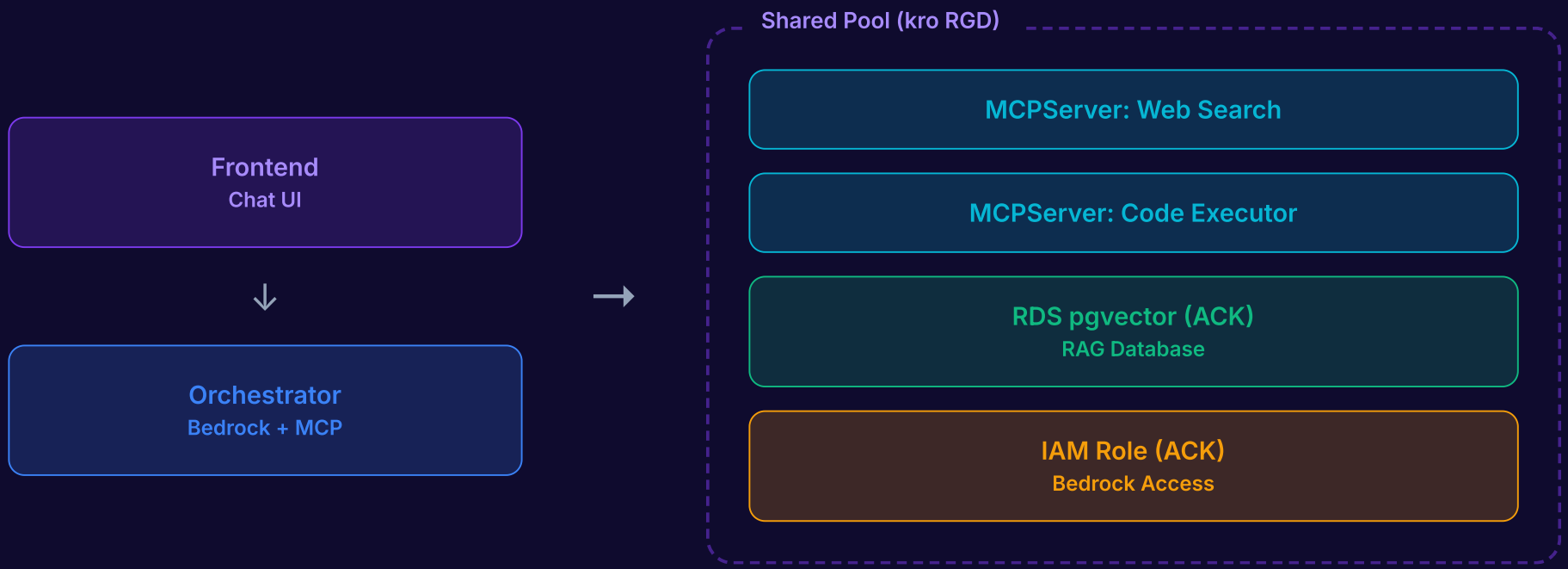
```yaml
# What the user writes
apiVersion: kro.run/v1alpha1
kind: MCPServer
metadata:
  name: mcp-web-search
spec:
  name: mcp-web-search
  image: my-registry/mcp-web-search:latest
  port: 8080

# kro populates status automatically
status:
  url: http://mcp-web-search.default:8080
  ready: true
```

kro creates the Deployment and Service, wires the ports, and exposes the URL in status.

DEMO: AGENT-AS-A-SERVICE

MCP + RAG composed with kro



One `AIAgentSystemSharedPool` instance creates ~15 resources: MCP servers, database, IAM, NodePools — all wired via CEL.

DEMO: KAGENT

kagent — CNCF Kubernetes Agent Framework

What we deployed

- kagent v0.8.3 on EKS via Helm
- Anthropic claude-sonnet-4-6
- 10 built-in agents (k8s, helm, istio, ...)
- MCPServer: everything-mcp-server
- Agent using `get-sum` MCP tool

Key CRDs

Agent · ModelConfig · RemoteMCPServer · MCPServer

```yaml
# MCPServer — deploys stdio MCP in-cluster
apiVersion: kagent.dev/v1alpha1
kind: MCPServer
spec:
  transportType: stdio
  deployment:
    cmd: npx
    args: ["-y", "@modelcontextprotocol/server-everything"]

# Agent — references MCPServer tools
apiVersion: kagent.dev/v1alpha2
kind: Agent
spec:
  type: Declarative
  declarative:
    modelConfig: default-model-config
    tools:
    - type: McpServer
      mcpServer:
        name: everything-mcp-server
        kind: MCPServer
        apiGroup: kagent.dev
        toolNames: [get-sum]
```

DEMO: SYMPOZIUM

Sympozium — K8s Agent Orchestration

What we deployed

- Sympozium v0.8.9 on EKS via Helm
- MCPServer: everything (stdio)
- Agents run as ephemeral K8s Jobs
- MCP bridge sidecar in agent pods
- Policy enforcement via webhooks

Key CRDs

SympoziumInstance · AgentRun · MCPServer · SkillPack · SympoziumPolicy

```yaml
# MCPServer — has status.url ✓
apiVersion: sympozium.ai/v1alpha1
kind: MCPServer
spec:
  transportType: stdio
  toolsPrefix: everything
  deployment:
    image: node:22-slim
    cmd: npx
    args: ["-y", "@modelcontextprotocol/server-everything"]
status:
  ready: true
  url: http://everything.sympozium-system.svc:8080

# Instance references MCP servers inline
apiVersion: sympozium.ai/v1alpha1
kind: SympoziumInstance
spec:
  mcpServers:
    - name: everything
      url: http://everything.sympozium-system.svc:8080
```

THE UNIFIED REGISTRY

MCPServerUnified — One API, any source

RGD wraps kagent MCPServer

```yaml
apiVersion: kro.run/v1alpha1
kind: ResourceGraphDefinition
metadata:
  name: mcpserver-unified
spec:
  schema:
    kind: MCPServerUnified
    spec:
      name: string | required=true
      namespace: string | required=true
    status:
      url: "http://${mcpService.metadata.name}
        .${mcpService.metadata.namespace}
        :${string(mcpService.spec.ports[0].port)}"
  resources:
    - id: mcpServer    # externalRef: kagent MCPServer
    - id: mcpService   # externalRef: companion Service
```

Live on our cluster

```yaml
# Instance pointing to kagent MCPServer
apiVersion: kro.run/v1alpha1
kind: MCPServerUnified
metadata:
  name: everything-mcp-server
spec:
  name: everything-mcp-server
  namespace: kagent
```

```
# kubectl get mcpserverunified
NAME                    URL                                READY
everything-mcp-server   http://everything-mcp-server
                        .kagent:3000   True
```

kagent MCPServer has no `status.url` — kro reads the Service, extracts the port, and composes it.

THE VISION

A Discoverable Agentic Registry

Agents query the cluster for available MCP servers. Capabilities become **first-class citizens** in Kubernetes.

KEY TAKEAWAYS

What We Showed Today

- AI Agents are a natural fit for Kubernetes CRDs
- Projects like kagent, Agent Sandbox, and Sympozium already provide MCP CRDs
- Different projects = different status contracts
- kro unifies them into a consistent API
- A cluster-scoped registry makes MCP servers discoverable
- Same pattern works for Skills, Sandboxes, and cloud resources

Links

kagent: github.com/kagent-dev/kagent
Agent Sandbox: github.com/kubernetes-sigs/agent-sandbox
Sympozium: github.com/sympozium-ai/sympozium
kro: github.com/kro-run/kro

@csantanapr

MCP Dev Summit 2026

Thank You

Kubernetes-Native Agent Discovery: A Unified Registry for MCP Servers and Skills

Carlos Santana · AWS
Connect on LinkedIn →