agentctl

A CLI for managing AI agents and MCP servers like infrastructure

Andrew Espira — Founding Engineer @ Kustode

Conf42 Cloud Native 2026

Part 1 — The problem

You run:

- **Ollama** locally for inference
- **LiteLLM** proxy routing to multiple providers
- **3+ MCP servers** (Cursor, healthcare billing, GitHub)
- **Custom agents** in terminal tabs

No unified view. No lifecycle. No health checks. No tagging.

> kubectl for pods. docker for containers. helm for charts. **Nothing for AI services.**

Part 2 — Registration & lifecycle

One binary, typed metadata, persistent state in `~/.agentctl/state.json`.

```
agentctl agent add --name kustode-denial-agent \
  --type http --endpoint http://localhost:8080/api/agent \
  --health http://localhost:8080/health \
  --confidence 0.92 --tags kustode,rcm,ml

agentctl agent start kustode-denial-agent
agentctl agent healthall
agentctl mcp add --name billing-mcp --transport stdio \
  --command "node dist/index.js"
```

Same verbs for agents, MCP servers, API calls. Name or ID-prefix resolution.

Part 3 — Auto-discovery

`agentctl discover` scans three sources in parallel:

- **Processes** — ~30 signatures: Ollama, LM Studio, LiteLLM, LangServe, CrewAI, `node.*mcp`, dev servers
- **Ports** — 11434, 1234, 4000, 8080, 3000, 5173, 9090 …
- **Configs** — Claude Desktop, Cursor `~/.cursor/mcp.json`, VS Code

```
agentctl discover --auto
```

Idempotent. Already-registered items are skipped.

Part 4 — The dashboard

```
agentctl dashboard
```

Live TUI — refreshes every 3 seconds.

- Agents: running / stopped / error count
- MCP servers: transport, health, last probe
- Recent calls, status, latency
- Confidence scores and SENTINEL tier

`htop` for your AI service layer.

Part 5 — Confidence-gated decisions

Each agent carries a score in **[0, 1]** that maps to a tier:

Tier	Range	Action
🟢 green	≥ 0.85	auto-proceed
🟡 yellow	0.60 – 0.85	notify-proceed
🔴 red	< 0.60	escalate

```
agentctl sentinel enable kustode-denial-agent --mode active
```

Shadow mode first. Active mode once you trust it.

Part 6 — SRE observability

Treat AI services like any other service.

- **Prometheus** `:9090` — RED + SLI recording rules, burn-rate alerts
- **Grafana** `:3000` — fleet, SLO, SENTINEL, MCP dashboards
- **Jaeger** `:16686` — OTLP tail-sampling: keep errors + slow + red-tier traces
- **Alertmanager** `:9093` — route by team, severity, SLO

```
agentctl metrics serve --addr :9091
agentctl metrics trace --endpoint http://localhost:4318
```

Part 7 — Simulated incident

Inject red-tier events, watch the system react.

- **Burn-rate alert** fires on `agent:error_budget_burn_rate:1h`
- Tail-sampled error traces land in Jaeger
- SENTINEL quality dashboard degrades
- Alertmanager routes to the right team

Closing the loop: **observability gates automation**.

Thank you

Install `go install github.com/espira/agentctl@latest`

Code github.com/espira/agentctl

Contact LinkedIn / Andrew Espira

Zero dependencies. Zero daemon. Pure Go stdlib + one SQLite read for Cursor.