Running Agentic AI in Production

From Chat to Orchestrated Multi-Agent Systems

Tobias Weiss

Agenda

1. The Evolution: Chat → Agent → Multi-Agent
2. Model Orchestration — LiteLLM as the Central Proxy
3. Agent Framework — OpenCode
4. Agent Orchestration — Oh-My-OpenAgent
5. Skills, Plugins & MCP — Extensibility
6. ACP — The Agent Client Protocol
7. Infrastructure — Ansible Automation
8. Wrap Up

Chat → Agent → Multi-Agent: Three Stages of AI Integration

Stage	What	Example
Chat	One prompt, one response	ChatGPT in the browser
Agent	Tool use, context-aware	OpenCode CLI
Multi-Agent	Orchestration, delegation	Oh-My-OpenAgent + ACP

From experimental chat to reproducible production workflows.

Production AI Infrastructure Must Be Secure, Controllable, Integrable

Requirements:

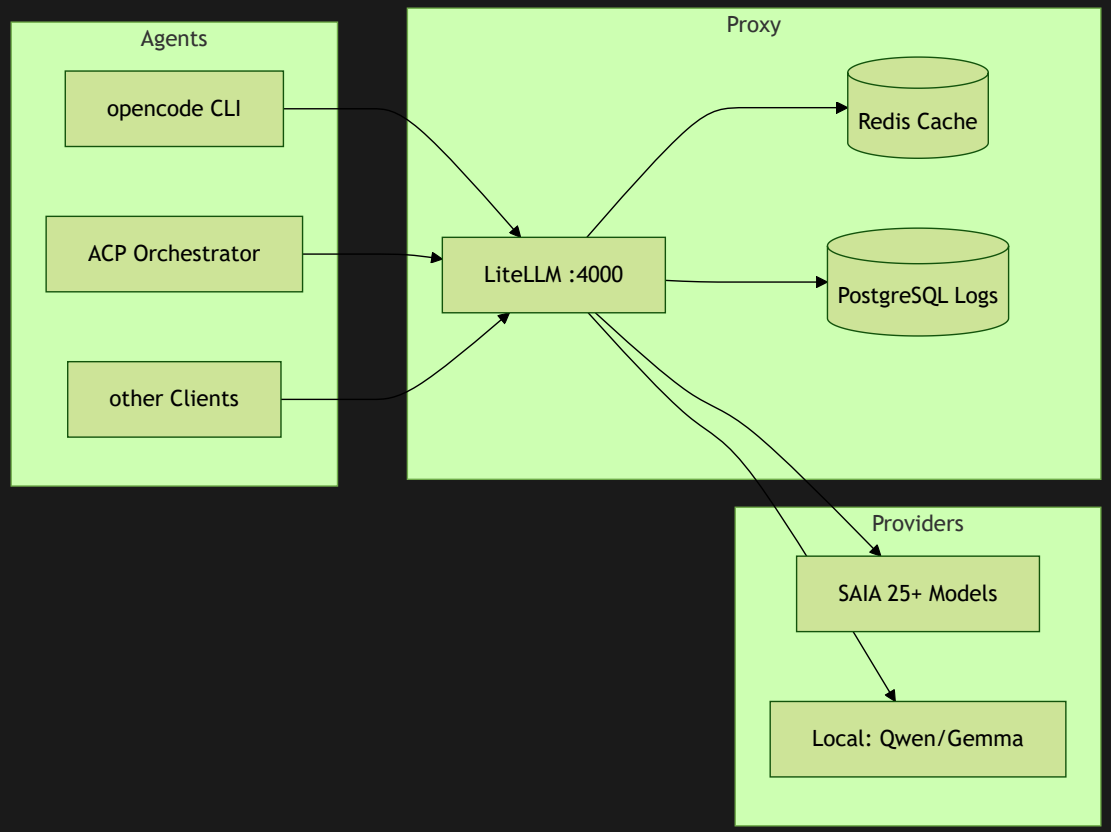
- Secure — Data sovereignty, GDPR compliance
- Controllable — Traceable agent decisions
- Integrable — APIs, existing systems, CI/CD

What we built (2026):

- Local GPU cluster: Qwen3.5 397B, Gemma 4 26B
- LiteLLM proxy as unified API layer (25+ models)
- Neo4j Knowledge Graph with 4 indexed repositories
- ACP-based multi-project orchestration

Model Orchestration — LiteLLM Proxy

One Endpoint, 25+ Models — Zero Vendor Lock-In



- Agents never know which provider serves them
- Swap models without code changes

LiteLLM Handles Fallback, Caching, Logging, and Rate Limiting

- No vendor lock-in: swap models without code changes
- Automatic fallback: SAIA down → Local — zero interruption
- Unified logging: every API call in PostgreSQL (cost, latency, tokens)
- Redis caching + rate limiting: identical prompts computed once, budget per model

Three Providers Cover Every Use Case

Local GPUs for sovereignty and fast tasks, SAIA for primary agent work and heavy reasoning.

Provider	Models	Highlights
Local GPU	Qwen3.5 397B, Gemma 4 26B	Data sovereignty, no VPN for local tasks
SAIA	GLM 5 Turbo, GLM 5.1, GLM 4.7, GLM 4.6V	Primary agent models, vision
SAIA	Mistral Large 3, GPT OSS, Devstral 2, DeepSeek R1, Qwen3.5 122B	Flagship reasoning, code-gen, chain-of-thought

Agent-to-Model Mapping — Each Agent Gets Its Optimal Model

Category	Model	Why
ultrabrain	GLM 5 Turbo	Hardest reasoning tasks
deep	GLM 4.7	Complex implementation
visual-engineering	Qwen3.5 122B	UI/UX, styling, animation
quick	Gemma 4 26B	Trivial fixes (fast, free)
writing	GLM 5.1	Documentation, articles

6 more categories follow the same pattern: right model, right cost. Up to 30 background agents simultaneously.

Agent Config Is Decoupled From Provider Infrastructure

Every agent model is a LiteLLM alias. The agent doesn't know where the model runs.

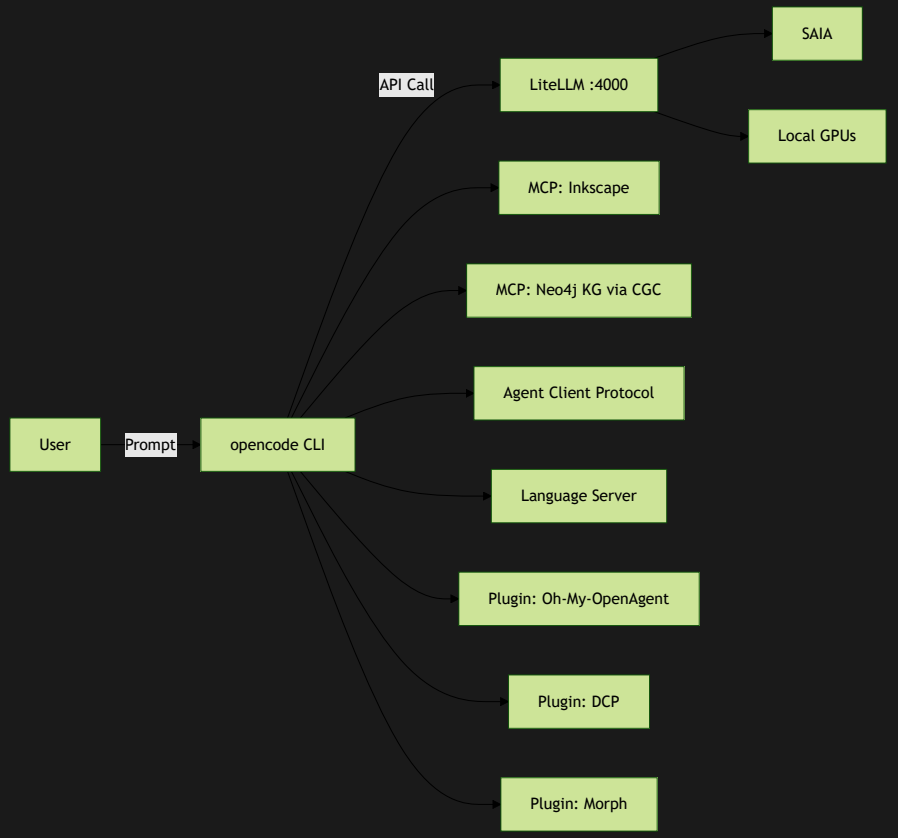
```
{
  "provider": {
    "litellm": {
      "models": {
        "glm-5-turbo": { "name": "GLM 5 Turbo (SAIA)" },
        "saia/gpt-oss-120b": { "name": "SAIA GPT OSS 120B" },
        "qwen3.5-397b": { "name": "Qwen3.5 397B (local)" }
      },
      "options": {
        "baseURL": "http://127.0.0.1:4000/v1",
        "timeout": 120000,
        "maxRetries": 5
      }
    }
  }
}
```

Agent Framework — OpenCode

OpenCode: CLI Agent That Reads, Writes, and Tests Code Autonomously

- Multi-model routing via LiteLLM (one config, any model)
- 30+ parallel background agents with context compaction
- LSP integration + AST-grep (real code understanding, 25 languages)
- Plugin system: Oh-My-OpenAgent, DCP, Morph

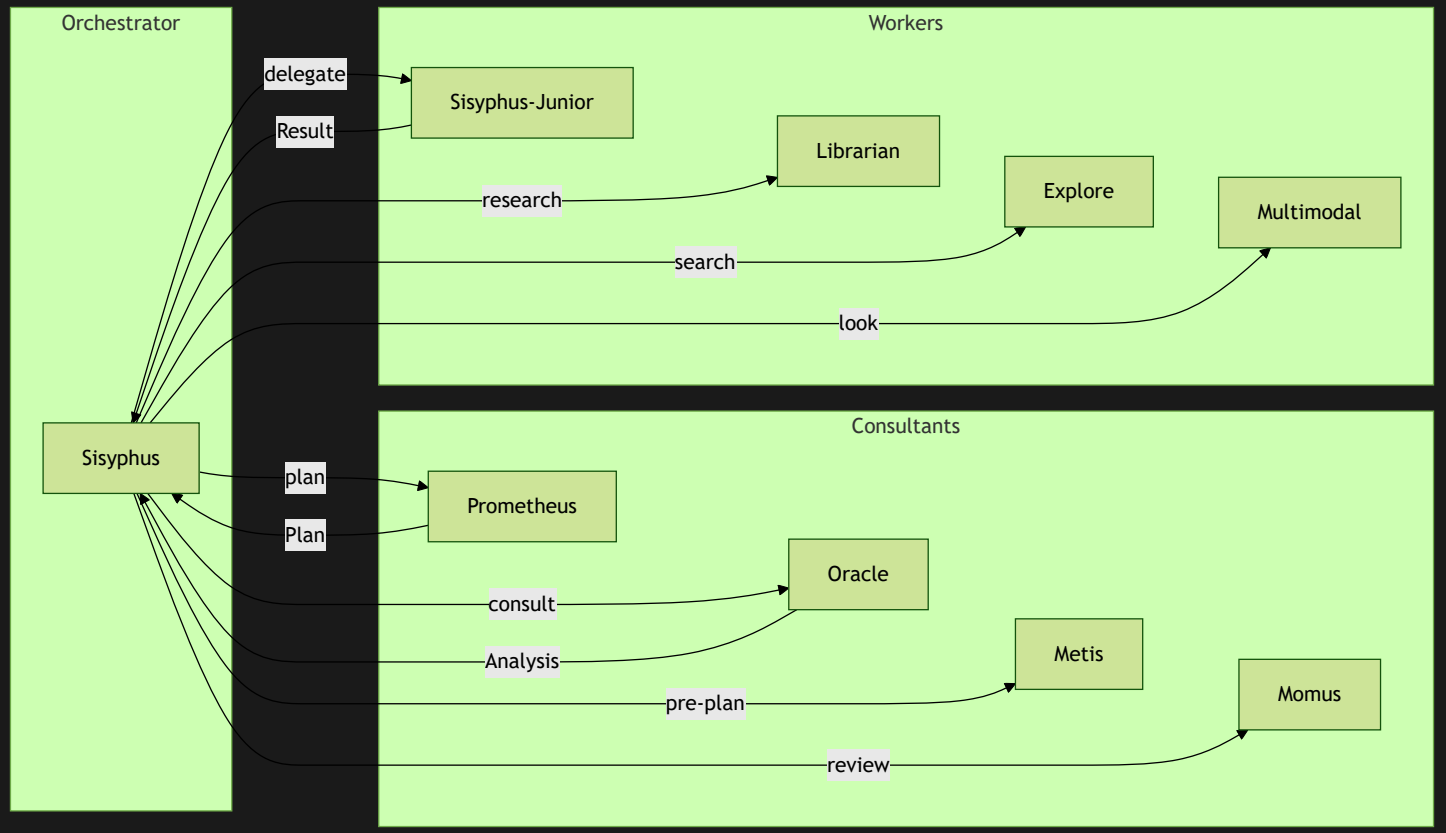
OpenCode Integrates Models, MCP Servers, Plugins, and LSP in One CLI



Three plugins extend OpenCode: Oh-My-OpenAgent (multi-agent), DCP (context management), Morph (runtime config).

Agent Orchestration — Oh-My-OpenAgent

Sisyphus Delegates to Specialised Agents by Role



- Each agent runs on its optimal model via LiteLLM
- 10 agents cover orchestration, planning, research, code, and review

Orchestrator + Consultant Agents — The Brain

Agent	Role	Model	Why This Model
Sisyphus	Orchestrator	GLM 5 Turbo	Strong reasoning for delegation
Prometheus	Planner	GLM 4.7	Structured work breakdowns
Oracle	Consultant	GPT OSS 120B	High-IQ read-only analysis
Metis	Pre-planning	GLM 5 Turbo	Ambiguities and edge cases
Momus	Reviewer	Qwen3.5 122B	Plan quality assurance

Worker + Research Agents — The Hands

Agent	Role	Model	Why This Model
Sisyphus-Junior	Worker	Devstral 2 123B	Code-gen, cheaper than orchestrator
Atlas	Worker	GLM 5 Turbo	Implementation execution
Librarian	Reference search	Qwen3.5 35B	External docs / API research
Explore	Code search	Gemma 4 26B	Fast local grep (cheap)
Multimodal	Vision	GLM 4.6V	Image analysis, screenshots

Real Workflow: 7 Steps, 6 Models, 1 Endpoint

User: "Fix the scheduler bug in the neo4jknowledgebase repo"

1. Sisyphus (GLM 5 Turbo) — Detects bugfix intent
2. Explore (Gemma 4 26B) — Finds scheduler code
3. Librarian (Qwen3.5 35B) — Researches embedding API
4. Oracle (GPT OSS 120B) — Root cause: model removed from config
5. Sisyphus-Junior (Devstral 2 123B) — Implements fix
6. Sisyphus (GLM 5 Turbo) — Verifies build passes

Cost: ~6 different models, ~15 API calls, all through one LiteLLM endpoint.

Skills, Plugins & MCP — Extensibility

MCP Connects Agents to the Outside World

MCP Server	Function	Transport
Inkscape	SVG creation and manipulation	local
CodeGraphContext	Code-indexed knowledge (Neo4j KG)	local (SSH tunnel)
Playwright	Browser automation, screenshots	built-in
Git	Repository operations	built-in
File System	Read/write access to workspace	built-in

CodeGraphContext indexes 4 repos (131 files, 736 functions) via Neo4j.

Skills: Reusable Workflow Templates

Skill	Description
graphwiz-reporter	Autonomous: KG + RSS → research → published article
test-driven-development	Test first, then implement
systematic-debugging	Structured error analysis before fixes
dispatching-parallel-agents	Parallel task distribution
verification-before-completion	Evidence before assertions

6 more skills for git worktrees, code review, planning, brainstorming, email, and graphics.

ACP — Agent Client Protocol

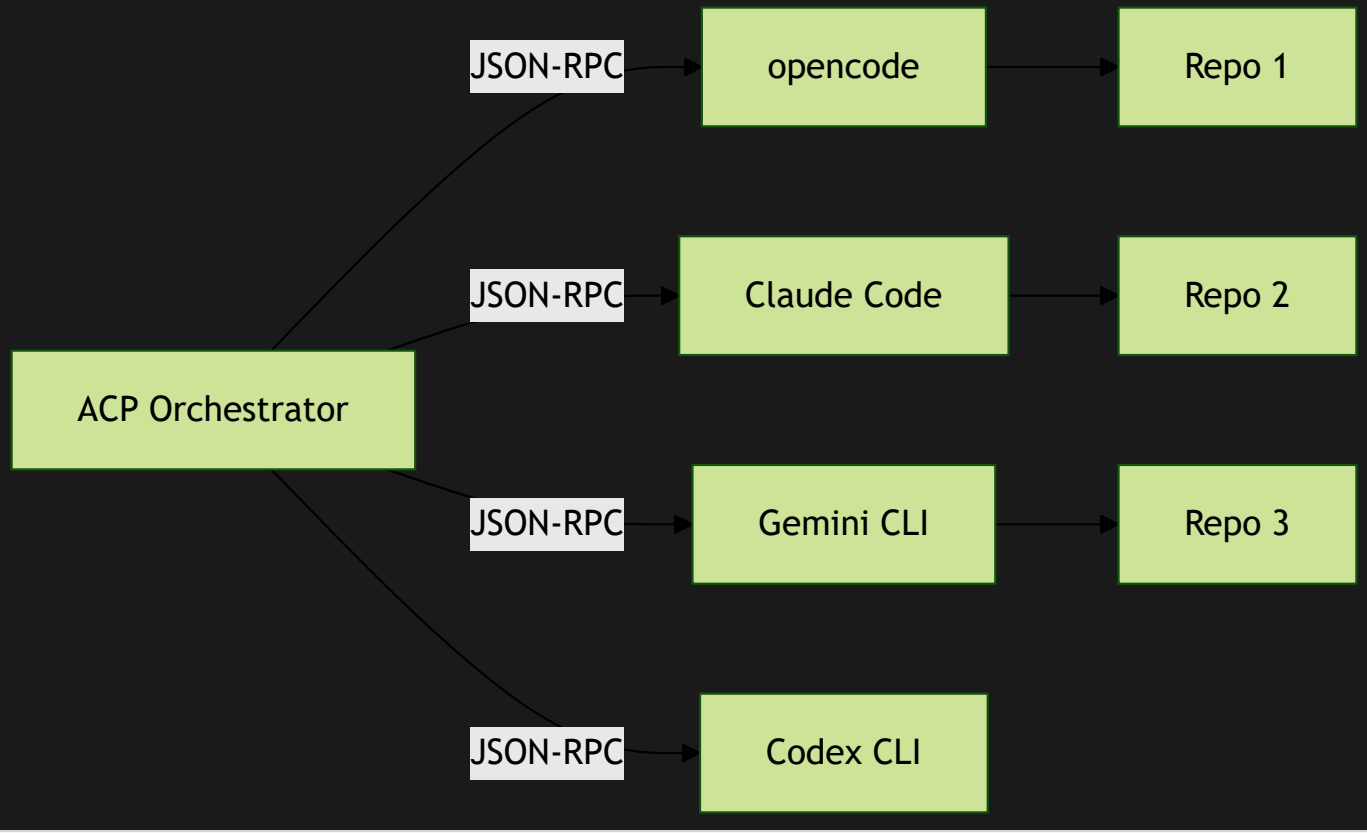
ACP Is the LSP Moment for AI Agents — One Protocol, Any Editor, Any Agent

ACP (Agent Client Protocol, by Zed Industries): JSON-RPC standard for communication between clients and agents.

```
Client (Editor/CLI) → JSON-RPC (stdio/HTTP) → Agent (Claude/Codex/Gemini/OpenCode/...)
```

Status (April 2026): 30+ agents, 40+ clients.

ACP in Practice — Multi-Project Orchestration




```
# Batch prompt across all Python projects
acp run "check for outdated dependencies" --tags=python

# Autonomous improvement loop
acp loop next-graphwiz-ai --max-iter 100 --rotate 25
```

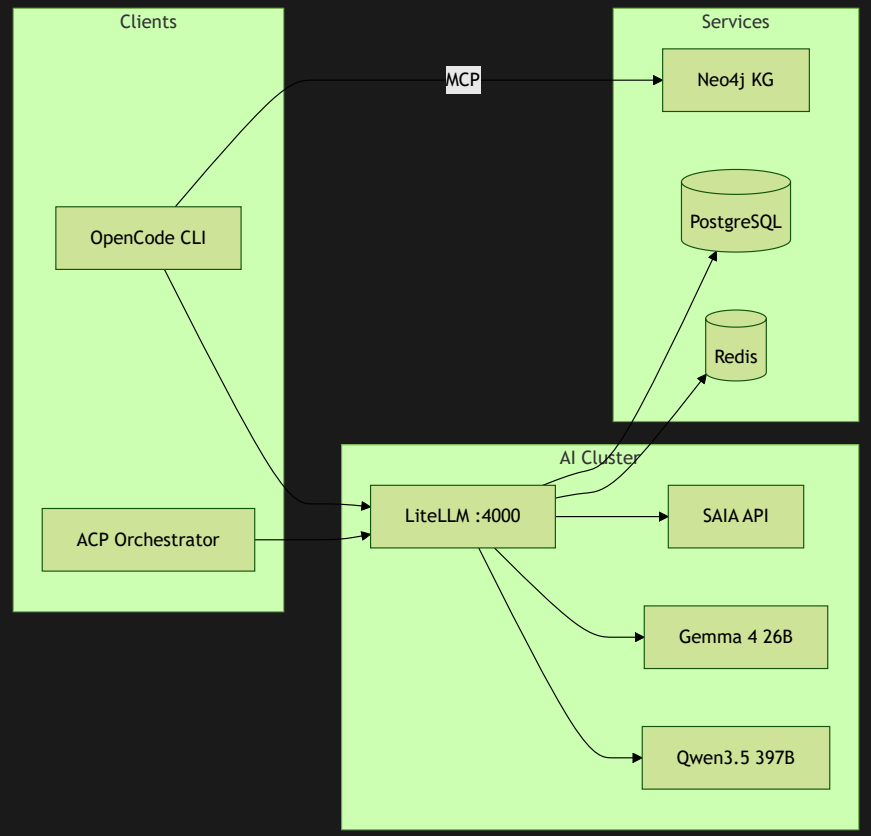
Infrastructure — Ansible Automation

9 Hosts, Everything as Code — Ansible Playbooks

Playbook	Function
litellm-proxy	Docker Compose: LiteLLM + PostgreSQL + Redis
opencode-deploy	Build & install OpenCode from source (Go)
opencode-sync	Sync agent configs to all hosts
knowledge-graph	Neo4j + CodeGraphContext deployment
vpn-hub / vpn-peers	WireGuard mesh networking
traefik	Reverse proxy + TLS (Let's Encrypt)

```
ansible-playbook site.yml --limit ai_cluster
```

Traefik → LiteLLM → GPUs → MCP: The Full Infrastructure Stack



Wrap Up

The Architecture: Three Decoupling Layers

Layer	Tool	Role
Models	SAIA, Local GPUs	25+ LLMs, 2 providers
Orchestration	LiteLLM Proxy	One API, fallback, logging
Agent	OpenCode CLI	Reads, writes, tests code
Multi-Agent	Oh-My-OpenAgent	10 specialised agents
Cross-Editor	ACP	Standardised agent protocol
Extension	MCP + Skills + Plugins	Tool integration
Infrastructure	Ansible	Reproducible deployment

LiteLLM decouples models from agents. Oh-My-OpenAgent decouples orchestration from implementation. ACP decouples agents from editors.

Resources — Core Tools

- SAIA: https://docs.hpc.gwdg.de/services/saia/index.html
- OpenCode: https://github.com/anomalyco/opencode
- Oh-My-OpenAgent: https://github.com/code-yeongyu/oh-my-openagent/
- LiteLLM Proxy: https://docs.litellm.ai/docs/proxy/proxy
- ACP: https://agentclientprotocol.com

Resources — Extensions & Infrastructure

- CodeGraphContext: https://github.com/tobias-weiss-ai-xr/CodeGraphContext
- SAIA Plugin: https://codeberg.org/graphwiz-ai/opencode-saia-plugin
- Superpowers Skills: https://github.com/obra/superpowers
- Neo4j Knowledge Graph: https://neo4j.com/docs/
- MCP Protocol: https://modelcontextprotocol.io/
- Ansible: https://docs.ansible.com/

Questions & Discussion

Thank you!

Discussion topics:

1. Governance rules for agentic systems?
2. Running your own models on local hardware?
3. Centralised vs. decentralised agent configuration?
4. Cost management across multiple providers?

License