Agent Orchestration Harnesses

The Plugin Layer That Makes Coding Agents Work

Comparing multi-agent plugins for OpenCode, Claude Code, and beyond

Focus: Solo dev / Multi-project / Production apps / Vibe coding

May 2026

Why Orchestration?

A base coding agent (OpenCode, Claude Code, Codex) is one brain. These plugins give it a team.

Task Decomposition

Break big goals into plannable, parallelizable steps. One prompt shouldn't mean one monolithic action.

Agent Specialization

Different roles for different work: planner, coder, reviewer, tester, security auditor. Each with tuned prompts and model routing.

Execution Gating

Nothing ships without review. Code written by one agent is reviewed by another, tested by a third. Quality gates enforce discipline.

Context Isolation

Agents don't clobber each other's files. Worktree isolation, file reservations, or serial phased execution prevent collisions.

Session Persistence

Pick up where you left off. Learnings, plans, and progress survive context death and carry across sessions.

Multi-Project Coherence

Switch between projects without losing conventions, patterns, or accumulated knowledge. Per-project config that persists.

The Contenders

OPENCODE PLUGINS

Oh-My-OpenAgent

11 agents, Sisyphus orchestrator

OmO Slim

Lightweight fork, token-efficient

OpenSpec

Architecture planning agent

OpenCode-Swarm

Architect-led gated phases

OpenAgentsControl

Plan-first, approval-based

Swarm-Tools

Learning swarm + hivemind

KDCO Workspace

16-component bundle

CLAUDE CODE / CROSS-PLATFORM

Ruflo

100+ agents, 3 topologies

Claude MPM

47 agents, PM coordinator

Composio AO

Fleet orchestrator, CI-aware

Symphony

OpenAI spec, board-to-agent

AgentsMesh

Enterprise multi-agent platform

Architecture Patterns

Four distinct approaches to the same problem

NAMED SPECIALISTS

OmO, OmO Slim, Claude MPM

Fixed roster of named agents (Sisyphus, Prometheus, Atlas...) with predefined roles, model routing, and orchestration pipelines. The "virtual dev team" approach.

GATED SERIAL PHASES

OpenCode-Swarm, OpenAgentsControl

Architect plans, coder implements, reviewer checks, tester validates — in strict sequence. Nothing advances until the gate passes. Safety over speed.

PARALLEL SWARM

Ruflo, Composio AO, Swarm-Tools

Spawn N agents in isolated worktrees, each with file reservations. Coordinate via message queues. Learn from outcomes. Maximum throughput.

DISPATCH FROM BOARD

Symphony, AgentsMesh

Project management board becomes the control plane. Agents pull tickets, execute, open PRs. The orchestrator monitors CI, rebases, and resolves conflicts.

Head-to-Head: Harness Comparison

Harness	Base Agent	# Agents	Pattern	Parallelism	Learning	Solo-Dev Fit	Complexity
Oh-My-OpenAgent	OpenCode	11	Named Spec.	Aggressive	Sessions	Good	High
OmO Slim	OpenCode	6-8	Named Spec.	Background	Sessions	Very Good	Medium
OpenSpec	OpenCode	1 (planner)	Planning only	None	Specs	Good (addon)	Low
OpenCode-Swarm	OpenCode	~20	Gated Serial	None (serial)	Config	Good	Medium
OpenAgentsControl	OpenCode	2 core	Plan-First	None	Context	Very Good	Low
Swarm-Tools	OpenCode	Dynamic	Parallel Swarm	Full (workers)	Hivemind	Good	Medium
KDCO Workspace	OpenCode	16 comp.	Bundle	Worktrees	Plugins	Medium	High
Ruflo	Claude Code	100+	3 Topologies	Full (swarm)	Self-learning	Overkill?	Very High
Claude MPM	Claude Code	47+	Named Spec.	Agent Teams	Skills	Good	Medium
Composio AO	Any CLI	Fleet	Parallel Fleet	Full (worktrees)	CI feedback	Overkill?	High
Symphony	Codex	Per-ticket	Board Dispatch	Full	Board state	Medium	Medium
AgentsMesh	Any CLI	Platform	Enterprise	Full	Platform	Overkill	Very High

Solo-Dev Fit = simplicity-to-power ratio for single-person workflow | Complexity = setup + ongoing config burden

Leader Quadrant

Purpose / Vision vs. Ability to Execute

QUADRANT KEY

Leaders

Broad orchestration vision AND proven execution. Full pipelines, model routing, production-tested.

Strong Executors

Do their specific job very well. May be narrower in scope but reliable and battle-tested.

Visionaries

Ambitious architecture, still maturing. High ceiling, evolving stability.

Niche / Focused

Solve one part of the puzzle excellently. Best as complements, not standalone.

Your Stack: Solo Vibe-Coding Dev

Indie dev | Production apps | 3-5 active projects | Flow state priority

Low Config Overhead

Install and go. Can't spend a day configuring 16 components before writing code.

Project Switching

cd into a folder and the harness already knows the project. Per-directory config.

Quality Without Ceremony

Planning + review built in, but not 6 approval gates. Trust the solo dev.

Learning Across Sessions

Patterns from Project A should inform Project B. Accumulated knowledge matters.

FIT SCORES FOR YOUR WORKFLOW

OmO Slim	Best balance: named agents without OmO-full complexity. Token-efficient, per-folder.
OpenAgentsControl	Plan-first + approval fits solo quality needs. MVI keeps costs low.
OpenCode-Swarm	Gated phases enforce discipline. Architect mode great for planning before coding.
Oh-My-OpenAgent	Maximum power, but setup cost and collision bugs hurt solo flow.
Claude MPM	Strong for Claude Code users. 47 agents may be more than solo dev needs.
Swarm-Tools	Unique learning/hivemind. Best if you want accumulated cross-project knowledge.
Ruflo	Enterprise-grade power but high complexity. Overkill unless running many parallel tasks.

Recommended Harness Strategy

Layered approach: start lean, add complexity only when the project demands it

TIER 1: DAILY DRIVER *Start every project here*

OmO Slim or OpenAgentsControl

OmO Slim gives you named specialists (Explorer, Oracle, Librarian) with minimal token overhead. OpenAgentsControl gives plan-first discipline with just 2 core agents. Both are per-folder config — cd into project, harness knows the context. Choose OmO Slim if you want agent variety; choose OAC if you want approval gates and MVI token efficiency.

TIER 2: WHEN YOU NEED STRUCTURE *Complex features, multi-file refactors*

OpenCode-Swarm + OpenSpec

Engage OpenSpec to plan architecture first, then let Swarm's gated phases (architect → coder → reviewer → tester) execute with discipline. The serial gating prevents the collision bugs you'd hit with OmO-full's parallel execution. Best for production-critical changes.

TIER 3: BIG PARALLEL PUSHES *Major refactors, multi-repo work*

Swarm-Tools (for learning) or Ruflo (for scale)

When a task genuinely benefits from parallel agents: Swarm-Tools if you want the hivemind learning system (patterns accumulate across projects); Ruflo if you need raw scale and don't mind the config overhead. Reserve this tier — most solo-dev work doesn't need fleet orchestration.