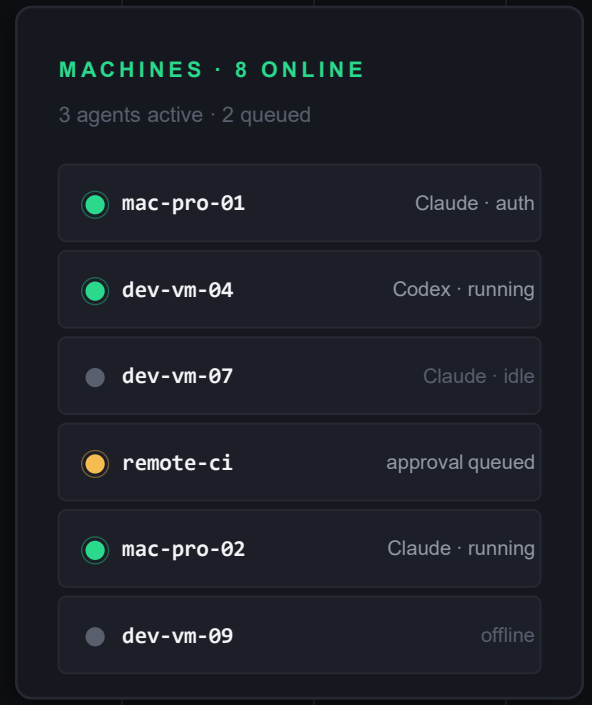
Controlum

The control plane for your AI coding agents.

Run, coordinate and govern AI coding agents across every machine your team owns.

Then turn runtime telemetry into engineering intelligence.



STAGE

Working MVP · v3.2

IN PRODUCTION

we run it on client work

PLATFORMS

macOS · Linux · Windows

THE SHIFT

AI agents became infrastructure. The operating layer is missing.

Claude Code and Codex aren't experiments anymore — teams already run dozens of agent sessions across laptops, VMs and remote shells. Every previous infrastructure shift got an operating layer. This one hasn't, yet.

LAYER	LAYER	LAYER	LAYER	LAYER
Containers	Services	Devices	Code	AI agents
OPERATING LAYER	OPERATING LAYER	OPERATING LAYER	OPERATING LAYER	OPERATING LAYER
Kubernetes	Datadog	Tailscale	GitHub	—

Every infrastructure shift gets an operating layer. **AI-native engineering needs one too.**

THE MISSING LAYER

Agents now write the code. Nobody is operating them.

Without a runtime layer, every team is rebuilding the same scaffolding by hand — and discovering the same gaps the hard way.

No visibility into the runtime

Sessions live in headless terminals across machines. You can't see what's running, where, or whether it's stuck waiting on you.

No coordination across machines

Skills, prompts, secrets and agent topology are copied by hand. Every host drifts. Workspaces can't be resumed elsewhere.

No governance at the runtime edge

Dangerous tool calls — rm -rf, leaked keys, unsanitised curl | sh — are caught only after the agent has already executed them.

AND THE COST IS REAL

~3×

more production incidents per merged change[1]

+154%

growth in PR size with AI assistance[1]

33%

of developers don't trust AI-generated code[1]

[1] DORA — State of AI-assisted Software Development 2025; Qodo State of AI Code Quality 2025. Full sources on the final slide.

THE SOLUTION

The operating layer for AI-native engineering.

A lightweight daemon on every machine, a single coordination plane, and the four primitives an agent runtime needs. Intelligence emerges from the data this layer captures — but the layer itself is the product.

Run

Launch and coordinate Claude / Codex sessions across every machine your team owns.

Observe

Live runtime visibility into sessions, prompts, tools and state — including sessions you didn't start through Controlum.

Govern

Policies, approvals and per-session sandboxes catch dangerous tool calls before they execute.

Standardize

Versioned skills, prompts, secrets and agent topology — defined once, propagated everywhere.

...AND THEN

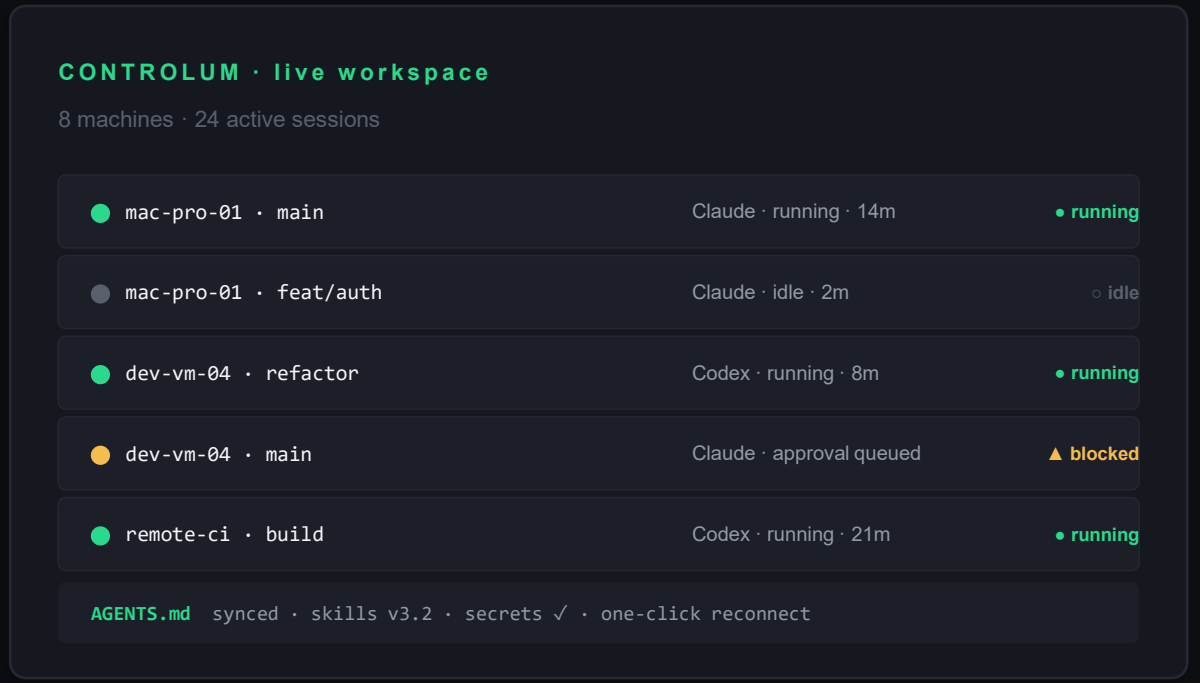
Learn — runtime telemetry compounds into engineering intelligence: rework cycles, context drift, prompt quality. The moat layer. *But the runtime comes first.*

WHAT THIS LOOKS LIKE

The workflow nobody else restores.

AI work happens in transient sessions on scattered machines. Lose your laptop, switch contexts, hand off mid-task — and the in-flight state evaporates. Controlum treats agent workspaces as durable infrastructure.

- **Resume agent sessions across machines**
- **Discover sessions not launched via Controlum**
- **Restore full AI workspaces — context, skills, secrets**
- **Sync agent context and skills globally**
- **Operate distributed coding agents like infrastructure**

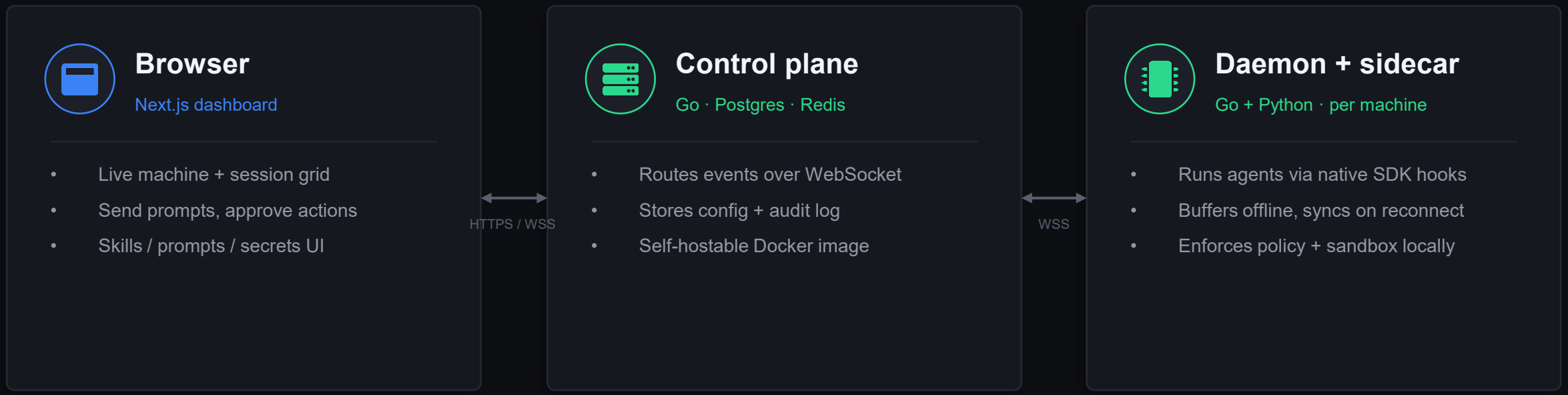


This is the demo. Nothing else on the market reconstructs distributed agent state.

HOW IT WORKS

A thin daemon. A shared brain.

Three layers, one connection model. The daemon does the work locally; the control plane coordinates; the browser is just a window.



Why SDK hooks, not raw stdin: precise lifecycle signals (prompt submitted, tool pre/post, idle, blocked) — and the agent's own TUI stays intact.

PRODUCT

A working MVP — and our own daily driver.

We run Controlum in production ourselves — it powers our own paid client-delivery work. The roadmap is written by the team that lives inside the problem every day.

Machines

Auto-detected hardware, OS, package managers and installed CLIs per host.

Live sessions

Structured event stream + optional xterm.js terminal view.

Session discovery

Surfaces every Claude / Codex session — even those not started via cu.

Skills & prompts

Versioned SKILL.md and CLAUDE.md / AGENTS.md libraries.

Secrets

AES-GCM at rest, X25519 + ChaCha20 in transit to the daemon.

Policy engine

Rule classifier at the PreToolUse hook + custom rules + audit log.

Sandboxing

Per-session Docker isolation with bridge / none / allowlist networking.

Marketplace

One-click import of curated skills and subagents from upstream repos.

All eight capabilities exist in the codebase today, across a Go · Python · TypeScript monorepo, on macOS / Linux / Windows.

MACRO CONVERGENCE

The ecosystem is converging on an agent runtime layer.

Not because of hype — because every adjacent piece of the stack is shifting in the same direction. Five concrete, observable patterns:

Coding agents become default

Claude Code · Codex · Cursor background agents · OpenHands · Devin-like systems

Context becomes infrastructure

AGENTS.md · skills systems · MCP · persistent memory · workflow-defined prompts

Multi-agent workflows emerge

Parallel agents · task delegation · autonomous retries · remote execution

Runtime governance becomes mandatory

Approvals · sandboxing · audit logs · secrets isolation · policy enforcement

Workflows shift IDE → runtime

From editor-centric to runtime-centric. The agent, not the developer's IDE, is now the active unit.

Five independent shifts pointing at the same gap. That's not a trend — that's a category forming.

THE EMPTY LAYER

Every layer of the stack has its tool. One doesn't.

Mapped against the layers of the engineering stack, the absence is conspicuous.

LAYER	TOOL	STATUS
Code hosting	GitHub · GitLab	solved
CI / CD	GitHub Actions · CircleCI	solved
Containers	Kubernetes · Docker	solved
Observability	Datadog · Grafana	solved
Engineering analytics	LinearB · GetDX · Faros	partial
AI agent runtime	—	EMPTY

Controlum fills the runtime layer. Not competing with analytics vendors — operating one layer below them.

THE MOAT

Owning the runtime owns the data.

Every vendor in adjacent categories reads the same external signals: git, tickets, surveys. The runtime layer sees something they structurally cannot — and the data asset compounds as fast as agents do.

WHAT EVERYONE ELSE SEES

- Git commits, PR diffs, merge timestamps
- Ticket transitions, sprint flow
- Survey-based satisfaction signals
- Self-reported time savings

WHAT THE RUNTIME LAYER SEES

- Versioned agent-context snapshots (AGENTS.md / skills)
- Raw session transcripts, cwd, tool calls
- Active-vs-idle session timing
- Prompt lineage, retries, abandonments

Adjacent vendors won't reach into the OS to get this data — it's outside their architectural model. The gap is structural.

WHY NOW

The conditions for this category exist now — not later.

Six independent signals pointing the same direction. Categories form when usage outruns tooling — and we're past that point.

We are our own first customer

Controlum runs in production today — powering our paid client-delivery work. The hardest possible early proof.

The behaviour already exists

Teams are running Claude Code & Codex across many machines now. We're instrumenting a habit, not creating one.

The pain is named, not solved

DORA 2025 explicitly flagged observability, rework and trust as the open problems of AI-assisted engineering.

The category has no name yet

"Agent-ops" has no incumbent and no standard. Naming and defining the category is itself the opportunity.

The data asset compounds

Versioned agent-context + transcripts get more valuable with every session — a moat that widens over time.

Bottom-up entry is frictionless

One-line install, native CLI, self-hostable. Adoption needs no procurement cycle and no workflow change.

BUSINESS MODEL

Land with a free daemon. Expand on the operating layer.

Bottom-up adoption: a developer installs cu in one line. Teams pay for coordination and governance. Enterprise expands into the intelligence layer.

RUNTIME

Free

Individual

- Single user, unlimited machines
- Live sessions + skills library
- Self-hostable, full CLI

OPERATING LAYER

Team

Per seat / month

- Shared skills, prompts, secrets
- Policy engine + audit log
- Cross-machine session restore
- Coordination + sandboxing

GO-TO-MARKET WEDGE

INTELLIGENCE LAYER

Enterprise

Custom

- SSO, RBAC, multi-tenancy
- Runtime telemetry exports
- Rework / prompt-quality / drift
- Context-capital benchmarking

ROADMAP

From daemon to operating system.

Five phases, each a strict prerequisite for the next. We're not adding features — we're claiming a layer at a time.

1 — Runtime visibility

Make AI agent activity observable.

- Session discovery
- Live runtime telemetry
- Cross-machine orchestration
- Prompt / tool lifecycle hooks
- Shared skills and secrets

2 — Operational governance

Make AI agents governable.

- Policy engine
- Sandboxing
- Approvals and audit trails
- Team-wide configuration
- Multi-tenant deployment

3 — Distributed agent ops

Make AI-native engineering operationally scalable.

- Workspace restoration
- Session continuity
- Task / session lineage
- Agent coordination
- Runtime replay

4 — Intelligence layer

Turn runtime telemetry into proprietary engineering intelligence.

- Rework-cycle detection
- Active vs idle analysis
- Prompt-quality scoring
- Defect attribution
- Context drift analysis

5 — Category platform

Define the operating standards for AI-native engineering.

- Cross-team benchmarking
- Industry reports
- Context-capital analytics
- Agent performance datasets
- AI-native operational standards

NOW + NEXT | LATER | FUTURE

WHY THIS SEQUENCING

The order is the strategy.

Every phase is a strict prerequisite for the next — not a wishlist. Each layer creates the substrate the next one needs.

1	**Visibility comes first**	You cannot govern, coordinate or optimize agents you cannot see. The runtime has to surface itself before anything else is possible.
2	**Governance depends on runtime control**	Policies and approvals only work once the runtime layer can intercept tool calls. You can't enforce what you can't see.
3	**Operations emerge after standardization**	Session continuity and workspace restoration require shared context infrastructure — versioned skills, secrets and agent topology.
4	**Intelligence requires historical runtime data**	The moat compounds only after the runtime becomes the source of truth. Analytics on stale, gameable signals isn't a moat.
5	**Standards emerge last**	Industry benchmarks and operational standards only matter once enough teams operate on the same substrate. Standards follow scale, not the other way around.

Controlum

The operating layer is up for grabs.

A working MVP we run in production, a defensible runtime data asset, and a category that doesn't yet have a name. We're looking for the partners and capital to turn it into the default.

LET'S TALK

controlum.pw
app.controlum.pw
github.com/dimkk/controlum

Capital

to fund production launch and go-to-market

Design partners

AI-native teams running agents at scale

Talent

founding engineers and a distribution lead

SOURCES DORA 2025 · SPACE (ACM Queue) · DX Core 4 (GetDX) · Faros AI · CodeRabbit · Qodo · Octoverse 2025 · contextarch.ai