

Agent Autoloop in Practice: What a 6,000 Star Unattended Agent Dispatcher Does, and What It Does Not

What happens to your coding agent after you close the terminal? Agent Autoloop answers that for 6,362 stargazers in twenty days, and the answer is a cron entry, a deterministic kernel and an approval card.

Agent Autoloop (KKKKhazix/agent-autoloop) is an MIT-licensed, self-hosted dispatcher that runs CLI agents such as Claude Code and Codex from cron rules, GitHub and GitLab webhooks, HTTP callbacks or issue polling. It exits the agent process between rounds and pings you on Feishu, Slack, Discord or desktop when a decision is yours. Created 3 November 2026, it has 6,362 stars and sits at release v1.9.4 of 20 November 2026.

Put plainly: the scheduling is solid, and the thing it schedules succeeds on 25 to 29 percent of hard coding tasks according to OpenAI's November 2026 numbers. Run it on low-stakes, reversible work and it earns its keep.

A kernel that does not read the clock

The design decision I respect most is also the least marketed one. Per the README, the kernel has no runtime dependencies, does not read the system clock, does not use random numbers and makes no outbound requests: plugins connect inward, credentials stay in the plugins, and identical inputs produce identical outputs. All state is derived from an append-only fact log, with log and index written in the same transaction before the write is acknowledged. That is the same discipline I argued for when I compared which agent frameworks actually survive a crash, and it is rare in a project this young.

The published benchmarks (Apple M5 Pro, 24 GB, Node 24.21.0) give 88.78 ms cold start, 59.94 MB idle RSS, 2,394 persisted writes per second, dispatch p99 of 1.82 ms, and 170.49 MB RSS holding a thousand simultaneous long-polls. The repo ships the reproduction commands: `node sim/random.ts 24301 200000` for a 200,000-step simulation of dropped packets and crashes, `node sim/crash.ts 1000 12648430` for a thousand hard kills with no loss of acknowledged records. The README also admits what was not tested: a full power loss was never simulated, and after a format change the first startup rebuilds from the log, which took 43.1 seconds at 1.1 million facts.

Two kernel mechanisms matter more than the throughput numbers. Instruction and data stay separated: external text such as an issue title or body is written to a file as reference material and never concatenated into the prompt; the prompt carries only validated types, ids and links, and an incoming event that tries to set `cwd`, `agent` or `access` is rejected outright. Then the hop-count fuse: every event an agent emits back into the system increments a depth counter that survives human questions and scheduled rechecks, cannot be reset by the agent, and stops dispatch at eight levels with a card to you.

> **What you need to run it**
> Node 24 or newer, Git, and a CLI agent already signed in. One line installs it: `curl -fsSL https://raw.githubusercontent.com/KKKKhazix/agent-autoloop/main/install.sh | sh` on macOS/Linux, `irm https://raw.githubusercontent.com/KKKKhazix/agent-autoloop/main/install.ps1 | iex` on Windows. MIT licence, TypeScript, no cloud server and no inbound ports. The docs recommend API keys rather than personal subscription plans for automated workloads.

Where the zero-token wait actually pays

The loop is four moves. An event arrives (cron, webhook, HTTP, poll) and becomes a standard event. Rules in `~/.agent-autoloop/rules.json` decide which agent handles it in which directory, merging new events on the same subject into one round. The agent works one round, writes its result into its final message and exits. If it needs a consequential call (deleting data, taking a service down, spending money, messaging people), the hub sends a card and the process is gone: no agent running, no tokens burning, for three days if that is how long you take. Your answer resumes the same session.

That suspend-at-zero-cost property is the real product. Everything else, the recipes for morning briefs, inbox triage, website watch and PR delivery via a `deliver` label, is configuration on top of it.

The operational commands are better than I expected from a three-week-old repo. `agent-autoloop check ` validates a draft without touching the running hub. `agent-autoloop rules explain` prints the effective rules and next scheduled runs. `agent-autoloop emit –dry-run` previews who an event would go to without contacting the hub or writing a log line. `agent-autoloop why ` reconstructs any round from the fact log. Pausing a rule skips its schedule and refuses events, and resuming lists what was skipped plus a `replay –since` command that hands each skipped event over exactly once. If `rules.json` is broken or deleted, the hub keeps running the last valid rules and sends you a card about it.

Is Agent Autoloop safe to run unattended?

Not in the sense most people mean. The SECURITY.md, as summarised by aipolix's write-up of the launch, states plainly that prompt injection from external issue text is not solved, that read access can still reach secrets on the machine, and that approval links act as bearer tokens for a limited window. Separating instructions from data reduces the injection surface. It does not close it, because the agent still reads the file.

Then there is the verification gap the README itself lists. Claude Code, Codex, ACP, ZCode and Agentica 1.6.1 have run multi-turn Q&A and cross-process session resumption. Feishu sending, button callbacks and card updates work. But all four webhook templates were

verified only against a local mock server, the GitLab webhook was built to the documented format and never connected to a real GitLab, and the recipe scripts and receipts were tested with local HTTP and a fake Feishu CLI, never end to end on a live account. Gemini CLI 0.67.0 and OpenCode 1.1.2 completed an ACP handshake and nothing beyond it. The connection commands for pi, OpenClaw, Hermes, Cursor and CodeBuddy were transcribed from vendor docs and, in the author's words, not yet run one by one.

WHAT WORKS

- Suspended runs cost nothing: process exits, state lives in the log, session resumes on your answer.
- Deterministic kernel, zero runtime dependencies, crash sim of 1,000 hard kills with no acknowledged record lost.
- Events, agents, channels and CI/CD delivery are all swappable plugins; a generic `command` adapter takes any CLI that prints a final reply.
- Hop-count fuse at depth 8 that the agent cannot reset, plus rejection of events carrying `cwd`, `agent` or `access`.
- Honest docs: the README names what was never tested instead of hiding it.

ROUGH EDGES

- CLI output and default notification strings are still Chinese; the English README dates to 21 November 2026 and recipe instructions remain in Chinese.
- Webhook templates validated only against local mocks; GitLab never connected for real.
- Prompt injection from external issue text explicitly unsolved; approval links are time-limited bearer tokens.
- Version churn: v1.0.0 on 3 November to v1.9.4 on 20 November, and a third-party tracker already lists v1.10.0 on 23 November while GitHub releases show v1.9.4.
- State summaries slow down as history grows; a post-format-change rebuild took 43.1 s at 1.1M facts.

The failure rate the scheduler cannot fix

Autoloop removes the human from the terminal. It does not improve what the agent does once alone there. OpenAI's November 2026 jagged-frontier research reported 25 percent success on writing inference and serving-stack code and 29 percent on building eval, RL and agent environments, against 79 percent on monitoring tasks. A 2026 dataset of 12,400 agent runs found an 18.4 percent mean tool-call failure rate, 39.1 percent of which was schema malformation and 23.4 percent parameter hallucination. I went through the wider adoption picture in the November 2026 adoption numbers.

Put those together. A dispatcher that reliably starts a 29-percent-success task every morning at 09:00 is a dispatcher that reliably produces wrong output most mornings. That is fine if the output is a markdown brief in `briefs/YYYY-MM-DD.md`. It is expensive if the output is a merged pull request, and the `deliver` recipe does walk a PR through review, CI, merge, deployment and live-version check. Autoloop mitigates this sensibly: review and delivery config are read only from `main`, and PR code is never checked out or executed on the dispatching machine. The approval card is still the entire brake.

MY READ

The quality signal here is the v1.9.4 changelog, not the star count. The fix was that an answer to a question asked inside a scheduled recheck round came back labelled `timer:N` instead of the originating source event id. Nobody ships a fix that specific unless they are running the thing against real traffic and tracing causality through the log. That is a maintainer I would bet on for correctness, which is a separate bet from whether the agent on the other end is good enough to leave alone.

The thing I cannot assess from the outside is maintenance load. Ten minor versions in seventeen days from what appears to be a single account, with 1,182 forks and an issue-and-PR count that gitnova puts at 395 while the GitHub API reports five open. That gap is probably a counting difference between open and total, but either way the inbound volume is high relative to one maintainer.

My take: run it if your unattended work is read-mostly and reversible. Morning briefs from local notes and commits, inbox triage that drafts but never sends, website watch that wakes an agent on first failure and stays quiet on repeats, issue triage that edits locally without committing or pushing. Those recipes ship with the project and the blast radius is a file you can delete. Skip it, for now, if you intended to point it at production remediation or automatic PR merges, and skip it if you cannot read Chinese CLI output while debugging at two in the morning. If you do run it, instrument the halt path before the first cron fires rather than after.

If you are weighing an unattended agent loop against a durable-execution framework for work that touches production, that choice turns on your rollback story more than on the scheduler, and it is a conversation worth having before the first cron entry goes in.



WRITTEN BY

Artur Markus

Software engineer in Basel building AI infrastructure and agentic systems, with a background in pharmaceutical automation, GxP compliance, and validation. I help teams turn AI from a chatbot into a reliable operational layer.

Book a meeting →