Fast Loops Beat More Autonomy as Agent Stacks Go Portable

Coding Agents Alpha Tracker

2026-03-26

Fast Loops Beat More Autonomy as Agent Stacks Go Portable

By Coding Agents Alpha Tracker • March 26, 2026

The clearest signal today is convergence: DHH, Mike Krieger, and Armin Ronacher all point to the same leverage point in coding agents—fast terminal loops, tight verification, and small tool surfaces. Also inside: Claude Code auto mode rollout details, Cursor's self-hosted agents, portable LangSmith skills, and the most reusable workflow tricks.

TOP SIGNAL

Stop optimizing for more agent freedom before you optimize the loop. DHH says agents became production-useful once they could stay inside a fast terminal cycle of run code, run tests, rewrite, retry, while Armin Ronacher argues that coding sessions are exactly the kind of measurable interaction models keep getting reinforced on, which is why coding harnesses remain the safest substrate to build on even for adjacent tasks [1, 2]. Mike Krieger lands on the same product lesson: ship the smallest V1 fast, because agents add features easily but still need human judgment on what to cut [3].

TOOLS & MODELS

- **Claude Code auto mode — now rolling out to teams.** Anthropic says auto mode uses tested classifiers to make approval decisions as a safer middle ground than either constant prompts or fully skipping permissions; it is now available to Claude for Team users via `claude --enable-auto-mode`, then `Shift + Tab`, and Anthropic engineer catwu says it has become a daily-driver workflow for most of their team [4, 5, 6]. Read: engineering blog [4]

- **Cursor self-hosted cloud agents.** Same cloud-agent harness, but run on your own infrastructure so code and tool execution stay inside your network [7]. Read: self-hosted cloud agents [7]
- **LangSmith Fleet skills.** Skills can now be pulled straight into Claude Code, Cursor, or Codex from the LangSmith CLI, which means workspace knowledge moves into local coding agents without copy-paste [8]. Try it: LangSmith Fleet [8]
- **T3 Code vs. Conductor.** Theo's current practitioner read: Conductor still wins on UX, especially worktrees, but T3 Code wins on Codex support, performance, and openness; Claude support is slightly worse in T3 Code [9, 10]
- **datasette-llm 0.1a1.** Simon Willison shipped purpose-based model routing for Datasette plugins: one model for enrichment, another for SQL query assistance, exposed through `register_llm_purposes()` and `llm.model(purpose=...)` [11]. Release: datasette-llm 0.1a1 [11]

WORKFLOWS & TRICKS

- **Minimal V1 first, then rewrite fast.** Mike Krieger's current loop is simple: build the smallest version that proves the problem, get it out quickly, then do V2 immediately if the first cut overbuilt. His example: Cowork V1 shipped in 10 days, and he says rewrites now take days instead of the old year-long rewrite death march [3].
- **Force self-verification in the prompt.** Krieger's line to steal is `prove to yourself and me it works as intended` before a PR. Pair that with a harness that actually exercises agent behavior, not just unit tests, because agent-native failures show up in weird end-to-end states you would never have written a narrow test for [3].
- **Keep the main loop responsive, and package your rules.** Krieger strongly prompts Claude Code not to do everything itself, but to delegate heavy work to sub-agents so the main run loop stays interactive. He also packages durable product principles as Claude Code skills, and LangSmith Fleet now makes that same skill pattern portable across Claude Code, Cursor, and Codex [3, 8].
- **Treat context like scarce RAM.** Armin's durable pattern: partial file reads, overflow files on disk, grep large outputs instead of dumping them into context, and explicit `file changed, reread it` messages whenever humans or other agents mutate state [2].
- **Start with fewer tools than you think.** Armin says PI works with four tools; custom tool sprawl and proprietary blobs in context make agents worse, not better. His other bias is toward common interfaces: SQL over a custom DSL, JS-like syntax over Lisp/Scheme, plus informative errors when the mini-language cannot do something [2].
- **Push cross-cutting concerns below the agent.** Geoffrey Huntley's practical anti-slop move: move logging and tracing into middleware or effects so forgetful agents do not need to remember it in control flow. His

summary is blunt: less agent work, better outcomes [12, 13].

PEOPLE TO WATCH

- **Mike Krieger.** High signal because he is talking from Anthropic Labs and his own prototyping loop, not theory: rebuilds in hours, ships minimal V1s fast, packages product principles as skills, and keeps experimental teams tiny when exploring agent-native products [3, 14].
- **Armin Ronacher.** One of the clearest voices on model-robust agent design right now: he explains why coding harnesses transfer, where they break, and he is still openly skeptical that current agent-generated code clears a high quality bar [2, 15, 16].
- **DHH.** Useful because Basecamp is already using multiple agents internally, and he gives the blunt version of what changed: terminal tools and fast feedback loops, not nicer chat. He also says an internal CLI was quickly pushed to roughly 65% agent-written, then refined toward 97% over time [1].
- **Mario Zechner + Simon Willison.** Best counterweight to the speed cult: Zechner, who created the Pi agent framework used by OpenClaw, argues for hard limits on daily generated code and hand-written architecture and APIs; Simon agrees the real problem is cognitive debt, even if he questions the write-it-by-hand part [17].

  Give yourself time to think about what you're actually building and why. Set limits on how much code you let the clanker generate per day, in line with your ability to actually review it [17]

WATCH & LISTEN

- **3:50-7:57 — Armin on the durable bet.** Best explanation today for why coding agents keep compounding: coding sessions are measurable, heavily reinforced, and teach the file/bash/test patterns that transfer to non-coding tasks [2].

Armin Ronacher Leaning In To Find Out - PyAI Conf 2026 (3:50)

- **11:36-15:37 — Armin on context-efficient tools.** Worth it for the concrete patterns alone: partial reads, overflow files, grep-first retrieval, and why hidden state changes can make an agent confidently lie about validation [2].

Armin Ronacher Leaning In To Find Out - PyAI Conf 2026 (11:36)

- **40:54-43:26 — Mike on the product boundary problem.** Good framing of the design space between tightly permissioned agents and wide-open tools like OpenClaw: the useful product is somewhere between gated and YOLO [3].

Building Is the Easy Part Now | Mike Krieger on What AI Changed (40:54)

PROJECTS & REPOS

- **Pi agent framework.** This repo matters because it powers OpenClaw, and its creator is now one of the loudest voices arguing for slower, more reviewable agent workflows [17]. Repo: pi-mono [17]
- **datasette-llm 0.1a1.** New base plugin for wiring purpose-based model routing into other Datasette extensions, including `datasette-enrichments-llm`; practical if you want task-specific model selection without hardcoding one model everywhere [11]. Docs: README [11]
- **T3 Code.** The traction signal today is practitioner demand: Theo says people keep asking for the T3 Code vs. Conductor tradeoff, and his current answer is OSS plus much better Codex support and much better performance [9, 10].

Editorial take: the teams getting real leverage are shrinking the problem, not romanticizing autonomy — faster loops, smaller tool surfaces, stricter verification, and tighter human review are still the winning pattern [1, 2, 3, 17].

Sources

1. Bring your AI agents to Basecamp – REWORK

2. Armin Ronacher Leaning In To Find Out - PyAI Conf 2026
3. Building Is the Easy Part Now | Mike Krieger on What AI Changed
4. X post by @AnthropicAI
5. X post by @claudeai
6. X post by @_catwu
7. X post by @cursor_ai
8. Skills in LangSmith Fleet
9. X post by @theo
10. X post by @theo
11. datasette-llm 0.1a1
12. X post by @GeoffreyHuntley
13. X post by @GeoffreyHuntley
14. X post by @danshipper
15. X post by @mitsuhiko
16. X post by @mitsuhiko
17. Thoughts on slowing the fuck down