Lean Prompts, Explicit Context, and Secure Execution

Coding Agents Alpha Tracker

2026-05-28

Lean Prompts, Explicit Context, and Secure Execution

By Coding Agents Alpha Tracker • May 28, 2026

Serious coding-agent users are simplifying the front end while adding stronger control underneath: handwritten AGENTS files, fresh threads, remote verification, trace-driven evals, and isolated execution. This brief pulls together Theo's Lakebed workflow, LangChain's latest agent infra, Codex/MCP changes, and the clips/repos worth stealing from.

TOP SIGNAL

- **The best coding-agent workflows are getting simpler up front and stricter underneath.** Theo says his five-day Lakebed push ran mostly on GPT-5.5 with a handwritten `Agent MD`, 1-2 sentence prompts, fresh threads, and remote browser-based verification—instead of skill sprawl or rigid plan modes [1]. Jediah Katz makes the same point from benchmarks: simpler harnesses can generalize well, while benchmark-tuned setups can win by brute-forcing impractical tool use [2, 3]. LangChain's Context Hub/Engine demos and Logan Kilpatrick's vibe-coding vs agentic-engineering split point to the same production standard: keep context explicit and versioned, then use traces, evals, checklists, and human review to control the system [4, 5, 6].

TRY THIS

- **Write `AGENTS.md` like a letter, then prompt like a teammate.** Handwrite the file yourself; describe what you're building, why it exists, and the general constraints/philosophy, but skip file paths and rigid technical mandates [1]. Then keep task prompts to 1-2 sentences, lead with the end goal, and when the model struggles, add a concrete example instead

of more prose—Matthew Berman says the best coding-agent prompts are short, behavior-focused, and closer to `this thing isn't working like I think it should ... go fix it` than to a long interface spec [1, 7]. Open a fresh thread for each concern so old context does not bias the next task [1].

- **Turn remote execution into a verification loop.** Theo's setup: host T3 code remotely, connect it with Tailscale, and work from browser/phone/tablet so the agent keeps running after the laptop closes [1]. Then configure Codex computer use once in the app and call it through CLI/T3 code so the agent can deploy the app, open the live site, and verify behavior before it reports done [1].
- **Use traces to generate fixes *and* evals, not just dashboards.** LangSmith Engine runs on a schedule against production traces, clusters issues into prioritized buckets, proposes prompt/agent-file/code fixes, opens PRs, and creates online evaluators plus offline regression examples from errored inputs [5]. LangChain's main lesson from dogfooding: the hard part is filtering for meaningful issues, so store team priorities in an `agent overview` memory file—e.g. care about hallucinations, ignore latency, or the reverse—and let that steer what the system surfaces [5].
- **Separate scratch space from durable memory.** In Context Hub, create an agent repo with an `AgentsMD` contract, put long-lived guidance under `/memories/...`, and keep temporary thread-local files in a separate state backend [4]. Let agents edit those memory files via normal file tools, keep the full version history visible to humans, and gate risky actions like `Send Email` behind an `interrupt` approval step [4].

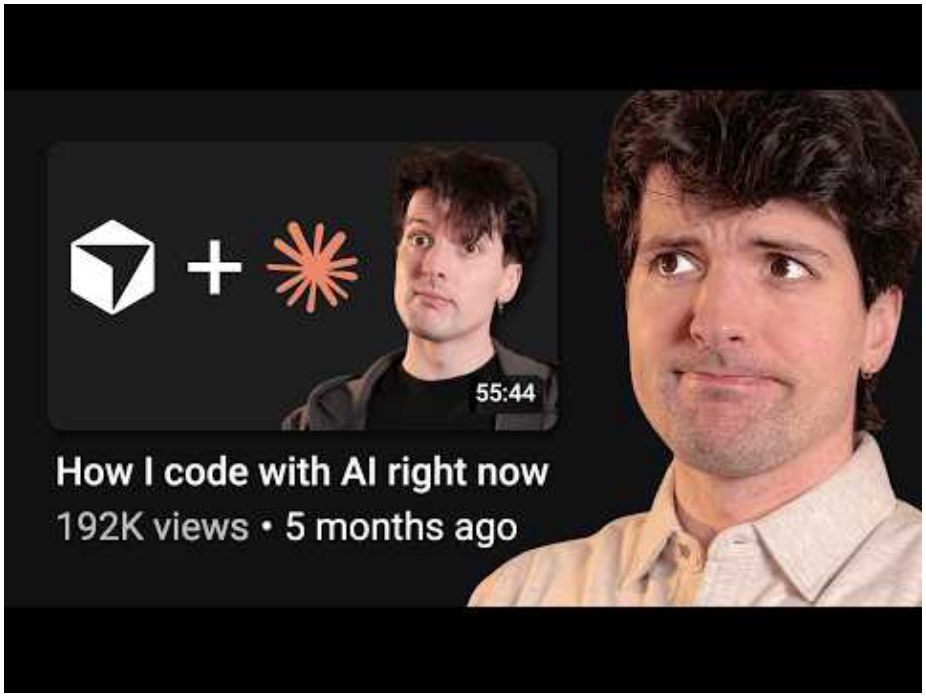
WHAT SHIPPED

- **Antigravity 2.0**: subagents, async task management, cron-style scheduled tasks, local shell-script JSON Hooks (docs), and voice input via real-time transcription [8, 9, 10, 11, 12, 13].
- **Deep Agents v0.6**: Delta channels cut checkpoint storage by up to 100x for long-running agents; LangChain shows 5.3GB falling to 129MB on a 200-turn coding session [14].
- **LangSmith Fleet**: public beta for isolated execution and computer use. Agents can analyze data, transform files, generate/write code, and run shell commands in a secure virtual computer; LangChain says Fleet + sandboxes already power an internal docs agent that listens to requests, triages them, and opens PRs automatically [15, 16].
- **Codex model cleanup**: GPT-5.2 and GPT-5.3-Codex are leaving Codex for ChatGPT-account users on June 2; GPT-5.5 becomes the default frontier model on free plans, while the deprecated models stay available via API. Adoption signal: GPT-5.2 is now under 1% of production usage [17, 18].

- **Private MCP tunnels for OpenAI products**: teams can keep MCP servers inside their own network while Codex, ChatGPT, and the Responses API connect over outbound-only HTTPS. Docs: secure MCP tunnels [19, 20].
- **Benchmark caution, not a product recommendation**: Jediah Katz says `mini-swe-agent`—roughly 150 lines in its core agent class—beats ClaudeCode and Codex on DeepSWE, but he also warns benchmark-optimized harnesses can be impractical for normal development if they force excessive tool calls or sloppier code [2, 3].

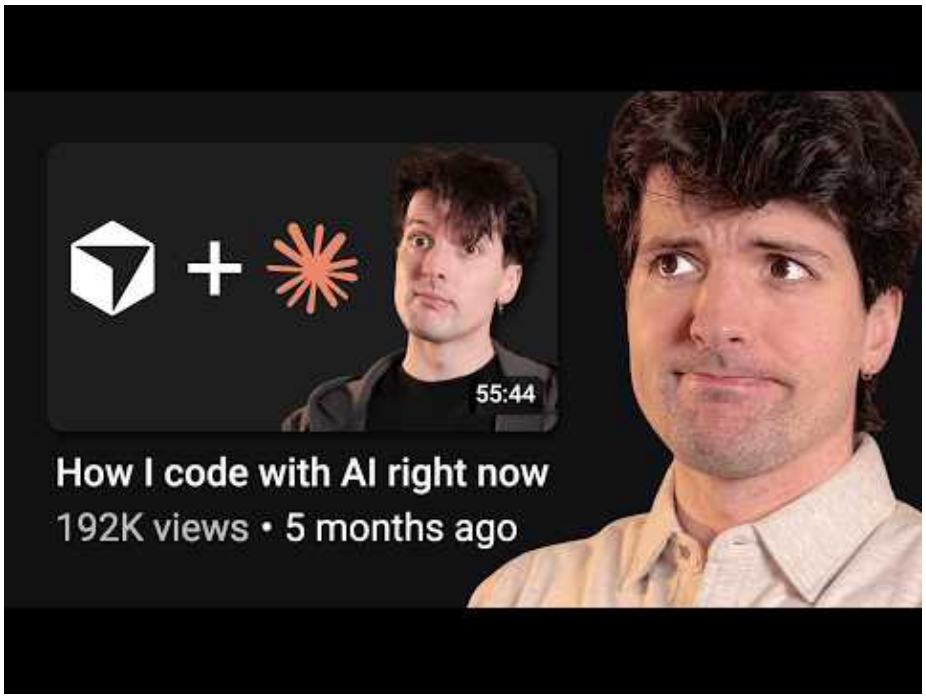
GO DEEPER

- **27:14-28:15 — Theo on handwritten `Agent MD`.** Worth watching if your agent instructions have turned into a brittle policy doc. His rule: no file paths, no rigid technical decisions—just a hand-written explanation of what you're building and why [1].



How I code with AI changed a lot (27:14)

- **39:47-40:45 — Theo on computer-use verification.** Good clip if you want agents to validate real deployments instead of handing you untested diffs. He sets up Codex computer use once, then reuses it through T3 code for remote verification [1].



How I code with AI changed a lot (39:47)

- **15:43-17:39 — LangSmith Engine on eval design.** Clear framing on why you need both targeted and end-to-end evals, why tool/context design still matters, and why prompt engineering is not dead inside production agent systems [5].

How We Built LangSmith Engine | Interrupt 26 (15:43)

- **Repo to study — `email_agent` Context Hub example.** https://github.com/langchain-samples/deepagents-with-langsmith/tree/main/agents/email_agent is a clean reference for `AgentsMD`, shared `/memories`, and human-visible version history [21, 4].
- **Repo/doc to study — SQLite `AGENTS.md`.** https://github.com/sqlite/sqlite/blob/master/AGENTS.md shows a strict maintainer policy for agent-driven contributions: no agentic code, yes agentic bug reports with reproducible tests, and proof-of-concept patches only as documentation [22].

Editorial take: the edge is moving away from clever prompt stacks and toward cleaner system design—short asks, explicit memory, real verification, and trace-backed eval loops. [1, 4, 5]

Sources

1. How I code with AI changed a lot
2. X post by @KLieret
3. X post by @jediahkatz
4. How to manage context the right way with LangSmith's Context Hub
5. How We Built LangSmith Engine | Interrupt 26
6. Interview: Building Apps with Google AI Studio

7. Finally a good benchmark (DeepSWE)
8. X post by @antigravity
9. X post by @antigravity
10. X post by @antigravity
11. X post by @antigravity
12. X post by @antigravity
13. X post by @antigravity
14. X post by @LangChain
15. X post by @LangChain
16. X post by @BraceSproul
17. X post by @thsottiaux
18. X post by @thsottiaux
19. X post by @OpenAIDevs
20. X post by @gdb
21. X post by @LangChain
22. sqlite AGENTS.md