Bun's 11-Day Rust Port Shows What Agent Verification Looks Like

Coding Agents Alpha Tracker

2026-07-15

Bun's 11-Day Rust Port Shows What Agent Verification Looks Like

By Coding Agents Alpha Tracker • July 15, 2026

A massive Bun migration offers a concrete blueprint for reliable agent workflows: pilot first, use adversarial review, treat failures as workflow bugs, and require evidence before merge. Also: new coding-agent tracing, Pi harness extensibility, and model-routing practices from active practitioners.

TOP SIGNAL

At agent scale, the breakthrough is the verification system—not a bigger prompt. In Theo's walkthrough, Jared's Bun rewrite reportedly moved a 500k-line Zig codebase to Rust in 11 days using about 50 continuously running Claude Code dynamic workflows, producing 6,502 commits and 1.78 million lines written or rewritten. [1] The transferable pattern was adversarial review, a language-independent test suite, and fixing the *process that generated* bad code rather than hand-patching each failure. [1]

TRY THIS

- **Pilot the workflow before scaling the task.** For a migration or broad refactor: (1) have the agent create a mapping guide for source-to-target patterns, (2) adversarially review it and read it yourself, then (3) run a three-file trial with one implementer, two reviewers checking behavior and guide compliance, and one fixer before unleashing the full job. That is the sequence used before Bun's bulk translation. [1]
- **Turn failures into a queue—and recurring failures into workflow changes.** Work crate-by-crate from compiler errors; when the agent finds a systematic issue, add a classification/fix workflow instead of starting over. For resource-hungry test suites, isolate runs with SystemD/cgroups;

merge only after the entire suite passes in CI across platforms and you confirm tests were not skipped. [1]

- **Audit `AGENTS.md` before persistent goals.** After a model release, run: `Review my AGENTS.md for stale rules or things we should revise or remove`, then clean it up—those rules load into every agent's context. [2] swyx's stronger warning: if you do not know what the file says before launching a task, you are effectively accepting indirect prompt injection; he describes a five-stage goal spending eight hours stuck refining stage 0 because repository instructions prevented it from moving on. [3]
- **Add visual proof before building more loops.** Jason Zhou's `verifier-setup` skill scaffolds a verifier sub-agent that drives the real app, embeds screenshots/video in every PR, and provides a one-command dev stack usable locally or in per-agent cloud sandboxes. His claimed review compression: watch the 20-second video, then merge. [4]

WHAT SHIPPED

- **LangSmith tracing for coding agents:** LangChain added tracing support for **Cursor, Copilot, Pi, and OpenCode**, including stable trace keys across agents, full run trees for turns/model calls/tools/subagents, and session token-cost tracking. Docs [5, 6]
- **Codex-to-LangSmith plugin:** Codex sessions can now emit inspectable traces covering turns, tool calls, model metadata, token usage, and nested subagents. The setup uses the Codex Marketplace plugin plus config flags and environment variables. [7]
- **Pi Agent's extensibility is the notable harness contrast.** AI Jason highlights a deliberately minimal default—bash plus file read/write/edit, no bundled subagents or MCP—then adds behavior through extension files for tools, hooks, context, session logic, UI, and model providers. One example package filters tool output before it reaches the model, reducing token use for `git log`-style commands by up to 96% in the demonstrated case. [8]
- **Model-routing field note from swyx:** for large projects, he uses Sol Ultra to plan, Fable 5 to critique, then Sonnet 5/Terra Ultra/SWE 1.7 for implementation and Devin Review via Kakuna for review. His comparison method is useful: give Fable and Sol the same prompt, then have each critique the other's plan; he reports Fable consistently preferred Sol's plan. [9, 10]
- **Autoreview skill:** Peter Steinberger recommends running autoreview as standard practice despite the token cost. Study the implementation. [11, 12]

GO DEEPER

- **17:18–20:04 — Theo on operationalizing a massive agent rewrite.** The useful part is not the migration headline: watch the compiler-error work queue, workflow-level fixes, and cgroup isolation for tests that exhaust system resources. [1]

*Well this really p*ssed me off (17:18)*

- **2:51–3:29 — LangChain's Codex trace anatomy.** A concise walkthrough of what observability should capture: accumulated context, outputs, model metadata, each tool call's inputs/outputs, and nested sub-

agent hierarchy. [7]
Trace Every Codex Session in LangSmith in Minutes (2:51)

- **Study `verifier-setup`.** It is a compact example of making PR evidence a first-class agent output rather than trusting an agent's completion message. [4]
- **Study OpenClaw's `autoreview` skill.** It is a practical starting point for adding a repeatable review pass to an agent workflow. [11]

Editorial take: agent autonomy scales when execution is paired with adversarial review, observable traces, and evidence that the real app worked—not when teams merely add more parallel agents. [1, 4, 5]

Sources

1. Well this really p*ssed me off
2. You aren't using Codex like me…
3. X post by @swyx
4. X post by @jasonzhou1993
5. X post by @LangChain
6. X post by @LangChain
7. Trace Every Codex Session in LangSmith in Minutes
8. Why I switched to Pi…
9. X post by @swyx
10. X post by @swyx
11. X post by @steipete

12. X post by @steipete