Agent runs grow from minutes to hours: Ronacher's checkpoint prompt, and Pi users banned from Antigravity

Coding Agents Alpha Tracker

2026-11-23

Agent runs grow from minutes to hours: Ronacher's checkpoint prompt, and Pi users banned from Antigravity

By Coding Agents Alpha Tracker • November 23, 2026

Armin Ronacher's session data shows agent runs getting much longer while his Pi harness stayed the same, and he shares how he keeps oversight. Also: which harnesses Google allows for Gemini subscriptions, Theo's case that open-weight models are falling behind, Sanfilippo's simulator feedback loop, and Mitchell Hashimoto's Rex.

Ronacher: longer runs leave human attention as the limit

Armin Ronacher's Pi agent still has the same core harness and tools it had last December. In that time, the number of corrections he makes has dropped with each new model [1]. At the 90th percentile, a task run took about 2.5 minutes with Opus 4.6. With current models it takes 14 minutes or more [1]. Tricks such as his PyAutoResearch plugin are no longer needed [1].

What hasn't changed is his own input. He sends 120–190 prompts a day, but agent-active time has grown from less than a working day to many times one. He has started 141 repos and regularly commits to only 10–15 of them [1]. His advice for working this way:

- **Add a review checkpoint before verification.** At around 70% of a session, before the agent starts verifying, he prompts it for a short executive summary and a typeset HTML page explaining what it did. That's how he keeps oversight now that he often doesn't read the code [1].
- **Push back on goal-completion.** Agents are rewarded for finishing the goal, which doesn't mean the software holds together as a whole. Without

pushback you can end up with a "Simpson car" of features [1].

- **Make long runs durable.** Overnight agents should restart themselves and support suspend and resume. With flex-scheduled tokens, suspend both the agent and its sandbox while no tokens are arriving [1].
- **Use code mode plus cheap classifiers.** In code mode, available in Codex, Pi and other harnesses, JavaScript orchestrates large batches of tool calls outside the main context [1]. The agent can build an on-demand workflow that sends each decision to a fast classifier such as Jeff, for example to triage 3,000 support requests [1].

Theo sees the same trend in model behavior. Opus 5.5 and Fable 5.2 use dedicated read and edit tools less than 1% of the time when they read or edit files. Instead they write throwaway Python or JS scripts that bundle the work [2].

Gemini subscriptions: T3 Code is allowed, the Pi shim is not

One user reports being banned after a single prompt through pi-antigravity [3]. Theo explains the rule: a Gemini subscription can only be used through Antigravity. T3 Code calls the Antigravity binary, which is allowed. Pi goes through a "fake antigravity shim," which is not. He compares this to how Claude subscriptions work [4]. If you use Pi or other third-party harnesses, use API billing for Gemini.

Theo: open-weight models fall behind on long agentic work

Theo says he has "fully stopped betting on good open weight models" [2]. His main argument is about one capability: staying coherent on long, chaotic tasks while managing subagents. He says Fable 5.2 is "so far ahead" on this, and Kimi K3.1 Pro can't do it [2]. His usage data comes from T3 Code only. There, open-weight models account for 0.5% of turns and 0.7% of tokens. The share of installs that have ever tried one fell from 16% to 6% [2].

In the video's sponsor segment, he says CI and local CPU limits were his bottleneck. His machines could only run three or four agents at once [2]. With Blacksmith's CLI, agents spin up test boxes and run CI without pushing a PR. His `/goal` runs went from 2–3 parallel tasks to 15+ [2]. This is sponsored content, but the takeaway applies generally: faster verification lets you run more agents in parallel.

Riley Brown: assign models by strength in T3 Code

Riley says Gemini is best at understanding video (visuals and sound), while Claude is best at writing the code that edits it. In T3 Code, Claude can start a Gemini subagent that analyzes 20+ videos and reports back so Claude can make the edits [5]. For short-form evals, he has Neon, Astra and Fable each edit the same video from the same context with one prompt [5]. He also found that

Claude artifacts open in T3 Code's browser, and that a connected iOS app can control that browser [5]. T3 Code is free [6].

He has also spent about a month letting Claude Code rent GPUs for Blender and fine-tuning. A longer video is coming, and he hasn't shared details yet [7].

Sanfilippo: give the agent a simulator to test against

Salvatore Sanfilippo argues that agents improve fastest with a tight feedback loop, and that you can build one for domains that lack it [8]. He built a chip-level Commodore 64 hardware simulator that models electrical interactions. It runs at about 30% of real C64 speed on an M3 Max [8]. He gave it to the model writing a dead-test cartridge, so it could break the simulated hardware in many ways and check that the cartridge responded correctly. He says that after some iterations it produced the best C64 test cartridge ever made [8].

He is also considering making perplexity the main quality check for his Dwarf Star GGUF quantizations and SafeTensors conversions. A rise in perplexity after changing kernels is a "big red flag" [9].

Hashimoto's Rex: a terminal and multiplexer built for agent users

Mitchell Hashimoto released Rex, a terminal built on the Ghostty core that merges the terminal with the multiplexer to replace tmux. Sessions survive closing and reopening, and you can open them from a phone or open a Windows terminal from a Mac or Linux machine [10]. He credits Claude Code with a big rise in terminal usage. He says the reasons are a simpler programming model and less sandboxing [10]. For the UI layer, which is still experimental, every element must be API-driven so agents can control it without going through accessibility hooks [10].

Quick notes

- **ThePrimeagen** endorses tree visualizations of prompts [11]. His example: adding web testing to his Odin app from his phone, where a simple task branched into two [12].
- **Ronacher** says his son, who isn't a software engineer, now sends him PRs. The quality difference is clear even though they use the same prompting setup [1].

Sources

1. Build the Workshop Before the Software Factory — Armin Ronacher, Earendil
2. If you like Open Models, I'm sorry

3. X post by @yuvadm
4. X post by @theo
5. X post by @rileybrown
6. X post by @rileybrown
7. X post by @rileybrown
8. Simulazione e rinforzo
9. Perplexity e Intelligenza negli LLM (dal blog di OpenAI)
10. Mitchell Hashimoto Talks Rex | TheStandup
11. X post by @ThePrimeagen
12. X post by @ThePrimeagen