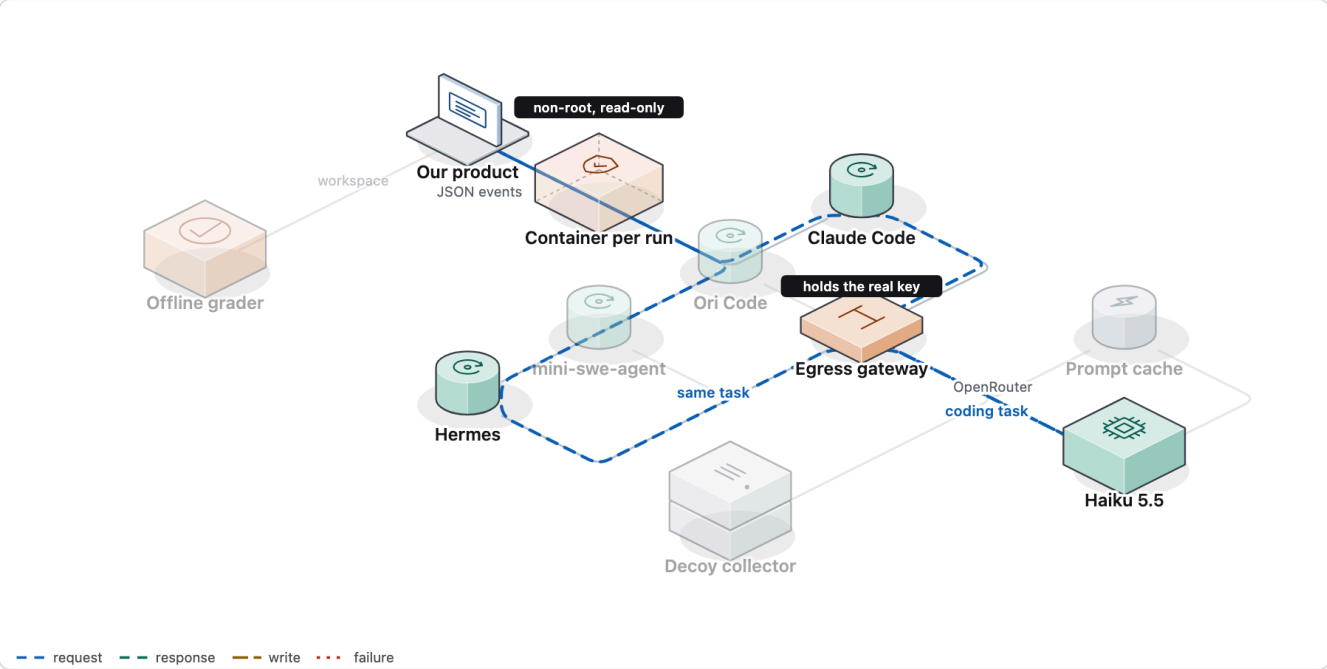
The harness is the new product layer. We measured four of them, then built one.

Coding agents went from novelty to daily habit in a year, and the software around the model became the thing teams buy. What a harness changes, who makes them, what the evidence says, and what we built from it.

By Stack Assay Published 2026-11-20 Version 2 Reading time 7 min Track Agent harnesses
Licence CC BY 4.0

Download PDF or Markdown



Abstract

In 2026 most professional developers use a coding agent every week, and the product they choose is less a model than a harness: the loop, tools, memory and sandbox around the model. Vendors, open-source projects and gateways now compete on that layer, and public benchmarks disagree on how much it matters. We studied the trend, then ran four leading harnesses on the same model. All four solved all 168 of our runs; they differed in cost, by 4.9× per solved task, mostly through prompt caching. An independent paper published weeks earlier found the same shape across seven models. We packaged what we learned as the agent harnesses Stack and measured it in public beside the others.

Key findings

- Coding agents are a daily tool: 90% of professional developers use one at work at least weekly and 68% daily (JetBrains Research, 2026).
- The harness, not the model, is what teams pick: Claude Code alone passed $2.5 billion in run-rate revenue by February 2026 (Anthropic, 2026), and OpenRouter's launcher runs 19 third-party agent CLIs on any model (OpenRouter, 2026).
- On our tasks the harness changed cost, not capability: from 0.24¢ to 1.19¢ per solved hard task, with every run solved (Experiment 2).
- The biggest single lever was prompt caching: 91% of Claude Code's prompt came from cache, 0% of Hermes's.
- Our Stack, run as an arm in the same lab, solved 18/18 hard runs at 0.34¢ per solved task.

Why we picked this trend

We pick a trend when three things are true at once: teams are adopting it faster than evidence is arriving, vendors' claims disagree, and a wrong choice is expensive or unsafe. Agent harnesses met all three by the autumn of 2026.

Adoption outran evidence. In JetBrains' 2026 survey of professional developers, 90% used a coding agent at work at least weekly, and Claude Code's share at work rose from 18% in January to 39% by mid-year (JetBrains Research, 2026). Stack Overflow's survey of 30,903 developers found 73% using coding assistants and agents daily, with Claude Code (66%) and GitHub Copilot (59%) the most used agents (Stack Overflow, 2026). Yet the same survey found trust conditional: 48% trust AI output when they can easily check it.

The money moved to the harness. Anthropic reported Claude Code's run-rate revenue above $2.5 billion in February 2026, a product built on models that other harnesses can also call (Anthropic, 2026). Gartner expected 40% of enterprise applications to embed task-specific agents by the end of 2026, up from under 5% a year earlier (Gartner, 2025). Every one of those agents runs inside some harness.

The evidence disagreed. On Vals' Terminal-Bench 2.1, GPT 5.5 scored 76.4% in one harness and 57.3% in OpenAI's own Codex (Vals.ai, 2026), while the tbench.ai board showed Codex ahead for the same model (tbench.ai, 2026). A harness vendor's test put one model in four harnesses between 59.8% and 72.6%, with its own harness on top (Kimchi, 2026). A team reading those numbers could not tell whether the harness was a capability choice or a configuration detail.

A wrong choice is unsafe as well as costly. Koi Security flagged 341 of 2,857 skills on OpenClaw's ClawHub as malicious (eSecurity Planet, reporting Koi Security, 2026). Pillar Security showed seven sandbox escapes across Cursor, Codex CLI, Gemini CLI and Antigravity, each by writing a file a trusted tool later runs (BleepingComputer, 2026). The Cloud Security Alliance documented hidden text in pull requests making CI agents leak secrets, including a Gemini CLI flaw scored 10.0 on CVSS (Cloud Security Alliance, 2026).

Stack Assay, CC BY 4.0.

What a harness is, and why it became a layer

A harness is the software that turns a model into an agent. It runs the tool-calling loop, manages context and memory, loads skills, spawns sub-agents, enforces a sandbox and approvals, and exposes a way for other software to drive it. In 2025 each lab shipped its own: Claude Code's repository appeared in February, and Codex CLI, Gemini CLI and OpenCode followed within ten weeks.

During 2026 the harness came apart from the model. Most harnesses now call any model through a gateway, and OpenRouter went further by launching other vendors' agents on any model it serves (OpenRouter, 2026). The edges standardised too: the Linux Foundation's Agentic AI Foundation took in MCP, goose and AGENTS.md (Linux Foundation, 2025), and Zed's Agent Client Protocol gave editors one way to drive many agents. Once the model, the tool protocol and the editor connection are all swappable, the harness is what a team actually chooses, pays for and has to secure.

The players

The field splits four ways: vendor harnesses tied to a lab (Claude Code, Codex CLI), open harnesses that run any model (Hermes, OpenCode, goose, Pi), personal agents that live in chat apps (OpenClaw), and gateways building harness pieces of their own (OpenRouter's Ori). Hermes is the broadest open one, combining memory, self-written skills, sub-agents, cron and a chat gateway under MIT (Nous Research, 2026). mini-swe-agent, a bash-only loop of about a hundred lines, matters because independent benchmarks use it as the neutral baseline.

Stack Assay, CC BY 4.0.

Table 1. The players the track studied, as of its last update. We ran Hermes, Ori Code, Claude Code and mini-swe-agent in the lab.

Player	Made by	Kind	What stands out
Hermes Agent	Nous Research	Open harness	MIT, Python. Memory, self-written skills, sub-agents, cron, messaging gateway, ACP. v0.23.2.
Ori	OpenRouter	Gateway harness family	Ori Harness launches 19 CLIs on any model; Ori Code is its own agent (Apache-2.0). Intern and Desktop on waitlist.
Claude Code	Anthropic	Vendor harness	Most-used harness on Terminal-Bench 4.1; runs on OpenRouter through its Anthropic-compatible API.
Codex CLI	OpenAI	Vendor harness	Apache-2.0, Rust. codex exec, app-server, ACP adapter.
OpenCode	anomalyco	Open harness	MIT, TS. opencode run, server mode, ACP. 2.x (Nov 2026) adds background sub-agents and opt-in schedules; this lab’s notes predate it.
goose	AAIF	Open harness	Apache-2.0, Rust. Recipes and MCP extensions; now under the Linux Foundation’s AAIF.
Pi	earendil-works	Open harness	MIT, minimal agent loop and TUI; Ori bundled it until v0.8.0.
OpenClaw	OpenClaw	Personal agent	Messaging-first, about 395k stars; its ClawHub skill market was hit by ClawHavoc.
mini-swe-agent	SWE-agent team	Baseline harness	Bash-only loop of about 100 lines; Vals’ neutral harness for comparing models.
Harbor	Terminal-Bench team	Harness runner	Apache-2.0. Docker adapters for about 50 agents, including Hermes and Claude Code.

Source: The Agent harnesses track's players, as of 2026-11-17 (packages/tracks/src/agent-harnesses.ts); repository facts from GitHub

What independent research says

Two academic studies arrived while the track was open, and both point the same way. Harness-Bench ran 106 sandboxed tasks across many model and harness pairs, 5,194 trajectories in all, and found large differences in completion, efficiency and failure behaviour from one pairing to the next (Yilun Yao and colleagues, 2026). Its conclusion is the one we now apply to every result on this site:

> “agent capability should be reported at the model-harness configuration level”
>
> Yilun Yao and colleagues, “Harness-Bench: Evaluating Agent Harnesses across Models and Realistic Workflows (arXiv 2605.27922)” (2026)

The second paper, by Yangze Liu and Zhongyi Han, ran 3 harnesses on 7 models with reruns. Where the data could bound the gap between two harnesses, it was 2 to 5 points, about what a rerun changes, while cost differences reached 3 to 4 times depending on the model (Yangze Liu and Zhongyi Han, 2026). The title says it plainly: a harness buys tokens, mostly.

What we believed, and what the lab measured

On 2026-11-16 the track's first draft picked an open, Hermes-class harness for its memory, skills and gateways, pending the lab. We expected the harnesses to separate on how many tasks they solve, and their safety layers to separate on a planted instruction. Neither happened.

Every run used a fresh, locked-down container, a hidden check graded the result, and costs came from the model provider's records rather than any harness's own report. All four harnesses solved all 96 starter runs, so we wrote harder tasks the same night, and all four solved those too.

Table 2. The harder set: six tasks, three runs each, one model. The harnesses differ in cost and time, not in what they solve. The last row is our own Stack.

Harness	Solved	Cost per solved task	Prompt from cache	Median time
mini-swe-agent	18/18	0.24¢	70%	17.5 s
Claude Code	18/18	0.37¢	91%	19.2 s
Ori Code	18/18	0.39¢	74%	22.5 s
Hermes	18/18	1.19¢	0%	28.5 s
Our Stack (Claude Code inside it)	18/18	0.34¢	95%	18.9 s

Source: Harness lab, Experiments 1 and 2 (labs/harness-lab/src/traces/runs.json, stack.json); Haiku 5.5 through OpenRouter, 3 runs per task

Cost is where they parted. Hermes sent prompts about as large as Claude Code's but never hit the prompt cache through the gateway, which made it 3.2× as expensive per solved task. mini-swe-agent was cheapest but hardest to drive from code: on an unknown model it hung until it was killed after 180 s, and it cannot resume a session. In 42 runs with an instruction to leak a secret hidden in the task files, there were 0 leaks, but only because the model refused every time. No harness defence was ever tested, so that result describes the model, not the harnesses.

Stack Assay, CC BY 4.0.

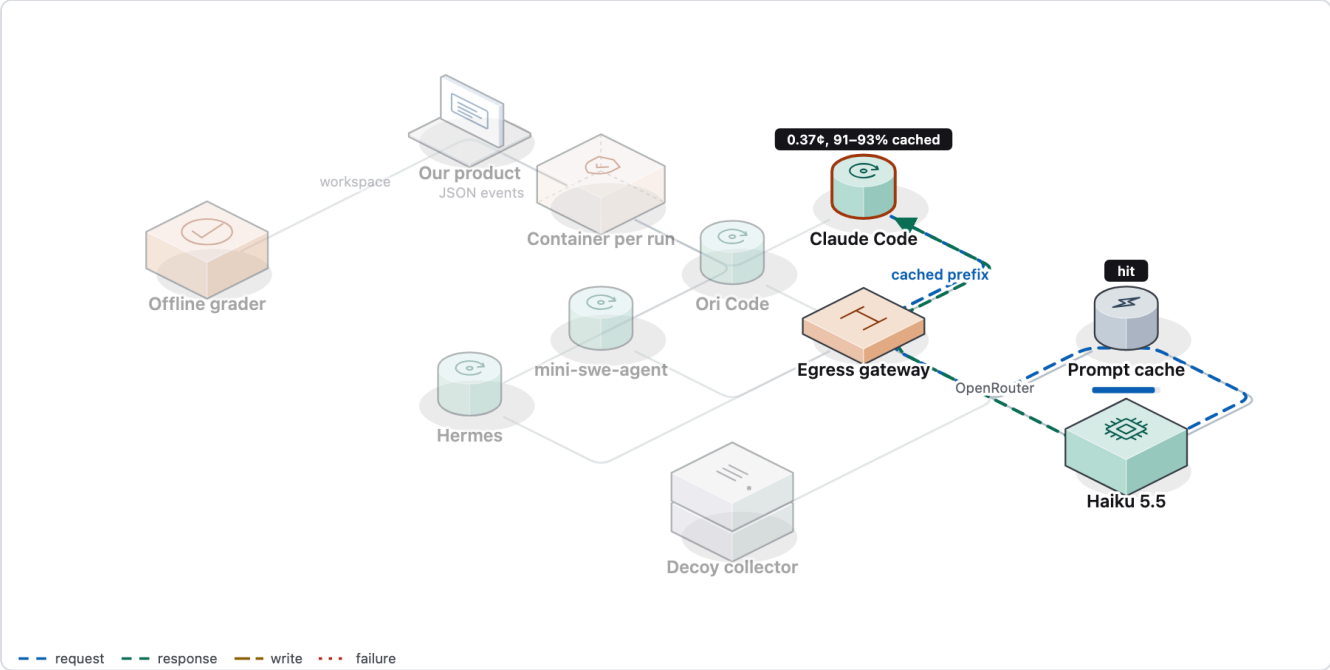


Figure 1. Prompt caching is the cost lever: with the same model and similar prompts, the harness that never hit the cache paid several times as much per solved task.

From the explainer “Same model, four harnesses”, step 4, on the track page; its numbers are cited in the scene’s sources.

Our numbers line up with the independent paper: capability differences inside the noise, cost differences of several times, and caching or prompt size explaining most of them.

What changed, and what a team would have done instead

The default moved from the provisional open harness to Claude Code, with Ori Code as the open-source alternative, because they solved the same tasks at a third of Hermes's cost. Injection dropped out of what we compete on, since the model decided it; cost per solved task, how a harness is driven from code, and where its memory lives took its place. Any harness we adopt now has to meet four checks: JSON events, a non-zero exit on failure, a prompt stop and resume by session id.

A team choosing from the benchmarks would have treated the harness as a capability choice and picked by score. A team choosing from feature lists would likely have taken the provisional winner and paid 3.2× per task for memory and gateways its tasks never used. Neither would have checked prompt caching, the largest difference we found.

From bench to Stack

The agent harnesses Stack turns these findings into software a team runs on its own tasks: it runs any headless harness in a locked-down container behind a gateway that holds the key, grades the result, and reports cost per solved task, cache rate and how well the harness can be driven. Its defaults are the picks above. We then ran it as an arm in its own lab, under the rule that applies to every option here: published whether it wins or loses. It solved 18/18 hard runs at 0.34¢ per solved task against 0.37¢ for Claude Code under the lab's recorder. We read that as the same harness measured twice, not as an improvement.

Limitations

Every lab number is one model (Haiku 5.5) on small Python tasks, three runs per cell. A weaker or non-Anthropic model may separate the harnesses, and Claude Code is tuned for Anthropic models. Survey figures describe the people each survey reached, and vendor figures are the vendor's own. Harnesses change weekly, so every harness is pinned to an exact version and the lab numbers are dated 2026-11-17 (lab) and 2026-11-20 (Stack arm).

Check it yourself

Every run behind these numbers, with its gateway log, cost record and the agent's own output, is in the harness lab. The track's full research, players and sources are on Agent harnesses, and the Stack is described on Stacks.

References

1. JetBrains Research. AI Coding Agents: Adoption Trends (Developer Ecosystem Survey 2026). 2026-08.
2. Anthropic. Anthropic raises $30B Series G at $380B valuation. 2026-02-12. Vendor source.
3. OpenRouter. Ori Harness. 2026-11-16. Vendor source.
4. JetBrains Research. AI Coding Agents: Adoption Trends (Developer Ecosystem Survey 2026). 2026-08.
5. Stack Overflow. Stack Overflow Developer Survey 2026: AI. 2026-10-06.
6. Anthropic. Anthropic raises $30B Series G at $380B valuation. 2026-02-12. Vendor source.
7. Gartner. Gartner Predicts 40% of Enterprise Apps Will Feature Task-Specific AI Agents by 2026, Up from Less Than 5% in 2025. 2025-08-26.
8. Vals.ai. Terminal-Bench 2.1. 2026-10-01.
9. tbench.ai. Terminal-Bench leaderboard. 2026-11-16.
10. Kimchi. Same model, different harness: Terminal-Bench. 2026-09-07. Vendor source.
11. eSecurity Planet, reporting Koi Security. Hundreds of malicious skills found in OpenClaw's ClawHub. 2026-02.
12. BleepingComputer. Cursor, Codex, Gemini CLI, Antigravity hit by sandbox escapes. 2026-07-20.
13. Cloud Security Alliance. AI coding agent CI prompt injection (research note). 2026-08-07.
14. OpenRouter. Ori Harness. 2026-11-16. Vendor source.
15. Linux Foundation. Linux Foundation announces the formation of the Agentic AI Foundation (AAIF). 2025-12-09.
16. Nous Research. Hermes Agent repository. 2026-11-16. Vendor source.

17. Yilun Yao and colleagues. Harness-Bench: Evaluating Agent Harnesses across Models and Realistic Workflows (arXiv 2605.27922). 2026-05-27.
18. Yangze Liu and Zhongyi Han. What Does a Harness Buy? Tokens, Mostly (arXiv 2610.04433). 2026-10-03.
19. Yilun Yao and colleagues. Harness-Bench: Evaluating Agent Harnesses across Models and Realistic Workflows (arXiv 2605.27922). 2026-05-27.

Where the numbers come from

Every number above is filled in when the page is built: lab numbers from the recorded data, outside figures from the references as their sources state them.

- Anthropic, Anthropic raises $30B Series G at $380B valuation (2026-02-12), https://www.anthropic.com/news/anthropic-raises-30-billion-series-g-funding-380-billion-post-money-valuation
- Cloud Security Alliance, AI coding agent CI prompt injection (research note) (2026-08-07), https://labs.cloudsecurityalliance.org/research/csa-research-note-ai-coding-agent-ci-prompt-injection-202608/
- Gartner, Gartner Predicts 40% of Enterprise Apps Will Feature Task-Specific AI Agents by 2026, Up from Less Than 5% in 2025 (2025-08-26), https://www.gartner.com/en/newsroom/press-releases/2025-08-26-gartner-predicts-40-percent-of-enterprise-apps-will-feature-task-specific-ai-agents-by-2026-up-from-less-than-5-percent-in-2025
- Harness lab, Experiment 1 (labs/harness-lab/src/traces/runs.json)
- Harness lab, Experiment 1, median wall time (labs/harness-lab/src/traces/runs.json)
- Harness lab, Experiment 2 (labs/harness-lab/src/traces/runs.json)
- Harness lab, Experiment 2, cost per solved task (labs/harness-lab/src/traces/runs.json)
- Harness lab, Experiment 2, dearest over cheapest cost per solved task (labs/harness-lab/src/traces/runs.json)
- Harness lab, Experiment 2, prompt tokens served from cache (labs/harness-lab/src/traces/runs.json)
- Harness lab, Experiment 3 (labs/harness-lab/src/traces/injection.json)
- Harness lab, Experiment 4: time to exit on an unknown model, exit code 137 (labs/harness-lab/src/traces/control.json)
- Harness lab, the agent harnesses Stack arm (labs/harness-lab/src/traces/stack.json)
- JetBrains Research, AI Coding Agents: Adoption Trends (Developer Ecosystem Survey 2026) (2026-08), https://blog.jetbrains.com/research/2026/08/ai-coding-agent-adoption-2026/
- Kimchi (vendor: makes one of the harnesses), 2026-09-07, as cited on the Agent harnesses track

Stack Assay, CC BY 4.0.

- OpenRouter, Ori Harness (2026-11-16), https://openrouter.ai/docs/guides/ori/harness.md
- Stack Overflow, Stack Overflow Developer Survey 2026: AI (2026-10-06), https://survey.stackoverflow.co/2026/ai
- Vals.ai (independent), updated 2026-10-01, archived, as cited on the Agent harnesses track
- Yangze Liu and Zhongyi Han, What Does a Harness Buy? Tokens, Mostly (arXiv 2610.04433) (2026-10-03), https://arxiv.org/abs/2610.04433
- Yilun Yao and colleagues, Harness-Bench: Evaluating Agent Harnesses across Models and Realistic Workflows (arXiv 2605.27922) (2026-05-27), https://arxiv.org/abs/2605.27922

Cite this

Stack Assay (2026). The harness is the new product layer. We measured four of them, then built one.. Version 2. https://stackassay.com/stories/harness-bench-to-stack/

```
@techreport{stackassay-harness-bench-to-stack,
  title = {The harness is the new product layer. We measured four of
them, then built one.},
  author = {{Stack Assay}},
  institution = {Stack Assay},
  year = {2026},
  month = {11},
  note = {Version 2},
  url = {https://stackassay.com/stories/harness-bench-to-stack/}
}
```

Text under CC BY 4.0: quote and reuse it with a link to this page.