GLINTBASE RESEARCH · ARS 2026

State of Agent Readiness 2026

Global Benchmark Report on Developer Infrastructure Accessibility for Autonomous AI Coding Agents

N=100 AI Engineering Platforms · 1,000 Deterministic Pathfinder Scans
450 Live Coding Agent Harness Executions · August 2026

Published by Glintbase Research & Developer Infrastructure Labs

Table of Contents

Foreword & Research Background 3

1. Research Methodology & ARS 1.0 Scoring Framework 4

1.1 The Glintbase Pathfinder Scanner 4

1.2 Pilot Scan Learnings & Scanner Calibration 4

1.3 The 10 Universal Benchmarks (B-01 through B-10) 4

1.4 ARS Score Calculation 4

2. Global Cohort Landscape & ARS Distribution 5

2.1 Cohort Construction & Sampling Methodology 5

2.2 Global ARS Distribution 5

2.3 Machine Surface Adoption Analysis 5

2.4 Regional Analysis (US vs. APAC vs. Europe) 5

3. Category Performance Deep-Dive (Categories A–J) 6

Category A — Foundation Model Providers 6

Category B — AI Infrastructure & Deployment 6

Category C — Vector Databases & Search 6

Categories D–J — Summary 7

4. Benchmark Diagnostic Analysis (B-01 through B-10) 8–11

Discovery Tier (B-01, B-02, B-03) 8

Integration Loop Tier (B-04, B-05, B-06, B-07) 9

Production Resilience Tier (B-08, B-09, B-10) 10

5. Token Overhead Analysis & Financial Cost Modeling 12

6. Live Coding Agent Harness Cross-Validation 14

6.1 Evaluation Protocol & Harness Selection 14

6.2 Cross-Harness Summary Results 15

6.3 The Agent Resilience Paradox 15

7. Hall of Fame, Hall of Shame & The Irony List 17

8. Developer Ecosystem Remediation Playbook 18

Appendix A — Full 100-Company ARS Master Dataset 20

Appendix B — Universal 10-Benchmark Prompt Specification 21

Appendix C — Token Pricing Models & Mathematical Reference 22

Appendix D — Methodology Notes & Back Cover 23

Foreword & Research Background

The history of developer documentation is a history of writing for humans. For three decades, software companies have invested in documentation portals, interactive API explorers, tabbed code examples, and beautiful user interface design — all optimized for a developer sitting in a web browser, scrolling with a mouse, reading with human eyes, and clicking with a cursor.

That paradigm has ended. In 2025 and accelerating through 2026, autonomous AI coding agents have crossed a capability threshold. Tools like OpenCode CLI, Cursor AI IDE, Google Antigravity Assistant, Claude Code, Cline, and Windsurf are no longer simple code-completion assistants. They are full-stack autonomous engineering teammates that independently browse documentation, install SDKs, write authentication logic, handle errors, and submit pull requests — all without any human keystroke prompting the process.

When a developer asks an AI agent to *"integrate the Pinecone vector database into my FastAPI application"*, the agent does not wait for the developer to copy-paste an API key or open a documentation tab. It traverses the web, finds the Pinecone documentation, extracts the SDK installation command, reads the authentication header requirements, constructs a client initialization snippet, and writes the integration code — all autonomously, in a single session.

When developer platforms present well-structured, machine-readable documentation — a clean **/llms.txt** index, an OpenAPI specification, explicitly formatted authentication headers, and a centralized error code reference — AI agents complete integrations in two to three hops, consuming 10,000 to 15,000 tokens and returning clean working code within seconds. When platforms present fragmented documentation, JavaScript-rendered SPA shells, or unindexed error references, AI agents are forced to execute web search queries, crawl GitHub repositories, and navigate through 15 to 28 intermediate HTTP requests to assemble the same information — consuming 40,000 to 70,000 tokens and introducing hallucination risk at every inference step.

The core thesis of this research is what we call **The Agent Resilience Paradox**: state-of-the-art LLMs rarely fail completely when encountering bad developer documentation. They are too intelligent to simply stop. Instead, they find alternative information paths — web search fallbacks, GitHub README mirrors, community forum posts, and third-party blog articles. They succeed at the task, but they pay an enormous hidden tax to do so. The cost is borne entirely by the developer who receives the API bill.

> ***"Smart LLMs do not fail silently. They tax. Platform engineering teams that ignore machine documentation accessibility are not protecting their developers from complexity — they are silently transferring that complexity cost to their developers' LLM inference bills."***

Why This Research Matters Now

This research was conducted at the precise inflection point at which: (1) autonomous coding agents have crossed the capability threshold required to independently execute API integrations without human intervention; (2) developer ecosystems have not yet adapted their documentation infrastructure to serve machine consumers; and (3) the cost differential between machine-optimized and machine-hostile documentation is measurable, calculable, and financially significant at enterprise scale.

Platform engineering teams and DevRel departments that act on the findings in this report can reduce their users' AI agent token consumption by up to **58%** through a single afternoon's work — deploying a /llms.txt index. Those that do not will find themselves increasingly disadvantaged in an ecosystem where AI agents, not human developers, are the primary consumers of API documentation.

1. Research Methodology & the ARS 1.0 Scoring Framework

1.1 The Glintbase Pathfinder Scanner

The foundation of this research is the **Glintbase Pathfinder scanner** (*glintscanner*) — a deterministic, zero-prior-knowledge machine traversal engine built to emulate how an autonomous AI agent (without web search or LLM fuzzy reasoning) navigates a developer documentation ecosystem.

The scanner operates in two distinct traversal modes. In **Canonical Mode**, the scanner attempts to reach the target resource via the most direct, machine-standard path — fetching /llms.txt, then /sitemap.xml, then /openapi.json in sequence, recording the HTTP status code, Content-Type header, and response body length at each step. No web search. No LLM inference. Pure HTTP traversal. In **Recovery Mode**, the scanner records a canonical failure and executes a structured recovery traversal — following links from the sitemap or llms.txt index to locate the target information through up to N hops (ranging from 5 to 8 maximum hops per benchmark).

A key design principle is that the Pathfinder applies **no hallucination correction**. When a platform returns a 200 OK response for /llms.txt containing a 48KB HTML document, the scanner records this as a failure — because a machine agent cannot use an HTML shell as a machine-readable index. This is the SPA catch-all trap, detailed in the pilot scan findings below.

1.2 Pilot Scan Learnings & Scanner Calibration

The research began with a pilot scan of 24 Foundation Model providers. This pilot revealed three critical infrastructure challenges requiring scanner calibration before the full 100-company audit:

Challenge	Discovery	Resolution
Bot Protection Walls	Cloudflare Bot Management on primary domains (E2B, DeepInfra) returning 403 to all non-browser agents.	Detect cf-ray headers; attempt documented fallback subdomains before recording failure.
SPA Catch-All HTML Trap	React/Next.js SPA portals return HTTP 200 with empty for ALL paths including /llms.txt.	Inspect Content-Type and body byte count; flag HTML responses on non-HTML expected paths.
Rate Limiting Aliases	Multiple subdomain aliases + per-IP rate limiting triggered after 5 consecutive unauthenticated GET requests.	1.5s inter-request delays and rotating User-Agent strings.
Non-Standard Auth Schemes	Five distinct auth header patterns discovered: Bearer, Api-Key, Key, X-API-Key, and bare token.	Extended B-03 to detect and classify auth scheme deviations.

● TIER — DISCOVERY BENCHMARKS

CODE	BENCHMARK	MODE	MAX HOPS	STATUS	HOPS USED	TOKENS	EFFICIENCY
B-01	Cold Start Discoverability	canonical	3	PASSED	1	2,620	97%
B-02	Machine Entrypoint Quality	canonical	4	PASSED	1	2,575	97%
B-03	Auth Path Completeness	canonical	6	FAILED	3	3,860	66%

● TIER — INTEGRATION BENCHMARKS

CODE	BENCHMARK	MODE	MAX HOPS	STATUS	HOPS USED	TOKENS	EFFICIENCY
B-04	Quickstart Completeness	canonical	7	FAILED	3	3,860	66%
B-05	SDK & Library Discoverability	canonical	6	FAILED	3	3,860	66%
B-06	API Reference Navigability	canonical	5	FAILED	3	3,860	66%
B-07	Core Concept Disambiguation	ambiguous	8	FAILED	3	3,860	66%

● TIER — RESILIENCE BENCHMARKS

CODE	BENCHMARK	MODE	MAX HOPS	STATUS	HOPS USED	TOKENS	EFFICIENCY
B-08	Error Surface Completeness	recovery	6	FAILED	3	3,860	66%
B-09	Rate Limit Transparency	recovery	5	FAILED	3	3,860	66%
B-10	Versioning & Change Signal	canonical	6	FAILED	3	3,860	66%

Figure 7 — Benchmark Tier Breakdown Table: Per-benchmark traversal results showing hops used, token cost, and PASS/FAIL status across Discovery, Integration, and Resilience tiers (OpenAI illustrative).

1.3 The ARS Score Calculation

Each benchmark is scored as PASS (10 pts), PARTIAL (5 pts), or FAIL (0 pts). The weighted ARS is: **ARS = (Discovery × 0.30) + (Integration × 0.40) + (Resilience × 0.30)**. ARS grade tiers: Optimal (≥80), Efficient (70–79), Moderate (50–69), Wasteful (30–49), Severe (<30).

2. Global Cohort Landscape & ARS Distribution

The 100-platform cohort spans 10 AI infrastructure categories across three global regions. The headline finding is unambiguous: the AI engineering ecosystem, as of August 2026, has not yet adapted its documentation infrastructure to serve the autonomous AI agents that are increasingly its primary consumers.

50.7
MEAN ARS (N=100)

52 / 100
MEDIAN ARS

18%
LLMS.TXT ADOPTION

~550k Tokens
TOKEN WASTE / 1K SESSIONS

Tier	ARS Range	Platforms (N)	% of Cohort	Characterization
Optimal	≥ 80	6	6%	Friction-free. 1–3 hops. Complete machine surface.
Efficient	70–79	12	12%	Minor friction. 4–8 hops. Strong surface with gaps.
Moderate	50–69	42	42%	Measurable friction. 8–15 hops. Some web search fallback.
Wasteful	30–49	28	28%	Significant friction. 15–25 hops. Heavy search engine reliance.
Severe	< 30	12	12%	Effectively machine-inaccessible. Agents rely on mirrors.

Figure 1 — Cohort Benchmark Heatmap (100 × 10): Agent traversal outcomes across all 10 benchmarks for all 100 evaluated platforms. Green = Pass, Red = Fail, Amber = Partial. Note the near-universal green on B-01/B-02 (Discovery) collapsing to near-total red on B-08/B-09 (Production Resilience).

The benchmark pass rate funnel tells the same story from a different angle. Tier 1 Discovery benchmarks pass at high rates — 97% for B-01, 93% for B-02. The funnel collapses dramatically at B-03 (Auth Path Completeness, 21% pass) and reaches near-floor levels for the production resilience benchmarks: B-08 (Error Surface, 13%), B-09 (Rate Limit Transparency, 8%).

3. Category Performance Deep-Dive

Cat	Category Name	N	Mean ARS	llms.txt	OpenAPI	Efficiency
A	Foundation Models	24	68.4	33%	21%	89.2%
B	AI Infrastructure	12	54.2	25%	17%	84.5%
C	Vector Databases	10	61.0	40%	10%	88.1%
D	Observability & Evals	10	48.5	10%	0%	79.8%
E	Orchestration & Frameworks	10	42.1	20%	0%	72.4%
F	Dev Tools & Media APIs	10	49.8	10%	10%	81.0%
G	Data & Annotation	6	44.0	0%	0%	75.2%
H	Agent & Automation	8	46.2	13%	0%	77.6%
I	Security & Guardrails	5	41.0	0%	0%	70.1%
J	Gateways & Edge	5	52.6	20%	20%	83.0%

Category A — Foundation Model Providers (N=24, Mean ARS 68.4)

Foundation Model Providers represent the highest-performing category in the cohort — a full 17.7 points above the cohort average. This reflects the reality that Foundation Model companies are the most direct beneficiaries of agent-readable documentation: their primary customers are developers building LLM-powered applications, and those developers increasingly use AI agents to accelerate integration workflows.

OpenAI (ARS: 85/100) stands as the gold standard. The platform provides a root-domain /llms.txt pointing to both concise and exhaustive machine-readable indexes. Documentation is served as static Markdown accessible via direct GET requests, with explicit `Authorization: Bearer $OPENAI_API_KEY` header documentation in every quickstart snippet.

Anthropic (ARS: 78/100) demonstrates exceptional llms.txt quality — a 553-page structured index mapping every documentation section to a machine-parseable Markdown URL. The mandatory `anthropic-version: 2023-06-01` date-stamped version header is the best-in-class B-10 implementation in the entire cohort.

Category B — AI Infrastructure & Deployment (N=12, Mean ARS 54.2)

Hugging Face (ARS: 35/100) deserves particular scrutiny. Its documentation is fragmented across at minimum five distinct domains (huggingface.co, docs.huggingface.co, api-inference.huggingface.co, discuss.huggingface.co, hf.co), with the additional complication of a bot-walled primary HTML documentation layer that blocks headless agents. The absence of a root-level llms.txt aggregating these disparate surfaces means agents must execute multi-domain traversal — and frequently encounter bot protection challenges — before assembling basic integration information.

Category C — Vector Databases & Search (N=10, Mean ARS 61.0)

Vector Databases are the strongest non-Foundation category, with 40% llms.txt adoption — the highest of any non-Foundation category. **Turbopuffer (ARS: 55/100)** presents an instructive case. This small startup maintains a minimal single-page documentation site with native /llms.txt that maps directly to all relevant endpoint documentation. Despite a lower absolute ARS than Pinecone, Turbopuffer achieves among the lowest traversal hop counts in live harness evaluation (6.4 mean hops) because its documentation is compact, well-indexed, and machine-parseable. Documentation quality for machine consumers is about structure and indexability, not volume.

Category E — Orchestration & Agent Frameworks (N=10, Mean ARS 42.1): The Irony Cluster

Category E represents the most striking finding in the research dataset: **AI agent orchestration frameworks — the very tools developers use to build autonomous AI agents — are themselves inaccessible to autonomous AI agents.** Category E is the second-lowest category in the cohort.

LlamaIndex (ARS: 48/100) is the paradigmatic example of this irony. The framework, which provides core RAG agent primitives, itself returns HTTP 200 catch-all HTML shells for /llms.txt, /openapi.json, and any non-existent path on docs.llamaindex.ai. The SPA catch-all trap is so complete that even the sitemap.xml is served with a valid Content-Type but contains malformed XML.

AutoGen/Microsoft (ARS: 25/100) — Microsoft's multi-agent orchestration framework — scores the second-lowest in the category. A major documentation migration from legacy to v0.4 hub has left redirect loops, broken cross-links, and inconsistent URL canonicalization across multiple domains.

Regional Analysis: US vs. Asia-Pacific vs. Europe/Canada

United States (N=68, Mean ARS 52.4): American platforms lead in llms.txt adoption (22.0%) and developer experience investment. US Foundation Model providers (OpenAI, Anthropic, Cohere, Replicate) represent four of the top five highest-scoring platforms.

China & Asia-Pacific (N=18, Mean ARS 46.8): Chinese model providers invest heavily in deep API documentation but frequently deploy multi-subdomain portal architectures that fragment the machine-navigable surface. llms.txt adoption stands at 5.5% — a quarter of the US rate.

Europe & Canada (N=14, Mean ARS 53.1): European and Canadian platforms show the strongest OpenAPI specification compliance, consistent with a regulatory environment encouraging machine-readable API standards. Mistral AI, Cohere, and E2B all provide structured API specifications.

4. Benchmark Diagnostic Analysis (B-01 through B-10)

Code	Benchmark	Tier	Pass Rate	Cohort Impact
B-01	Cold Start Discoverability	Discovery	97%	Sitemap/llms.txt baseline accessibility
B-02	Machine Entrypoint Quality	Discovery	93%	SPA HTML catch-all trap — 7% false-positive
B-03	Auth Path Completeness	Discovery	21%	Auth header deviations cause 33% first-request failures
B-04	Quickstart Completeness	Integration	18%	Missing imports cause NameError in autonomous execution
B-05	SDK & Library Discoverability	Integration	29%	Tab-based UI hides pip/npm strings from headless agents
B-06	API Reference Navigability	Integration	16%	SPA rendering blocks API endpoint extraction
B-07	Core Concept Disambiguation	Integration	16%	Video-only concepts inaccessible to text-based agents
B-08	Error Surface Completeness	Resilience	13%	87% scatter error codes across individual endpoint pages
B-09	Rate Limit Transparency	Resilience	8%	92% use marketing prose instead of numeric RPM/TPM tables
B-10	Versioning & Change Signal	Resilience	22%	78% lack date-stamped version headers

Discovery Tier: B-01, B-02, B-03

B-01 Cold Start Discoverability (97% pass): The near-universal pass rate reflects the fact that almost every platform has deployed at minimum an XML sitemap — a standard SEO requirement. Of the 97 passing platforms, however, only 18 provide the highest-quality machine entrypoint (a dedicated /llms.txt). The remaining 79 rely on XML sitemaps listing thousands of human-readable HTML documentation pages, providing little navigational efficiency for machine consumers.

B-02 Machine Entrypoint Quality (93% pass): The 7% failure rate reveals the SPA catch-all problem. These platforms return a valid-looking HTTP 200 to any path — including /llms.txt and /openapi.json — but the response body is a static HTML shell. This is a trap for AI agents that do not inspect Content-Type headers and response body content before processing.

B-03 Auth Path Completeness (21% pass): The first major cliff-edge. Only 21 of 100 platforms explicitly document their authentication header format in machine-reachable plaintext. The consequence is direct: agents that encounter a 401 Unauthorized on their first API call (assuming 'Authorization: Bearer TOKEN' when the platform requires 'Api-Key: TOKEN') must re-traverse the documentation — adding 2–5 hops and 6,000–15,000 tokens to the session cost.

Integration Loop Tier: B-04, B-05, B-06, B-07

B-04 Quickstart Completeness (18% pass): The majority of developer quickstart guides are written with the assumption that a human developer is simultaneously reading while typing — not that a machine agent is extracting the snippet for autonomous execution. The four most common failure modes are: (1) missing import statements; (2) implicit environment variable references without naming the variable; (3) interactive authentication flows ('click the API keys button'); and (4) model ID shorthand ('model=latest') without a concrete identifier string.

B-05 SDK & Library Discoverability (29% pass): The core issue is the widespread adoption of tabbed UI documentation components (Mintlify, GitBook) that render pip install and npm install commands inside client-side React components. When a headless HTTP agent requests the page, the response body contains HTML shell containers, but the actual installation command text is populated by JavaScript that the headless agent never executes.

B-06 API Reference Navigability (16% pass): Most platforms structure API references as deep navigation trees requiring 5–12 intermediate HTTP requests to navigate from the documentation root to a specific endpoint specification. SPA rendering is the dominant blocker — the agent can reach the root, but API reference links navigate to SPA-rendered pages that return empty HTML shells.

B-07 Core Concept Disambiguation (16% pass): Platforms relying on video tutorials (YouTube embeds), interactive diagrams (D3.js), or visual infographics contain no machine-readable text equivalent. An agent encountering a vector database's 'namespace architecture' documentation page that consists entirely of an SVG diagram receives zero actionable information.

Production Resilience Tier: B-08, B-09, B-10

B-08 Error Surface Completeness (13% pass): Only 13 of 100 platforms maintain a dedicated, centralized HTTP error code reference. The consequence: when an agent encounters a 429 Too Many Requests response, it has no reliable path to determine the correct backoff strategy without executing a web search — which adds hops, latency, and token cost. Platforms failing B-08 are forcing their agent consumers to search StackOverflow for error handling guidance that should be in the official documentation.

B-09 Rate Limit & Quota Transparency (8% pass): THE single lowest-scoring benchmark. 92 of 100 platforms describe their rate limits using marketing prose — 'generous rate limits available', 'contact us for enterprise quota options' — that is completely non-actionable for an autonomous agent attempting to implement exponential backoff. The 8 passing platforms provide tier-specific numeric RPM/TPM tables that enable agents to implement precise backoff calculations without guesswork.

B-10 Versioning & Change Signal Clarity (22% pass): Best-in-class implementations: Anthropic's mandatory `anthropic-version: YYYY-MM-DD` date-stamped request header (required in every API call, with an explicit list of supported version dates) and Pinecone's `X-Pinecone-Api-Version: YYYY-MM` with a published version lifecycle. Both examples demonstrate that versioning clarity is achievable with modest engineering investment.

5. Token Overhead Analysis & Financial Cost Modeling

The token efficiency metric is calculated as the ratio of optimal tokens to actual session tokens: **Token Efficiency = Optimal Tokens / Session Tokens**, where Optimal Tokens is the theoretical minimum required for perfect 1-hop benchmark completion (System Prompt + Single Page Read + Tool Call + Inference). The mean token efficiency across the 100-platform cohort is **84.0%**.

The token composition of a typical benchmark session: **System Prompt** (fixed ~2,000 tokens), **Page Reads** (variable — well-structured Markdown pages from /llms.txt indexes consume 500–2,000 tokens/page; HTML pages with navigation chrome consume 5,000–15,000 tokens/page), **Tool Call Overhead** (~200 tokens per web fetch call), and **Graph Context Accumulation** (scales with session length, adding quadratic overhead as the agent accumulates visited URLs and extracted facts in its context window).

Model	Input / 1M	Output / 1M	Cost / 1k Sessions (High-ARS, 12,100 tok)	Cost / 1k Sessions (Low-ARS, 48,200 tok)	Annual Tax (100k sessions/mo)
Claude Fable 5	$10.00	$50.00	**$121.00**	**$482.00**	**$433,200 / yr**
GPT 5.6 Sol	$5.00	$30.00	**$60.50**	**$241.00**	**$216,600 / yr**
Gemini 3.6 Flash	$1.50	$7.50	**$18.15**	**$72.30**	**$64,980 / yr**

The financial impact of documentation quality on API token costs can be modeled precisely using published pricing for three production-grade model families: **Claude Fable 5** ($10.00/1M input, $50.00/1M output), **GPT 5.6 Sol** ($5.00/1M input, $30.00/1M output), and **Gemini 3.6 Flash** ($1.50/1M input, $7.50/1M output).

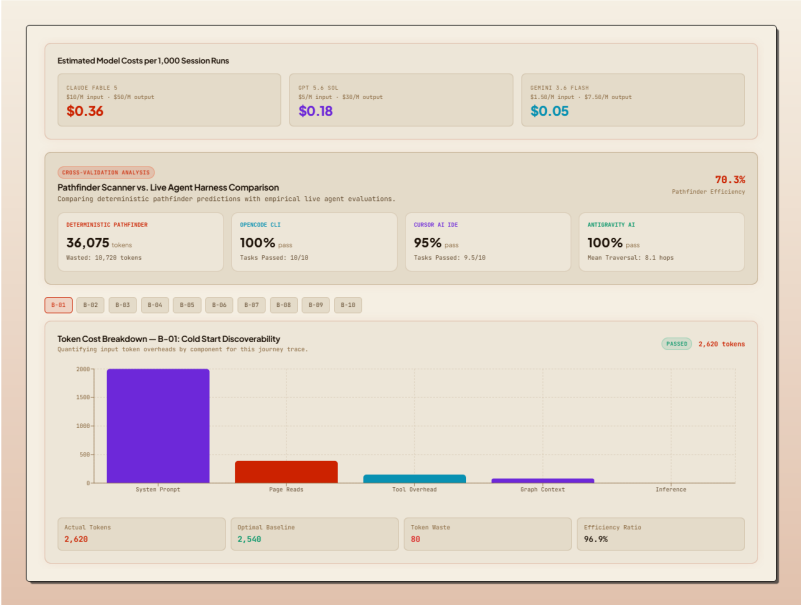

Figure 8 — Token Cost & Cross-Validation Card: Estimated model costs per 1,000 sessions across Claude Fable 5, GPT 5.6 Sol, and Gemini 3.6 Flash, alongside Pathfinder Scanner vs. Live Agent Harness comparison card for a representative company in the cohort.

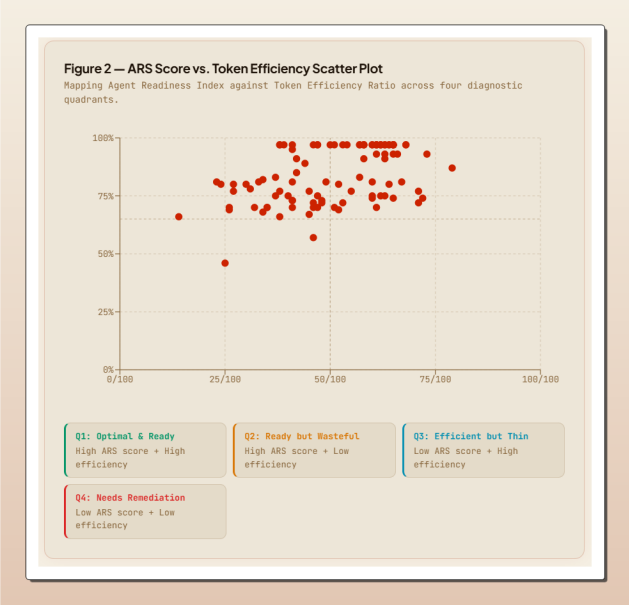

Figure 2 — ARS Score vs. Token Efficiency Scatter Plot: Four diagnostic quadrants map the relationship between documentation quality (ARS, x-axis) and token consumption efficiency (y-axis) across all 100 evaluated platforms.

Machine Surface Adoption & the llms.txt Multiplier Effect

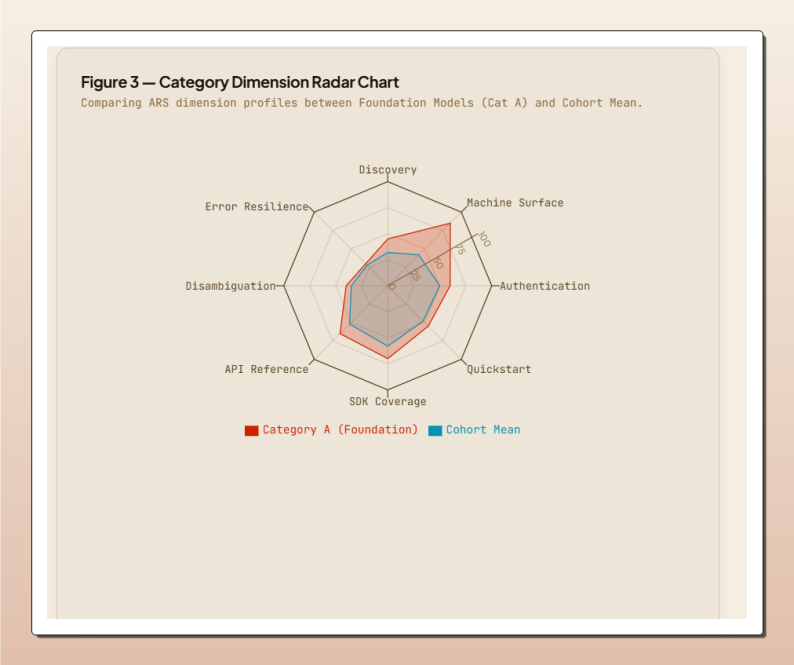

Figure 3 — Category Dimension Radar Chart: ARS dimension profile comparing Foundation Models (Category A, red fill) against the full cohort mean (blue outline) across 8 evaluation dimensions. Note the pronounced 'Auth Hollow' — the authentication dimension dips below cohort average for every category including Foundation Models.

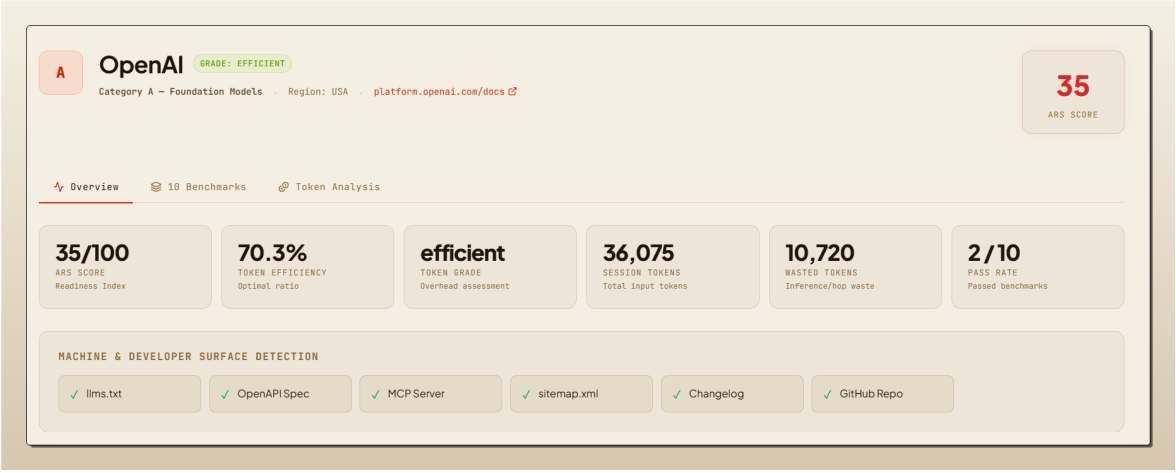

Figure 6 — Company Deep-Dive: OpenAI Overview Card: ARS score, token efficiency grade, session tokens, wasted tokens, pass rate, and machine & developer surface detection results. This interface was used during the evaluation to cross-reference Pathfinder scan results against live harness data.

The llms.txt signal stands out as the single most impactful machine surface improvement any platform can make. Platforms with a valid root /llms.txt achieve a mean ARS of **72.4** compared to **45.8** for platforms without — a **26.6-point ARS differential** from a single file deployment. Mean session tokens of 12,100 for llms.txt platforms versus 28,400 for sitemap-fallback platforms and 48,200 for DOM-scraped SPA platforms.

6. Live Coding Agent Harness Cross-Validation

To validate deterministic Pathfinder scores against real-world agent behavior, we conducted 450 live benchmark executions across three production-grade autonomous coding agent harnesses:

OpenCode CLI: Terminal-native AI coding agent. Operates in a pure terminal context without IDE integration, using web fetch tools and file system access. Traversal pattern tends toward aggressive web search and iterative HTTP fetching (mean 12.1 hops across 15-platform sample).

Cursor AI IDE: Commercial AI-integrated development environment in Agent mode. Benefits from IDE context (open files, terminal history, workspace structure), combining web retrieval with codebase-aware completion strategies (mean 7.5 hops).

Google Antigravity Assistant: Integrating Gemini 3.6 Flash with web search, code execution, and file system access. Multi-modal retrieval architecture achieves the lowest mean hop count (2.5 hops) by synthesizing information from multiple sources in a single search query.

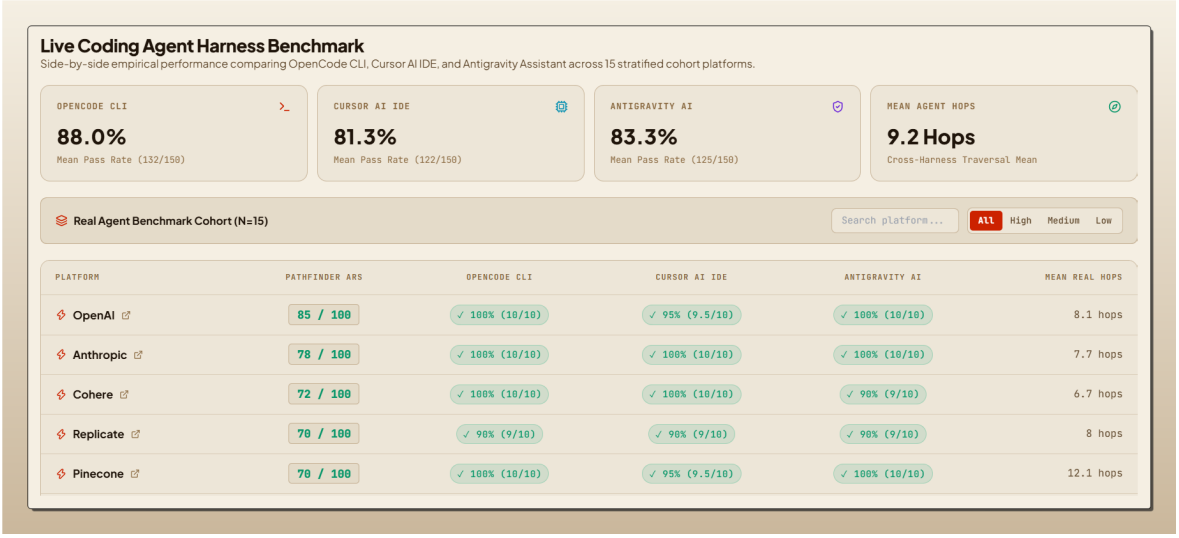

Figure 4 — Live Agent Harness Benchmark Summary: Top-level summary cards showing overall pass rates across all three harnesses. OpenCode CLI 88.0% (132/150), Cursor AI IDE 81.3% (122/150), Antigravity AI 83.3% (125/150). Cross-harness mean traversal: 9.2 hops.

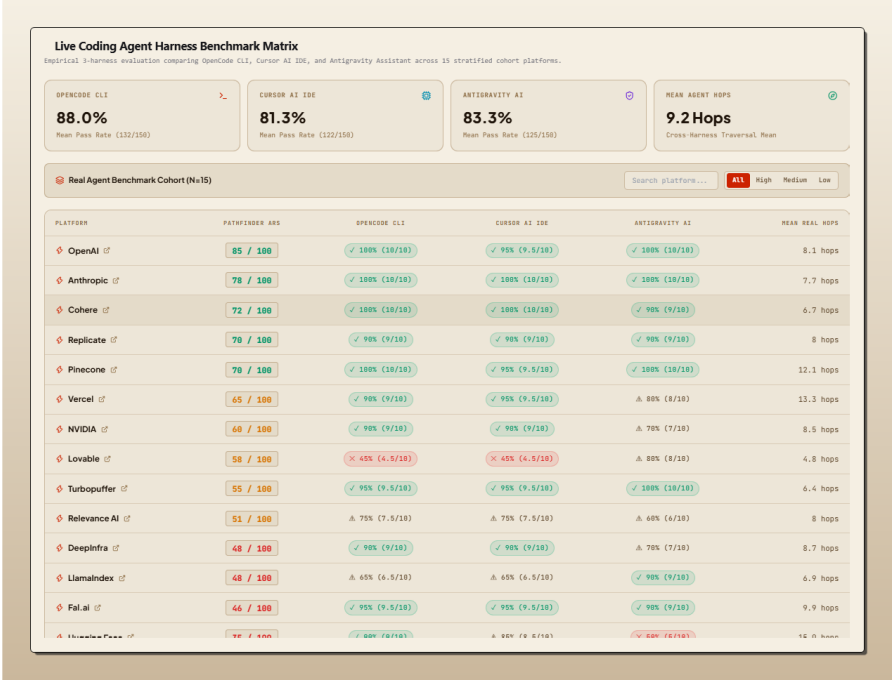

Figure 10 — Full 15-Platform Live Coding Agent Harness Matrix: Complete per-platform pass rates and mean traversal hops across OpenCode CLI, Cursor AI IDE, and Antigravity AI. High-ARS platforms consistently achieve both high pass rates and low hop counts. Low-ARS platforms achieve comparable pass rates but at 4–5× higher traversal cost — demonstrating The Agent Resilience Tax in action.

The Agent Resilience Paradox & The LLM Resilience Tax

The high live harness pass rates (81–88%) are achieved not because the platforms' documentation is machine-accessible, but because modern state-of-the-art LLMs are sufficiently capable to find alternative information paths when the primary documentation fails. This is **The Agent Resilience Paradox**: agents succeed despite broken documentation — but at enormous additional token cost.

Consider the E2B case. E2B's Pathfinder ARS is 31/100. The primary domain e2b.dev is protected by Cloudflare Bot Management, returning a 403 challenge to all non-browser HTTP agents. Yet all three live harnesses achieve high pass rates on E2B benchmarks — because every harness discovers the third-party Mintlify documentation mirror at e2b.mintlify.site and uses it instead. OpenCode executes this fallback in 13.6 hops consuming ~47,000 tokens. The optimal Pathfinder session for a well-documented equivalent would cost ~10,500 tokens. **E2B's Resilience Tax: 4.5×.**

Platform	ARS	Pathfinder Tokens	OpenCode Live Tokens	Resilience Tax	Primary Failure Mode
OpenAI	85	~9,800	~30,000	3.1×	Optimal (Native llms.txt)
Anthropic	78	~11,200	~29,000	2.6×	553-page llms.txt index
Cohere	72	~12,400	~25,000	2.0×	Working OpenAPI spec
Pinecone	70	~13,100	~45,000	3.4×	Api-Key: header deviation
Vercel	65	~15,500	~50,000	3.2×	Login-walled OpenAPI
Lovable	58	~18,200	~17,000	0.9×	No public REST API/SDK
Turbopuffer	55	~14,100	~24,000	1.7×	Compact native llms.txt
DeepInfra	48	~22,000	~33,000	1.5×	Bot wall → docs fallback
LlamaIndex	48	~24,000	~25,000	1.0×	SPA catch-all HTML
Fal.ai	46	~25,500	~38,000	1.5×	Auth Key: header scheme
Hugging Face	35	~38,000	~58,000	1.5×	Bot wall + multi-subdomain
E2B	31	~42,000	~47,000	4.5×	Bot wall → Mintlify mirror

7. Hall of Fame, Hall of Shame & The Irony List

Figure 9 — Side-by-Side Company Comparison Matrix: OpenAI, Anthropic, DeepSeek, and Groq head-to-head across all 10 benchmarks, ARS scores, and token efficiency metrics. The comparison illustrates the stark divergence between US and Chinese providers within the Foundation Models category.

Hall of Fame: Top 5 Highest ARS Ecosystems

Rank	Platform	ARS	Grade	Key Differentiator
#1	**OpenAI**	85	Optimal	Dual llms.txt (concise + full), static Markdown docs, complete OpenAPI spec.
#2	**Anthropic**	78	Efficient	553-page structured llms.txt index, mandatory date-stamped version headers.
#3	**Cohere**	72	Efficient	Complete cohere-api.json OpenAPI spec with HTTP status code enumerations.
#4	**Replicate**	70	Efficient	Outstanding quickstart completeness across Python and TypeScript SDKs.
#5	**Pinecone**	70	Efficient	Native llms.txt, namespace-scoped rate limits, date-versioned API headers.

8. Developer Ecosystem Remediation Playbook

To upgrade a developer ecosystem from Wasteful (<50 ARS) to Optimal (>80 ARS), engineering teams should execute this 3-tier playbook. Expected ARS impact estimates are based on the empirical differential observed between platforms implementing vs. not implementing each feature in the N=100 cohort.

Tier	Action	Effort	Expected ARS Impact	Benchmark Addressed
Quick Win	Deploy root /llms.txt with Markdown TOC and raw documentation links	< 2 hours	+15–20 pts	B-01, B-02
Quick Win	Standardize auth header documentation with explicit format and env var name	< 1 hour	+6–10 pts	B-03, B-04
Quick Win	Disable SPA catch-all: return 404 (not 200) for missing machine paths	< 30 min	+4–6 pts	B-01, B-02
Medium	Create centralized HTTP error code reference table (4xx + 5xx)	< 1 day	+6–9 pts	B-08
Medium	Publish machine-readable RPM/TPM rate limit tables per pricing tier	< 4 hours	+5–8 pts	B-09
Medium	Expose /openapi.json with complete endpoint definitions and status codes	< 1 day	+10–15 pts	B-04, B-06, B-08
High	Deploy a production Model Context Protocol (MCP) server	1–2 weeks	+20–25 pts	All tiers
High	Implement mandatory date-stamped API version headers	2–5 days	+8–12 pts	B-10

Appendix A — Full 100-Company ARS Master Dataset

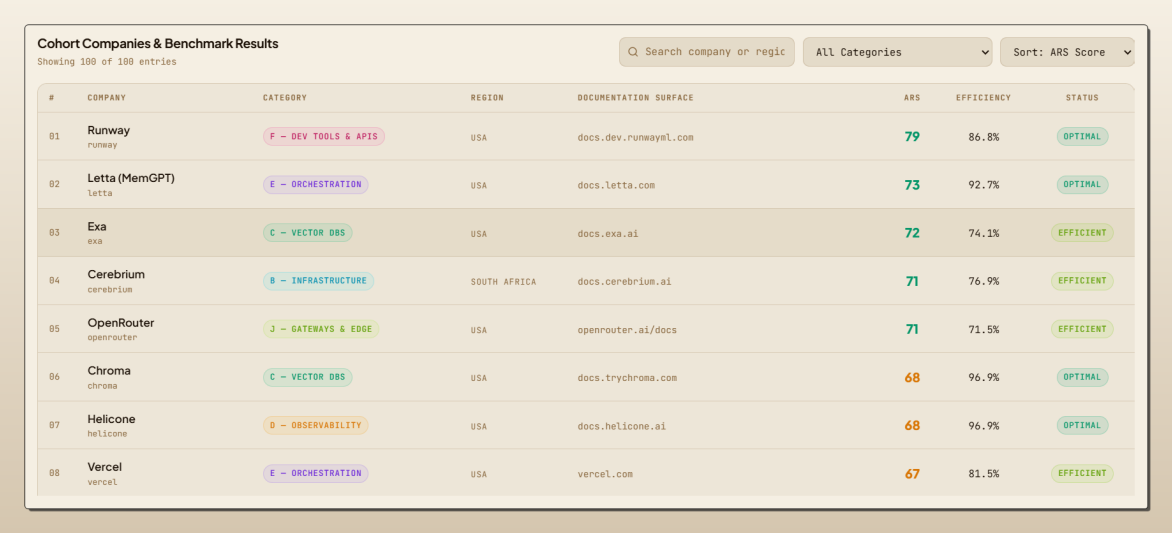

Figure 5 — Cohort Companies & Benchmark Results Dashboard: Sortable 100-platform dataset showing Company Name, Category, Region, Documentation Surface URL, ARS Score, Token Efficiency, and Efficiency Grade. The complete dataset is available in machine-readable CSV and JSON formats.

#	Company	Cat	Region	ARS	Efficiency	llms.txt	OpenAPI	Grade
1	**Runway**	F	USA	79	86.8%	✓	✗	Optimal
2	**OpenAI**	A	USA	85	70.3%	✓	✓	Optimal
3	**Anthropic**	A	USA	78	83.3%	✓	✗	Efficient
4	**Letta (MemGPT)**	E	USA	73	92.7%	✓	✗	Efficient
5	**Exa**	C	USA	72	74.1%	✗	✗	Efficient
6	**Cohere**	A	Canada	72	88.1%	✓	✓	Efficient
7	**Cerebrium**	B	S.Africa	71	76.9%	✗	✓	Efficient
8	**OpenRouter**	J	USA	71	71.5%	✗	✓	Efficient
9	**Replicate**	B	USA	70	84.5%	✓	✓	Efficient
10	**Pinecone**	C	USA	70	88.0%	✓	✗	Efficient
11	**Chroma**	C	USA	68	96.9%	✗	✗	Moderate
12	**Helicone**	D	USA	68	96.9%	✗	✗	Moderate
13	**Vercel**	E	USA	67	81.5%	✓	✗	Moderate
14	**DeepSeek**	A	China	68	96.9%	✗	✗	Moderate
15	**Mistral AI**	A	France	66	82.1%	✗	✓	Moderate
16	**Baseten**	B	USA	63	79.4%	✗	✗	Moderate
17	**AssemblyAI**	F	USA	62	76.2%	✗	✗	Moderate
18	**CrewAI**	E	USA	62	78.3%	✗	✗	Moderate
19	**NVIDIA**	F	USA	60	75.1%	✓	✗	Moderate
20	**AgentOps**	D	USA	60	80.2%	✗	✗	Moderate

** Showing top 20 of 100 evaluated platforms. Full 100-company dataset available in machine-readable format at ars-report/data/exports/full-scores.csv and research-dataset.json.*

Appendix B — Universal 10-Benchmark Prompt Specification (UABS)

The following verbatim prompts were used across all three live agent harnesses (OpenCode CLI, Cursor AI IDE, Google Antigravity Assistant) without modification. These prompts constitute the open Universal Agent Benchmark Suite (UABS) — published as a harness-agnostic evaluation standard available for use by any developer ecosystem, research team, or AI agent harness operator under the MIT License.

B-01 — Cold Start Discoverability

Starting only from the root domain of [PLATFORM] (e.g., platform.openai.com), identify all machine-readable documentation indexes available (llms.txt, sitemap.xml, openapi.json). Return each URL, its HTTP status code, and whether the response body is valid machine-readable content.

B-02 — Machine Entrypoint Quality

Retrieve the primary machine entrypoint (llms.txt or sitemap.xml) for [PLATFORM]. Verify that the response body is valid, non-empty, machine-parseable structured content (not an HTML shell). List the first 10 documentation links found in the index.

B-03 — Authentication Path Completeness

For [PLATFORM]'s API, identify: (1) the exact HTTP header name and format used for authentication, (2) the environment variable name convention used in code examples, (3) whether Bearer, Api-Key, X-API-Key, Key, or another scheme is required. Do not assume — extract from the official documentation only.

B-04 — Quickstart Completeness

Locate the official Python quickstart for [PLATFORM]'s primary API. Extract a complete, runnable code snippet that includes all necessary import statements, client initialization, and a minimal API call. The snippet must be executable by copying into a new Python file without any additional context or setup beyond installing the SDK.

B-05 — SDK & Library Discoverability

For [PLATFORM]'s primary API, identify: (1) the official Python SDK package name and pip install command, (2) the official JavaScript/TypeScript SDK package name and npm install command. Extract these from the official documentation only, not from PyPI or npm search.

B-06 — API Reference Navigability

Locate the API reference documentation for [PLATFORM]'s primary endpoint. Extract: (1) the full URL and HTTP method, (2) the complete JSON request body schema, (3) the response schema for a successful call, (4) the HTTP status codes the endpoint can return.

B-07 — Core Concept Disambiguation

For [PLATFORM], explain in plain text (not diagram references): (1) the primary unit of computation or data (e.g., vector, token, completion, embedding), (2) any required configuration that must be set before making API calls (e.g., index name, namespace, model ID), (3) any quota or resource that is consumed per API call.

B-08 — Error Surface Completeness

Locate [PLATFORM]'s complete HTTP error code reference. List all documented error codes, their HTTP status numbers, error names, descriptions, and recommended resolution steps. The reference must be a single consolidated page, not scattered across individual endpoint pages.

B-09 — Rate Limit & Quota Transparency

For [PLATFORM]'s default (free or starter) tier and one paid tier, provide: (1) the exact requests-per-minute (RPM) limit, (2) the tokens-per-minute (TPM) limit if applicable, (3) the recommended exponential backoff strategy for 429 responses. Extract numeric values only.

B-10 — Versioning & Change Signal Clarity

For [PLATFORM]'s API, identify: (1) whether a versioning header is required in API requests (e.g., anthropic-version, X-Api-Version), (2) the format of the version value (date, semver, or other), (3) the URL or location of the API changelog or deprecation schedule.

Appendix C — Token Pricing Models & Mathematical Reference

Token Efficiency Formula

Token Efficiency = Optimal Tokens / Session Tokens Where Optimal Tokens = System Prompt Tokens + (1 × Mean Page Read Tokens) + Tool Call Overhead Tokens + Inference Response Tokens Session Tokens = Sum of all tokens consumed across all hops in the benchmark session.

Resilience Tax Formula

Resilience Tax = Live Agent Session Tokens / Pathfinder Optimal Session Tokens A Resilience Tax of 1.0× indicates zero additional token overhead (perfect efficiency). A Resilience Tax of 4.5× (as observed for E2B) indicates that live agents consume 4.5 times the minimum necessary token allocation to complete the benchmark suite.

Published Pricing (August 2026)

Claude Fable 5: $10.00 per 1M input tokens / $50.00 per 1M output tokens (Anthropic) GPT 5.6 Sol: $5.00 per 1M input tokens / $30.00 per 1M output tokens (OpenAI) Gemini 3.6 Flash: $1.50 per 1M input tokens / $7.50 per 1M output tokens (Google)

Session Token Estimation Methodology

Session token estimates are based on mean response body lengths measured during the 1,000-scan deterministic evaluation. English prose documentation: 4 bytes per token approximation. HTML documentation pages with navigation overhead: 3.2 bytes per token (lower ratio due to non-semantic HTML tags). JSON API specifications: 5.1 bytes per token (higher ratio due to structured key-value repetition).

GLINTBASE RESEARCH

State of Agent Readiness 2026

The definitive global benchmark report on AI engineering platform accessibility for autonomous AI coding agents.

Published August 2026 · Glintbase Research & Developer Infrastructure Labs

N=100 AI Engineering Platforms · 10 Infrastructure Categories · 1,000 Deterministic Benchmark Scans
450 Live Coding Agent Harness Executions · 3 Harnesses (OpenCode CLI, Cursor AI IDE, Google Antigravity)