

# The Agent *Observability* Gap

Why agents that look healthy in a playground go sideways the moment they meet real users, and what a useful observability stack for agent-shaped systems has to cover.

---

**57%** of teams now run LLM agents in production, and 89% have stood up some observability.<sup>[6]</sup>

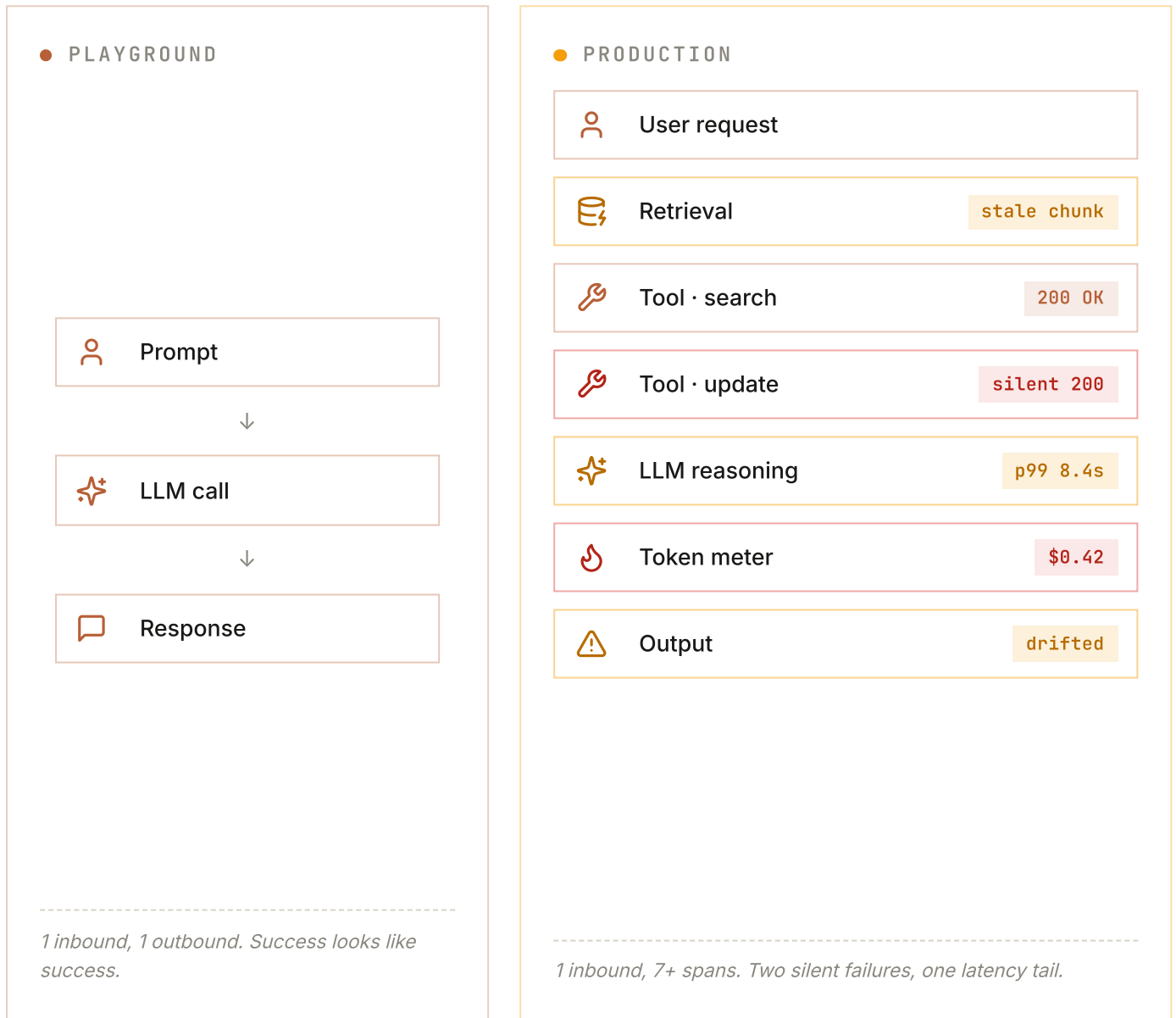
*Most inherited observability built for request-and-response web apps, a shape their agent outgrew long ago.*

---

— PLAYGROUND VS PRODUCTION

# Why the demo lies

A single user request in a playground is a one-shot prompt-and-response. The same request in production fans out into retrieval, tool calls, and retries, and most of those steps never surface in traditional logs.



**TAKEAWAY.** The same agent that looked healthy in the playground now depends on retrieval quality, tool correctness, retry loops, and spend per request. Any one of those can degrade without throwing an error, and without traces you'll find out from a user.

---

**FIVE FAILURE MODES**

# Failures that only appear in production

Agent failures rarely raise exceptions. These five are the ones most teams meet first, and the five that traditional logging is least equipped to spot.



## Retrieval Failure

Your RAG pipeline returns stale, wrong, or irrelevant chunks, the model answers confidently anyway.

*APM sees a successful vector query. It can't judge whether the retrieved context was actually right.*



## Tool-Call Failure

The agent picks the wrong tool, passes bad parameters, or the tool fails silently mid-chain.

*HTTP 200 hides semantic errors. Logs don't know which tool the agent should have called, or whether a prompt injection just redirected it.*



## Latency Spikes

p50 looks fine. p99 is 12 seconds on the exact flows your power users hit most.

*Request-level metrics average away the tail. Per-span latency across an agent chain is invisible.*



## Cost Blowouts

A reasoning loop burns 40k tokens on a single request. One user racks up \$200 before lunch.

*Infra dashboards track CPU, not tokens. Cost per trace, per user, per model is not a native concept.*



## Output Drift

Same input, different output. Quality silently degrades after a model upgrade or prompt tweak.

*There's no 'error' to log. You need evals and historical replay to catch behavioral regressions.*

— WHAT EACH TIER ACTUALLY SEES

# Why standard logging breaks down

APM and log aggregation were designed for stateless request-and-response apps. Inside one agent run there are seven or more spans, and the useful signal lives between them. Here is what each tier can observe.








<div data-bbox="145 645 204 705"></div> <h3>Traditional APM</h3> <p>Built for stateless HTTP</p> <div style="text-align: right;">20% of an agent run</div> <table border="0"> <tr> <td><b>SEES</b></td> <td><b>MISSES</b></td> </tr> <tr> <td> <ul style="list-style-type: none"> <li>⊙ HTTP request and response</li> <li>⊙ Top-level status codes</li> <li>⊙ Process CPU and memory</li> </ul> </td> <td> <ul style="list-style-type: none"> <li>⚠ Per-step reasoning</li> <li>⚠ Tool inputs, outputs, decisions</li> <li>⚠ Silent semantic failures on HTTP 200</li> </ul> </td> </tr> </table>	<b>SEES</b>	<b>MISSES</b>	<ul style="list-style-type: none"> <li>⊙ HTTP request and response</li> <li>⊙ Top-level status codes</li> <li>⊙ Process CPU and memory</li> </ul>	<ul style="list-style-type: none"> <li>⚠ Per-step reasoning</li> <li>⚠ Tool inputs, outputs, decisions</li> <li>⚠ Silent semantic failures on HTTP 200</li> </ul>
<b>SEES</b>	<b>MISSES</b>			
<ul style="list-style-type: none"> <li>⊙ HTTP request and response</li> <li>⊙ Top-level status codes</li> <li>⊙ Process CPU and memory</li> </ul>	<ul style="list-style-type: none"> <li>⚠ Per-step reasoning</li> <li>⚠ Tool inputs, outputs, decisions</li> <li>⚠ Silent semantic failures on HTTP 200</li> </ul>			
<div data-bbox="145 1025 204 1086"></div> <h3>Log aggregation</h3> <p>print() plus aggregation</p> <div style="text-align: right;">50% of an agent run</div> <table border="0"> <tr> <td><b>SEES</b></td> <td><b>MISSES</b></td> </tr> <tr> <td> <ul style="list-style-type: none"> <li>⊙ Whatever you remembered to log</li> <li>⊙ Error messages and stack traces</li> <li>⊙ Free-text breadcrumbs</li> </ul> </td> <td> <ul style="list-style-type: none"> <li>⚠ Causal chain between steps</li> <li>⚠ Prompts, completions, token cost</li> <li>⚠ Drift over time</li> </ul> </td> </tr> </table>	<b>SEES</b>	<b>MISSES</b>	<ul style="list-style-type: none"> <li>⊙ Whatever you remembered to log</li> <li>⊙ Error messages and stack traces</li> <li>⊙ Free-text breadcrumbs</li> </ul>	<ul style="list-style-type: none"> <li>⚠ Causal chain between steps</li> <li>⚠ Prompts, completions, token cost</li> <li>⚠ Drift over time</li> </ul>
<b>SEES</b>	<b>MISSES</b>			
<ul style="list-style-type: none"> <li>⊙ Whatever you remembered to log</li> <li>⊙ Error messages and stack traces</li> <li>⊙ Free-text breadcrumbs</li> </ul>	<ul style="list-style-type: none"> <li>⚠ Causal chain between steps</li> <li>⚠ Prompts, completions, token cost</li> <li>⚠ Drift over time</li> </ul>			
<div data-bbox="145 1406 204 1467"></div> <h3>Agent observability</h3> <p>Built for stateful graphs</p> <div style="text-align: right;">100% of an agent run</div> <table border="0"> <tr> <td><b>SEES</b></td> <td><b>MISSES</b></td> </tr> <tr> <td> <ul style="list-style-type: none"> <li>⊙ Full span tree with parent and child causality</li> <li>⊙ Prompts, completions, tool I/O, token cost per step</li> <li>⊙ Eval scores, drift over time, replay</li> </ul> </td> <td> <p><i>Nothing in the intended scope.</i></p> </td> </tr> </table>	<b>SEES</b>	<b>MISSES</b>	<ul style="list-style-type: none"> <li>⊙ Full span tree with parent and child causality</li> <li>⊙ Prompts, completions, tool I/O, token cost per step</li> <li>⊙ Eval scores, drift over time, replay</li> </ul>	<p><i>Nothing in the intended scope.</i></p>
<b>SEES</b>	<b>MISSES</b>			
<ul style="list-style-type: none"> <li>⊙ Full span tree with parent and child causality</li> <li>⊙ Prompts, completions, tool I/O, token cost per step</li> <li>⊙ Eval scores, drift over time, replay</li> </ul>	<p><i>Nothing in the intended scope.</i></p>			

**TAKEAWAY.** APM and logs are not wrong, they are just scoped for a different shape. Stateful, multi-step agents need span-level visibility, eval signal, and replay as primitives, not add-ons.

— SEVEN LAYERS, NOT ONE

# What a working observability stack covers

A useful stack for agent-shaped systems has to do all seven at once: traces, evals, cost, latency, drift, replay, and guardrails. Miss any one and you're debugging blindfolded on the others.






<p><b>1</b>  <b>Traces</b></p> <p>Full reasoning path and tool chain for every agent run.</p> <hr/> <p><b>89%</b> of teams have stood up some observability, but only 62% have step-level tracing. <sup>[6]</sup></p>	<p><b>2</b>  <b>Evals</b></p> <p>LLM-as-a-judge scores, golden datasets, regression checks.</p> <hr/> <p><b>20% to 40%+</b> auto-approve rate climbs with session count, evidence, not vibes, earns autonomy (Anthropic). <sup>[8]</sup></p>
<p><b>3</b>  <b>Cost</b></p> <p>Token spend attributed per model, per app, per user.</p> <hr/> <p><b>\$3.5B to \$8.4B</b> LLM API spend in 6 months (Menlo Ventures). Per-span attribution turns that curve into a line item. <sup>[12]</sup></p>	<p><b>4</b>  <b>Latency</b></p> <p>p50, p95, p99 broken down by span across the whole chain.</p> <hr/> <p><b>42% to 54%</b> YoY jump in AI monitoring adoption (New Relic 2025 Forecast). <sup>[9]</sup></p>
<p><b>5</b>  <b>Drift</b></p> <p>Behavioral change detection over time and across versions.</p> <hr/> <p><b>1.8% to 24.2%</b> hallucination-rate spread across frontier models on Vectara's HHEM, a 22-point gap. <sup>[4]</sup></p>	<p><b>6</b>  <b>Replay &amp; Debug</b></p> <p>Re-run any trace, inspect any step, fix with confidence.</p> <hr/> <p><b>dozens</b> of distinct agent failure modes Forrester catalogues. Replay turns each one into a reproducible bug. <sup>[11]</sup></p>
<p><b>7</b>  <b>Guardrails</b></p> <p>Prompt-injection, PII leakage, and policy violations caught in-flight.</p> <hr/> <p><b>#1</b> Prompt injection tops OWASP's 2025 LLM Top 10. Every agent with tool access is an injection target. <sup>[2]</sup></p>	

---

— PRINT, TICK, SHARE

# Is your LLM app production-ready?

Five yes-or-no questions, each tied to one of the pillars. Tick the boxes with a pen, then count the yeses and compare against the rubric.

-  01 **Can you trace every agent step?**  
*Without a full trace, debugging is guesswork and fixes are gambles.*
- 
-  02 **Can you see exactly where it fails?**  
*Retrieval, tool calls, and prompts all fail differently. You need to see which one broke.*
- 
-  03 **Can you inspect latency and cost per step?**  
*One slow tool or one runaway loop can tank UX and burn budget. Per-span visibility prevents both.*
- 
-  04 **Can you compare runs over time?**  
*Prompts and models change. Evals and replay tell you whether quality held or slipped.*
- 
-  05 **Can you catch drift before your users do?**  
*Silent degradation is the most expensive bug. Drift detection turns it into an alert, not a postmortem.*

---

## MATURITY RUBRIC

● 5 / 5

### Mature coverage

Traces, evals, cost, latency, drift, replay, and guardrails are all in place. You can debug on evidence.

● 3 or 4

### Partial coverage

You can debug most failures, but one or two classes still rely on guesswork.

● 0 to 2

### Building from scratch

Expect regressions, silent cost leaks, and long debugging loops until the basics land.

## — ROADMAP, NOT A WISHLIST

## You don't have to land all seven at once.

Wherever your scorecard landed, the next step is the same: instrument spans first, attribute per-step cost and latency second, then layer evals and drift. Ordered by dependency, not preference. The first step closes more blind spots than any other, and every later layer needs the one before it to mean anything.

<p><b>01</b> 📅 WEEK 1</p> <h3>Make the graph visible</h3> <p><i>Close the retrieval, tool, and cost blind spots in a week.</i></p> <ul style="list-style-type: none"><li>• Emit OTLP spans around every LLM call and every tool call.</li><li>• Capture prompt, response, tool name, arguments, and status on each span.</li><li>• Ship one dashboard that lists the slowest ten traces this hour.</li></ul>	<p><b>02</b> 📅 MONTH 1</p> <h3>Attribute latency and cost</h3> <p><i>Turn spans into per-step numbers you can budget.</i></p> <ul style="list-style-type: none"><li>• Tag spans with model, tenant, and app for dimensional breakdowns.</li><li>• Track token cost on every LLM span; alert on spikes, not totals.</li><li>• Break latency into p50, p95, and p99 per span, not per request.</li></ul> <hr/> <p>→ BUILDS ON WEEK 1 SPANS</p>	<p><b>03</b> 📅 MONTH 3</p> <h3>Catch quality regressions</h3> <p><i>Make drift the alert, not the postmortem.</i></p> <ul style="list-style-type: none"><li>• Build a small golden dataset, calibrate a judge against human labels, run it on every deploy.</li><li>• Track score distributions over time; flag statistically meaningful drops.</li><li>• Wire replay: pick any failed trace and re-run a step with edits.</li></ul> <hr/> <p>→ BUILDS ON MONTH 1 ATTRIBUTION</p>
--	--	---

**TAKEAWAY.** The order matters more than the timeline. You can't budget cost per step without per-step spans, and you can't baseline drift without a quality signal to baseline.

## — BEHIND THIS EXPLAINER

### Progress Observability Platform unifies traces, evals, cost, latency, drift, and replay in one pane.

Built for teams shipping LLM apps to production, where per-step visibility matters more than per-request status codes.

[telerik.com/ai-observability-platform](https://telerik.com/ai-observability-platform) →

---

## — REFERENCES (14)

# What this piece stands on

Grouped by evidence class, ordered by epistemic weight. Standards and primary research sit above surveys and analyst commentary; vendor and context material frame the category but don't carry the data.

## STANDARDS

---

- [01] **OpenTelemetry**. OpenTelemetry, Semantic Conventions for Generative AI (spans, attributes, events, still Development status). <https://opentelemetry.io/docs/specs/semconv/gen-ai/>
- [02] **OWASP Foundation**, 2025. OWASP Top 10 for Large Language Model Applications 2025 (prompt injection ranked LLM01). <https://genai.owasp.org/llm-top-10/>

## PRIMARY RESEARCH

---

- [03] **Berkeley FCL (Patil et al., arXiv:2407.00121)**, ICML 2025. Berkeley Function-Calling Leaderboard, paper and live leaderboard. <https://gorilla.cs.berkeley.edu/leaderboard.html>
- [04] **Vectara**, 2025, updated. Hallucination Evaluation Model Leaderboard (HEM), factual-consistency rates across frontier models. <https://github.com/vectara/hallucination-leaderboard>
- [05] **Anthropic**, 2025. Measuring AI Agent Autonomy in Practice, real usage data and auto-approve curves. <https://www.anthropic.com/research/measuring-agent-autonomy>

## INDUSTRY SURVEYS

---

- [06] **LangChain**, 2025. State of Agent Engineering 2025 (1,340 responses). <https://www.langchain.com/state-of-agent-engineering>
- [07] **McKinsey**, Nov 2025. The State of AI, Nov 2025 (1,993 companies). <https://www.mckinsey.com/capabilities/quantumblack/our-insights/the-state-of-ai>
- [08] **Langbase**. State of AI Agents, 3,400+ builders in 100+ countries (observability is the #3 cited blocker at 50%). <https://langbase.com/state-of-ai-agents>
- [09] **New Relic**, 2025. 2025 Observability Forecast, 1,700+ practitioners on AI monitoring adoption. <https://newrelic.com/resources/report/observability-forecast/2025>

## ANALYST REPORTS

---

- [10] **Gartner**, Jun 25, 2025. Gartner predicts more than 40% of agentic AI projects will be canceled by end of 2027. <https://www.gartner.com/en/newsroom/press-releases/2025-06-25-gartner-predicts-over-40-percent-of-agentic-ai-projects-will-be-canceled-by-end-of-2027>
- [11] **Forrester**, May 29, 2025. Why AI Agents Fail (And How To Fix Them), practical framework for agentic failure modes. <https://www.forrester.com/report/why-ai-agents-fail-and-how-to-fix-them/RES183446>
- [12] **Menlo Ventures**, Jul 2025. 2025 Mid-Year LLM Market Update (API spend \$3.5B Nov 2024 to \$8.4B mid-2025). <https://menlovc.com/perspective/2025-mid-year-llm-market-update/>

## VENDOR / CATEGORY

---

- [13] **Datadog**, 2025. Datadog expands LLM Observability with capabilities for agentic AI (category legitimacy). <https://www.datadoghq.com/about/latest-news/press-releases/datadog-expands-llm-observability-with-new-capabilities-to-monitor-agentic-ai-accelerate-development-and-improve-model-performance/>