

PLAYBOOK · OCTOBER 2026

The Agent Reliability Playbook

Evaluation, observability and guardrails for AI agents in production. What to measure, what to gate, and how to prove your agent is right, not just running.

Mohcine Madkour, PhD · Founder, EvalTrust
For heads of AI, engineering leaders, and AI governance and risk teams

Live demo dashboard (read-only, no login): dashboard.evaltrust.cloud

Leader's guide: [Your AI Agent Is Working. Is It Right?](#)

Technical deep dive: [Green Is Not Correct: Production Observability for AI Agents](#)

Open-source code: github.com/mohcinemadkour/agent-observability-demo

Running is not the same as right

AI agents rarely fail with an error. They fail with a fluent, confident, wrong answer. Traditional monitoring sees a healthy system; your users see a wrong decision.

≈ 89%

Run success

Runs that finished without an error, timeout or crash.

≈ 63%

Evaluation pass

Runs whose answer was actually correct, grounded and safe.

From the live demo dashboard that accompanies this playbook: a predictive-maintenance agent with a simulated model and deliberately injected failures. The numbers illustrate the pattern, not a benchmark.

The difference, roughly one request in four, is invisible to error-rate monitoring. Closing it takes four controls: **observe** every step, **evaluate** every answer, **guard** against known harms, and **gate** every release on evidence.

Why it matters now

Signal	What the evidence says
Agents are in production	57% of 1,340 practitioners surveyed have agents in production; quality is the #1 barrier (32%). LangChain, State of Agent Engineering, survey Nov–Dec 2025
Quality is rarely checked live	Only 37% run online evaluations; nearly 1 in 4 organizations with agents in production do not evaluate at all, although 89% have observability. LangChain
Projects get cancelled	Gartner predicts over 40% of agentic AI projects will be cancelled by end of 2027, citing costs, unclear value and inadequate risk controls . Gartner, June 2025
Pilots stall	95% of enterprise generative AI pilots stall before scaled adoption. MIT NANDA, The GenAI Divide 2025
Regulation is here	Texas TRAIGA in force since Jan 1, 2026: Attorney General enforcement, penalties up to \$200,000 for uncurable violations, safe harbor for following the NIST AI RMF. Context, not legal advice

Full source links on the last page.

Five ways agents fail without an alarm

Each failure below produces no exception, no HTTP 500 and no page to the on-call engineer. All five run continuously in the live demo, so you can see each one.

Failure	What happens	Business impact	How it is caught
Confidently wrong	The agent reads a critical alarm and answers "looks fine, no action needed".	Wrong decisions taken with full confidence	Groundedness evaluation
Skipped work	The agent stops early: no manual lookup, no service order.	Missed actions: no ticket, no order, no escalation	Required-action check; fewer steps than normal
Loops	A tool says "retry"; the agent obeys until a limit stops it.	Token spend climbs; users wait; nothing resolved	Loop guard and loop counter
Silent dependency failure	A tool errors (HTTP 503) or slows to 4 s; the agent works around it.	Degraded answers that look normal	Per-tool error rate and latency
Data leakage	An email and phone number from the request appear in the answer and the logs.	Privacy and compliance exposure	PII redaction and a PII output check

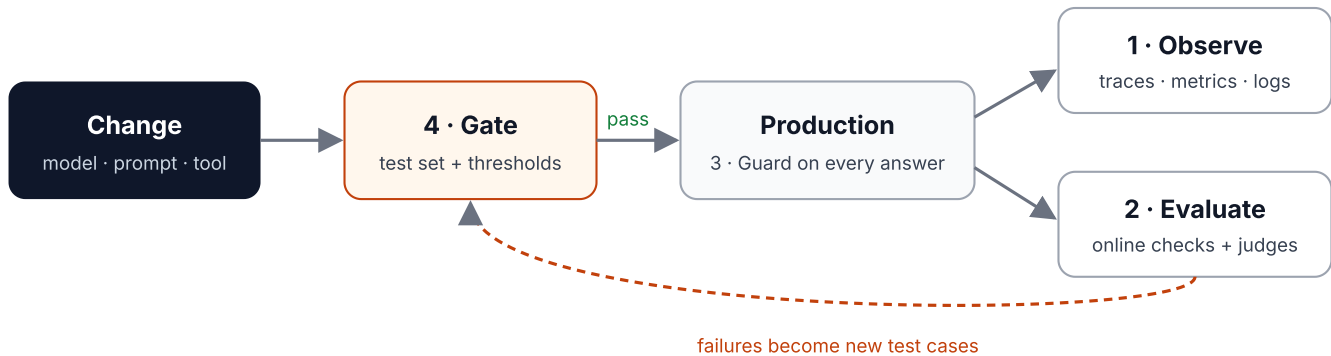
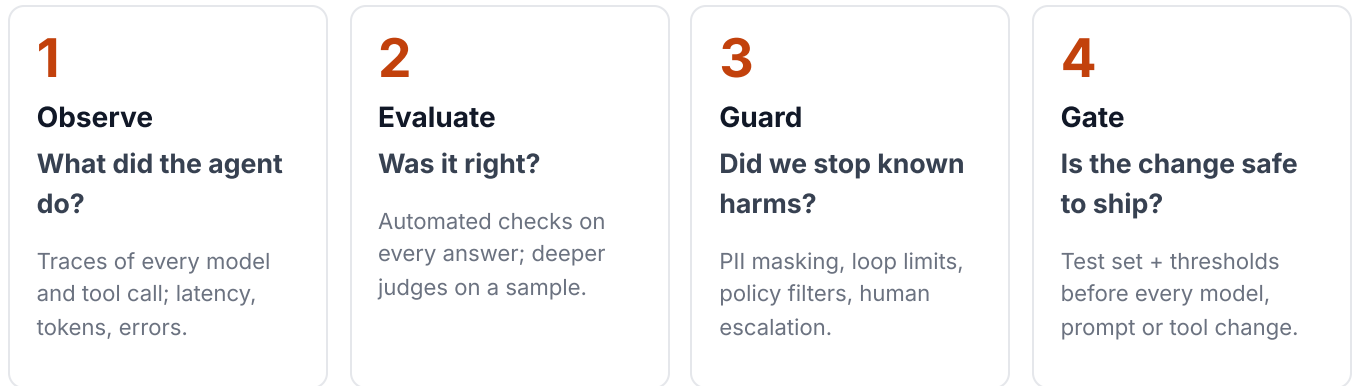
The pattern behind them

Traditional monitoring asks *"did it run?"* Agents need a second question: *"was it right?"* The first is answered by observability. The second only by evaluation, against a definition of "right" that the business writes down.

If your only quality signal is user complaints, you are evaluating in production with your customers as the test suite.

Four controls: observe, evaluate, guard, gate

Each control answers one question. Together they turn "we think it works" into evidence.



Two rules that make the framework work

1. **Define "correct" in writing, with the business.** For a maintenance agent: grounded in sensor data, a service order opened whenever severity is high or critical, no personal data in answers. Engineers cannot invent this alone.
2. **Use the same metrics before and after release.** The checks that gate a release also run on live traffic, so a drop in production is comparable to the bar you shipped against.

What to observe: the signal catalog

Instrument once with OpenTelemetry, the open standard every major backend accepts. These are the signals the demo agent emits; they are a sound default for any agent.

Group	Signals	Question answered
Agent runs	Runs by outcome (success, tool error, loop, model error); end-to-end latency; reasoning steps per run	Is it up, fast, and decisive?
Tools	Calls, error rate and latency per tool	Which dependency is failing or slow?
Model calls	Input and output tokens, latency p50/p95, by model and provider	What does it cost; is the model the bottleneck?
Evaluation	Pass rate, score, failures per check	Was it right, and why not?
Safety	Loop-guard triggers, PII detections, guardrail activations	Did we prevent known harms?
Traces	One trace per run: every model call, tool call and evaluation as linked steps	What exactly happened in this run?
Logs	Application logs carrying the trace ID	What did the code report at that moment?

Three design choices that pay off

- **A collector between the agent and the tools.** The agent knows one address. Backends (Grafana, Splunk, Datadog...) are swapped at the collector, never in the agent.
- **Redact centrally.** PII is removed in the collector before data reaches any backend, for every service at once.
- **Metrics from traces.** The collector derives rate, error and latency for every step automatically, so new tools and agents are covered without new code.

Observability shows what happened. It cannot tell you whether the answer was right; a trace of a wrong answer is entirely green.

What to evaluate: from rules to judges

Start with cheap, deterministic checks on every request. Add model-based judges and human review on a sample. Organize by layer so a failing score points to the cause.

Starter checks (run on 100% of traffic)

Check	Passes when	Catches
task_completed	The user's goal was achieved (not merely "no error")	Crashes; polite non-answers
grounded_in_evidence	The answer reflects the data retrieved; no contradicting or dismissive claims	Confidently wrong answers
required_action_taken	Mandatory actions happened (ticket, order, escalation)	Skipped work
no_pii_in_output	No personal identifiers in the answer	Data leakage
tool_call_budget	Steps and tool calls within budget	Loops and runaway cost

Evaluation by layer (as systems grow)

Layer	Metrics	Method
Retrieval	Context precision and recall, NDCG@k, MRR@k	Labeled queries
Generation	Faithfulness, answer relevancy, similarity to reference	LLM-as-judge at temperature 0
Tool use	Tool selection accuracy, argument correctness	Expected tool calls
Trajectory	Task completion, step efficiency, recovery after failure	Judges over the full trace
Multi-agent	Routing accuracy, handoff success, redundant work	Per-agent traces + labels
Safety	PII leakage, prompt-injection resistance, policy compliance	Rules, classifiers, red-team sets
Business	Cost per resolved task, escalation rate, acceptance rate	Joined with business systems

Attach every score to the trace it judges. An aggregate tells you quality dropped; the trace tells you which agent, retrieval and prompt caused it.

Prevent known harms. Ship only on evidence.

Guardrails: act before the user sees the answer

Guardrail	What it does	Signal to watch
PII masking	Removes personal identifiers from answers (and, separately, from telemetry)	PII detections per day
Loop and step limits	Stops repeated identical calls; caps reasoning steps	Loop-guard triggers
Policy filters	Blocks out-of-scope, unsafe or non-compliant responses	Blocks by policy
Grounding enforcement	Refuses or escalates when the answer is not supported by retrieved data	Ungrounded-answer rate
Human escalation	Routes high-risk or low-confidence cases to a person	Escalation rate and wait time
Prompt-injection defense	Treats tool output and documents as data, never instructions	Injection attempts detected

Release gates: an example

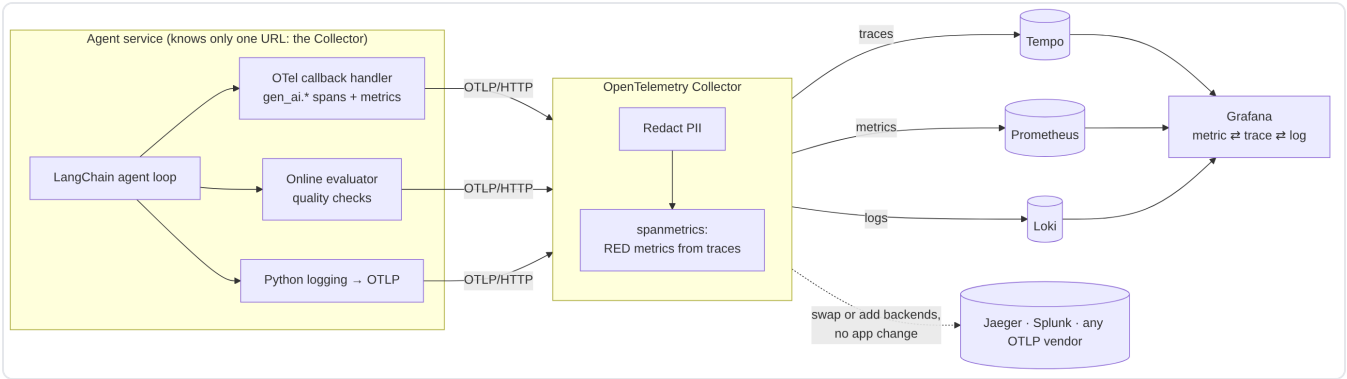
Every model, prompt or tool change runs against a fixed test set. Any blocking gate failing stops the release.

Gate	Threshold	Type
Faithfulness	≥ 0.70	Blocking
Answer relevancy	≥ 0.70	Blocking
Context precision (retrieval)	≥ 0.60	Blocking
Task completion	≥ 0.75	Blocking
PII leaks on test set	0	Blocking
p95 latency	$\leq$ agreed SLA	Advisory
Cost per task vs. current version	$\leq +10\%$	Advisory

Thresholds are examples from a CNC predictive-maintenance RAG system; set yours with the business owner and tighten them as the system matures.

An open, vendor-neutral stack

Everything in the demo is open source and runs in Docker on one small server. The same design scales to many agents and services.



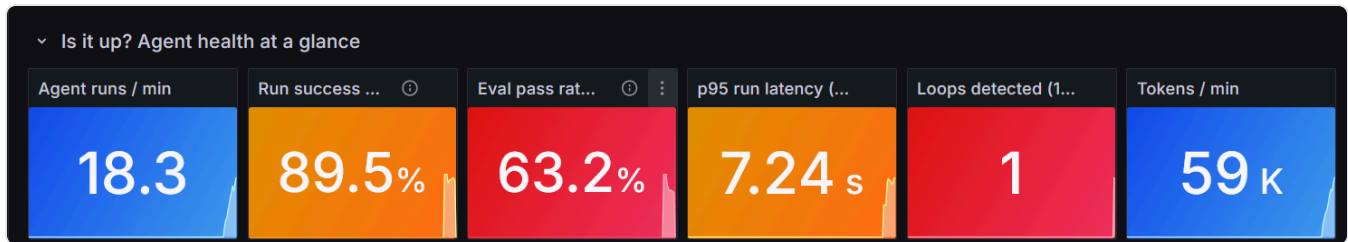
The agent sends traces, metrics and logs over OTLP to the OpenTelemetry Collector, which redacts PII, derives metrics from traces, and routes each signal to its backend. Grafana links metrics, traces and logs in one view.

Component	Role	Why this choice
OpenTelemetry SDK	Produces traces, metrics and logs in the agent	Industry standard; every backend accepts it
OpenTelemetry Collector	Receives, redacts, derives, routes	Keeps governance and vendor choice out of the app
Prometheus · Tempo · Loki	Store metrics, traces, logs	Open source, cost-efficient, widely used
Grafana	Dashboards and drill-down	One view, click from metric to trace to log
Evaluator	Scores answers; writes results onto the trace	Quality next to latency and cost

Swapping Tempo for Splunk, Datadog or Honeycomb is a collector configuration change; the agent is untouched.

Tour the live dashboard

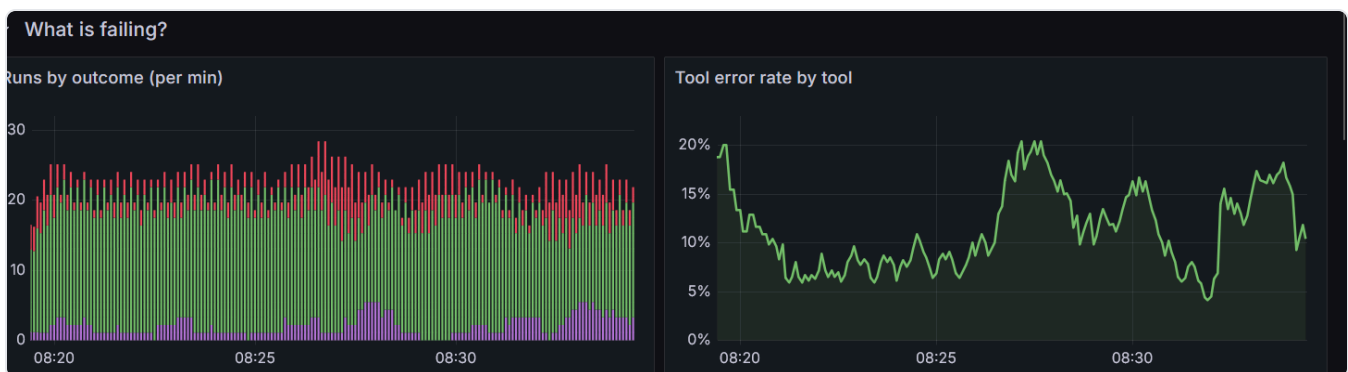
Open dashboard.evaltrust.cloud. No login needed. Read it top to bottom; each row answers one question.



Is it up? Runs per minute, run success, evaluation pass rate, p95 latency, loops, tokens. Read across: it is up and mostly error-free, but only about 63% of answers are right.

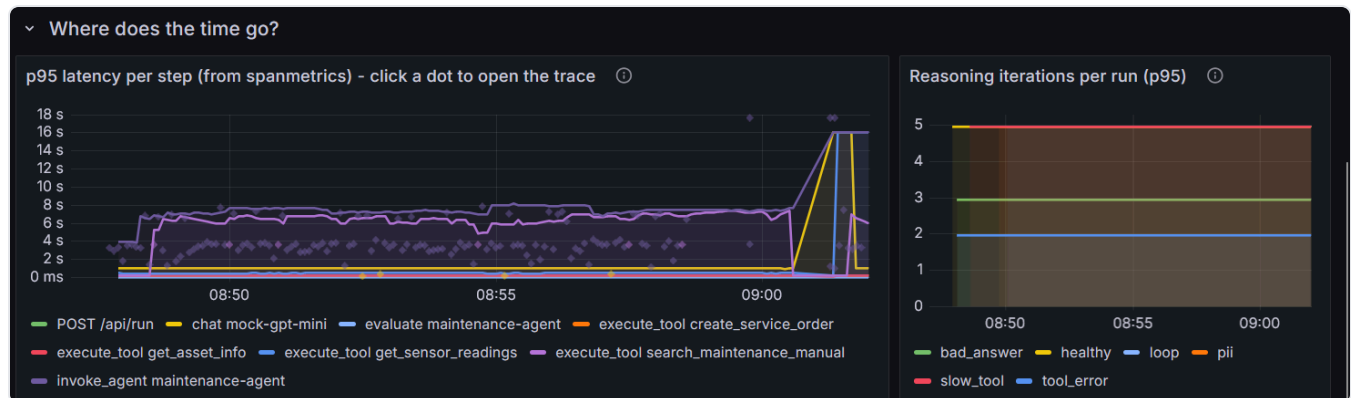


Green is not the same as correct. The shaded band between the blue (no errors) and orange (correct) lines is the blind spot. The bars show why: crashes (**task_completed**) are caught by monitoring; ungrounded answers, skipped actions and PII are not.

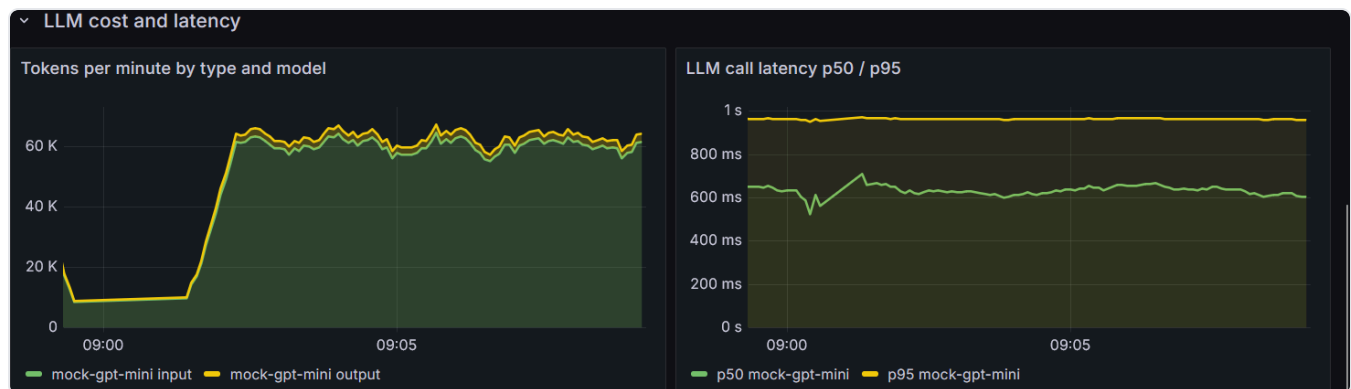


What is failing? Runs by outcome (red: tool errors, purple: loops) and error rate per tool. Loops never show at the tool level: the tool succeeds every time; the agent misbehaves.

Where time and money go



Where does the time go? p95 latency per step: one slow line (the manual search) while model calls stay fast. Dots are real runs; click one to open its trace. Right: the wrong answer takes 3 steps instead of 5, because the agent skipped work.



LLM cost and latency. Over 90% of tokens are input: an agent re-sends its growing history at every step. Cheapest savings: fewer steps and shorter context. Model latency is steady at about 600 ms.

Three things to try

1. In *Failed or wrong runs* at the bottom, open a **bad_answer** trace. Every step is green; only the **evaluate** step shows the failure.
2. On the latency graph, click a high dot to land on a slow run, then open its logs from the trace.
3. Open a **pii** trace and find **<redacted:email>**: removed by the collector before storage.

Screens from the demo; values change continuously. The demo uses a simulated model so it runs offline and reproducibly.

From one agent to many

The pipeline does not change when you add agents. What changes is what you instrument and what you evaluate.

Architecture	What to add
Orchestrator and workers	One agent span per worker under the orchestrator; routing decision as an attribute; routing accuracy
Graph workflows (e.g. LangGraph)	A span per node with the edge taken; cycle counts; per-node latency and cost
Handoffs between peers	Handoff events (from, to); handoff depth; "ping-pong" count, the multi-agent version of a loop
Agents as separate services	Trace context passed across every HTTP call and queue message, or each service produces a disconnected trace
Retrieval (RAG)	A span per retrieval: collection, top-k, similarity scores; empty-retrieval rate

Additional metric groups

Coordination

Routing per target agent, handoffs per run, per-agent share of latency and cost.

Cost and capacity

Cost per agent, per task type and per resolved task; rate-limit errors, retries, model fallbacks.

Guardrails and humans

Guardrail triggers by type; escalation rate; time waiting for human approval.

Drift and feedback

Evaluation scores per model version over time; user feedback and edits attached to traces.

Readiness checklist

Tick each statement that is true today.

- ☐ We can open any single agent run and see every model and tool call it made.
- ☐ We know the agent's error rate **and** its correct-answer rate, separately.
- ☐ "Correct" is defined in writing for our use case, with measurable checks.
- ☐ Evaluations run automatically on live traffic, not only before launch.
- ☐ There is a loop or step limit, and we count how often it triggers.
- ☐ Personal data is masked before it reaches logs, traces or answers.
- ☐ Every model or prompt change passes a test set with thresholds before release.
- ☐ We track cost per resolved task, not only total spend.
- ☐ A non-engineer can see the quality trend on a dashboard.
- ☐ We could hand an auditor or client evidence of how the agent behaved last month.

Score	What it means
7 – 10	Ahead of most teams. Focus on depth: judges, multi-agent signals, drift.
4 – 6	Foundations exist. Close the gaps before scaling to more users or agents.
0 – 3	The agent runs on trust, not evidence. Start with observability and five starter checks.

A 30-day path to evidence

Week	Focus	Outcome
1	Instrument the agent with OpenTelemetry; stand up the collector and dashboard	Every run visible: steps, latency, tokens, errors
2	Define "correct" with the business; build 5–10 automated checks	Evaluation pass rate next to success rate
3	Add guardrails (PII masking, loop limits, escalation) and alerts	Known failure modes prevented or flagged
4	Build a test set and release gates; set a monthly quality review	No change ships without evidence; a trend leaders can read

Questions leadership should be able to answer at any time

Question	Signal that answers it
Is the agent available and fast enough?	Runs per minute, success rate, p95 latency
Are its answers right?	Evaluation pass rate, next to success rate
Why did quality drop?	Failed checks by type, linked to the exact runs
What does it cost, and is that rising?	Tokens and cost per run, per agent, per resolved task
Is it safe?	Guardrail triggers, PII detections, policy violations
Did the last change help or hurt?	Gate results and evaluation trend per version
Can we prove it to an auditor or client?	Retained traces, evaluation records, change history

See it, then let's talk

EvalTrust helps teams in healthcare, finance, pharma and industry evaluate, monitor and govern AI agents and multi-agent systems on their way to production.

Free 30-minute Agent Readiness Review

Tell us what your agent does and how it is monitored today. We will tell you which gates, signals and guardrails you are missing, and what to fix first.

Email mohcine@evaltrust.cloud with the subject "Agent Readiness Review".

Resources

- **Live dashboard** (read-only): dashboard.evaltrust.cloud
- **For leaders:** [Your AI Agent Is Working. Is It Right?](#)
- **For engineers:** [Green Is Not Correct: Production Observability for AI Agents with OpenTelemetry](#)
- **Code:** github.com/mohcinemadkour/agent-observability-demo
- **Evaluation deep dive:** [Evaluating Agentic AI Systems: A Practitioner's Guide](#)

Sources

- LangChain, [State of Agent Engineering](#) (1,340 respondents, Nov 18 – Dec 2, 2025).
- Gartner, [Gartner Predicts Over 40% of Agentic AI Projects Will Be Canceled by End of 2027](#) (June 25, 2025).
- Computing, [MIT report: 95% of corporate generative AI pilots fail to deliver returns](#), on MIT NANDA, The GenAI Divide: State of AI in Business 2025.
- McDonald Hopkins, [Texas enacts broad artificial intelligence regulations](#); Spencer Fane, [Texas Enacts Comprehensive AI Governance Law](#). Regulatory content is context, not legal advice.

© 2026 EvalTrust · evaltrust.cloud · Demo figures come from a simulated model with injected failures and illustrate patterns, not benchmarks.