

ON-CALL · AI AGENTS · OBSERVABILITY

From Alert to Root Cause Building — and observing — an AI On-Call Agent

How to make an agent that investigates incidents for you, and how to trace, evaluate and improve it so on-call engineers can actually trust it.

Neelakshi Soni

Incident Management

THE INCIDENT: HUMAN RESPONSE

CRITICAL ALERT · ERROR SPIKE

- Timeouts increasing
- Logs fragmented across systems
- Multiple dashboards and runbooks

- Consumer lag / service degradation
- Logs fragmented; multiple runbooks
- Engineers spend time gathering context

→ TRANSFORM →

THE SOLUTION: AI ON-CALL AGENT

INVESTIGATION SUMMARY

- Pulls relevant incident history
- Checks metrics, logs and changes
- Collects evidence for a human review

- Evidence collected in one place
- Investigation steps are visible
- Findings support — not replace — on-call judgment

What the investigation needs

01

Memory

Past incidents and postmortems — plus learning captured from the current incident.

02

Tools

Metrics, logs, deployments, diffs, code, and database health.

03

Playbooks

Standard operating procedures for known issue types.

alert fires



classify



recall



investigate



notify on-call

Recall prior context → gather current evidence → record what the incident teaches us

Evidence gathered step by step

Intermediate findings stay visible as the investigation progresses.

- | | | |
|---|--|--|
|  | Classify alert | Identify the symptom and affected service |
|  | Recall incident history | Find similar incidents and prior fixes |
|  | Inspect metrics and logs | Compare error rates, traffic, and log patterns |
|  | Check service and database health | Look for dependency failures or saturation |
|  | Review deployments and diffs | Check changes in the relevant time window |
|  | Summarize evidence | Separate observed facts from hypotheses |
|  | Notify on-call | Provide findings and next steps for human review |

INTERMEDIATE FINDINGS

- Metric change observed
- Relevant log lines collected
- Recent change identified

FINAL HANDOFF

Evidence → possible explanations → suggested next checks

Human on-call engineer reviews and decides.

Specialist investigations, visible results

Each investigation contributes evidence before the consolidated handoff



Parallel work makes progress visible; the on-call engineer remains responsible for the decision.

Questions you can't answer yet



Why did the investigation take 6 minutes?

Which tool call or reasoning step was the bottleneck?



Why did it burn 400k tokens?

Which step stuffed raw logs straight into the prompt?



Why was the root cause wrong?

Bad retrieval? Wrong playbook? A flaky tool? Model reasoning?



Is it getting better or worse?

Did last week's prompt change help or hurt accuracy?

Agents are non-deterministic and multi-step, so **logs alone don't tell the story.**

Four signals, one timeline

Classic observability still applies — agents just add a fourth pillar on top.



Metrics

Error rate, latency, cost/run, tokens/run — the numeric pulse of the system.



Logs

Structured events with context: what the agent saw, decided, and said.



Traces

The end-to-end path of one investigation — every tool call and LLM step, timed.



Evals

Was the output actually correct? Scored by rubric, LLM judge, or human review.

The glue: a **trace ID** that threads through all four, so you can pivot from a metric spike to the exact tool call that caused it.

WALKTHROUGH

An on-call agent, mid-investigation

Scenario: payments API latency spike — root cause is a database retry storm.

● **PAGERDUTY** — **payments-api p99 latency > 2.0s** · **prod** · **sev-2**

🤖 Investigating — pulling the last 15 minutes of metrics and recent deploys for payments-api.

🔧 `get_service_metrics(service='payments-api', window='15m')`
→ `db_pool.active=100/100` `db_pool.wait_ms=1,840` `retry_rate=38%`

🔧 `get_recent_deploys(service='payments-api')`
→ `deploy a1b2c3 (12m ago): db.timeout_ms reduced 5000 → 1000`

🎯 Root cause: the new 1s DB timeout is shorter than p95 query time (1.6s), so calls time out, retry, and exhaust the connection pool. Recommend: revert timeout to 5s or raise pool size.
Confidence: High — attached trace + 2 supporting signals.

Every step, timed and priced

TRACE · db-retry-investigation-77a1

- **agent-investigation** 6.20s\$0.0091
 - get_service_metrics 0.41s
 - get_recent_deploys 0.33s
 - llm:root_cause_synthesis 3.12s\$0.0067
 - post_to_incident_channel 0.18s

SPAN · llm:root_cause_synthesis

model gpt-4.1-mini

latency 2.12s

tokens 1,850 → 210 (Σ 2,060)

cost \$0.0067

prompt oncall-agent-v3

output.root_cause

"db.timeout_ms (1000) < p95 query
time (1600ms) → retry storm →
pool exhaustion. Confidence: 0.93"

Grading the investigation, not just the system

An LLM judge compares the agent's root cause to the verified postmortem cause.

NEW EVALUATOR

Root-cause correctness judge

LLM-as-judge

claude-haiku-4.5

You compare the agent's stated root cause to the verified postmortem cause.

Return CORRECT only if the mechanism matches (not just the affected service). Otherwise return INCORRECT, citing what the agent missed.

SAMPLE VERDICTS

✓ **CORRECT** · confidence 0.93

Agent: "DB timeout shorter than p95 query time → retry storm → pool exhaustion."
Matches postmortem exactly.

✗ **INCORRECT** · confidence 0.81

Agent: "Upstream network blip."
Postmortem: DB pool exhaustion from a timeout misconfiguration.
Missed: deploy correlation step.

From trace events to an actionable dashboard

One trace ID connects the agent run, tool spans, telemetry ingestion, storage, evaluation, and UI.



CORRELATED SIGNALS



KEY TAKEAWAY

Trace what happened → ingest it reliably → connect logs, metrics and evals → visualize the run → measure whether the diagnosis was right.

KEY TAKEAWAYS

Make the agent explain itself

- 1 Trace every step**

Log each tool call, retrieval, and reasoning hop — not just the final answer.
- 2 Thread one ID through everything**

Metrics, logs, traces, and evals should all pivot on the same trace/session ID.
- 3 Score correctness, not just completion**

An agent can finish fast and still be wrong — grade the output against ground truth.
- 4 Auto-attach findings to the alert**

The investigation should be waiting for the on-call engineer, not started by them.
- 5 Watch accuracy over releases**

Treat the agent like a model in production: track drift, not just uptime.