OBSERVABILITY SUMMIT 2026

TRACE EVERY DECISION WITH AGENTIC AI OBSERVABILITY

Why Are Your AI's Decisions Hard to Explain?

Dhiraj Kumar Jain
Vikash Agrawal

Amazon CloudWatch (AWS)

aws

Your agent made a decision.
Can you explain why?

The Explainability Gap

You Can See What — But Not Why

- You know the agent called a tool , but not why it chose that tool
- Reasoning is inside the model , invisible unless you capture it
- Non-deterministic: Same input, different path every time
- Traces, metrics, logs , none answer 'Why did it choose X over Y?'

The Cost of Unexplainable Decisions

- Hallucinations: Confidently wrong — metrics say everything is fine
- Cost explosions: Confused agent burns token budget in seconds
- Trust erosion: Teams stop deploying agents they can't explain
- Compliance: 'Why did the AI do that?' — you need an answer

Trace the Reasoning, Not Just the Action

THE GOAL: EXPLAIN ANY AI DECISION IN UNDER 60 SECONDS

- Capture WHY the agent chose this action — not just what it did
- Record what information it had at the moment of decision
- Link each decision to its inputs, outputs, and downstream effects
- Make every agent choice auditable after the fact

Four Signals That Explain Every Decision

1. Reasoning Chain → 'What was it thinking?'
2. Tool Interactions → 'What evidence did it use?'
3. Token Spend → Healthy decisions are cheap. Confused ones are expensive.
4. Confidence + Outcome → 'How certain was it? Did it work?'

OpenTelemetry for Decision Tracing

OTel Maps Perfectly to Decision Tracing

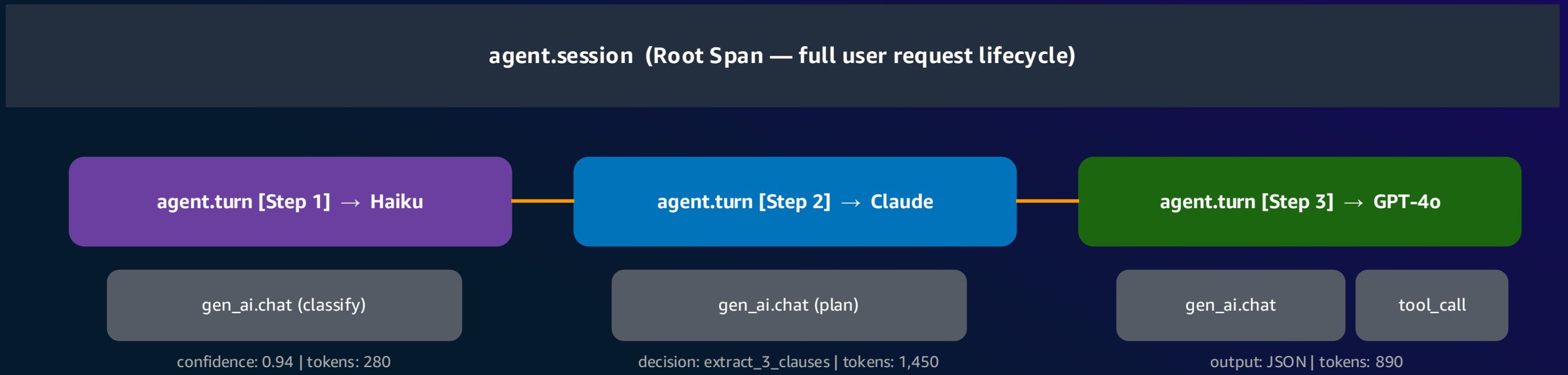
- Spans = Decisions (start, end, context, outcome)
- Events = Reasoning (full thought process, attached to the span)
- Links = Causality (this retry happened because THAT decision failed)
- Attributes = Queryability (filter by model, cost, type, confidence)

One Decision = One Span

THE TRACE HIERARCHY FOR AGENT DECISIONS

- agent.session (root) → agent.turn (the decision) →
  - gen_ai.chat (the reasoning) + agent.tool_call (the action)
- Span events: Full reasoning text, alternatives considered
- Span links: Connect retries to their original failed decision

Decision Trace: Span Hierarchy



Span Events (attached to each turn)

- Reasoning text: Full chain of thought
- Tool I/O: Request and response payloads
- Alternatives: Options considered and rejected

Span Attributes (queryable)

- gen_ai.system = 'anthropic' | 'openai'
- gen_ai.usage.input_tokens = 1450
- agent.decision.type = 'plan'
- agent.confidence = 0.94

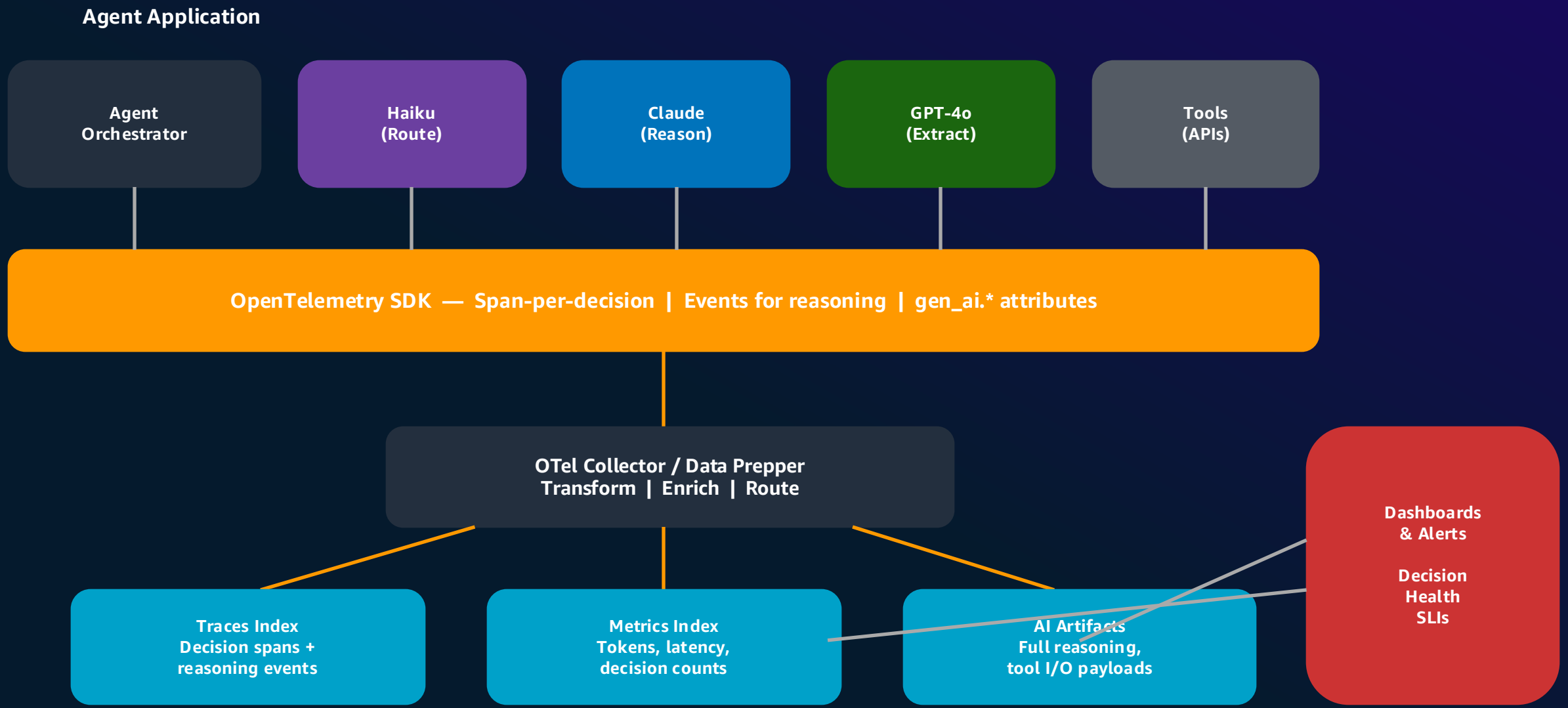
aws

Make It Queryable, Start Simple

ATTRIBUTES THAT MATTER + THE TWO PATTERNS THAT GIVE YOU 80%

- Tag every span: gen_ai.system, model, tokens, decision.type, confidence
- Pattern 1: One span per decision (the foundation — detects loops)
- Pattern 2: Reasoning text as span event (the debugger — explains why)
- That's it. Start here. Add metrics and context propagation later.

Instrumentation Pipeline



When Decisions Go Wrong

Two Stories from Production

- Hallucination: Agent sent refund confirmation — no refund issued.
  - → Traced backwards: Turn 2 read wrong customer record. Found in 3 min.
- Cost explosion: 50K tokens burned in 90 seconds.
  - → Trace shows same decision repeated 12x. Circuit breaker stops it at 3.

The Decision Dashboard

KNOW IN 3 SECONDS IF DECISIONS ARE HEALTHY

- Decision success rate by type — your top-level SLI
- Decisions per task (distribution) — long tail = confused agents
- Token cost per decision type — find expensive patterns
- One click from any anomaly → the decision trace that explains it

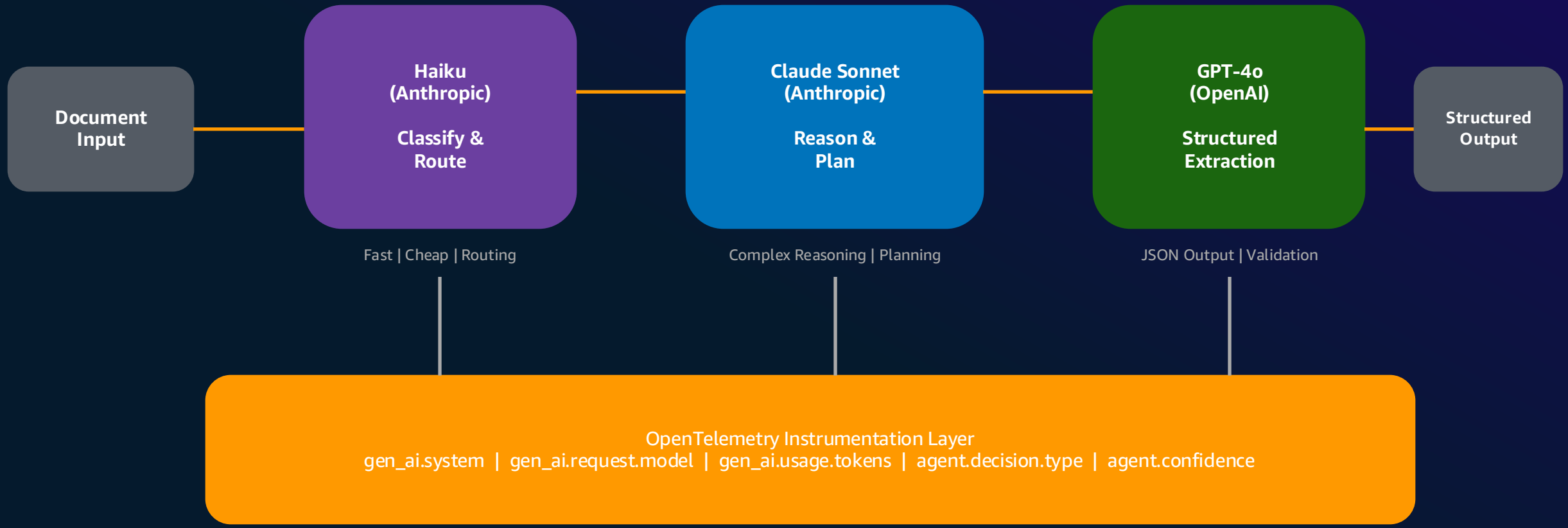
Live Use Case:
Three Models, One Trace

The System: Multi-Model Document Agent

HAIKU ROUTES. CLAUDE REASONS. GPT-4O EXTRACTS.

- Enterprise document processing — contracts, invoices, queries
- Haiku (Anthropic): Fast classification and routing
- Claude Sonnet (Anthropic): Complex reasoning and planning
- GPT-4o (OpenAI): Structured data extraction (JSON output)

Multi-Model Agent Architecture



Tuesday 2 PM: Accuracy Drops from 95% to 65%

THREE MODELS. WHICH ONE BROKE?

- GPT-4o team: 'Our model hasn't changed'
- Claude team: 'Our reasoning looks correct'
- Without tracing: Finger-pointing for days
- With decision tracing: Root cause in 15 minutes

Following the Trace to Root Cause

- Span 3 (GPT-4o): Bad output — but trace upstream...
- Span 2 (Claude): Plan was correct given its input — trace further...
- Span 1 (Haiku): Classified 'invoice' as 'contract' — confidence: 0.52!
- Root cause: Stale prompt after cache refresh. The failing model wasn't the culprit.

Three Things to Remember

1. Decisions are invisible by default — you must trace them explicitly
2. One span per decision + reasoning as event = 80% of the value
3. In multi-model systems, always trace upstream — the culprit hides there

One span per decision.
Start Monday.

Thank you!