SAMIR GENAI ACADEMY

PROFESSIONAL STUDY GUIDE SERIES

Phase 04 · Advanced Agentic Patterns

20

LLMOps & Agent Monitoring

Observability, tracing, cost control & production reliability

Module 20 of 25 · **1 hour**

Instructor: **Samir Kumar** · Edition 2026

STUDY GUIDE

01 Overview

> Observability, tracing, cost control & production reliability. This guide gives you the core theory, the tools that matter, a hands-on build, working code, and the pitfalls to avoid — everything you need to learn the topic to a professional standard and apply it in real projects.

1 hour	4 + 4	4	20/25
LESSON LENGTH	CONCEPTS + BUILD STEPS	FRAMEWORKS & TOOLS	PHASE 04

02 Learning Objectives

- Instrument agents with tracing and structured logs.
- Track cost, latency, and quality in production.
- Set alerts and budgets to prevent runaway spend.
- Build an observability stack for agents.

03 Core Concepts

Observability for agents

Agents are non-deterministic and multi-step, so you need traces: a tree of every LLM call, tool call, and decision per request. Tracing turns 'it sometimes fails' into a concrete, inspectable trajectory you can debug.

The three pillars

Cost (tokens x price per call, aggregated), latency (per step and end-to-end), and quality (evals, user feedback, error rates). Instrument all three from day one; you can't improve what you don't measure.

Standards & tooling

OpenTelemetry provides vendor-neutral tracing; platforms like LangSmith, Langfuse, and Arize Phoenix add LLM-aware dashboards, eval integration, and prompt versioning on top.

Reliability controls

Add timeouts, retries with backoff, fallbacks to cheaper models, rate limiting, and hard spend budgets with alerts. These keep a viral spike or a looping agent from causing an outage or a surprise bill.

04 Visual Diagram

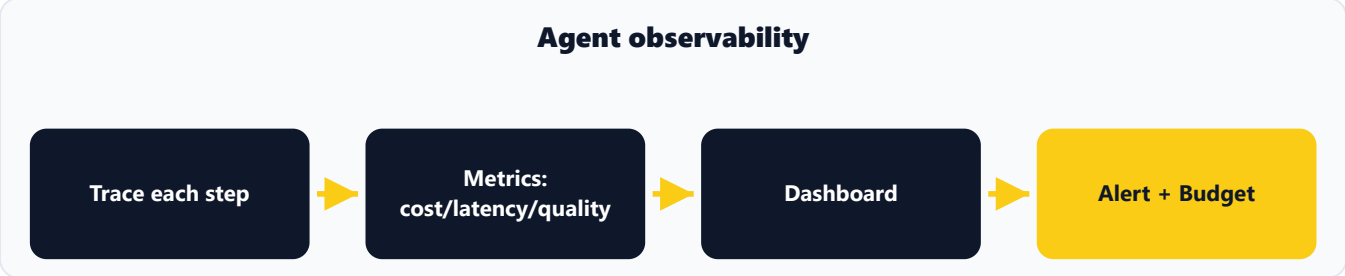

Figure 20 — Trace every LLM/tool call, emit the three pillars of metrics, and alert on budgets.

05 Formula Spotlight

Request cost

$$\text{cost} = \Sigma\ (\ \text{tokens}_{\text{step}} \times \text{price}_{\text{model}}\ )$$

$\text{tokens}_{\text{step}}$	input + output tokens for each LLM call in the trajectory
$\text{price}_{\text{model}}$	the per-token price of the model used for that call
Σ	summed over every step in the agent's run

An agent makes many calls per task, so cost is the sum across the whole trajectory — not one call. Tracking it per step shows exactly where spend concentrates and which step to optimise first.

06 Key Frameworks & Tools

OpenTelemetry · Langfuse / LangSmith · Arize Phoenix · Grafana

07 Hands-On Build

🛠 OpenTelemetry Monitoring Stack for Agents

1. Add OpenTelemetry tracing around LLM and tool calls.
2. Emit cost and latency metrics per step.
3. Pipe traces to Langfuse/Phoenix and build a dashboard.
4. Set a spend budget and alert thresholds.

08 Code Walkthrough

Tracing an agent step with OpenTelemetry — PYTHON

```python
from opentelemetry import trace

tracer = trace.get_tracer("agent")

with tracer.start_as_current_span("llm_call") as span:
    span.set_attribute("model", "claude-opus-4-8")
    resp = call_model(prompt)
    span.set_attribute("input_tokens", resp.usage.input_tokens)
    span.set_attribute("output_tokens", resp.usage.output_tokens)
```

09 🧪 Lab Assistant

Lab 20 · Trace an agent run

Prerequisites:pip install opentelemetry-sdk · A tracing backend (Langfuse/Phoenix)

1. **Wrap LLM/tool calls in spans with token attributes**
   > Expected: A trace tree per request
2. **Pipe traces to a dashboard**
   > Expected: Cost and latency visible per step
3. **Set a spend budget + alert**
   > Expected: Runaway loops trigger an alert

Troubleshooting

- No traces → confirm the exporter is configured.
- Surprise bill → you skipped the budget guard.

10 Pitfalls & Key Takeaways

⚠ COMMON PITFALLS

- Shipping agents with no tracing and debugging from logs alone.
- No spend budget — a looping agent runs up a huge bill.
- Measuring latency end-to-end only, missing the slow step.

✓ KEY TAKEAWAYS

- Tracing makes non-deterministic agents debuggable.
- Instrument cost, latency, and quality from day one.
- Budgets, retries, and fallbacks deliver production reliability.

📺 **Companion video:** Watch the module 20 lesson in your Student Dashboard, then use this guide to revise, copy the code, and complete the build. Download both for offline study.

11 Further Resources

OpenTelemetry docs • Langfuse / LangSmith docs • Arize Phoenix