arize

Arize AI - Agent Mastery Course

arize

Module 2: Agent Engineering

Agent Engineering

- Agent engineering involves designing your system to accomplish a task
- With agents, your job as an engineer is to think through:
  - What is the ideal user experience for this product?
  - How does an agent help here?
  - What form should this agent take:
    - What context does it need?
    - What tools does it need?
    - Should this agent be in the foreground or background?

What is Agent Observability?

- Observability is the ability to understand what is happening inside an agent system from the outside.
- It turns opaque “black box” reasoning into something visible, interpretable, and diagnosable.

Why it matters:

- Debugging agent behavior
- Identifying bottlenecks such as slow tool calls or repeated loops
- Tracking hallucinations and errors
- Providing transparency for stakeholders

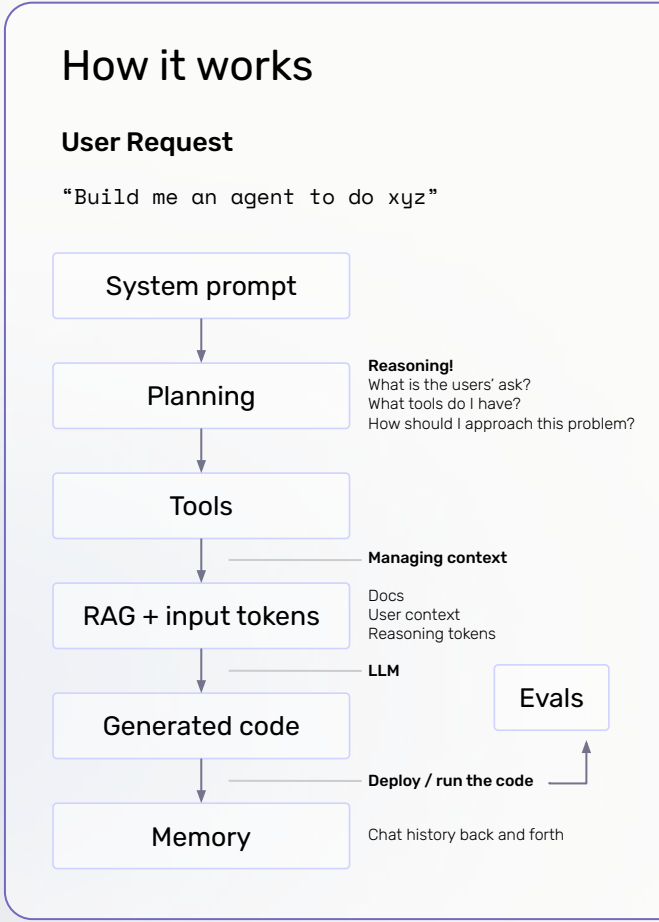
Observability—Why it Matters

Allows you to change your system and verify those changes took effect.

For example:

- Changing architecture (serial to parallel agent)
- Changing your prompt
- Changing your model
- Context engineering

Coupled with evals, you can verify you are making your system better



Traces, Spans, and the Three Pillars

Observability relies on three categories of signals:

Traces and spans

End-to-end request lifecycles

Represented by attribute JSONs

Metrics

Numerical measurements like latency and success rate

Logs

Detailed event-level records

The Trace: The complete delivery route

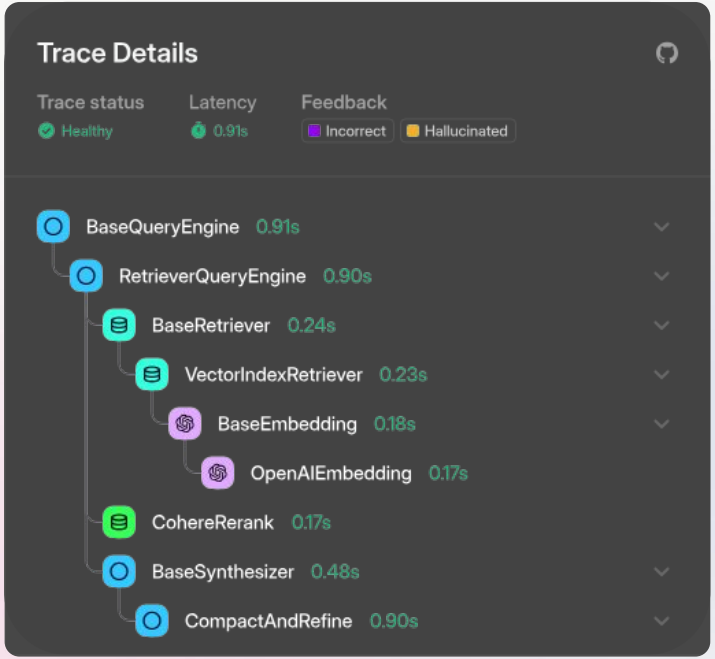
For agents, traces are most important, since they reveal reasoning and decision-making.

Each trace is composed of spans, which represent individual steps.

- Prompt spans: inputs and outputs of LLM calls
- Tool spans: external API or function calls
- Decision spans: reasoning or planning steps

Example Trace Tree

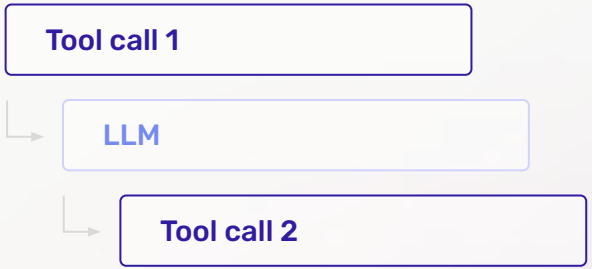
A trace can be visualized as a tree of spans, showing how an agent combined reasoning and tools to produce an answer.



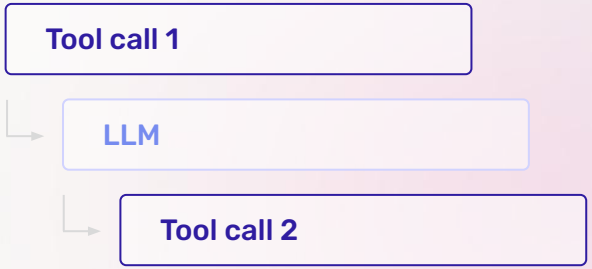
Example Trace Tree

Session 1

Trace 1



Trace 2



Trace Data

```
{"trace":"ID1"
      {"span":"id1", "input": "...",
      "output":"...", "tool":"toolX"}
      {"span":"id2", "input":"..."},
      "output":"..."}
      {"span":"id3", "input": "...",
      "output":"...", "tool":"toolY"}
```

You are an Evaluator of agent decisions.
Please determine if the agent is making the right tool call choices, choosing a path to solve the problem.

{trace_data}

Standards: OTel, OI, and Auto-instrumentors

- **OpenTelemetry (OTel)**: standard for collecting traces, metrics, and logs.
  - Provides a common language across systems.
  - Ensures consistency when agents call APIs, databases, or services.
- **OpenInference (OI)**: extends OTel for LLM and ML systems.
  - Adds span types like prompts, tools, and evaluations.
  - Brings richer semantics for agent reasoning.
- **Auto-instrumentors**: pre-built connectors under OI.
  - Automatically capture spans for frameworks like LangChain, AutoGen, and LlamaIndex.
  - Enable visibility with minimal setup while allowing customization when needed.

Lab 2: Set up observability for our Agent

- Navigate to labs/lab2.md