LangSmith Tracing & Core Concepts

What Does LangSmith Trace?

LangSmith acts as a complete observability layer for LLM applications. It automatically logs detailed execution data from every run, enabling developers to analyze behavior, detect failures, and optimize performance.

Traced Components

LangSmith captures the following key metrics and data points:

1. **Input and Output:** Every input (prompt) and output (model response) is logged, allowing developers to see the full conversation context. *Example:* If a user asks, "Summarize this article," the input prompt and generated summary are both stored.
2. **All Intermediate Steps:** Tracks how a query passes through chains, tools, and agents. *Example:* For an agent using a search tool, LangSmith records both the search query and the tool's result.
3. **Latency:** Measures how long each component takes to execute, helping identify slow or inefficient steps.
4. **Token Usage:** Records token consumption for each LLM call to monitor efficiency and manage API costs.
5. **Cost:** Tracks cost per run (when supported by model provider APIs), making it easier to estimate and control expenses.
6. **Error Tracking:** Logs any failures, exceptions, or unexpected model responses with full traceback information for debugging.
7. **Tags:** Each run can be labeled with tags (e.g., `production`, `test`, `evaluation`) to categorize experiments.
8. **Metadata:** Stores additional contextual data, such as user ID, timestamp, or session details, for deeper analysis.
9. **Feedback:** Collects human or automated feedback on outputs to measure quality and guide future improvements.

Example: Tracing an Agent Run

Example Scenario: An AI customer support agent answers a question using LangChain + LangSmith.

- Input: "What is my order status?"
- Steps:
  - Tool 1: `DatabaseQueryTool` → fetch order details
  - Tool 2: `ResponseFormatter` → format answer
- Output: "Your order #1234 has been shipped."

LangSmith records each tool call, timing, tokens, and final output, so the developer can later inspect performance or errors.

Core Concepts of LangSmith

LangSmith structures data hierarchically for better organization and insight.

1. **Project:** A logical grouping of related traces or runs. Think of it as a "workspace" for a particular LLM application or experiment. *Example:* `Project: AI Chatbot Evaluation`
2. **Trace:** A trace is a record of one complete execution of a workflow. It includes all steps, sub-calls, and tool invocations made during that process. *Example:* A single user question and the full reasoning path the agent took.
3. **Run:** Each individual execution within a trace (e.g., one LLM call or one function call). Runs are the atomic units of LangSmith's observability model. *Example:* A "summarize_text()" function call inside a larger document analysis trace.

LangSmith Concept Hierarchy

Project ⊃ Trace ⊃ Run

Each Project contains multiple Traces, and each Trace is made up of multiple Runs.

Why It Matters

By capturing all this data, LangSmith provides:

- Complete visibility into how LLM applications behave in real time.
- Easier debugging and optimization for multi-step agents.
- Reliable performance tracking across deployments and experiments.