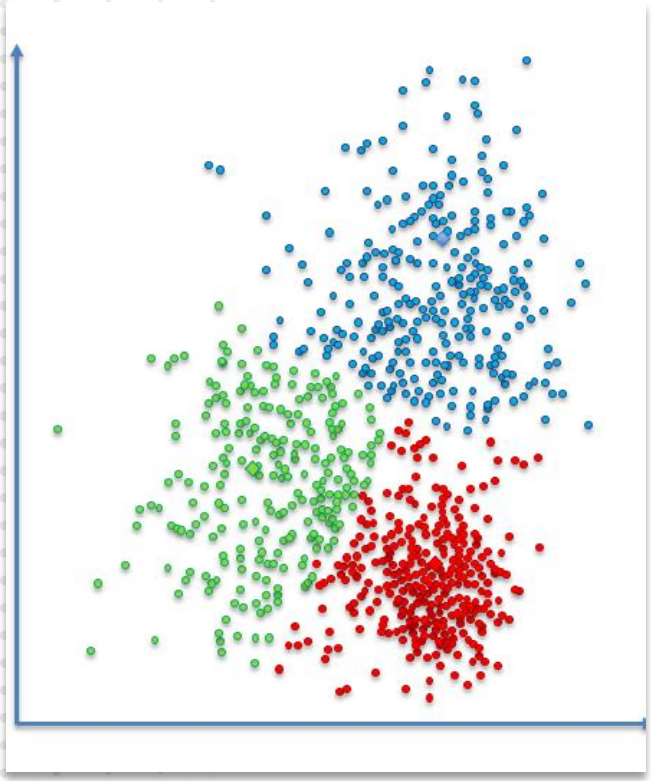


Lesson #6: LLM Monitoring and Observability

Objectives:

- Monitoring and Observability Overview
- LangSmith

Monitoring and Observability Platforms

Models and Transformers
- Llama
- OpenAI, Claude, etc
- DeepSeek

Agent Orchestration Platforms
- LangGraph
- AutoGen
- CrewAI

Development Environments
- Colab
- VS Code
- Cursor / Windsurf

Agentic Protocols
- APIs
- MCP
- A2A

Local LLM Management Platforms
- Ollama
- LM Studio
- Jan

Cloud Development Platforms
- AWS Bedrock
- Azure AI Studio
- Google Vertex AI

RAG & Agentic Frameworks
- LlamaIndex
- LangGraph
- Haystack

Monitoring and Observability
- LangSmith
- Weights & Biases
- Helicone

Mature | Strong Contenders | Emerging

Pearson

Seeing Inside the Black Box: Monitoring AI Agents in Production

Your AI agent just made 47 API calls, reasoned through 12 steps, and gave a wrong answer. How do you debug that????

Quick statistic: "73% of AI projects fail due to lack of proper monitoring and observability"

The AI Agent Monitoring Challenge

- **Complex reasoning chains**: Agents make multi-step decisions we can't see
- **Non-deterministic behavior**: Same input ≠ same output
- **Tool orchestration**: Agents use multiple tools, APIs, and data sources
- **Emergent failures**: Issues arise from interaction patterns (e.g. workflows), not individual components

What Makes AI Agent Observability Different

Reasoning is not transparent:

- Need to see the "thought process" not just inputs/outputs
- Understanding decision trees and branching logic (why was a certain tool selected?)
- Tracking confidence levels and uncertainty

AI uses multi-modal interactions

Behaviour is dynamic (agents adapt based on context)

LangSmith - Purpose-Built for AI Observability

What is LangSmith?

- Observability platform specifically designed for LLM applications and AI agents
- Created by LangChain team - understands the AI agent ecosystem
- End-to-end visibility from user query to final response
  - Allows tracing of every step of the agentic workflow

LangSmith – How it Works

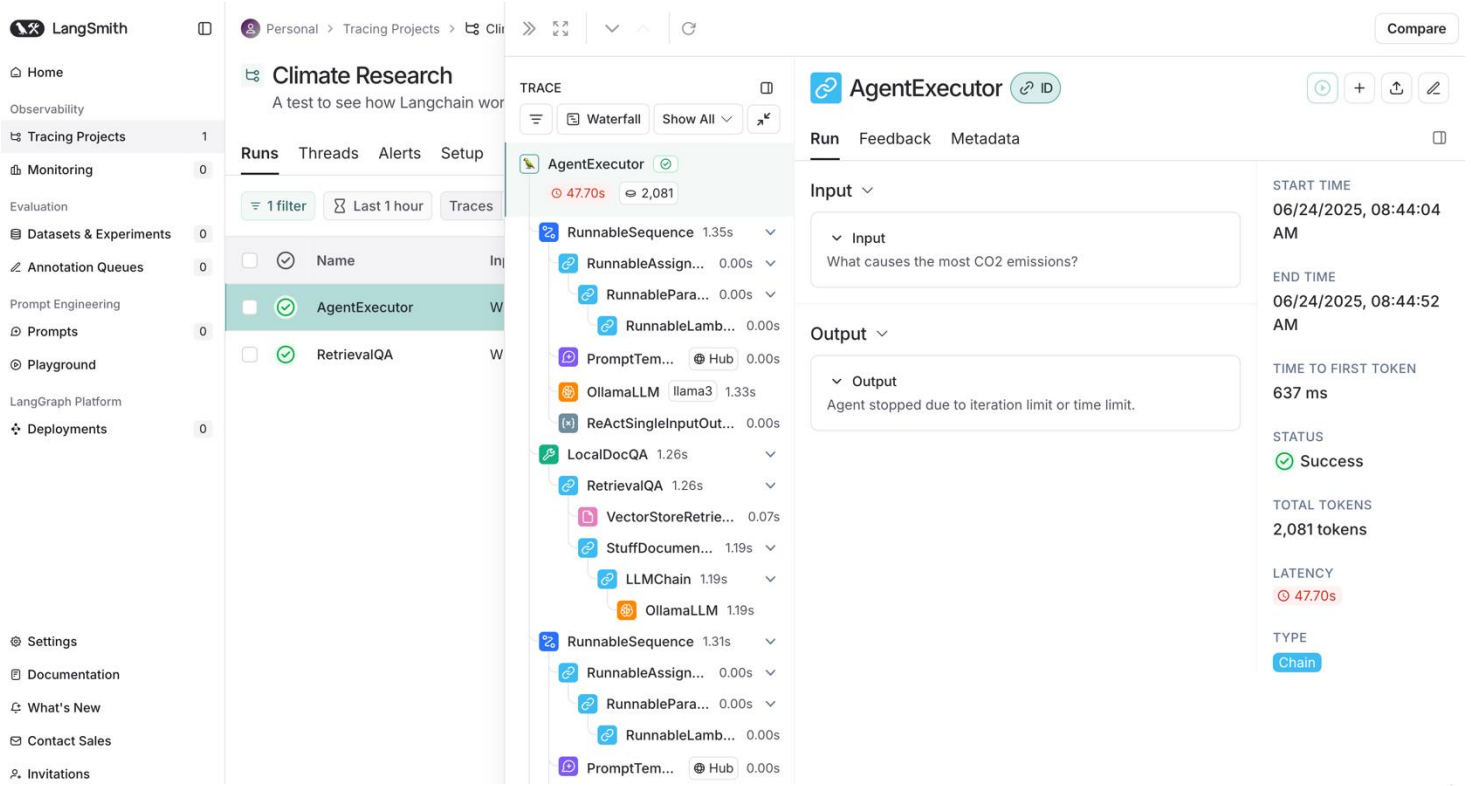
- LangChain has a built-in callback architecture that all its components use.
- Whenever something meaningful happens (e.g. like calling an LLM, executing a tool, running a chain) it triggers callback events.
- The Callback Manager is like an event hub. It sends those events to all registered handlers.

```
LANGSMITH_TRACING=true
LANGSMITH_ENDPOINT="https://api.smith.langchain.com"
LANGSMITH_API_KEY="<your-api-key>”
LANGSMITH_PROJECT=”Climate Research"
```

- Also works without a LangChain framework, using “wrappers”

LangSmith in Action

DEMO



Pearson