Langfuse

Understanding and monitoring your AI applications

What is LLM Observability?

- **Visibility into AI application behavior** during development and production
- **Tracking LLM calls, inputs, outputs, and performance** metrics
- **Understanding conversation flows** and user interactions
- **Debugging failures** and optimizing performance
- **Quality monitoring** and cost management

Why Do You Need It?

- AI applications are **non-deterministic** and hard to debug
- **Token costs** can quickly spiral out of control
- **Quality issues** may only surface in production
- **Performance bottlenecks** are difficult to identify

Introducing Langfuse

What is Langfuse?

- **Open-source LLM engineering platform**
- **Framework-agnostic** observability
- **Production-ready** tracing and analytics
- **Self-hosted or cloud** deployment options
- **MIT licensed** with active community

Core Features

- **Tracing**: Nested execution tracking
- **Analytics**: Performance and cost metrics
- **Evaluations**: Quality assessment tools
- **Sessions**: Conversation tracking
- **Real-time monitoring**: Live application insights

Key Benefits

- **Complete visibility** into LLM operations
- **Cost tracking** and optimization
- **Quality monitoring** and evaluation

Integrations

- **LangChain** (Python & TypeScript)
- **LiteLLM** for multi-provider support
- **Custom SDKs** for any framework

Langfuse Architecture

Core Concepts

- **Traces**: Complete execution flows
- **Spans**: Individual operations within traces
- **Generations**: LLM calls with input/output
- **Sessions**: User conversation groups
- **Observations**: All captured events

Trace Hierarchy

```
Trace: "User Chat Session"
├── Generation: "Initial LLM call"
├── Span: "Tool execution"
│   ├── Generation: "Tool reasoning"
│   └── Generation: "Tool result"
└── Generation: "Final response"
```

Session Tracking

```
Session: "User_123_Chat"
├── Trace: "Question about Python"
├── Trace: "Follow-up on functions"
└── Trace: "Code review request"
```

Getting Started with Langfuse

1. Installation

```
# Install Langfuse SDK
pip install langfuse

# With LangChain integration
pip install langfuse langchain
```

2. Environment Setup

```
# .env file
LANGFUSE_SECRET_KEY=sk-lf-your-secret-key
LANGFUSE_PUBLIC_KEY=pk-lf-your-public-key
LANGFUSE_HOST=https://cloud.langfuse.com
```

3. Account Setup

- Sign up at cloud.langfuse.com
- Create a new project
- Generate API credentials
- Add to your environment

Basic Integration

```python
from langfuse import Langfuse

# Initialize Langfuse client
langfuse = Langfuse()

# Create a trace
trace = langfuse.trace(
    name="chat_completion",
    user_id="user_123",
    session_id="session_abc"
)

# Add a generation
generation = trace.generation(
    name="llm_call",
    model="gpt-4",
    input={"prompt": "Hello, world!"},
    output={"response": "Hi there!"}
)

generation.end()
```

LangChain Integration

Callback Handler Pattern

```python
from langfuse.callback import CallbackHandler
from langchain_google_genai import ChatGoogleGenerativeAI

# Create Langfuse callback handler
langfuse_handler = CallbackHandler(
    session_id="user_session_123"
)

# Initialize LLM
llm = ChatGoogleGenerativeAI(
    model="gemini-2.5-flash-preview-04-17"
)

# Invoke with tracing
response = llm.invoke(
    "What is machine learning?",
    config={"callbacks": [langfuse_handler]}
)
```

Automatic Tracing

```python
from langchain.chains import LLMChain
from langchain.prompts import PromptTemplate

prompt = PromptTemplate.from_template(
    "Explain {topic} in simple terms"
)

chain = LLMChain(
    llm=llm,
    prompt=prompt,
    callbacks=[langfuse_handler]
)

# Automatically traced
result = chain.invoke({"topic": "recursion"})

# Async operations too
await chain.ainvoke({"topic": "databases"})
```

What Gets Tracked

Automatic LangChain Tracking

- **Chain operations**: start, end, errors
- **LLM calls**: inputs, outputs, model parameters
- **Agent actions**: reasoning steps, tool calls
- **Token usage**: input/output token counts
- **Execution time**: latency for each operation
- **Error details**: stack traces and error messages

Metadata Captured

- **Model information**: name, temperature, parameters
- **Session context**: user ID, conversation ID
- **Performance metrics**: execution time, token costs
- **Quality indicators**: success/failure rates

Dashboard and Analytics

Real-Time Dashboard

- **Live trace updates** as operations execute
- **Performance metrics** with charts and graphs
- **Cost tracking** with token usage breakdown
- **Error monitoring** with failure rates
- **User sessions** with conversation flows

Trace Exploration

- **Nested view** of execution hierarchies
- **Timeline visualization** of operations
- **Input/output inspection** for debugging
- **Performance drill-down** to identify bottlenecks
- **Session grouping** for user journey analysis

Key Metrics

- **Latency**: P50, P95, P99 response times
- **Throughput**: Requests per minute/hour
- **Cost**: Token usage and estimated costs

Collaborative Features

- **Shared dashboards** for team visibility
- **Annotations** for trace comments
- **Alerts** for performance thresholds

Key Takeaways

- **Observability is essential** for production AI applications
- **Langfuse provides comprehensive** LLM tracing and analytics
- **LangChain integration** makes tracing automatic and seamless
- **Graceful degradation** ensures tracing never breaks your app
- **Dashboard insights** help debug and optimize applications
- **Session tracking** enables conversation-level analysis
- **Production patterns** support scalable observability

Remember

Observability transforms AI development from **guessing** to **understanding**. Start with basic tracing and build sophisticated monitoring as you scale.

Monitor Everything!