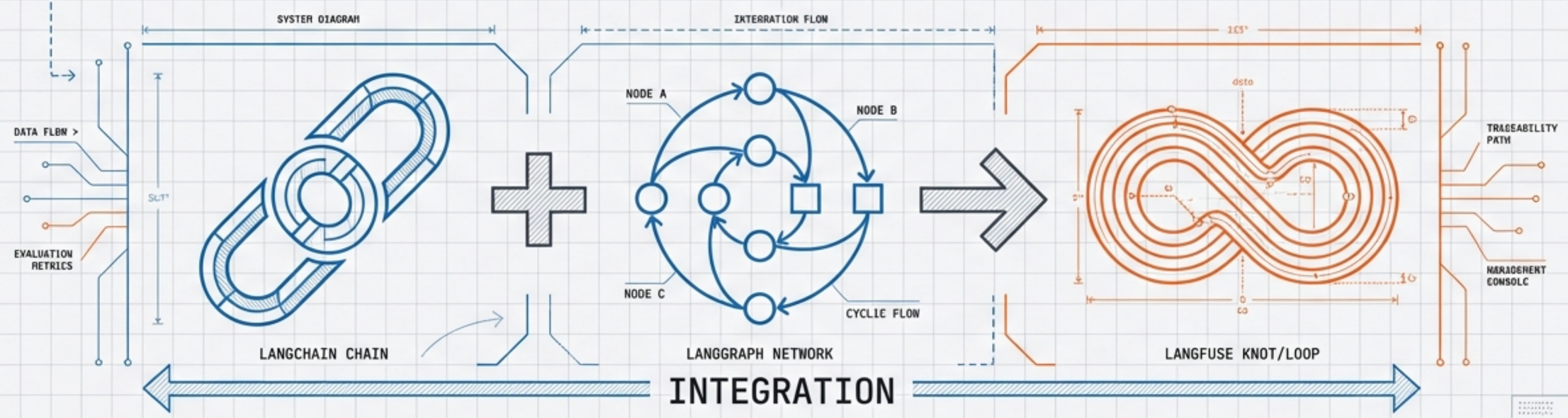
The Complete Guide to Langfuse for LangChain & LangGraph

Tracing, Evaluation, and Management for Python Applications




```
def trace_chain(data):
    ...
    langfuse.trace(
        langfuse.trace, ddea, "...")
    )
```

ENGINEERING LIFECYCLE SERIES • VOL. 1

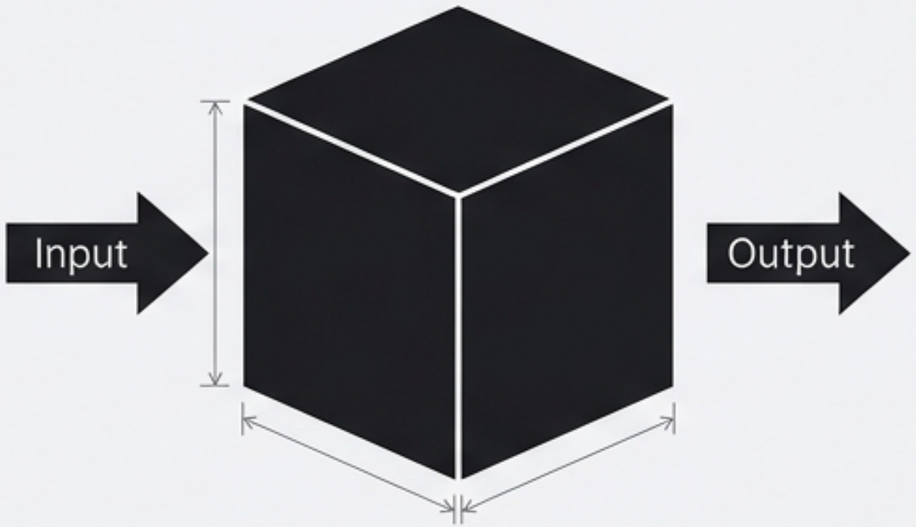
PROJECT: AI-OPS	
DATE: OCT 26, 2024	
EKGINEER: SUISSE DESIGN	

The Observability Imperative

Moving from “it works on my machine”
to production reliability.

BEFORE

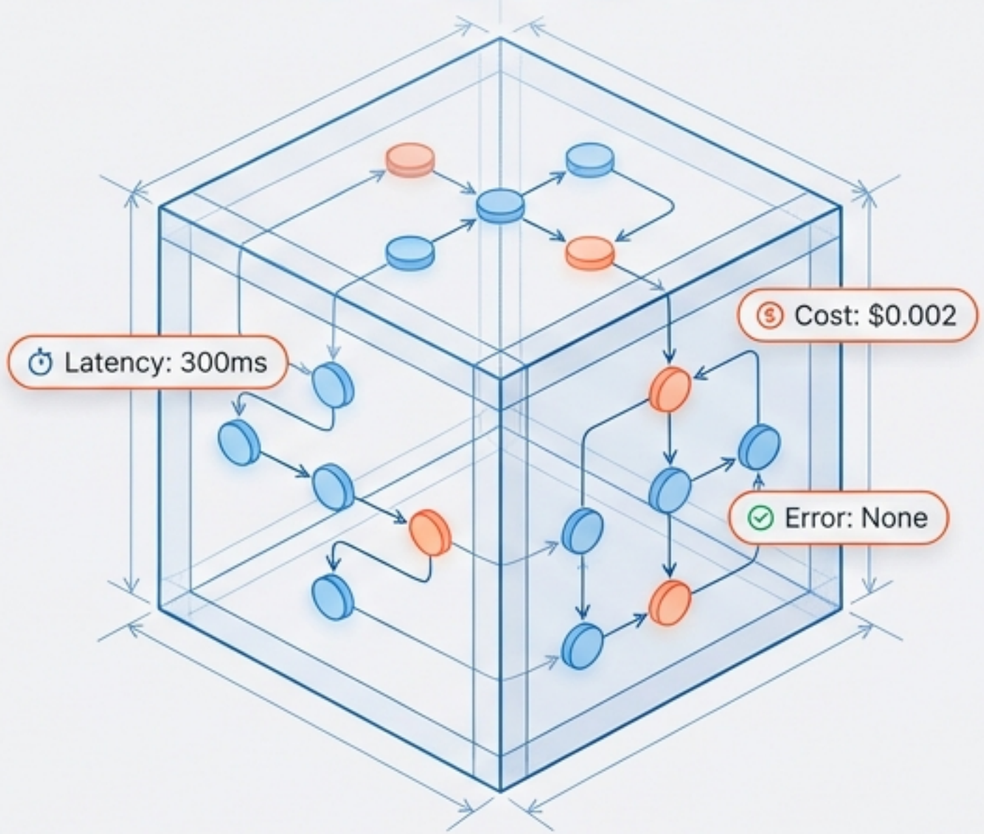
The Black Box



- Unclear failure points
- Unknown costs
- Non-deterministic errors

AFTER

Glass Box



1. **DEBUG:** Trace complex logic.
2. **MONITOR:** Real-time health metrics.
3. **IMPROVE:** Dataset evaluation loops.

Initialization & Environment Setup

00

Step 01: Installation

```
%pip install langfuse langchain langgraph langchain_openai
```

Step 02: Environment Variables

```
LANGFUSE_PUBLIC_KEY = pk-lf-...
LANGFUSE_SECRET_KEY = sk-lf-...
LANGFUSE_BASE_URL   = https://cloud.langfuse.com
OPENAI_API_KEY      = sk-proj-...
```

Step 03: Client Initialization

```
from langfuse import get_client
langfuse = get_client()
# Verify connection: langfuse.auth_check()
```

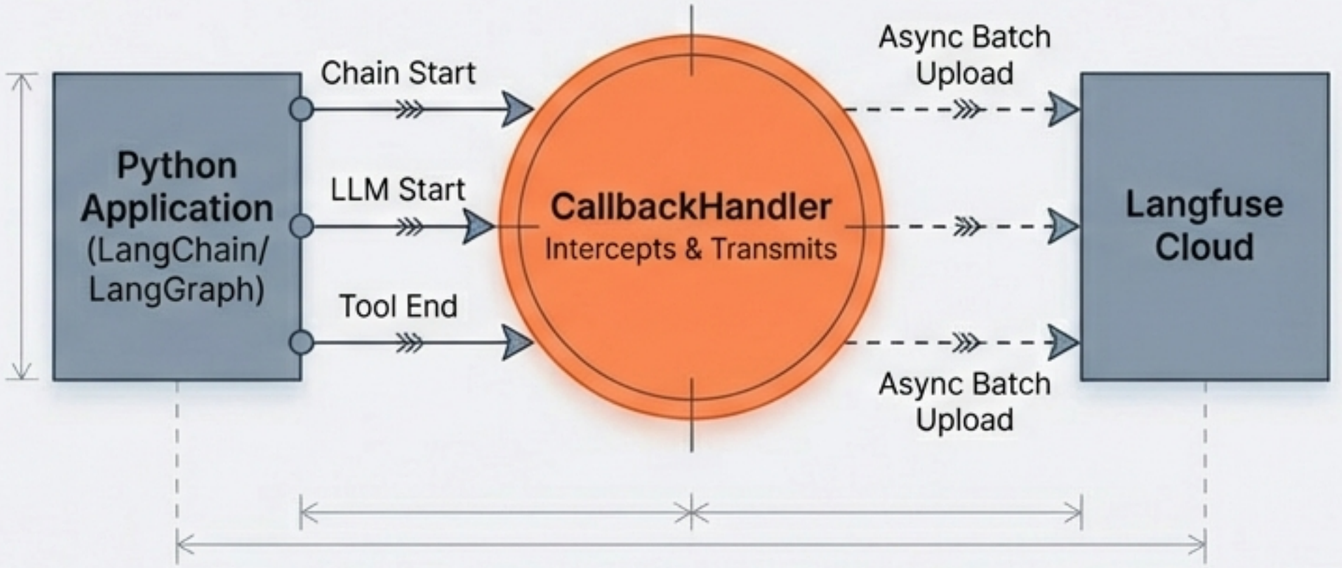
The Core Mechanism: CallbackHandler

The CallbackHandler is the bridge between your application logic and the Langfuse observability cloud. It intercepts events—Start, End, Error—and transmits them asynchronously.

```python
from langfuse.langchain import CallbackHandler

# Initialize the handler once
langfuse_handler = CallbackHandler()
```

JetBrains Mono



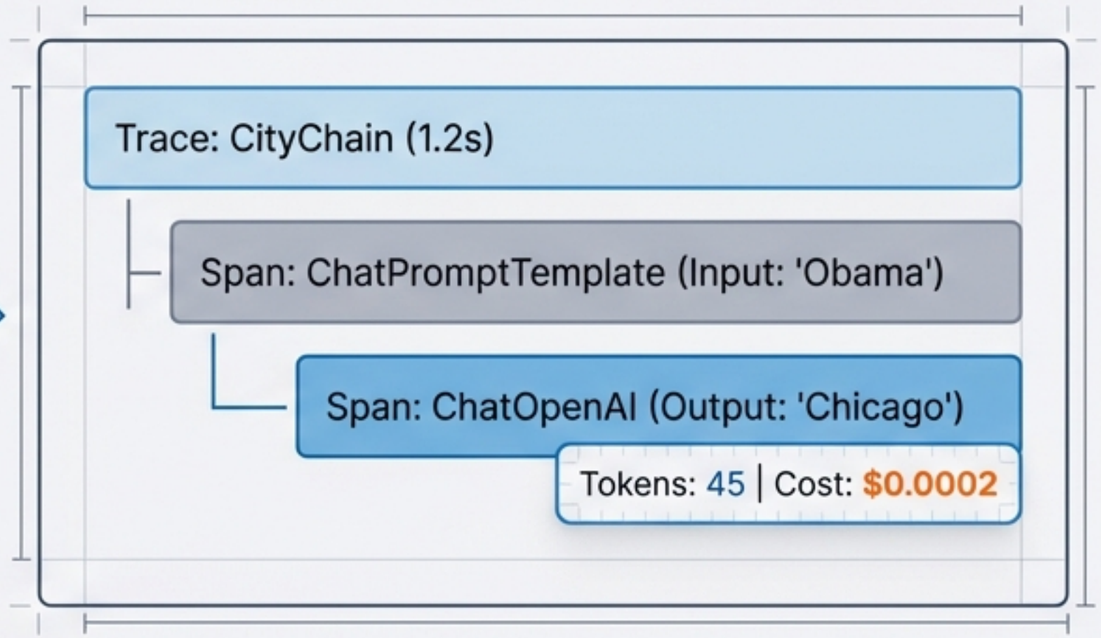
Tracing Linear Chains (LCEL)

THE INPUT (Python)

```
chain = prompt | model | StrOutputParser()

# Pass the handler in the config
chain.invoke(
    {'person': 'Obama'},
    config={'callbacks':[langfuse_handler]}
)
```

THE OUTPUT (Waterfall Trace)

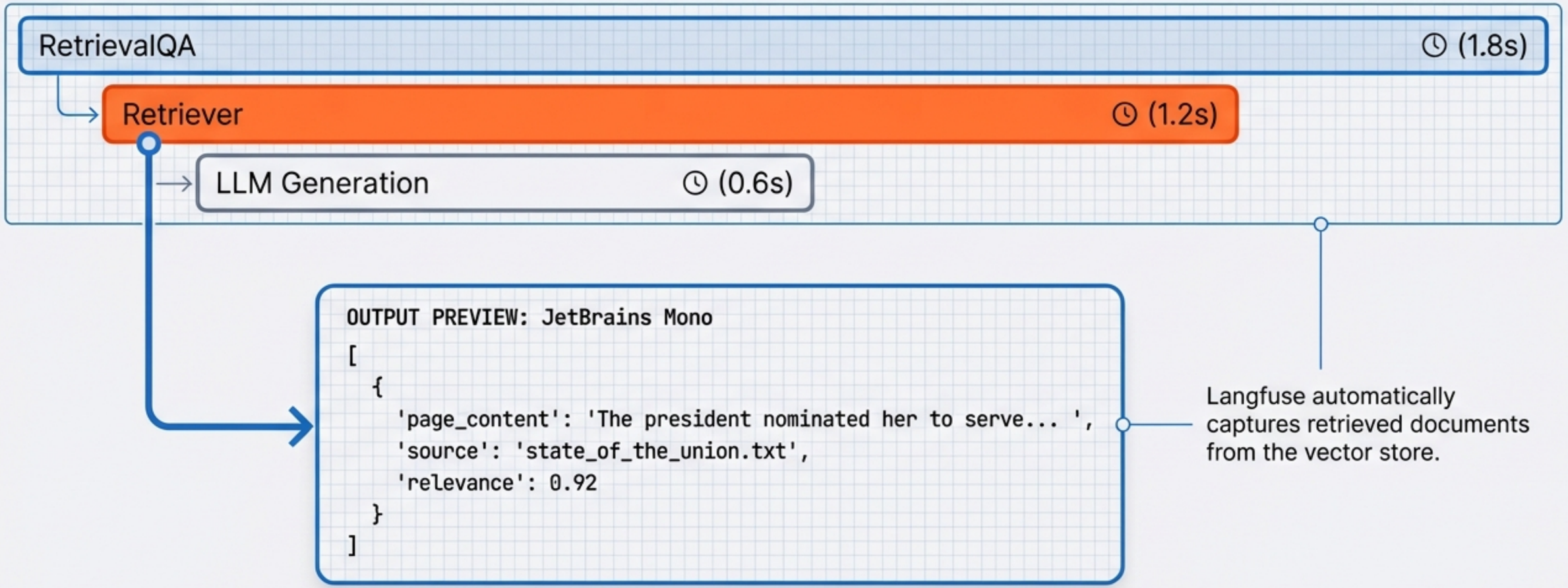


Debugging Retrieval Pipelines (RAG)

Inspecting retrieved context to identify hallucinations.

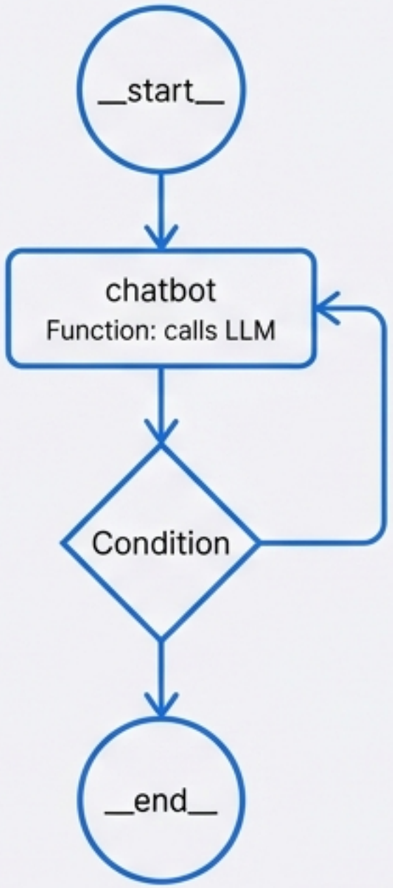
```
chain = RetrievalQA.from_chain_type(..., retriever=docsearch.as_retriever())
chain.invoke(query, config={'callbacks':[langfuse_handler]})
```

Trace Visualization



Stepping Up to LangGraph: Stateful Agents

Cyclic StateGraph



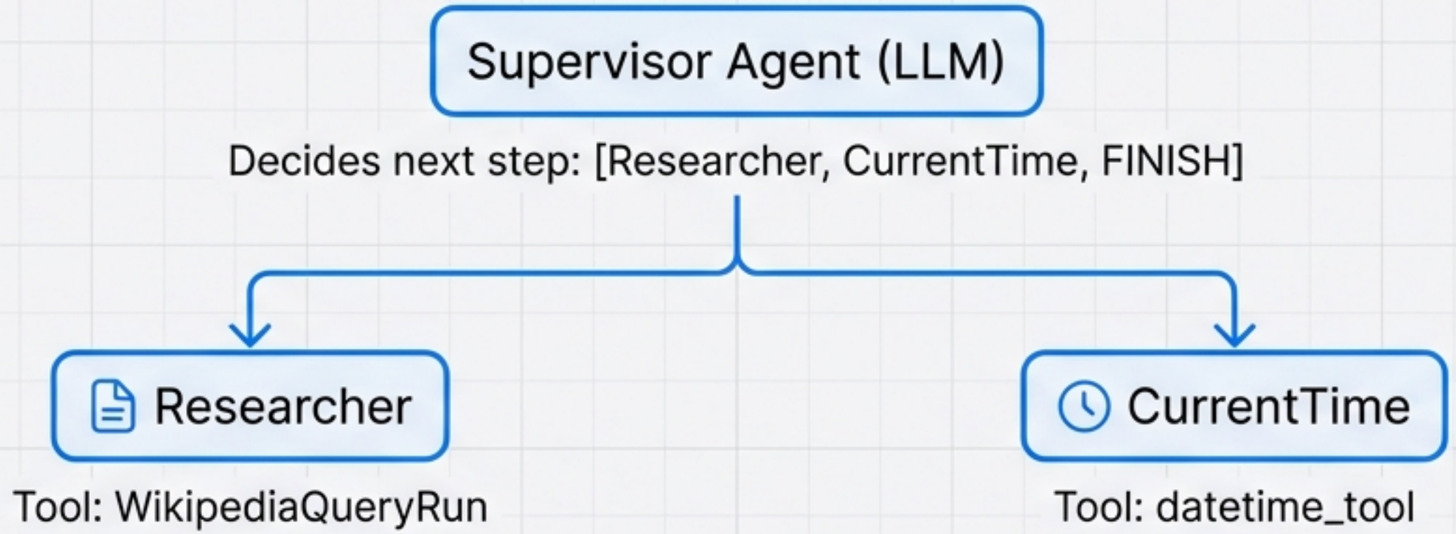
Implementation details

```
# 1. Compile the graph
graph = builder.compile()

# 2. Stream with callbacks
inputs = {'messages': [HumanMessage(content='Hi')]}
graph.stream(
    inputs,
    config={'callbacks': [langfuse_handler]}
)
```

The handler automatically traces the graph structure, preserving the parent/child relationship of steps.

Orchestrating Multi-Agent Systems



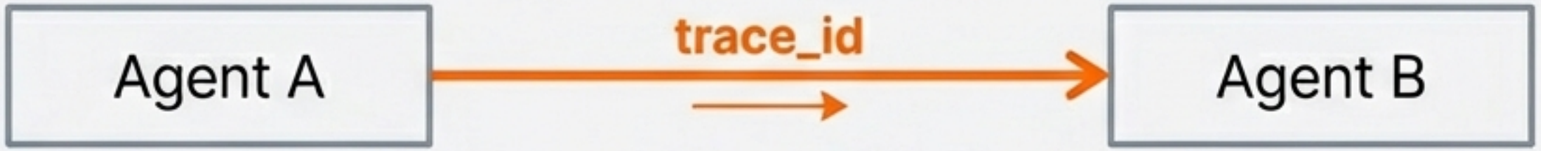
Trace Tree

```
▼ 🤖 Supervisor (Trace Root)
  ├─ 🤖 Router (LLM Decision: 'Researcher')
  ├─ 🛠️ Researcher (Tool Execution)
  │   └─ >_ wikipedia.search('Photosynthesis')
  └─ 🤖 Supervisor (Next Step: 'FINISH')
```

Langfuse visualizes the delegation of tasks across agent nodes.

Advanced Tracing: Linking & Custom Spans

Linking Distributed Executions




```
trace_id = Langfuse.create_trace_id()
...
trace_context={'trace_id': trace_id}
```

Manual Instrumentation (Custom Spans)

```
with
langfuse .start_as_current_observation(
    name='🤖-sub-research-agent',
    trace_context=...
) as span:
    # Execute custom logic here
    result = sub_agent.invoke(...)
    span.update_trace(output=result)
```

Wraps logic inside a custom span, useful for tools or sub-agents.

Online Evaluation (Production Metrics)

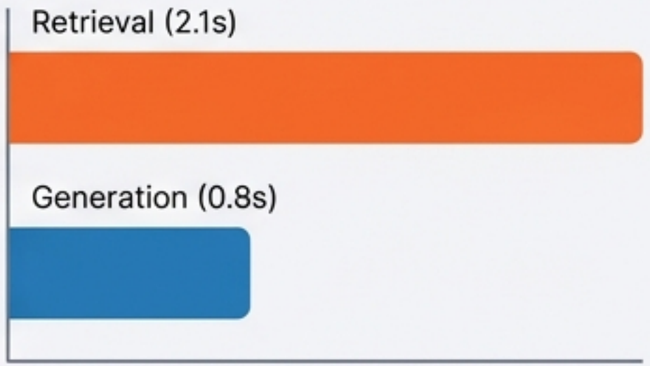
Monitoring the three vital signs of AI applications.

Costs (Token Usage)

Automatic Tracking

Latency (Bottlenecks)

Step-level breakdown



Explicit User Feedback

Correct answer

```
span.score_trace(
  name='user-feedback',
  value=1,
  comment='Correct answer'
)
```

Automated QA: LLM-as-a-Judge

Scalable evaluation without human intervention.

Agent Output

Draft: "Here is the investment offer..."

`Is this Spam?`

Judge Model

Model: `gpt-4o-mini`

Prompt: "`Check for toxicity/spam`"

Langfuse Score

Score: Toxicity

0.0 (Safe)

The content is promotional but not toxic.

Offline Evaluation (Datasets)

Regression testing before deployment.

1. Dataset Upload

HuggingFace -> Langfuse Dataset

```
langfuse.create_dataset_item(input=..., expected_output=...)
```

2. Experiment Run

Iterate items & Trace

```
with item.run() as root_span:
```

3. Comparative Report

Input Question	GPT-3.5	GPT-4o
What is Langfuse?	❌ Fail	✅ Pass
How to install?	✅ Pass	✅ Pass

GPT-4o Accuracy: 98%

Prompt Management as a CMS

Decoupling prompts from application code.

The Workflow

1. Create/Edit in UI

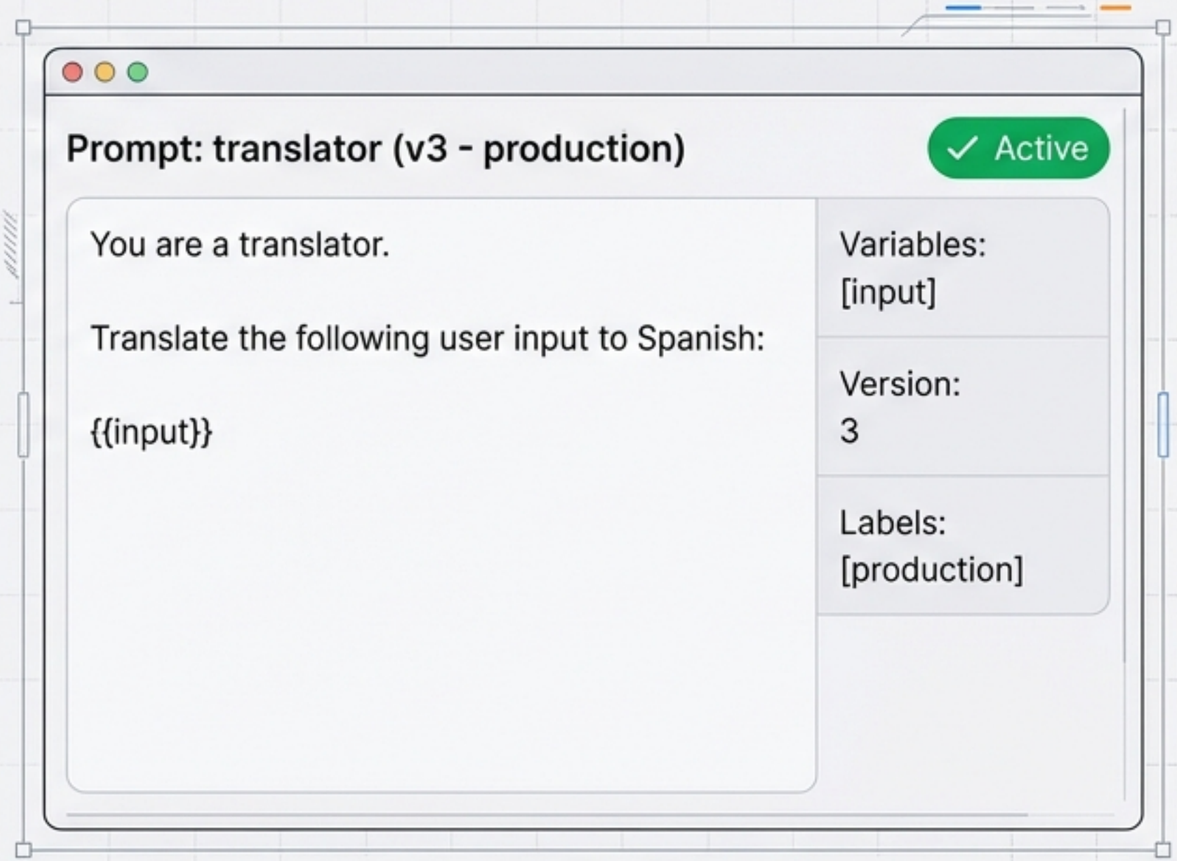
```
langfuse.create_prompt(name='translator', ...)
```

2. Fetch in Code

```
langfuse.get_prompt('translator')
```

3. Convert Format

```
.get_langchain_prompt()
```



Prompt Editor UI Simulation

Implementation Cheatsheet

Goal	Action / Syntax
Trace a Chain	`chain.invoke(..., config={'callbacks':[handler]})`
Trace a Graph Agent	`graph.compile().with_config({'callbacks':[handler]})`
Link Distributed Agents	`trace_context={'trace_id': predefined_id}`
Score a Run (Feedback)	`span.score_trace(name='feedback', value=1)`
Test a Dataset	`with item.run() as root_span: ...`

Start Tracing Today

cloud.langfuse.com

```
$ pip install langfuse langchain langgraph
```

Cookbooks

Integration examples for Python & JS

Documentation

langfuse.com/docs

GitHub

Open Source (MIT License)

Scan for Cookbooks