

# Thực hành LangSmith

## LangSmith là gì?

LangSmith là một nền tảng giúp bạn: [LangSmith](#)

- **Theo dõi (tracing):** Ghi lại toàn bộ quá trình thực thi của ứng dụng LLM.
- **Đánh giá (evaluation):** Đánh giá chất lượng đầu ra của mô hình.
- **Kỹ thuật prompt (prompt engineering):** Quản lý và tối ưu hóa các prompt.
- **Giám sát (observability):** Theo dõi hiệu suất và hành vi của ứng dụng. [LangSmith](#)

LangSmith có thể được sử dụng độc lập hoặc tích hợp với **LangChain** để xây dựng và triển khai các ứng dụng LLM chất lượng cao.

## Bước 1: Cài đặt và cấu hình

### 1.1 Cài đặt thư viện cần thiết

```
pip install langsmith openai
```

### 1.2 Tạo tài khoản và API Key

1. Truy cập [LangSmith](#) và đăng ký tài khoản.
2. Sau khi đăng nhập, vào phần **Settings** và tạo một **API Key**. [Introduction | LangChain](#)  
[+5LangSmith+5YouTube+5LangSmith+1Medium+1](#)

### 1.3 Thiết lập biến môi trường

```
export LANGSMITH_TRACING=true
export LANGSMITH_API_KEY="your-langsmith-api-key"
export OPENAI_API_KEY="your-openai-api-key"
```

## Bước 2: Tạo ứng dụng mẫu và ghi lại quá trình thực thi

### 2.1 Định nghĩa ứng dụng mẫu

```

from langsmith import traceable
from langsmith.wrappers import wrap_openai
from openai import OpenAI

# Bọc OpenAI client để ghi lại các cuộc gọi
openai_client = wrap_openai(OpenAI())

# Hàm truy xuất tài liệu (giả lập)
def retriever(query: str):
    return ["Harrison worked at Kensho"]

# Hàm chính thực hiện RAG
@traceable
def rag(question: str):
    docs = retriever(question)
    system_message = f"Answer the user's question using only the provided information below:\n\n{docs[0]}"
    response = openai_client.chat.completions.create(
        messages=[
            {"role": "system", "content": system_message},
            {"role": "user", "content": question},
        ],
        model="gpt-4o-mini",
    )
    return response.choices[0].message.content

# Gọi hàm và in kết quả
answer = rag("Where did Harrison work?")
print("Answer:", answer)

```

## 2.2 Xem trace trên LangSmith

Sau khi chạy mã, truy cập [LangSmith](#) để xem chi tiết trace của ứng dụng. [LangSmith+3LangSmith+3YouTube+3](#)

# Bước 3: Đánh giá ứng dụng với LangSmith

## 3.1 Tạo tập dữ liệu đánh giá

```

from langsmith import Client

```

```

client = Client()
dataset = client.create_dataset(
    name="agent-qa-demo",
    description="Dataset for evaluating the RAG application."
)

# Thêm dữ liệu vào tập
examples = [
    {"input": {"question": "Where did Harrison work?"}, "output": "Harrison worked at Kensho."},
    {"input": {"question": "What company did Harrison work for?"}, "output": "Kensho"},
]

for example in examples:
    client.create_example(
        inputs=example["input"],
        outputs={"answer": example["output"]},
        dataset_id=dataset.id
    )

```

### 3.2 Chạy đánh giá trên tập dữ liệu

```

from langsmith.evaluation import run_on_dataset
import functools

# Định nghĩa hàm tạo ứng dụng
def create_rag_app():
    return functools.partial(rag)

# Chạy đánh giá
results = run_on_dataset(
    dataset_name="agent-qa-demo",
    llm_or_chain_factory=create_rag_app(),
    evaluation="qa",
    client=client,
    project_name="rag-evaluation-demo"
)

```

### 3.3 Phân tích kết quả

Sau khi đánh giá hoàn tất, bạn có thể xem kết quả chi tiết trên giao diện LangSmith, bao gồm:

- Các trace của từng lần chạy.
  - Điểm số đánh giá.
  - Phản hồi từ người dùng (nếu có).
- [Introduction | LangChainLearn R, Python & Data Science Online+1YouTube+1](#)

## Bước 4: Tùy chỉnh và mở rộng

- **Tích hợp với LangChain:** Nếu bạn sử dụng LangChain, LangSmith có thể tích hợp trực tiếp để ghi lại trace của các chain và agent.
- **Tạo dashboard tùy chỉnh:** LangSmith cho phép bạn tạo các dashboard để giám sát các chỉ số quan trọng như chi phí, độ trễ và chất lượng phản hồi.
- **Thu thập phản hồi từ người dùng:** Bạn có thể thu thập phản hồi từ người dùng cuối để cải thiện ứng dụng.

---

Phiên bản #1

Được tạo 7 tháng 5 2025 10:18:08 bởi Đỗ Ngọc Tú

Được cập nhật 7 tháng 5 2025 10:23:44 bởi Đỗ Ngọc Tú