

RAG 평가

강사장철원

□ LangSmith 소개

RAG 평가

Section 1. LangSmith

Section 1-1. LangSmith의 개념

LangSmith의 개념



langsmith



전체

이미지

동영상

쇼핑

뉴스

짧은 동영상

도서

⋮ 더보기



LangChain

<https://www.langchain.com> > langsmith ⋮

LangSmith

LangSmith is a unified observability & evals platform where teams can debug, test, and monitor AI app performance — whether building with LangChain or not ...

Introduction to LangSmith

Learn the essentials of LangSmith — our platform for LLM ...



How to select examples from a ...

LangSmith datasets have built-in support for similarity search ...

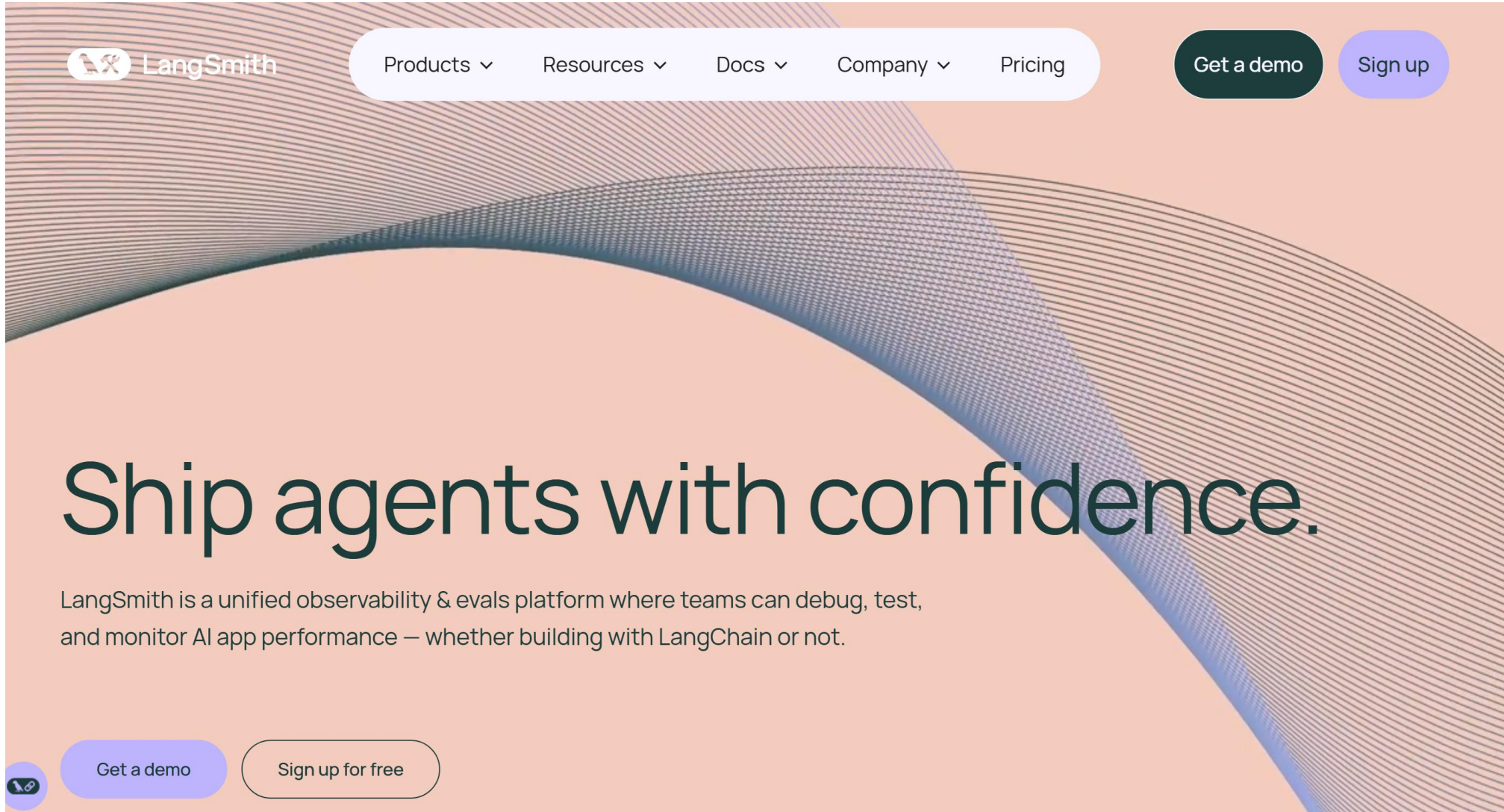


[langchain.com](#) 검색결과 더보기 »

Section

LangSmith의 개념

LangSmith의 개념



Section

LangSmith의 개념

가격 정책



LangChain

Products ▾

Resources ▾

Docs ▾

Company ▾

Pricing

Get a demo

Sign up

Plans for teams of any size

Get all the LangChain products – pay for what you use

Developer

For hobbyist projects by solo devs.

Starting at

\$0 / month

then pay as you go

Start for free

Plus

Batteries-included tooling for building reliable agents fast.

Starting at

\$39 / month

then pay as you go

Sign up

Enterprise

For teams with advanced deployment, security, and support needs.

Custom

Get a demo

LangSmith의 개념



How would you like to start?

Select a workflow to begin exploring LangSmith's capabilities

Trace an Application

Analyze and debug applications using traces.



Test Prompts in the Playground

Iterate and test on prompts across any model or provider.



Run an Evaluation

Measure app performance: identify failures, compare changes, ensure reliability.



Build and Deploy Agents

Deploy agentic applications to production with LangGraph Platform.



[Skip →](#)

LangSmith의 개념

Personal > Tracing Projects > default

default

Runs Threads Alerts Setup

1 filter Last 7 days Traces LLM Calls All Runs

<input type="checkbox"/>	<input checked="" type="checkbox"/>	Name	Input
<input type="checkbox"/>	<input checked="" type="checkbox"/>	Sample Agent Trace	What is a document lo...

TRACE

Sample Agent Trace

ChatOpenAI 0.95s

search_latest_knowle... 0.76s

VectorStoreRetriever 0.52s

ChatOpenAI 3.44s

Sample Agent Trace

Run Feedback Metadata

Input

Input

What is a document loader?

Chat History

Output

Output

BEEP BOOP! A document loader is a component that retrieves data from a specific source and returns it as a LangChain "Document." You can find more information about document loader integrations [here] (/docs/modules/data_connection/document_loaders/). Each loader is designed to retrieve data from a particular source and return it in a format that can be processed by LangChain.

START TIME

07/12/2025, 07:29:20 AM

END TIME

07/12/2025, 07:29:28 AM

TIME TO FIRST TOKEN

STATUS

Success

TOTAL TOKENS

540 tokens

LATENCY

7.31s

TYPE

Chain

Compare

LangSmith

Section 1. LangSmith

Section 1-2. Tracing - LangChain

Tracing

The screenshot shows the 'Tracing Projects' interface. The sidebar on the left contains several icons, with the 'Tracing Projects' icon (a circuit board) highlighted by a red box. The top bar displays the user's profile 'Personal' and the current view 'Tracing Projects'. A 'Setup resource tags' button and a 'DEVELOPER' role indicator are also present. A '+ New Project' button is highlighted with a red box in the top right. The main area features a search bar labeled 'Search by name...' and a 'Columns' button. Below this is a table with one visible project named 'default'. The table has a header 'Name' with a sort icon. At the bottom, there is a pagination control showing 'Page 1' and a 'Show 10' dropdown.

Personal > Tracing Projects

Setup resource tags DEVELOPER

Tracing Projects

Analyze and debug applications.

+ New Project

Search by name...

Columns

Name ↑↓
default

Page 1 < > Show 10

Section

Tracing - LangChain

Tracing

GET STARTED WITH LANGSMITH

Set up observability

Trace, debug and monitor your application

With LangChain

Without LangChain


1. [Generate API Key](#)

2. Install dependencies

Python

TypeScript

```
1 pip install -U langchain langchain-openai
```


 Copy

3. Configure environment to connect to LangSmith.

Project Name


pr-warmhearted-making-60

```
1 LANGSMITH_TRACING=true
2 LANGSMITH_ENDPOINT="https://api.smith.langchain.com"
3 LANGSMITH_API_KEY="<your-api-key>"
4 LANGSMITH_PROJECT="pr-warmhearted-making-60"
5 OPENAI_API_KEY="<your-openai-api-key>"
```

 Copy

4. Run any LLM, Chat model, or Chain. Its trace will be sent to the set project.

```
1 from langchain_openai import ChatOpenAI
2
3 llm = ChatOpenAI()
4 llm.invoke("Hello, world!")
```

 Copy

Tracing

보관 주의

GET STARTED WITH LANGSMITH

Set up observability

Trace, debug and monitor your application

With LangChain

Without LangChain

1. New key created:

[Redacted]



2. Install dependencies

Python

TypeScript

1

```
pip install -U langchain langchain-openai
```



Copy

3. Configure environment to connect to LangSmith.

Project Name

pr-warmhearted-making-60

1

```
LANGSMITH_TRACING=true
```

2

```
LANGSMITH_ENDPOINT="https://api.smith.langchain.com"
```

3

```
LANGSMITH_API_KEY=
```

[Redacted]

4

```
LANGSMITH_PROJECT="pr-warmhearted-making-60"
```

5

```
OPENAI_API_KEY="<your-openai-api-key>"
```



Copy

4. Run any LLM, Chat model, or Chain. Its trace will be sent to the set project.

Section

Tracing - LangChain

코드 실행(1)

```
[10]: import os

os.environ["LANGCHAIN_TRACING_V2"]="true"
os.environ["LANGCHAIN_ENDPOINT"]="https://api.smith.langchain.com"
os.environ["LANGCHAIN_API_KEY"]=""
os.environ["LANGCHAIN_PROJECT"]="pr-shadowy-scow-41"

[11]: from langchain_huggingface import HuggingFacePipeline
      from transformers import AutoTokenizer, AutoModelForCausalLM, pipeline

[12]: # 토큰라이저 & 모델 불러오기
      model_id = "skt/kogpt2-base-v2"
      tokenizer = AutoTokenizer.from_pretrained(model_id)
      model = AutoModelForCausalLM.from_pretrained(model_id)

[13]: # 파이프라인 생성
      text_gen = pipeline(
          "text-generation",
          model=model,
          tokenizer=tokenizer,
          device=-1,
          truncation=True,
          max_length=50,
          do_sample=True,
          temperature=0.7,
      )
```

Device set to use cpu

코드 실행(2)

```
[14]: # 프롬프트 예시
prompt = "산 속에 토끼 한 마리가 살고 있었습니다. 그러던 어느 날 "

# LangChain으로 실행
response = llm.invoke(prompt)
```

```
[15]: print("생성된 문장:", response)
```

생성된 문장: 산 속에 토끼 한 마리가 살고 있었습니다. 그러던 어느 날 뭔가 일이 벌어졌습니다. 토끼는 어디론가 사라졌습니다. 그 토끼는 어디선가 날아온 건데, 이상하게도 이상한 소리를 내며 날아가고 있었습니다. 그 소리는 바로 '꿈틀거리는 토끼'였습니다."

1942년 4월, 나는 토끼에게 달려들었다.

토끼는 내가 미처 생각하지 못했던 바로 그 소리를 냈습니다.

"아! 이것은 꿈틀거리는 토끼입니다."

"그래요! 꿈틀거리는 토끼입니다."

토끼는 나를 향해 달려가더니 나를 향해 손을 뻗었습니다.

그 순간

실행 결과

Personal > Tracing Projects > pr-shadowy-scow-41

pr-shadowy-scow-41

RunsThreadsAlertsSetup

1 filterLast 7 daysTracesLLM CallsAll Runs

	Name	Input	Output
	HuggingFacePipeline	["산 속에 토끼 한 마리가 ...	산 속에 토끼 한 마
	QAEvalChain	[{"query":"데이터 무제...	[{"results":"질문:

TRACE

Waterfall

HuggingFacePip...skt/kogpt2-bas...3.44s324

HuggingFacePipeline

RunFeedbackMetadata

Prompt & Completion

산 속에 토끼 한 마리가 살고 있었습니다. 그러던 어느 날 산 속에 토끼 한 마리가 살고 있었습니다. 그러던 어느 날 뭔가 일이 벌어졌습니다. 토끼는 어디론가 사라졌습
니다. 그 토끼는 어디선가 날아온 건데, 이상하게도 이상한 소리를 내며 날아가고 있
었습니다. 그 소리는 바로 '꿈틀거리는 토끼'였습니다."
1942년 4월, 나는 토끼에게 달려들었다.
토끼는 내가 미처 생각하지 못했던 바로 그 소리를 냈습니다.
"아! 이것은 꿈틀거리는 토끼입니다."
"그래요! 꿈틀거리는 토끼입니다."
토끼는 나를 향해 달려가더니 나를 향해 손을 뻗었습니다.
그 순간

START TIME
07/11/2025, 06:08:26 PM

END TIME
07/11/2025, 06:08:29 PM

TIME TO FIRST TOKEN

STATUS
Success

TOTAL TOKENS
324 tokens

LATENCY
3.44s

TYPE
LLM

RAG 평가

Section 1. LangSmith

Section 1-3. Tracing - LangGraph

Section

Tracing - LangGraph

Tracing

```
[1]: import os

os.environ["LANGCHAIN_TRACING_V2"]="true"
os.environ["LANGCHAIN_ENDPOINT"]="https://api.smith.langchain.com"
os.environ["LANGCHAIN_API_KEY"]=[REDACTED]
os.environ["LANGCHAIN_PROJECT"]="pr-shadowy-scow-41"

[2]: from langgraph.graph import StateGraph, END

# 분류 노드
def classification(state):
    text = state.get("user_input", "")

    if any(word in text for word in ["좋아요", "네", "좋다", "응"]):
        label = "positive"
    elif any(word in text for word in ["싫어요", "아니요", "별로", "싫다"]):
        label = "negative"
    else:
        label = "neutral"
    return {**state, "label": label}

# 응답 노드
def positive_answer(state):
    return {**state, "response": "좋게 생각해주셔서 감사합니다!"}

def negative_answer(state):
    return {**state, "response": "무엇이 불편하셨나요?"}

def neutral_answer(state):
    return {**state, "response": "조금 더 자세히 말씀해 주세요."}

def get_label(state):
    return state.get("label", "")
```

Tracing

```
# 그래프 설계
```

```
graph = StateGraph(dict)
```

```
graph.add_node("classification", classification)
```

```
graph.add_node("positive", positive_answer)
```

```
graph.add_node("negative", negative_answer)
```

```
graph.add_node("neutral", neutral_answer)
```

```
graph.set_entry_point("classification")
```

```
graph.add_conditional_edges("classification", get_label, {  
    "positive": "positive",  
    "negative": "negative",  
    "neutral": "neutral"  
}))
```

```
# 각 노드 끝나면 종료
```

```
graph.add_edge("positive", END)
```

```
graph.add_edge("negative", END)
```

```
graph.add_edge("neutral", END)
```

```
# 5. 실행기 구성
```

```
app = graph.compile()
```

```
# 6. 테스트 실행 (한글 입력)
```

```
user_input = input("한글 입력: ")
```

```
final_state = app.invoke({"user_input": user_input})
```

```
print("응답:", final_state.get("response", ""))
```

한글 입력: 이 서비스 너무 좋아요

응답: 좋게 생각해주셔서 감사합니다!

Section

Tracing - LangGraph

Tracing

Personal > Tracing Projects > pr-shadowy-scow-41

pr-shadowy-scow-41

Runs Threads Alerts Setup

1 filter Last 7 days Traces LLM Calls All Runs

	Name	Input	Output
<input checked="" type="checkbox"/>	LangGraph	이 서비스 너무 좋아요	positive
<input type="checkbox"/>	HuggingFacePipeline	["산 속에 토끼 한 마리가 ...	산 속에 토끼 한 마
<input type="checkbox"/>	QAEvalChain	[{"query":"데이터 무제...	[{"results":"질문:

TRACE

Waterfall

LangGraph 0.01s

classification 0.00s

get_label 0.00s

positive 0.00s

LangGraph ID

Playground Add to

Run Feedback Metadata

Input

User Input
이 서비스 너무 좋아요

Output

Label
positive

Response
 좋게 생각해주셔서 감사합니다!

User Input
이 서비스 너무 좋아요

START TIME
07/11/2025, 06:17:06 PM

END TIME
07/11/2025, 06:17:06 PM

TIME TO FIRST TOKEN

STATUS
Success

TOTAL TOKENS
0 tokens

LATENCY
0.01s

TYPE
Chain

RAG 평가

Section 1. LangSmith

Section 2-1. Monitoring

Monitoring

Personal > Monitoring

Monitoring

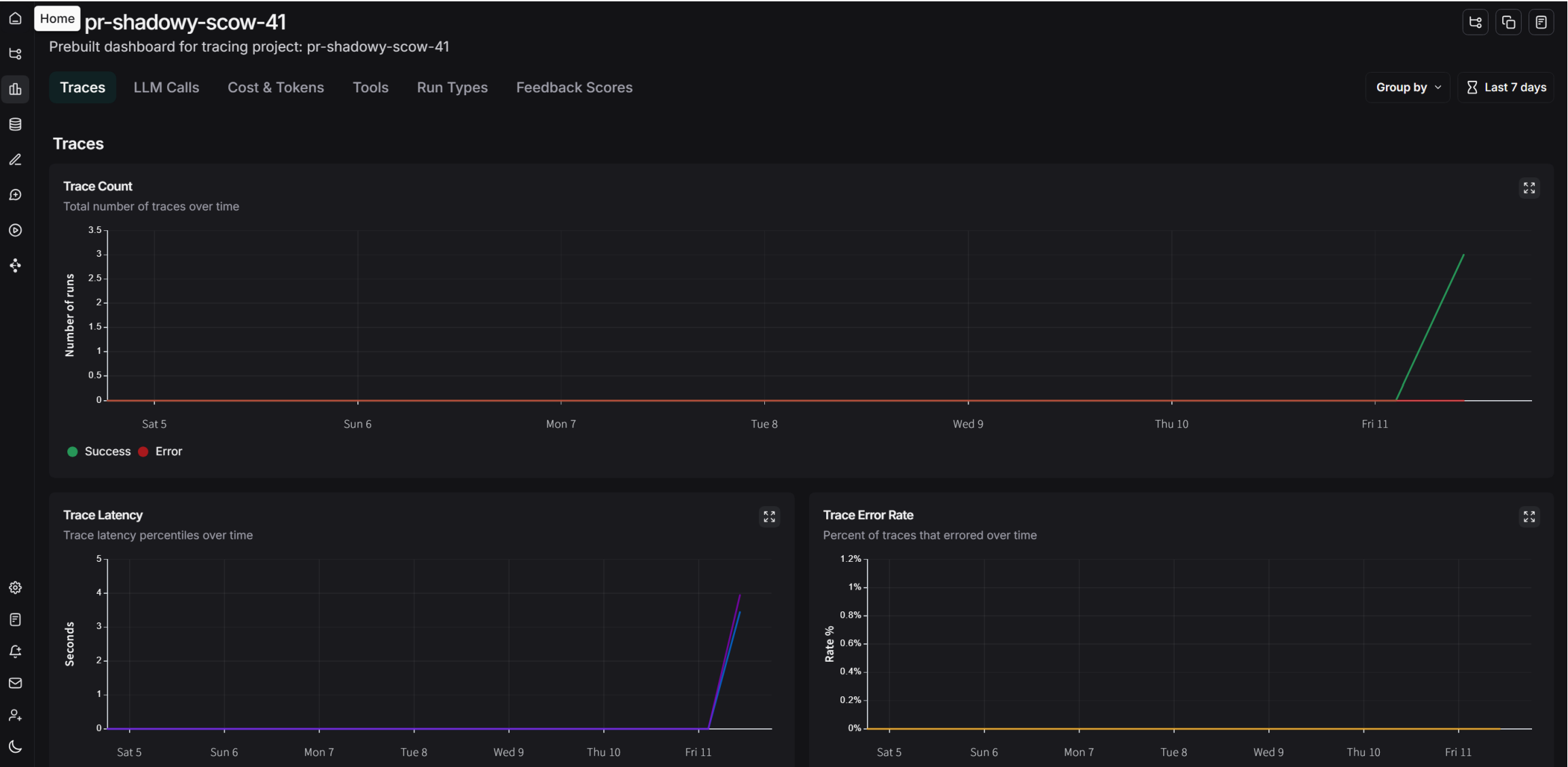
Monitor trends and performance in real time.

Prebuilt DashboardsCustom DashboardsAlerts

Search by name...

Tracing Project Name ↑↓	Most Recent Run (7D) ↓	Run Count (7D)
default	2025. 7. 12. 오전 7:29:20	1
pr-shadowy-scow-41	2025. 7. 11. 오후 6:17:06	3

Monitoring



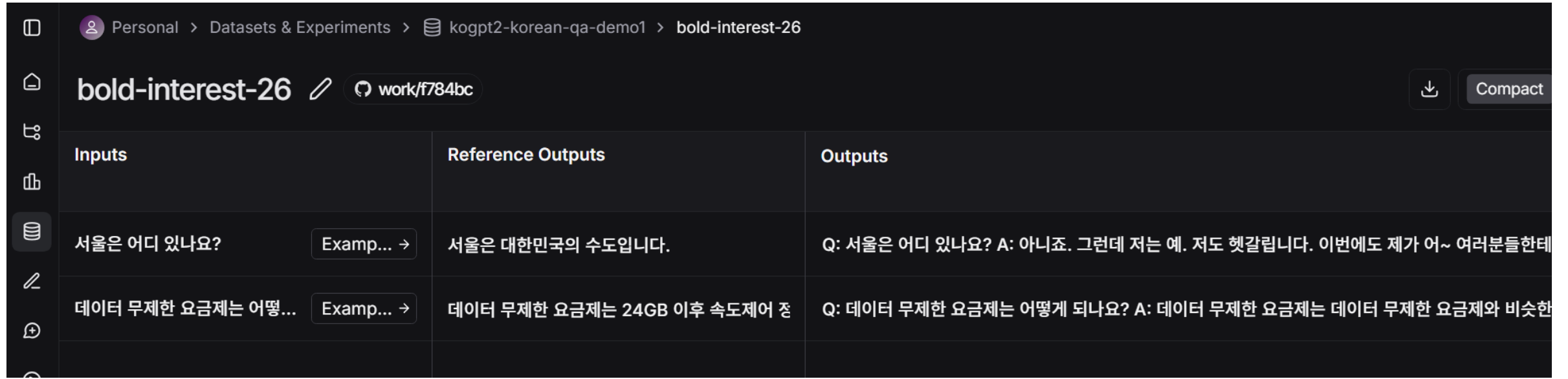
RAG 평가

Section 1. LangSmith



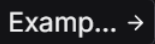
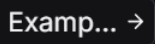
Section 3-1. Evaluation

Evaluation

평가를 할 수 있긴하지만...



The screenshot shows the Google Cloud AI Platform interface. The breadcrumb navigation is: Personal > Datasets & Experiments > kogpt2-korean-qa-demo1 > bold-interest-26. The dataset name 'bold-interest-26' is displayed with an edit icon and a version 'work/f784bc'. There are download and 'Compact' buttons. Below is a table with three columns: Inputs, Reference Outputs, and Outputs. The first row shows an input '서울은 어디 있나요?' with an example button, a reference output '서울은 대한민국의 수도입니다.', and an output 'Q: 서울은 어디 있나요? A: 아니죠. 그런데 저는 예. 저도 헛갈립니다. 이번에도 제가 어~ 여러분들한테'. The second row shows an input '데이터 무제한 요금제는 어떻...' with an example button, a reference output '데이터 무제한 요금제는 24GB 이후 속도제어 쯤', and an output 'Q: 데이터 무제한 요금제는 어떻게 되나요? A: 데이터 무제한 요금제는 데이터 무제한 요금제와 비슷한'.

bold-interest-26  work/f784bc  Compact		
Inputs	Reference Outputs	Outputs
서울은 어디 있나요? 	서울은 대한민국의 수도입니다.	Q: 서울은 어디 있나요? A: 아니죠. 그런데 저는 예. 저도 헛갈립니다. 이번에도 제가 어~ 여러분들한테
데이터 무제한 요금제는 어떻... 	데이터 무제한 요금제는 24GB 이후 속도제어 쯤	Q: 데이터 무제한 요금제는 어떻게 되나요? A: 데이터 무제한 요금제는 데이터 무제한 요금제와 비슷한

- evaluator API가 까다롭고 불투명
- 오직 OpenAI 기반 evaluator에 맞춰져있음
- 커스텀 evaluator 사용 불편
- evaluate_strings와 같은 비직관적 추상 메소드 강제

감사합니다.

Q & A