LangGraph vs LangChain vs LangFlow vs LangSmith : Which One To Use & Why?

FuturMinds
11.4K subscribers

Subscribe

2.5K | Share | Save | Download | ...

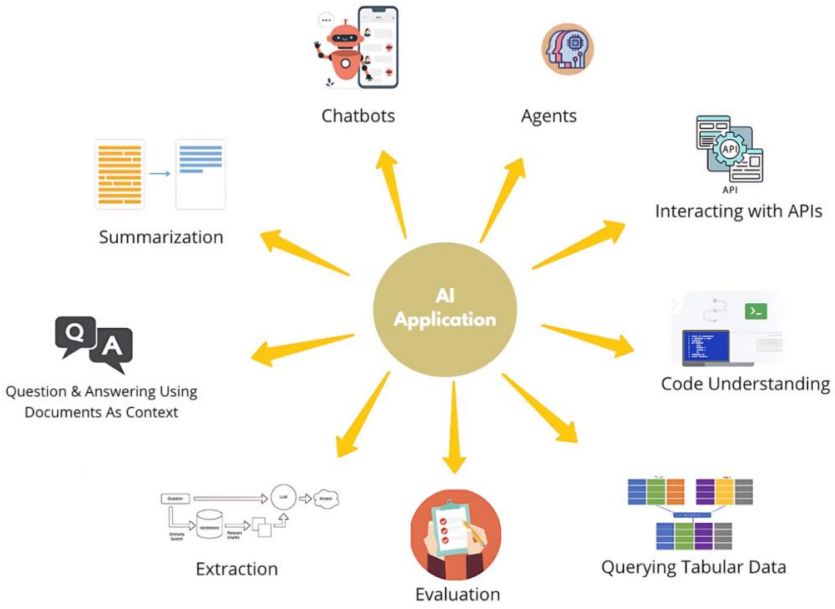
70,256 views Aug 21, 2024

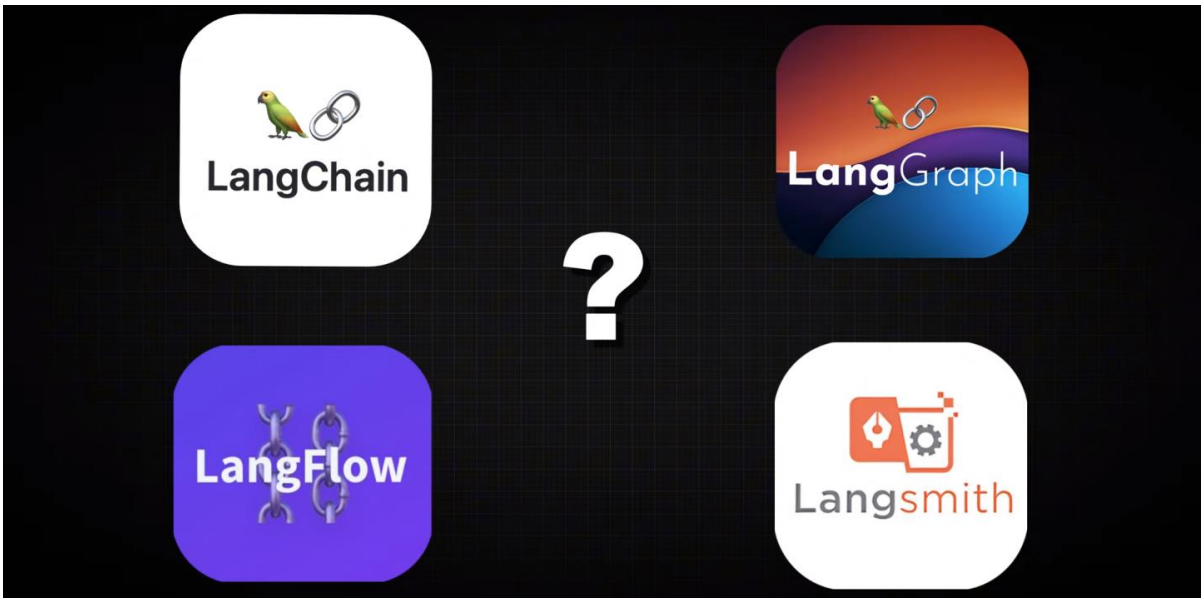
Are you curious about the differences between LangChain, Langflow, Langgraph, and Langsmith? In this video, I'll explain what is LangChain, what is Langflow, what is Langgraph, and what is Langsmith, breaking down each tool's features, use cases, and why they were created.

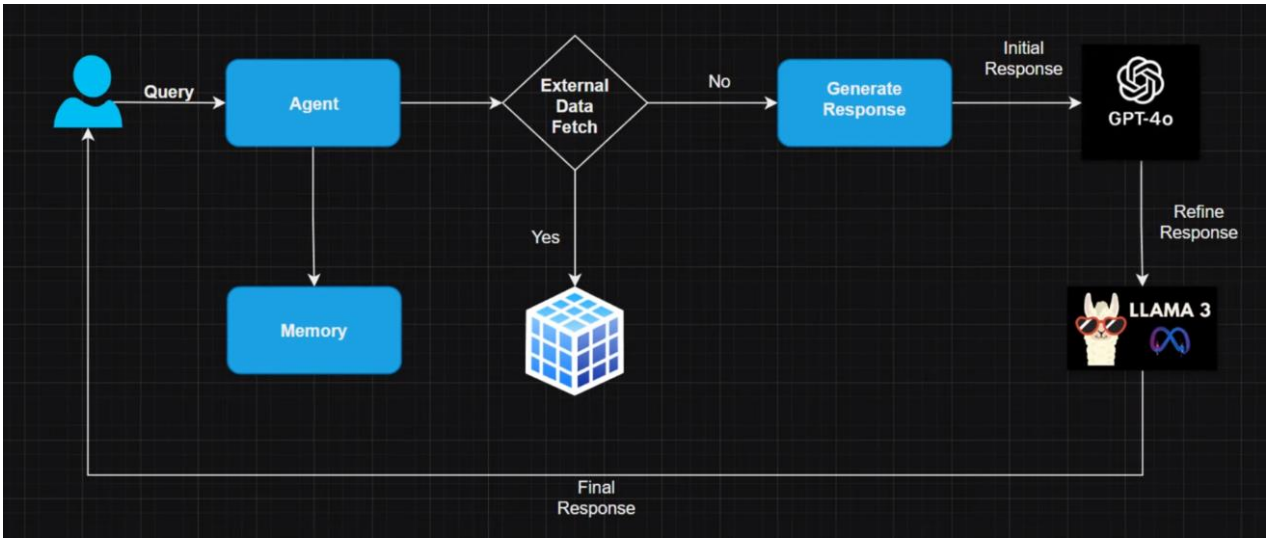
We'll explore what Langsmith is used for and how it integrates with LangChain for monitoring and debugging workflows. Learn how Langgraph simplifies agent management and discover the differences in Langchain vs Langgraph and Langgraph vs Langchain for building AI applications. I also cover the Langchain vs Langflow comparison, showing how Langflow's visual interface can speed up prototyping for non-coders.

If you're wondering about Langchain vs Langsmith, this video also breaks down how Langsmith helps in tracking and optimizing your applications. Whether you're a developer or just starting with AI workflows, this video will clear up the confusion around these tools, helping you understand which one to choose for your next project.

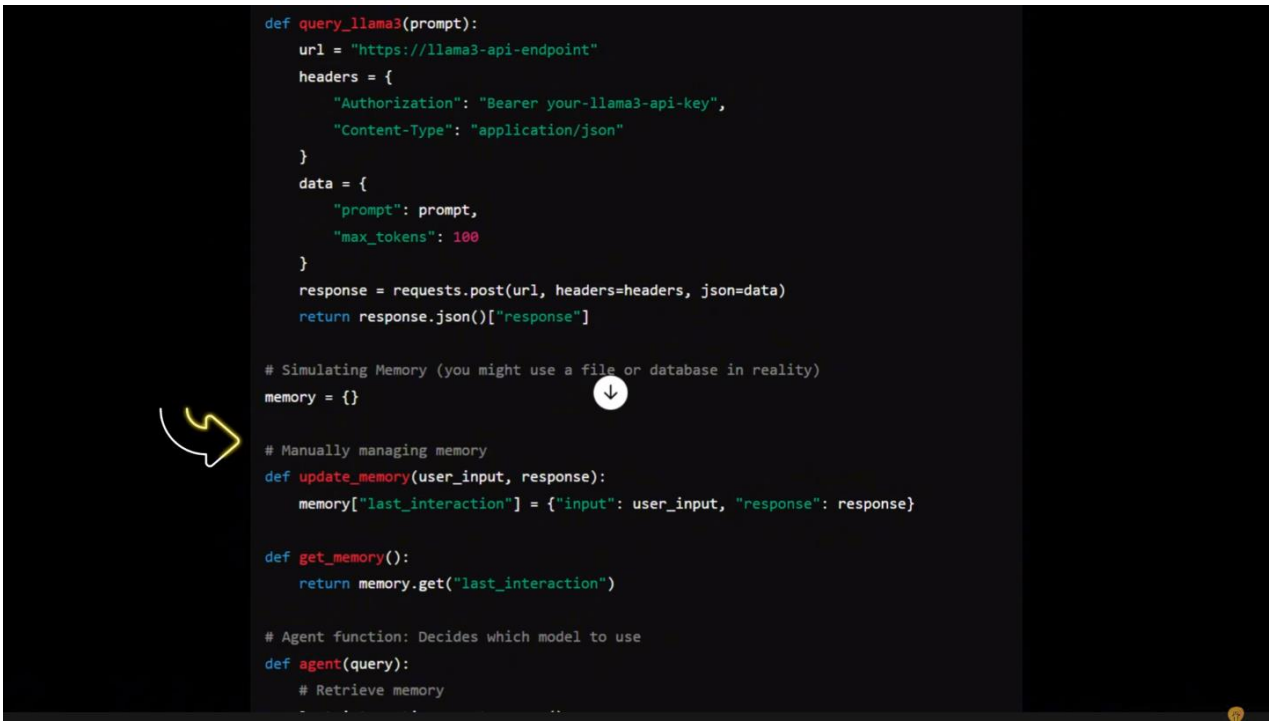
📌 Timestamps:



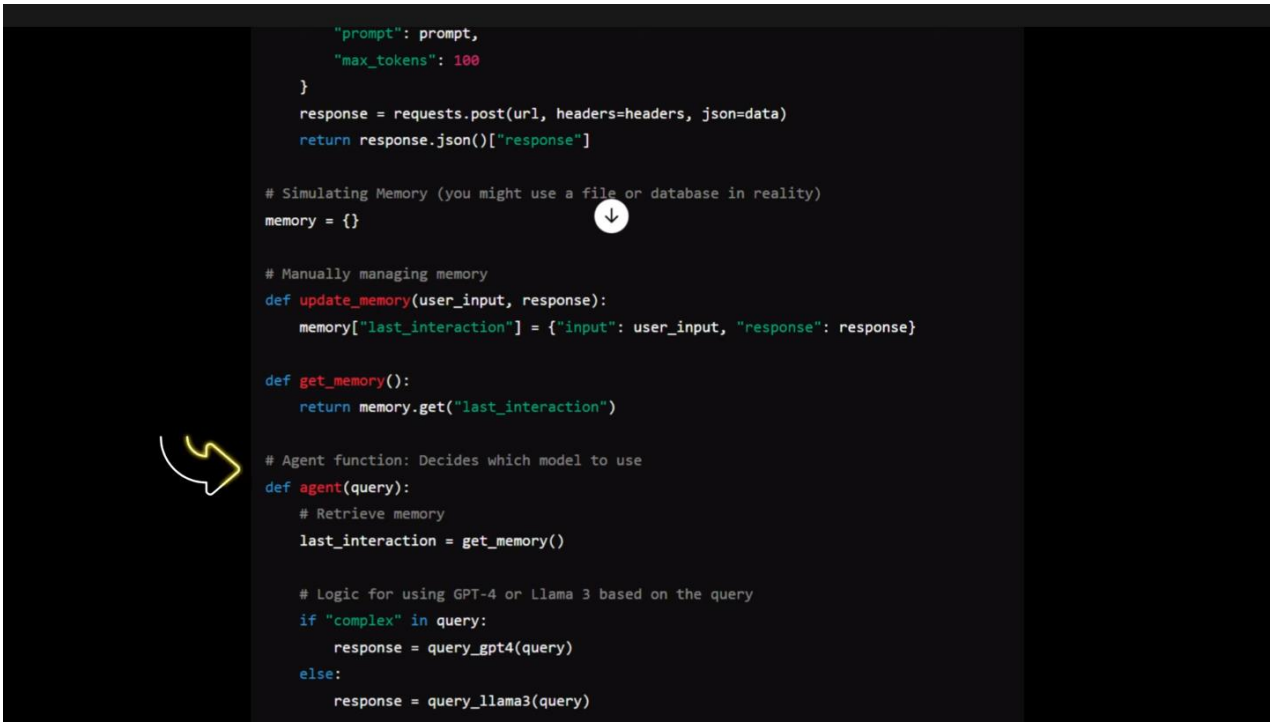





```python
import openai
import requests

# GPT-4 API function
def query_gpt4(prompt):
    openai.api_key = "your-gpt4-api-key"
    response = openai.Completion.create(
        engine="gpt-4",
        prompt=prompt,
        max_tokens=100
    )
    return response.choices[0].text

# Llama 3 API function
def query_llama3(prompt):
    url = "https://llama3-api-endpoint"
    headers = {
        "Authorization": "Bearer your-llama3-api-key",
        "Content-Type": "application/json"
    }
    data = {
        "prompt": prompt,
        "max_tokens": 100
    }
    response = requests.post(url, headers=headers, json=data)
    return response.json()["response"]
```



```python
def query_llama3(prompt):
    url = "https://llama3-api-endpoint"
    headers = {
        "Authorization": "Bearer your-llama3-api-key",
        "Content-Type": "application/json"
    }
    data = {
        "prompt": prompt,
        "max_tokens": 100
    }
    response = requests.post(url, headers=headers, json=data)
    return response.json()["response"]

# Simulating Memory (you might use a file or database in reality)
memory = {}

# Manually managing memory
def update_memory(user_input, response):
    memory["last_interaction"] = {"input": user_input, "response": response}

def get_memory():
    return memory.get("last_interaction")

# Agent function: Decides which model to use
def agent(query):
    # Retrieve memory
```



```python
        "prompt": prompt,
        "max_tokens": 100
    }
    response = requests.post(url, headers=headers, json=data)
    return response.json()["response"]

# Simulating Memory (you might use a file or database in reality)
memory = {}

# Manually managing memory
def update_memory(user_input, response):
    memory["last_interaction"] = {"input": user_input, "response": response}

def get_memory():
    return memory.get("last_interaction")

# Agent function: Decides which model to use
def agent(query):
    # Retrieve memory
    last_interaction = get_memory()

    # Logic for using GPT-4 or Llama 3 based on the query
    if "complex" in query:
        response = query_gpt4(query)
    else:
        response = query_llama3(query)
```

You can write the code to manually do all the above things as above or you can use a framework like LangGraph

Build

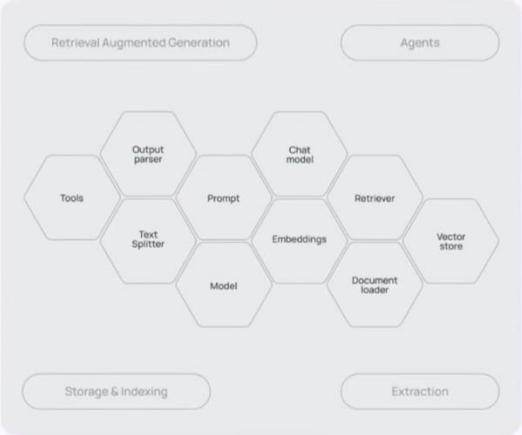
LangChain is a framework to build with LLMs by chaining interoperable components. LangGraph is the framework for building controllable agentic workflows.

Run

Deploy your LLM applications at scale with LangGraph Cloud, our infrastructure purpose-built for agents.

Manage

Debug, collaborate, test, and monitor your LLM app in LangSmith - whether it's built with a LangChain framework or not.



Build your app with LangChain

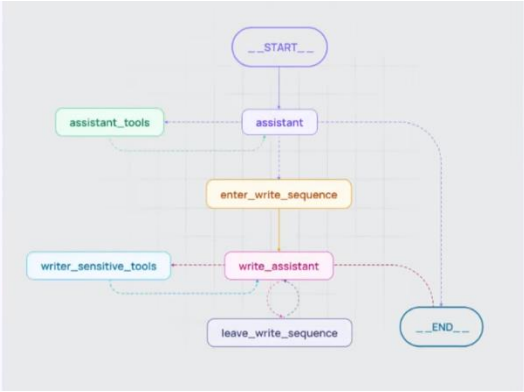
Build context-aware, reasoning applications with LangChain's flexible framework that leverages your company's data and APIs. Future-proof your application by making vendor optionality part of your LLM infrastructure design.

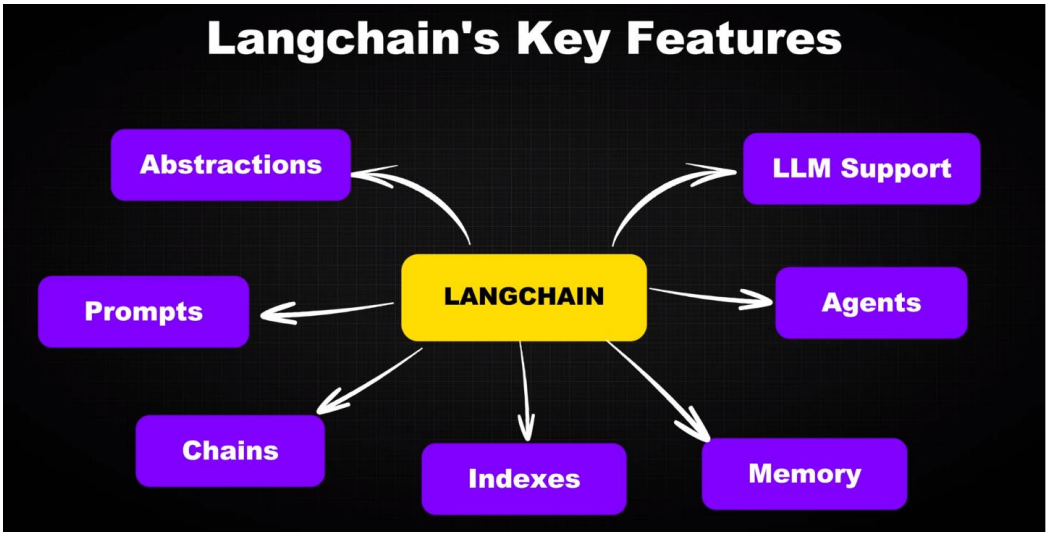
Learn more about LangChain ↗

Run at scale with LangGraph Cloud

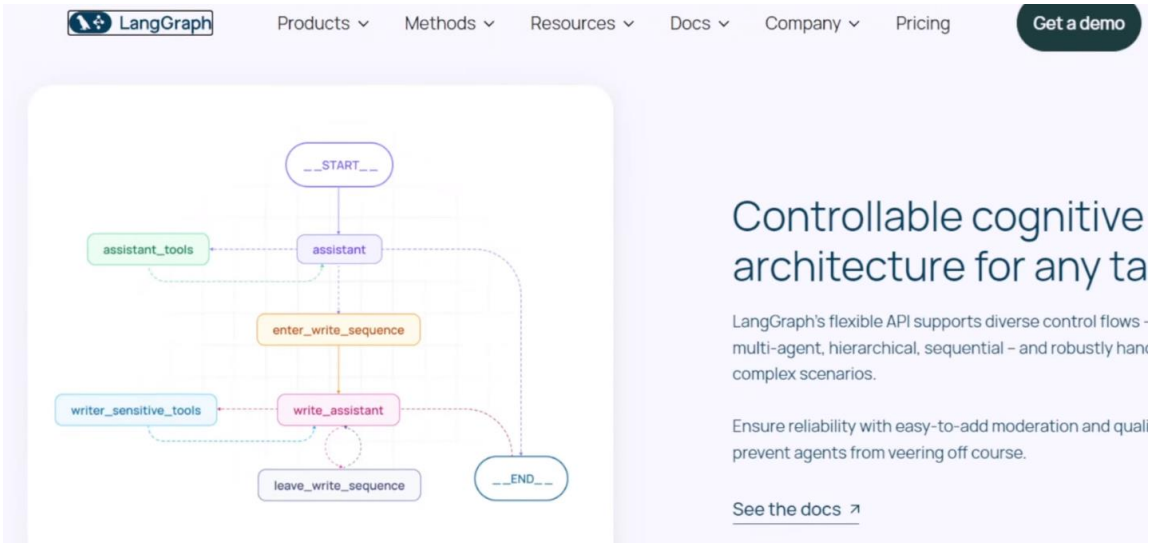
Deploy your LangGraph app with LangGraph Cloud for fault-tolerant scalability - including support for async background jobs, built-in persistence, and distributed task queues.

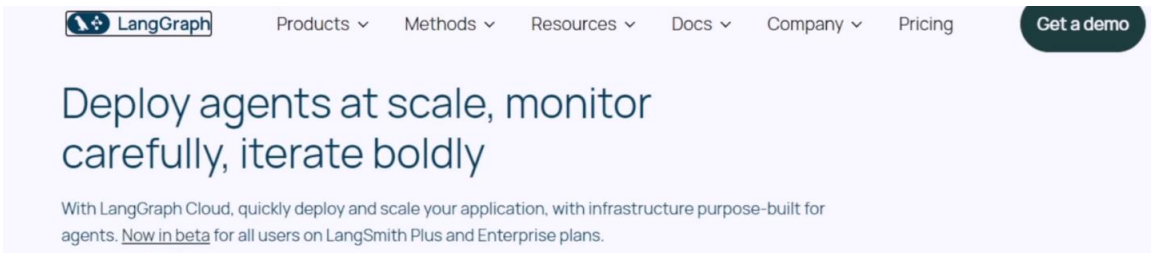
Learn more about LangGraph ↗

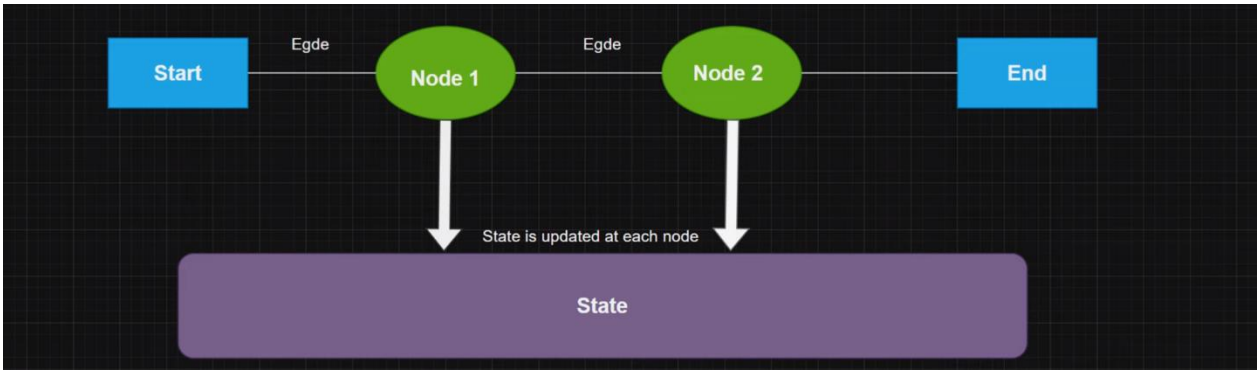




```python
from langchain import OpenAI, LlamaCpp, Memory, Agent
from langchain.chains import SequentialChain

# Step 1: Define the models
gpt4_llm = OpenAI(api_key="your-gpt4-api-key", model="gpt-4")
llama3_llm = LlamaCpp(model_path="path-to-your-llama3-model")

# Step 2: Set up memory
memory = Memory()

# Step 3: Create an agent that decides whether to use GPT-4 or Llama 3
def agent_fn(query, memory):
    last_interaction = memory.load()

    # Simple agent decision: Use GPT-4 for complex queries
    if "complex" in query:
        return gpt4_llm(query)
    else:
        return llama3_llm(query)

agent = Agent(agent_fn, memory=memory)

# Step 4: Set up LangChain to manage the sequence of tasks
chain = SequentialChain([agent], memory=memory)

# Example usage
```







Installation

```
pip install -U langgraph
```

Example

One of the central concepts of LangGraph is state. Each graph execution creates a state that is passed between nodes in the graph as they execute, and each node updates this internal state with its return value after it executes. The way that the graph updates its internal state is defined by either the type of graph chosen or a custom function.

Let's take a look at a simple example of an agent that can use a search tool.

```
pip install langchain-anthropic
```

```
export ANTHROPIC_API_KEY=sk-...
```

Optionally, we can set up LangSmith for best-in-class observability.

```
export LANGSMITH_TRACING=true
export LANGSMITH_API_KEY=lsv2_sk_...
```

```
from typing import Annotated, Literal, TypedDict

from langchain_core.messages import HumanMessage
from langchain_anthropic import ChatAnthropic
from langchain_core.tools import tool
from langgraph.checkpoint.memory import MemorySaver
from langgraph.graph import END, StateGraph, MessagesState
from langgraph.prebuilt import ToolNode
```

```
# Define the tools for the agent to use
@tool
def search(query: str):
    """Call to surf the web."""
    # This is a placeholder, but don't tell the LLM that...
    if "sf" in query.lower() or "san francisco" in query.lower():
        return "It's 60 degrees and foggy."
    return "It's 90 degrees and sunny."


tools = [search]

tool_node = ToolNode(tools)

model = ChatAnthropic(model="claude-3-5-sonnet-20240620", temperature=0).bind_tools(tools)

# Define the function that determines whether to continue or not
def should_continue(state: MessagesState) -> Literal["tools", END]:
    messages = state['messages']
    last_message = messages[-1]
    # If the LLM makes a tool call, then we route to the "tools" node
    if last_message.tool_calls:
```

Install Langflow Locally

> **CAUTION**
> Langflow **requires** Python version 3.10 or greater and pip or pipx to be installed on your system.

Install Langflow with pip:

```
python -m pip install langflow -U
```

Install Langflow with pipx:

```
pipx install langflow --python python3.10 --fetch-missing-python
```

Pipx can fetch the missing Python version for you with `--fetch-missing-python`, but you can also install the Python version manually. Use `--force-reinstall` to ensure you have the latest version of Langflow and its dependencies.

Having a problem?

If you encounter a problem, see Common Installation Issues.

To get help in the Langflow CLI:

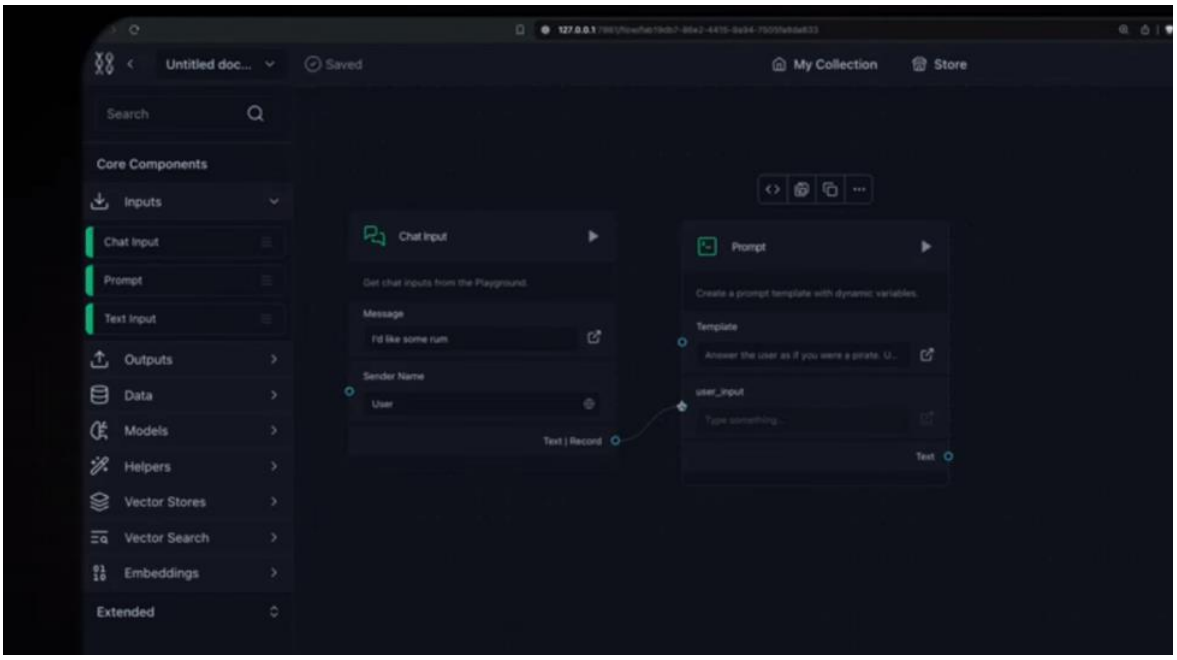
```
python -m langflow --help
```

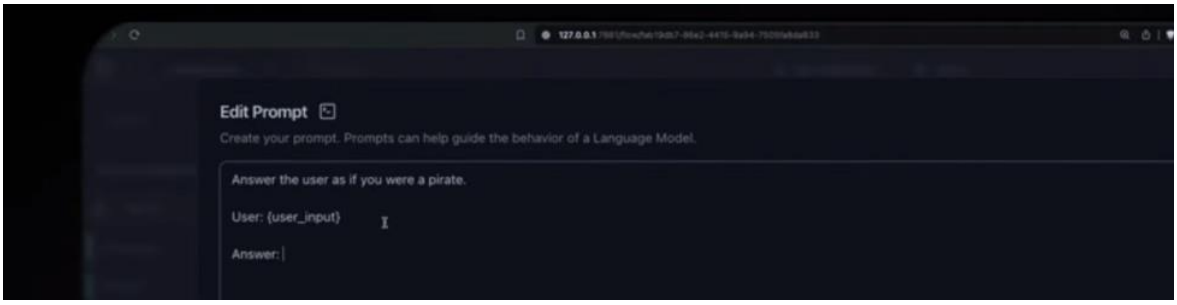
⛓ Run Langflow

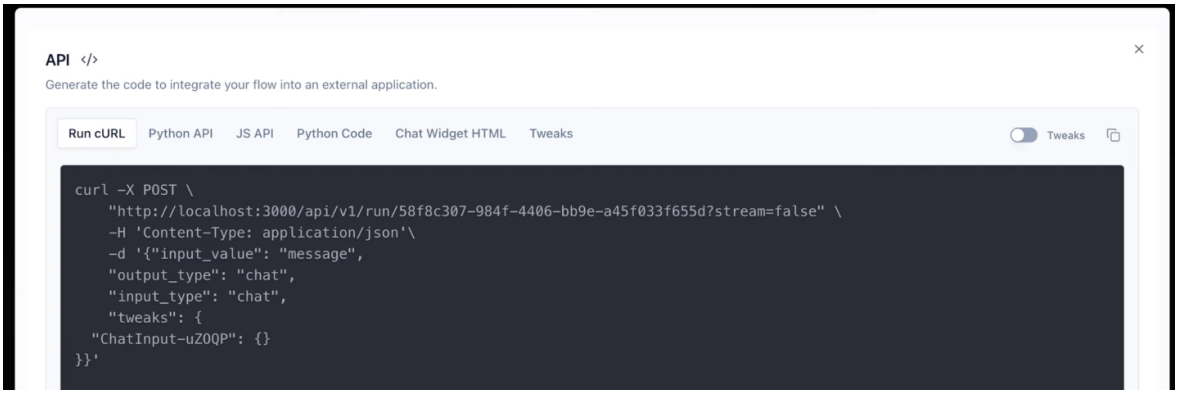
1. To run Langflow, enter the following command.

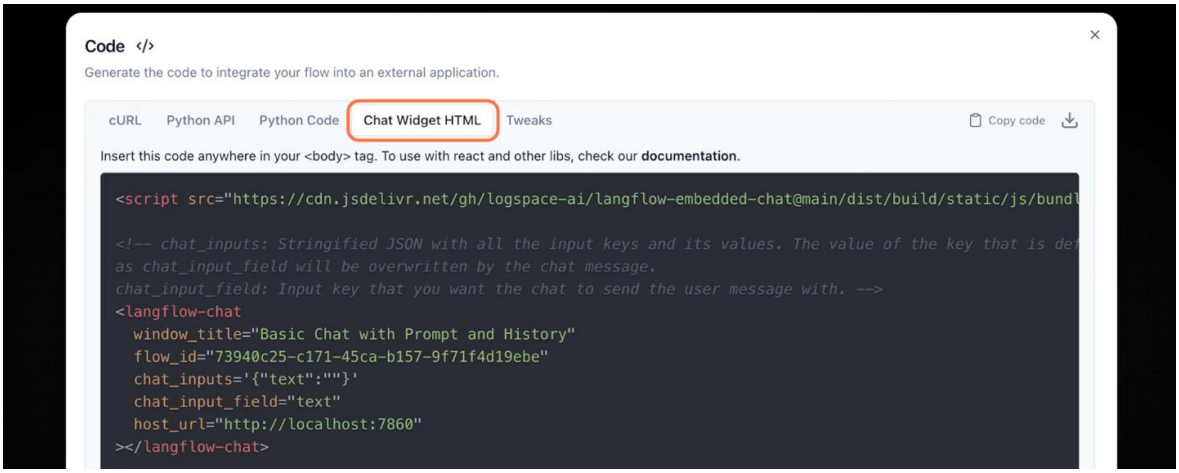
```
python -m langflow run
```

2. Confirm that a local Langflow instance starts by visiting `http://127.0.0.1:7860` in a Chromium-based browser.


Untitled doc...
Saved
My Collection
Store
Search
Core Components
Inputs
Chat Input
Prompt
Text Input
Outputs
Data
Models
Helpers
Vector Stores
Vector Search
Embeddings
Extended
Chat Input
Get chat inputs from the Playground.
Message
I'd like some rum
Sender Name
User
Text | Record
Prompt
Create a prompt template with dynamic variables.
Template
Answer the user as if you were a pirate. U...
user_input
Type something...
Text


Edit Prompt
Create your prompt. Prompts can help guide the behavior of a Language Model.
Answer the user as if you were a pirate.
User: {user_input}
Answer:

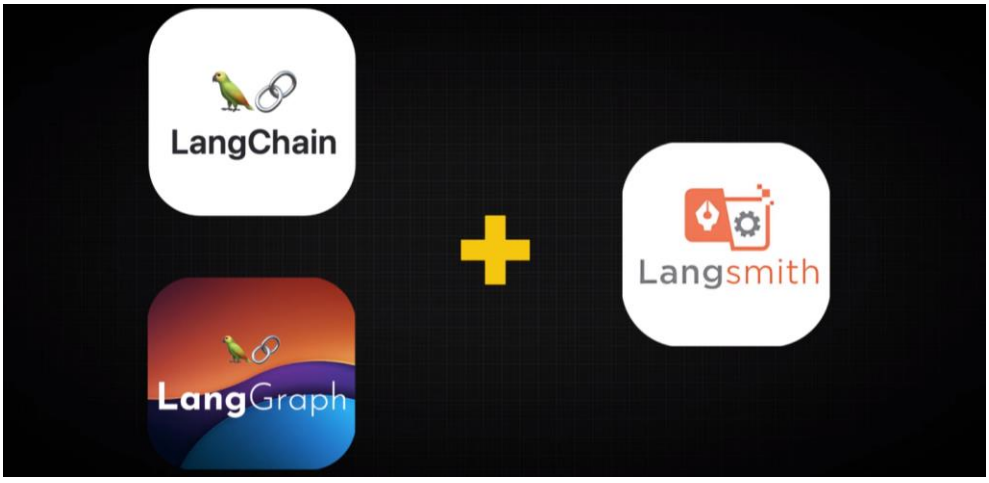

API
Generate the code to integrate your flow into an external application.
Run cURL
Python API
JS API
Python Code
Chat Widget HTML
Tweaks
Tweaks
curl -X POST \
"http://localhost:3000/api/v1/run/58f8c307-984f-4406-bb9e-a45f033f655d?stream=false" \
-H 'Content-Type: application/json'\
-d '{"input_value": "message",
"output_type": "chat",
"input_type": "chat",
"tweaks": {
"ChatInput-uZOQP": {}
}}'


Code
Generate the code to integrate your flow into an external application.
cURL
Python API
Python Code
Chat Widget HTML
Tweaks
Copy code
Insert this code anywhere in your <body> tag. To use with react and other libs, check our documentation.
<script src="https://cdn.jsdelivr.net/gh/logspace-ai/langflow-embedded-chat@main/dist/build/static/js/bundl
<!-- chat_inputs: Stringified JSON with all the input keys and its values. The value of the key that is def
as chat_input_field will be overwritten by the chat message.
chat_input_field: Input key that you want the chat to send the user message with. -->
<langflow-chat
window_title="Basic Chat with Prompt and History"
flow_id="73940c25-c171-45ca-b157-9f71f4d19ebe"
chat_inputs='{"text":""}'
chat_input_field="text"
host_url="http://localhost:7860"
></langflow-chat>

The platform for your LLM development lifecycle

LLM-apps are powerful, but have peculiar characteristics. The non-determinism, coupled with unpredictable, natural language inputs, make for countless ways the system can fall short. Traditional engineering best practices need to be re-imagined for working with LLMs, and LangSmith supports all phases of the development lifecycle.

Sign Up ↗



Comparing Experiments

2 experiments compared by ANSWER_ACCURACY against a Baseline of rag-qa-gpt4-1106-d7a86cae

Input	rag-qa-gpt4-1106-d7a86cae (Baseline)	rag-qa-gpt4o-2cc85e63 (6 ↑ 3 ↓)
I'm passing {'a':1} and want to create an output map of {'a':1,'b':2, 'c':3}. H... Example #2056	In LCEL, you can achieve this by using the `RunnablePassthrough.assign()` method to add additional ke... ANSWER_ACCURACY 0.5 → 10.68s SUCCESS 61,356 $0.617	You can achieve this by using the `RunnablePassthrough` an... ANSWER_ACCURACY 0.5 → 9.18s SUCCESS 66,259
With a RAG chain in LCEL, why are documents retrieved automatically f... Example #3306	In the RAG chain example, when we construct `chain = ({'context': retriever, 'question': RunnablePassthr... ANSWER_ACCURACY 1 → 16.52s SUCCESS 61,512 $0.621	When you construct the chain using `{'context': retriever, 'qu... ANSWER_ACCURACY 1 → 16.59s SUCCESS 66,416
How can I configure the temperature of an LLM when invoking the LCEL... Example #361e	In LCEL, you can configure the temperature of a large language model (LLM) such as GPT-3 using the `... ANSWER_ACCURACY 1 → 33.09s SUCCESS 61,552 $0.623	You can configure the temperature of a language model (LL... ANSWER_ACCURACY 0.8 → 13.99s SUCCESS 66,276
How can I dynamically route logic based on input using a RunnableLam... Example #3a42	In LangChain, you can dynamically route logic based on input by creating a custom function that uses a... ANSWER_ACCURACY 1 → 21.12s SUCCESS 61,459 $0.621	To dynamically route logic based on input using a `Runnable... ANSWER_ACCURACY 1 → 23.05s SUCCESS 66,737
I am passing a map with {'num': 1} to a LCEL chain. How can I add an extr... Example #3b6a	To add an extra key `num2` that adds 1 to the value of `num`, and then assign the new map to a key nam... ANSWER_ACCURACY 0.8 → 17.64s SUCCESS 61,466 $0.62	To achieve this, you can use the `RunnablePassthrough.assi... ANSWER_ACCURACY 0.8 → 16.17s SUCCESS 66,201
I am passing text key 'foo' to my prompt and want to process it with a fu... Example #3ec2	To process text with a custom function in LCEL before feeding it into a prompt, you can use a `Runnable... ANSWER_ACCURACY 0.9 → 18.76s SUCCESS 61,441 $0.619	You can achieve this by introducing a custom function using... ANSWER_ACCURACY 1 → 13.44s SUCCESS 66,287

Get started with LangSmith

LangSmith is a platform for building production-grade LLM applications. It allows you to closely monitor and evaluate your application, so you can ship quickly and with confidence. Use of LangChain is not necessary - LangSmith works on its own!

1. Install LangSmith

Python | TypeScript

```
pip install -U langsmith
```

2. Create an API key

To create an API key head to the Settings page. Then click **Create API Key.**

3. Set up your environment

Shell

```
export LANGCHAIN_TRACING_V2=true
export LANGCHAIN_API_KEY=<your-api-key>

# The below examples use the OpenAI API, though it's not necessary in general
export OPENAI_API_KEY=<your-openai-api-key>
```

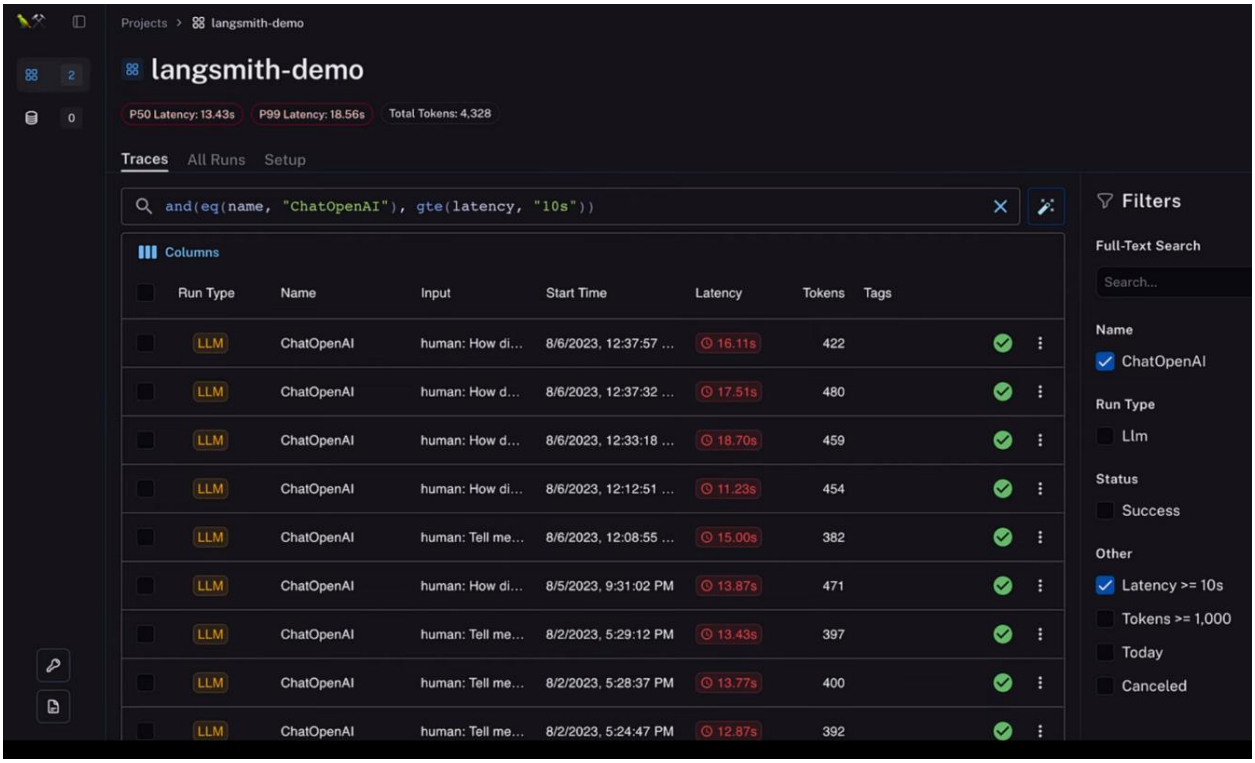
4. Log your first trace

> **TRACING TO LANGSMITH FOR LANGCHAIN USERS**
>
> There is no need to use the LangSmith SDK directly if your application is built entirely on LangChain (either Python and JS).
>
> We've outlined a tracing guide specifically for LangChain users here.

We provide multiple ways to log traces to LangSmith. Below, we'll highlight how to use `traceable`. See more on the Annotate code for tracing page.

Python | TypeScript

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

# Auto-trace LLM calls in-context
client = wrap_openai(openai.Client())

@traceable # Auto-trace this function
def pipeline(user_input: str):
    result = client.chat.completions.create(
        messages=[{"role": "user", "content": user_input}],
        model="gpt-3.5-turbo"
    )
    return result.choices[0].message.content

pipeline("Hello, world!")
# Out:  Hello there! How can I assist you today?
```


Projects > langsmith-demo
langsmith-demo
P50 Latency: 13.43s
P99 Latency: 18.56s
Total Tokens: 4,328
Traces
All Runs
Setup
and(eq(name, "ChatOpenAI"), gte(latency, "10s"))
Columns
Run Type
Name
Input
Start Time
Latency
Tokens
Tags
LLM ChatOpenAI human: How di... 8/6/2023, 12:37:57 ... 16.11s 422
LLM ChatOpenAI human: How d... 8/6/2023, 12:37:32 ... 17.51s 480
LLM ChatOpenAI human: How d... 8/6/2023, 12:33:18 ... 18.70s 459
LLM ChatOpenAI human: How di... 8/6/2023, 12:12:51 ... 11.23s 454
LLM ChatOpenAI human: Tell me... 8/6/2023, 12:08:55 ... 15.00s 382
LLM ChatOpenAI human: How di... 8/5/2023, 9:31:02 PM 13.87s 471
LLM ChatOpenAI human: Tell me... 8/2/2023, 5:29:12 PM 13.43s 397
LLM ChatOpenAI human: Tell me... 8/2/2023, 5:28:37 PM 13.77s 400
LLM ChatOpenAI human: Tell me... 8/2/2023, 5:24:47 PM 12.87s 392
Filters
Full-Text Search
Search...
Name
ChatOpenAI
Run Type
Llm
Status
Success
Other
Latency >= 10s
Tokens >= 1,000
Today
Canceled


LangChain Inc. > Projects > chat-langchain
chat-langchain
See: https://chat.langchain.com/
Traces
LLM Calls
Monitor
Setup
Latency
P50
P95
Trace Latency (s)
LLM Latency (s)
LLM Calls per Trace
Tokens / sec
Metadata