Getting Started with LangSmith (1/8): Tracing

8 MIN · YOUTUBE · HTTPS://WWW.YOUTUBE.COM/WATCH?V=FA9B4D8ISPQ
https://www.youtube.com/watch?v=fA9b4D8IsPQ

SUMMARY

Langmith is an observability and evaluation platform designed for AI applications, allowing users to trace and monitor their applications effectively. In this introduction, Robert demonstrates how to set up Langmith, create tracing projects, and utilize its features to track the performance of an AI application that explains complex concepts in simple terms.

- *Langmith provides a user-friendly interface for monitoring AI applications through tracing projects.*
- *Users must create an account and generate an API key to send traces from their applications to Langmith.*
- *The platform is framework agnostic, enabling users to trace applications regardless of their underlying architecture.*
- *Tracing is initiated by adding a traceable decorator to functions within the application.*
- *Langmith offers detailed insights into application performance, including input/output tracking and step-by-step breakdowns.*
- *Users can visualize latency issues through a waterfall view, helping identify bottlenecks in the application.*
- *Langchain's open-source libraries, Langraph and Lang Train, can simplify the tracing process by working seamlessly with Langmith.*
- *The demo showcases an AI application that answers questions by conducting web searches and summarizing results in an accessible manner.*

Hello, my name is Robert and I'm an engineer at Langjang. Welcome to our intro to Langmith. Langmith is an observability and evaluation platform for AI applications. And today we'll be using Langmith to peak under the hood of application that we've made using a feature called tracing. Let's get started. To use Langmith, the first thing you need to do is create an account. I've already connected my Google account, so I'll be using that.

Once you've logged into Langmith, you should see a screen that looks like this. There's a lot going on here, but don't worry, we'll be covering it all in future videos. For now, let's focus on this tab on the left called tracing projects. A tracing project is just a collection of traces or logs associated with your application. and each project corresponds roughly onetoone with the application that you're building. The natural next question is how can we send traces from our application to Langmith so that they can be recorded in these projects. To do so, you first need to create an API key. You can create one by heading to the settings and clicking this create API key button in the top right. Personal access token is fine.

I've already created some API keys for this demo, so I'll be using that. Next, let's hop over to my Jupyter notebook to see our application. This notebook contains a basic AI application that I've created. It's designed to explain a concept like I'm five. It's going to take in a question, do a web search on the answer, and then

summarize that answer so that even a 5-year-old could understand it, at least theoretically. It's then going to give me that answer so that I can understand it. We're going to use Langsmith to trace this project. And I want to highlight that Langmith is framework agnostic. The Langchain engineering team strongly believes that you should be able to effectively monitor and observe your AI applications no matter how you've built them under the hood. So, we've made Langmith easy to set up and use no matter how you've chosen to build your application. The first step is to import some environment variables into your project. This includes Langmouth tracing, Langmith endpoint, and your Langmith API key. You can also optionally specify the Langmith project you would like to use. My project uses OpenAI and a tool called Tavali to search the web. So, I'm going to import some API keys for them as well. Let's load them all in through AM file.

Next, before we start tracing, let's set up our project. So, I'm going to give our application access to the web and I'm going to define our prompt. You can see that the prompt for our application is that it's an expert in explaining complex topics in a way that's easy to understand. And its job is to answer the provided questions so that even a 5-year-old could understand it. Now, we're ready to actually create our application and trace it with LinkSmith.

So once you've set up linksmith by importing the environment variables, it's very easy to trace your application. The only thing you need to do is to add the traceable decorator to any function that you want to trace. In this case, I have a couple of functions that I want to trace for my application. The first is the web search step where I search the web. The second is the explanation step where I ask OpenAI to summarize the answer. And then finally, I want to trace our entire process from end to end. The full explain like I'm five experience.

Let's put that together and actually test our application. And I'll give it a real question of what is complexity economics. And after running for a little bit, you can see it's come up with a pretty decent answer. It's used simple, friendly terms like Lego blocks to explain the concept of complexity economics. Let's move back to LinkSmith to see what happened in our tracing project. Back in Linksmith, let's check out our projects.

We can see that there's been a new project generated for us, the explain like I'm 5 bot project representing our application. Clicking into it, we can see what happened when we ran our application. There's a new trace that has been generated representing the question that we asked our chatbot. If we want some more detail, let's click in. Here we can see a high-level breakdown of everything that happened when we ran our application. At the top level, we can see the input, which is the question, what is complexity economics?

As well as the final output, which is the answer our AI generated using Lego blocks as an analogy. If you look into the lefth hand sidebar, you can also see a detailed breakdown of every single step that happened within your application. This is known as the run tree and it gives you a breakdown of each individual run that comprises your application's overall flow. For us, there's two steps that we had in our application. A search step where we searched the web and an explanation step where we asked OpenAI to summarize the answers. If you want to see more detail on a particular step, you can click into it as shown. For example, in our OpenAI step, we can now dig into exactly what happened when we called OpenAI. We can see the exact input we provided, including our system prompt and the context we supplied from the web, as well as the AI's output, which was

the final answer we eventually returned to the user.

This isn't the only thing that LinkSmith shows in its tracing capabilities. If you hover over any individual step, you can also see some helpful telemetry on that step. This includes how many tokens that step cost you or the overall latency of that step. And a great way to view latency is through the waterfall view accessible through this button. This gives a nice visual representation of where the highest sources of latency in your project are. For example, if we wanted to make our chatbot a little bit more responsive to the user, we should probably start by looking at this explanation step because the slowest step with the highest latency was our call to open AAI.

That's tracing with Langmith. And next, I want to go back to our project to show how using one of our open-source libraries, Langraph or Lang Train, can make tracing even easier. If you're not familiar, Langchain and Ling Graph are Langchain's open- source libraries for building AI agents. And while they're not required, they do work out of the box with Langmith. If you are using Langraph or Lang Chain, you don't even need to set the traceable decorator in your application. All you need to do is import those same environment variables we imported earlier and Langmith will work out of the box. To demonstrate this, I've created the same application using lang graph. Let's go through the setup steps.

You can see that once I've compiled my application, it follows the exact same flow that we showed earlier. You ask a question, conduct a web search, and then summarize the answers using OpenAI. That final answer is output to the user. Let's see it in action. We're going to ask the same question about complexity economics and we'll be able to view the trace in Langmith afterwards. So, it took a little bit to run, but you can see it's also generated a pretty decent answer to the question, what is complexity economics? Let's take a look one last time back at LinkSmith.

Back in our project, you can see we have a new trace representing our call to our new application implemented using ling graph. If we click in, we can see that we have all the same tracing capabilities. We can see the input, the output, and every step in between. That's tracing with LinkSmith. I want to thank you for joining me today and I'll see you in the next