Tracing Claude Code to LangSmith

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

SUMMARY

Tanish from Langchain introduces the integration of Claude Code with Langsmith, showcasing how users can observe the inner workings of their Claude Code sessions. By tracing each step, users can gain insights into LLM calls and tool interactions, enhancing their understanding of workflow execution.

- *Claude Code integration with Langsmith allows real-time observability of workflows.*
- *Users can trace LLM calls and tool interactions, providing detailed insights into the execution process.*
- *Demonstrated a simple agent that retrieves system prompts and traces the workflow.*
- *Showed how to modify the agent to fetch real-time weather data using an external API.*
- *The integration automatically generates transcripts of user and AI messages, including tool calls.*
- *A custom hook script is used to fetch transcripts and send traces to Langsmith.*
- *Setup involves creating a stop hook file and configuring environment variables in Cloud Code's settings.*
- *Tracing can be enabled on a per-project basis or globally, making it flexible for users.*

Are you curious about what cloud code is doing behind the scenes? Or do you want observability in the critical workflows that you've set up with claude code? Hey, I'm Tanish from Langchain and we built a claude code to LinkSmith integration so that you can see each step that cla takes whether that be an LLM call or tool calls. it's pretty fascinating to see the entire trace. So I want to show you what this looks like. I have a project here. It's a very very very simple agent that I build with link chains create agent.

it's actually just the default agent that I've basically copied from the docs. I want to ask cloud code. I have a cloud code session running here and I want to ask cloud code a question about it. So let's ask what is the system prompt in this file. Pretty simple question.

All right, there we go. And then it it responded with the answer. let's take a look at what that trace looks like because I have the tracing set up to Langmith. And here it is. this is this is as I would expect. So, the first thing here is an LLM call that takes my prompt that I put in, sends it to the LLM, and the LM reasons about what the next steps that claude code should take are. So you can see it it invokes this read file tool.

and then we can see the inputs and outputs of the read file tool as well. So this points to the the file that we were just looking at. and then this is the this is the contents of the file. Kind of interesting. You get the system reminder here as well as part of the tool output. and it warns warns the LLM about potential malware in the file. And the last step here is is taking this we we send the whole chat conversation into into the LM call. But the most important part here is the contents of the file that I asked Claude about. and you can see it responds with the response that we saw in in the CLI. so that was a really easy example, but let's try doing something more

complicated. what I want to do now is right now we have this like hardcoded value that's returned. I actually want claude code to set it up so that I get the actual real-time weather in San Francisco returned instead. And we can do that using this open- source website called open mateo that returns weather in real time. So let's have cloud code use that and return the weather in San Francisco in real time.

I'll type in my prompt. All right. So these are my instructions to claude code. I'm going to send this through and we'll come back once it's done generating the code. All right, looks like that finished up and this looks as I would expect it to. let's take a look at the resulting trace to see how cloud code actually did this.

So, okay, there we go. This is the prompt that I had asked it and provided this website as well. we can see that I actually like looking through the the in visual choices here. I can see that the first thing the first the first thing was an LLM call and so it planned out how it was going to how it was going to generate the results here. so the first thing was in order to update the get weather function, it's going to make a web fetch call to the website that I provided to understand how the API endpoint worked. And so this is the web fetch call that was made. you can see that it returns returns the API endpoint here.

Cool. And then next is the call to claude that takes the data from the web fetch call as well as the rest of the conversation history and it generates code for what the the API call should look like what the fun you know how to update that function. and then this is the edit tool which actually goes in and edits the file and then I think actually the yeah so so then it returns actually what the the new file contents is and then the last step is cloud code cloud does a check and says okay this looks good return back to the user.

So pretty cool to peel the curtain and look behind the scenes there. I want to actually cover a little bit of how this works as well, how this integration works. So these are like the main components here. so two things to call out. So claude code automatically generates transcripts of your conversation history with it. So this is an example of of what part of a transcript looks like. But the key thing to to note is that we get user messages kind of both details about the message itself and the contents of the message and likewise we get AI messages as well as tool calls in this format. So we can use this data to create traces in lang. And then the second piece is cloud code has a feature called hooks u one of which is a stop hook that is triggered after cloud code responds.

And so we've used a combination of these two in this custom hook hook script that we've created that gets triggered by the stop hook and then goes and fetches the transcripts from from the files that cloud code generates and then goes and then sends the traces to linksmith in the expected format. So it's a pretty neat integration and very lightweight to run. I'll go over a little bit of the details on how this is set up as well. It's pretty easy to set up. So, two things I have, you you can access the documentation. we'll link it down below. first thing is this stop hook file. and this contains the core logic for how to parse the transcripts from from cloud code and send them to Langsmith. And then the second thing you need to set up is this global hook which basically is a stop hook that points to the file that we just created.

And then you can enable tracing either on a per cloud code project. So cloud code project is a repo that you have cloud code initialized in or a global basis. and really it's as simple as adding adding a few environment variables

to cloud codes local settings.local.json JSON file. so you need your Linksmith API key. You want the project that you want to trace to, and then if you want to turn it off at any time, you can always set this to to false. So that that's really it for for the tracing. You set up the hook script as well as the global hook at a global level for Cloud Code across your machine and then you can enable tracing optionally per project.

So, that's really all there is to it. it's super interesting to peel back the layers and see what Cloud Code is doing. and I hope you give it a try.