LangSmith Studio: The first agent IDE

8 MIN · YOUTUBE · HTTPS://WWW.YOUTUBE.COM/WATCH?V=PLPJOFVQ4_M
https://www.youtube.com/watch?v=pLPJoFvq4_M

SUMMARY

Lang Graph Studio is a new integrated development environment (IDE) designed for building complex agentic applications on top of the Lang Graph framework. It facilitates the development process by providing visualization tools, debugging capabilities, and interactive features that enhance the coding experience, particularly when working with language models.

- *Lang Graph Studio is built on the Lang Graph framework, aimed at creating agentic applications.*
- *The IDE allows for local development and requires Python for configuration.*
- *Users can visualize agent structures, including nodes and branches, enhancing understanding of complex systems.*
- *It supports real-time interaction with agents, allowing users to modify states and simulate different scenarios on-the-fly.*
- *Debugging features enable step-by-step execution, providing insights into agent behavior at each node.*
- *Users can perform prompt engineering directly within the IDE, facilitating rapid iterations without restarting the entire agent.*
- *The tool is designed to accelerate the development cycle for applications utilizing language models.*
- *Feedback and feature requests from users are encouraged to further enhance the platform.*

today I'm really excited to announce and show off lang graph studio which is the first agent ID and so we've built this to work on top of Lang graph Lang graph is a framework that we released a few months ago and I've been working on for over a year that is aimed at building complex agentic applications it is extremely low-level it's extremely controllable it comes with a built-in persistence layer and we've seen it really accelerate the pace at which team

can build and productionize these agentic types of applications building these applications is often very code heavy but there's new elements to it that aren't present in traditional software engineering you're interacting with llms it's a much more iterative process You're Building these agents which are these complex systems it's nice to be able to visualize that and and so we've been working on something we think complements the the code development experience and we're calling it langra Studio

and it's really an IDE for building these types of agentic applications so I want to show it off and show how you can use it so after you download it you can open it up it's a desktop app it runs locally so I can open it up and I need to log into link Smith so right now everyone even with free accounts gets access to lingraph studio so let me open it back up and I can now see that I

can basically open up different folders so what's in these folders you'll need a l graph graph definition and that will need to be available in a python file currently this only works with python and then there'll be a few other configuration things you need to set up you can either set this up as part of the UI experience or you can put this as a Json file that you then load so let's go to L graph example Lang graph example

is based off of a repo you can find it online it's a GitHub repo and if we look at what it contains if we go there we can see that we have this agent. piy file so this is where our agent is defined and we can see that down at the bottom we have this graph which is a compiled agent so we've got this agent uhp file with the agent in it and then we also have this Lang graph.

Json file and this is what we can either configure in the UI or we can just put in Json and so we have three things in here we have the dependencies so this points the dependencies and it basically says install everything in this folder there's a requirements. text in this folder so we see that and we smartly pull that in and build the agent environment with the requirements in this requirement. text so some Lang chain Imports and then toil which we'll

use for search we have this this graphs thing which points to agent. Pi and then the graph variable this basically tells L graph Studio where to find the agent you can deploy multiple agents here if you want as well and then we have this environment file so this points to environment variables that we're going to need to load to run this agent and so if we look at the example environment file we can see that we need three keys

we need an anthropic API key we need an open AI API key and then we need a tavil API key and so these two anthropic and open AI are for the language models that we're going to use and tavil is for the search engine so if we go back to L graph Studio we can see that we've now loaded up the agent and let me move this up here and so the first thing that we see is this nice visualization of the

agent we can see the nodes that are defined we can see where it starts we can see where it ends we can see the different branches that come out of each node and this is a great way to visualize any type of complex agent that you're building we can then start to interact with this agent so let's send in a message let's say hi what's the weather in SF so the agent starts running and it sees and it realizes that needs to call

toil so it calls toil with a search term of weather in San Francisco it gets back a response and then it gets out a and then it generates a final answer so this is a run of an agent already this is helpful because I can see as the agent is progressing what exactly is happening I can see the streaming of tokens back I can then see when it's finished I can see the tool call I can see exactly what

the tool call is it's a nice and explorable way so I can click in I can see the results of the tool call I can see the tool parameters represented nicely this is already helpful for just understanding what's happening live as the agent is running but what's really cool is I can then start to interact with this so one thing I can do is I can modify the state of the agent directly so if I wanted to do some simulation

where I said something like okay let me edit this and let's say what would have happened if the agent instead generated weather in San Francisco AI weather or something like this then I can click so I can modify it directly there and I can then click this Fork button that'll then start up a separate parallel thread where it continues from this spot in time as if it had searched for this term so this is a simple example there's only three

steps but as your agents get longer and longer you can imagine how nice it is to be able to do this halfway through an agent run you can also run this graph in a sort of debug mode so let's see what that means so I can click here I can create a new thread let me create the same input so what's the weather in SF and I can go up here to interrupts I can click interrupt on all and this will

basically mean before every node it stops so let's click submit we can see that after the first node it's already asking me to continue so I click continue and it now generates the agent node it then stops and so if I wanted to modify it here I could I could edit it and then I can click continue runs the action node this is going to make the call and then after that I need to click continue again and it keeps on

continuing so this is used useful for being able to walk through the agent step by step there's another really cool part that we can do and this is where we modify the underlying code so let's say that I wanted to do some prompt engineering let's say I wanted the final response to always be in Spanish so I could you know if without this I would modify the prompt and I would then rerun the agent from scratch

which is fine but again for long running agents that's not the fastest iteration cycle you want to really drill out on what would have happened with this node exactly if I reran with a different prompt so let's see how we can do that I'll open up my code editor and I have a nice little system prompt right here and this says just right now be a helpful assistant let me add some new instructions when you are responding

to the user respond in Spanish so I modified The Prompt I can now go back back to langra Studio I can see that it's reloading it's now finished reloading I can go back here and I can click this fork and retry button so this will rerun this node with the updated code so I don't have to simulate the first two calls anymore I can say hey let's let's keep these nodes consistent exactly what happened there I want to

exactly happen but I just want to rerun this node and so I rerun it if I want to change it again let's now change this to Old English so I can change that I can go back here reloads very quickly I can click this fork and retry button and it responds and now it's talking in Old English in the olden tongue so this shows off really cool human in the loop interaction patterns that I think are really really helpful

when developing agentic applications llms represent a new component of applications that we're all building and so so we deserve Frameworks to help build those types of applications and that's what we've built with Lang graph but we also deserve tooling and editors to help interact and develop these as we're going along and that's what we've built with langra Studio I've been using this for the past week I think it's incredibly helpful for speeding up the development speed of

these more complex types of applications I think there are a lot of other features that we can add in here and I'm really excited to see what everyone builds with it how they use it and the feature requests that they have so please try it out and then let us know any feedback any feature requests this is one of the coolest things that we've launched and I'm really excited for everyone to try it out thanks