Inside LangSmith's No Code Agent Builder

19 MIN · YOUTUBE · HTTPS://WWW.YOUTUBE.COM/WATCH?V=YVINK_ZIRT0
https://www.youtube.com/watch?v=yVinK_ZIrt0

SUMMARY

Lang Chain introduces a no-code agent builder that allows users to create automated agents quickly by simply describing their needs in natural language. This new platform aims to democratize the development of agents, making it accessible for both technical and non-technical users, while leveraging a powerful underlying architecture called deep agents.

- *The no-code agent builder enables users to create agents by providing a brief description, eliminating the need for technical expertise.*
- *Deep agents architecture allows for flexible and powerful agent creation, supporting both simple tasks and complex multi-agent systems.*
- *The platform features a user-friendly UI/UX that abstracts away complex prompting, making it easier for users to interact with and refine their agents.*
- *Agents can run autonomously in the background, responding to triggers such as new emails or Slack messages, while also allowing for human oversight through an "interrupt" feature.*
- *Sub-agents can be utilized for long-running or context-intensive tasks, improving efficiency by delegating complex operations.*
- *The platform supports iterative development, enabling users to test and refine their agents through a chat interface.*
- *Feedback from users is crucial for optimizing agent performance and enhancing the overall user experience.*
- *Lang Chain aims to provide a modular platform where users can bring their own tools and triggers, fostering a customizable environment for agent development.*

There's so many agents and workflows and automations that that I want for myself, but I would never spend the time or I don't have the time to build them out. But with agent builder, I can spend a minute to describe what I want and then instantly get an agent that that can run for me. At Lang Chain, we've historically focused on developer tools. That's what we're known for. And so that's why it's pretty exciting to be here today talking about something different out of our wheelhouse, a noode agent builder. And so in this conversation, we're gonna kind of like walk through why we built it, some of the features of it, and and go into all that detail. My name is Harrison. I'm the co-founder CEO, joined here by Bryce and Sam, who led a lot of this development effort. So, do you guys want to introduce yourselves quickly?

>> Yeah. my name is Bryce. I lead the Appi team at Langchain. Historically, that's meant internal agents and workflows. but now we're starting to work on this noode platform to power the enterprise. >> I'm Sam. I'm a product manager at Langchain. I work on Langraph platform which is our deployment platform and runtime for agents but also working on new efforts around agent security and no code. >> All right. So as said historically we've focused on building more procode tools for technical developers. Why did we decide to create this no

code agent builder?

>> Yeah like like you said that's the history of the company. Pretty long history of of building technical >> three years not that long. >> True technical tools for technical people to build agents in production. but we we found sort of a common set of of patterns emerge over time and we find that those patterns can basically be distilled into agentic architectures and once you have actually the agentic architecture defined you can easily allow non-technical people to build new agents via configuration on top of those architectures and so that looks like taking an architecture like deep agents which this is built on and adding your own tools adding your own off adding your own system prompt and you make it really easy to build a flexible and powerful agent >> since you mentioned kind of like the architecture and deep agents like what is deep agents what is this architecture that we're building on top of >> I think deep agents at a high level is what we've seen in many popular autonomous agents right so cloud code manis codeex they're basically react agents right tools and system prompts but with a slight spin where we give it this like sub aents concept which itself is just tools and system prompts so we can delegate off longunning tasks to these sub agents they can go manage do their research and then prov provide a concise response to the parent agent. And then we also give it tools for to-do lists and file system. So file system is used for memory. We see in like cloud code, it uses the file system. These models are clearly being tuned to work well with the file system. And the same with to-do list.

if you provide, you know, these these agents a structure for the task it should execute. In our case, this to-do list tool, they tend to do better and follow that structure. >> Awesome. So we've got this core kind of like agent architecture which we now think people can build on top of in a in a no code way. There's a bunch of different ways to do no code. How did you guys think about the UIUX for this particular noode product?

>> Yeah, I think it does come down to the deep agents package. There there's different ways to approach it. Like you said, there are more like workflow builder type solutions that are more deterministic and they have a defined flow and and that is very powerful for building exactly the flow that you want. But a lot of times agents in production need to be able to react on the fly to new information. And deep agents is basically just like we're describing as sort of a loop with these tools.

taking in information having a system prompt and so that very flexible architecture allows sort of two things. The agents to be more powerful in prod because it can respond to new info on the fly but also super easy to build because all you have to do effectively is say sort of what tools do I want? What sub agents do I want optionally? And what instructions should I give the agent? >> Yeah, I think that's kind of what we want to emphasize here is is you know we've been building agents for the last what three years. we've seen everything from the, you know, most trivial React agents, which is tools and prompts, to like highly complex multi-agent systems. and that's what's led us to deep agents is this accumulation of everything from the simplest agents to the most complex.

and we've managed to put them into this deep agent package. And we think that it's going to be really easy for anyone with no technical experience to build agents because, this really powerful architecture, which is deep agents, under the hood is actually fairly simple, right? It's just tools and prompts, but then specialized tools and

prompts. yeah. >> So, so this is like a pretty simple, you described it as a configuration file, and it could just be like a string and and and a list of tools, but but there's more than that in the UIUX. Like there was that kind of like way of chatting to create in like why that like you could just have like a Google form, right, and have someone like fill that out, but it's not that. Like why?

>> Yeah. I think the big thing here is is most people don't know how to prompt well, right? Even people who are technical are not great prompters. >> I'm not a great prompter. >> Yeah. you know, we all struggle with prompting and they're also just tedious to write. You know, like look at the cloud code prompt. It's what 10,000 tokens long. So nobody wants to write a full prompt. and what we found is is we can have we can, you know, abstract that away to the LLM. So that what the user does is they just provide a natural language description of their agent and then an LLM does all the busy work of writing out the system prompt, picking what tools to use, deciding if it needs a sub agent or not. and you know just using natural language to do all this makes it much easier on both the nontechnical user and the technical user.

>> I think an interesting part here as well talking about like prompts and building prompts. It's a very iterative process. how and and here you've kind of got a platform where you both build the agents and use the agents and there's kind of like some interesting you know, like you might want to iterate on it in both scenarios like you might be using it and then want to iterate on it. might be like building it and the the first meta prompt comes out incorrect and you might want to iterate on it like how like how did you guys think about that and and what are some still open questions there if there are any?

>> Yeah, that was clearly very obvious from the get-go was both paradigms are important and sort of the paradigm of initial creation allowing them to use natural language making it very obvious how they could do that. but also updating the agent over time. in terms of interacting with the agent, chat is is something that people have gotten very used to with with products like chatbt becoming becoming pretty ubiquitous. And so, the main way to in initially test your agent, as we saw in the demo, is to just chat at it and see how it responds, see what tools it calls. Then you as the human user can very easily understand what is it doing.

you can see that and you can start to think, you know, what maybe in the instructions would allow it to improve what it's doing and and follow a better path. from them. You can easily update that in the editor that we've created. >> This is maybe a little bit philosophical, but like prompt instructions memory. Are these the same thing? >> It's a good question. we've been debating as to whether or not we should have the system prompt be stored in memory and kind of have it work like DSPI where it's constantly updating its own system prompt. so yeah, that's I think that's something we're experimenting with, whether it should be a hard-coded prompt that only change when the user explicitly tells it to or if the agent should be able to, you know, constantly update its own prompt on the fly based on whatever feedback it gets from the user.

>> What I mean, what what do you think? sort of prompt versus memory. How do you think about the intersection? >> I don't know if I have a super strong take right now. I think I think one really interesting way to view memory is just like as a file system. And I think a prompt is a file. And so I think that makes it very very natural to put in there. I think you know memory you can also think of more complex data structures that go

beyond a prompt. I think there's also if we think about sharing agents right like if I if I create an agent and share it I think there is if if I update the base instructions I probably want that to be reflected with the agent I share with you. But if you interact with the agent or I interact with the agent and it starts to remember things that are different than the base instructions, those probably shouldn't be shared, right? So I do think there is some difference. I don't 100% know like yet like what that means technically or or what the barrier exactly is. so I I I don't know. I think I think that's probably one of the more interesting things to to figure out. Speaking of like figuring things out, this is a new kind of like platform for building agents. A new set of agents are being built on top of it. What have you learned about how people want to use their agents once they've built it?

>> Yeah, I mean part of how we've developed this product so far is by dog fooding it within Langchain pretty extensively. so we have as of today almost almost 100 agents running on this platform used by the employees at our company. to to start most people want to do pretty simple things. They want an assistant for their email inbox. They want an assistant for some Slack channel they have commonly have to use. so a lot of them in in the early days look sort of like simple productivity type assistants. Over time, even though it's a new product, we're starting to see more complex sort of like multi-turn agents being brought to production, but also agents that take action, that send send messages, that send emails, that create linear tickets or labels or whatever it might be.

>> Yeah, I think that, you know, the the most common way people want to interact with these agents is through chat. But then there's also instances where people don't want to interact with their agent. They want to create their agent, set up some sort of autonomous trigger, right? when you get a new email or a Slack message and have it run for them in the background. So, we have both, right? We have chat where you can go and if you have a question you ask your agent, you send it. but then it can also just run for you in the background. That way, you know, you create your agent once and then, you know, maybe you check your inbox once a day or you get a Slack DM and the agent is always active and running for you in the background.

>> So, okay. So, on on that note, you mentioned triggers. That's a new thing that's new in this platform. Like, what are triggers? Why do we introduce them? What are some use cases? And then kind of related to that and with this idea of kind of like balancing autonomy with like human oversight, there are a bunch of tools that require there's this human in the loop aspect to a lot of these tools like requiring approval editing.

Can you talk about why we have that and how that balances also with some of these triggers and and all of this like interaction between autonomy and running in the background but also still having the human in the loop and and how to balance all of that. Yeah, we have found that some people do just want agents that they can chat to like chat GBT, but more often than not, people are starting to want ambient agents, which is a concept we've been talking about for a few months now of sort of these agents that run in the background.

>> And are those are those like workflows basically that they're trying to automate? Like is that a way to think about it or are they more broad even? >> I think a lot of the time they are workflows. They are sort of, you know, receive email, see if he wants to ignore it via a mark Gmail mark as red tool or respond to it. but a lot of the times they are something that should just sort of like run daily and you know pull in a bunch of information from various sources and then decide how to act and the action that it will take will be different each time. So I

think it's sort of a combination of workflows but also something much more flexible than that.

>> Yeah, I think we have workflows and that's going to be the most common you know task people implement like read my emails and then decide if you you should respond or not. but because deep agents is so good at handling these simple tasks right like reading and responding to emails. but because they have this sub agents concept, right, this like worker concept where you can send the more complex tasks off to have it handle that and then returned with this concise response. They're also really good at handling like agentic tasks, right? Where you get an email and maybe responds or maybe goes and does research and then responds. And deep agents allows you to do the this wide range of tasks all within like the same simple agent.

>> Well, okay. So, that's an interesting point. Sub agents, when should I use them? Yeah, I think sub aents are best utilized when you have a longunning or context inensive task, right? So research where it needs to go and search over 5, 10, 100 different websites, that's going to be a lot of tokens. You don't necessarily need the parent agent to know about every single one of those web pages. So instead, you delegate it off to a sub agent. It goes it it reads the million different tokens from all these different websites, generates a concise report with all the answers that the parent agent asks, and then the parent agent only sees that final report with the answers it cares about, and it doesn't need to see any of those intermediate steps that led to the final reports.

>> And I I think the fact that you asked that question is important too, like when to use sub agents. Most people don't know how to answer that. As for myself, like I have no idea in a lot of cases, you know, when to use a sub agent. And so I think that's also that speaks to the agent creation and the agent editor flow that we've created where I just have to describe the task effectively and our agent creator agent is is the one that can decide you know th this might require a lot of search and so like we should split out this part of the task into a sub agent and over time the product will also get better at that.

>> We've talked we've mentioned a few times kind of like human in the loop and autonomy and that type of things. There there are like these human in the like taking actions is scary and can be scary sometimes and so there are these human in the loop controls on some of the tools. Could you guys just talk about like what those are and then how you interact with with these things >> in the agent builder. We have this concept of interrupts which is something we're able to to take from Langraph where the agent can essentially propose an action and what to take. Say it's the send email tool. You don't necessarily want your agent to just send an email on your behalf without you reading over what it's going to send.

so in the agent builder we have a way for you to mark a tool as a tool that requires an interrupt and how that works is the agent will then call that tool right and it can see the tool from the agent's perspective there's no such thing as interrupt it thinks it can just call it and it'll execute so it calls the tool it proposes some input if it's the send email say it's you know a a subject and a body but then before actually executing that action right sending the actual email the agent will pause and it'll give you the opportunity to read over and review the action it's trying to take you can modify it. You can send a response to the agent and ask it to change it in some way. so we're we're allowing the agent to take these so-called like destructive actions, but first requiring your approval and giving you the opportunity to edit or or change that before it actually set.

>> What What does that look like for it to give me the the ability to control it or edit and like what if it's kicked off in the background by one of these triggers that we've talked about? >> Yeah, that's why we've built part of the product is is sort of an agent inbox. It's it's a a view of all conversations from this agent and many of those conversations are are done and have been fully processed. There's there's no action to take, but many of them have been interrupted and so you as the user of the agent can just easily filter through your agent inbox, see which ones need you to take action and quickly go through each of those and we've sort of embedded as we saw in the demo that interrupt flow into the actual chat experience and that's a pretty natural way to give it feedback. so you can easily go through and and take action quickly on all of them.

>> Exactly. Yeah, we've kind of borrowed this concept from an email inbox where you know you you receive the these emails. You can you can go through in our case threads or conversations. You can go through and look at past threads. you can see which ones require your your attention, require actions for you to take, right? These interrupted tools. and then you can you know go and chat with the agent resolve the interrupts and manage it all through like a very familiar concept to all of us who have used email before.

>> a few points you mentioned langraph which is obviously a developer framework. Can I take these agents and run them in langraph? >> Yeah. agents that you build in in the agent creator under the hood are just langraph assistants. and the the agent that powers the agent builder is just an instance of the deep agents architecture. So any agent you create in the agent builder you can very trivally take the configuration which defines that agent and go and create a new assistant in a separate production deployment. So you can use the agent builder to prototype or iterate on agents you might want to take to production afterwards. And then once you're happy with it and you use this this agent builder UX to iterate on it, you can take that configuration, go and create a new assistant in your production deployment. And that's all it takes to take an agent from agent builder to production.

>> And it's also why in the in the early days of this product, we want to make it really easy to get started. So we're hosting the deep agent for all of our users that use this product in the cloud. But over time, we'll allow you to bring your own deep agent. will allow you to host some other kind of graph architecture and and use that graph to power new assistants via nontechnical users as well. >> A lot of technical concepts here. you know we've talked about tools, integrations, o let let like putting even aside kind of like you know langraph and some of the developer frameworks. How how do you think about making all of these things accessible to a different audience than we've approached in the past?

>> I I think you have to just totally break down your mental framing of the product. I found that like as a technical person myself, I'm I'm often coming up with frankly the wrong ideas for this product because I'm I'm designing it in a way that I want it to be designed and and we we can talk as well. I think there is an interesting aspect of this product that it does make it easier for technical people to build agents as well. But we we are primarily purposing it for those nontechnical people and that's sort of how we came to this core UX of like using natural language to create the agent, having a a very friendly looking docbased editor to edit it and so forth.

>> Yeah, exactly. I think the the biggest thing is just we we don't want to force people to go through the tool list

and pick that write out the system prompt. so we're really centering all around these these agents that work in the background for you or you can chat with that that can build this agent for you. and Sam brought up a good point where like like yes, this is a non-technical application where you don't need to know how to code or like really how LLMs work to use it. but that being said, it's also really useful for technical people, right? there's so many agents and workflows and automations that that I want for myself, but I would never spend the time or I don't have the time to build them out.

but with agent builder, I can spend a minute to describe what I want and then instantly get an agent that that can run for me. So like yes, it's going to you know help with with the non-technical users to automate some tasks, but then also for technical users, you know, we have so many ideas for things we want to build, but we would never build them if we had to write it all out in code. >> And I mean like writing the prompts is really hard. So yeah, thank you for building something where I don't have to write prompts. maybe maybe one kind of like you know interesting hottake question to kind of end it like you know this is a new thing that we're going into. I think there's there's a lot of really cool ideas that I think we've put in here. What is one area where you actually feel like we don't have the right answer for necessarily yet and you'd love kind of like feedback from users as they use it in in that direction? I I think we've built the products to be very modular like you know bring your own tools use your own off eventually bring your own triggers to today it's we we want sort of like a very easy onboarding experience so we're providing all of that for you I'm really excited to get this out in people's hands and see sort of like what tools they want to be added which which we can add for them but also like what sort of experience do they want their core platform team to have to to add new tools themselves to add new triggers themselves also that the non-technical user by the time that they show up in the product for the first time, all the modules are in place for them already.

>> Yeah, I think for me that there's like two main areas which which are still kind of an unknown. The first is like how you optimize the agent. So, initially when you go to agent builder, you give it a description, answer some follow-ups, and we give you this initial version. but then what is the the best way to improve that agent afterwards? Is it a chat like a chatbot agent which can go and modify your prompt? Is it similar to like a canvas experience where you highlight some text in the system prompt and give feedback? is it totally different where you just chat with the agent and then can like give a thumbs up or thumbs down on a message and then we have some autonomous system figure out what your intent is? so like for for me that that's still kind of a big unknown and like we have ideas there. but the only way we're going to really figure out what what the best experience is is by getting feedback from everyone who uses it and seeing what works and what doesn't work for them.

>> Cool. Well, I I think that's a great call to action. We'd definitely love people to to try it out. I think this is one of the cooler, more interesting things that we've done. So, yeah, kudos to to you guys for a great ship and thanks for walking us all through it. >> Thank you.