Zero-Shot Learning - Episode 7: Jeff Wang from Cognition

The work runs itself

Host: Nancy Wang, Chief Technology Officer, 1Password
Guest Co-host: Richard Liu, Head of API Products, Anthropic
Guest: Jeff Wang, President of New Enterprise, Cognition

Note: Dev Tagare, regular co-host and Senior Director of Engineering for Gemini Enterprise and Business at Google, is absent this episode. Richard Liu joins as guest co-host.

Introduction: Windsurf, Devin, and the new agent platform

Nancy Wang: Hi everyone, and welcome back to Zero-Shot Learning, the podcast about the reality of developing with AI from the people actually doing the work. I'm Nancy Wang, Chief Technology Officer at 1Password. Normally, I co-host this show with Dev Tagare, Senior Director and Head of Engineering for Gemini Enterprise and Business at Google. Dev couldn't make it today, so Richard Liu, who leads API products at Anthropic, is joining me as guest co-host. Our guest is Jeff Wang, President of New Enterprise at Cognition. Jeff previously led Windsurf, which is now part of Cognition. During this episode, you'll hear us refer to Windsurf and Devin Desktop. Those names describe the connected products and platforms that Jeff's team has built. We discuss how cloud agents work, how agent swarms change software development, and what happens when those systems fail. This conversation is especially useful because Jeff is candid about the infrastructure, security, and adoption problems that teams face when they put agents to work. This is Zero-Shot Learning.

Jeff Wang: Yeah, thanks for having me. Let's have some fun together.

Nancy Wang: And I want to parse that a little bit. So Windsurf, Cognition, and Devin. Can you unpack that for our audience?.

Jeff Wang: Yeah it's a combination of a couple of products now, but we try to sell it now as one platform. Engineers have a lot of ways they want to interact with code and write code. with Windsurf you had the IDE, but now it's with Windsurf 2.0. It's more like an agent command center. You're like moving agents around. we also have a CLI as well. We also have a CLI. Claude Code popularized this form factor, and people use it because they don't have to modify the files as much anymore with the models getting better. And then now Devin is this like remote agent. This agent that runs in the cloud has all the dependencies and access to the relevant data that you need to do the task, and you could scale that across the whole company.

Nancy Wang: That's amazing. And I'm sure you have some thoughts as well. Richard, what do API products at Anthropic mean?

Richard Liu: Yeah, so I've seen a couple of different parts of the stack previously was at Google with Gemini and looked at a lot of the infrastructure layers and APIs. Now really I'm working on the API layer at Anthropic. And so that's looking at the security, the enterprise readiness and the identity stack around how our APIs are served to customers. And really it's about all of the harnesses around the core model. That makes a lot of the possibilities of things like agents and our Agents SDK and Claude Managed Agents possible. And so super excited to be part of this conversation.

Where cloud agents run

Nancy Wang: So I noticed, you started talking about Devin and you mentioned Claude Managed Agents Claude Managed Agents. Where do these agents run?

Jeff Wang: They're running on VMs that are in the cloud. Sometimes our customers request to be deployed in their VPC as well. But you have with like Windsurf 2.0, you can run agents locally as well. So you have the option to work locally or in the cloud. But what we've really discovered is cloud agents are like the future. If you have an agent that has access to

everything, it needs to be productive and you roll it out to your whole company and you could scale it to like thousands of agents. Like we're seeing a lot of people using dozens of agents at once in parallel. this is like the new world that we're living in.

Richard Liu: I think what we saw at Anthropic, at least, was previously it was mostly Claude Code, which is you're running the model and this these capabilities directly on your laptop. And then we evolved and introduced the Agents SDK, which allows you to bring your own infrastructure and be able to run these agents there. And I think the next evolution, which is where we're seeing a lot of the direction go, is a truly serverless solution where a lot of this is hosted in the cloud, in CSPs like Amazon or Google. and it's fully managed. So the customer just ships their configuration for the agent and we handle the rest.

Nancy Wang: And in fact, it was really interesting to note, as a security provider amongst the ISVs we've also had to evolve as well because, where we've traditionally been, 1Password is we have a local vault which sits on your laptop or your mobile device and stores credentials there. But certainly as agents, especially like Devin and Claude Managed Agents are now running in the cloud, maybe even on sandboxes. To your point, you said VMs. we've also had to extend our credential broker capabilities into sandboxes, into cloud environments. so maybe on that note, Jeff, walk us through, an example of a task that is really well suited for cloud agents like Devin. That might not be for an IDE assistant.

Jeff Wang: And it's really funny. All this has happened because the models have gotten really good at very long-running tasks. I like the joke. I have a son who's four and a half years old, and all of a sudden he's able to do a lot of things by himself now. And it was like the Opus 4.5 moment where it crossed the threshold, where it can do so many different things, though. But the problem is, the agent doesn't have access to some of the necessary like data or access to write code or a modify code as well. And if you think about Devin, it is this agent that has access to everything to be really productive. So a good example. And by the way, because it's in a VM you could verify the code is working as well. So if you say you're doing a codebase migration or a code upgrade, there's a way to have a before and after state. the code is running beforehand. Hopefully you have a unit test or success criteria that shows the code is running well. And then when you do this, this new version, like let's say you do a COBOL migration or a Java upgrade, even in the completed state, Devin can run the actual application on the VM and do like some UI testing. Have a list of things to check. You could run windows environments. Now you can run Android environments. Say you have an Android app and you want to test if it's working. You can just deploy that in Devin and it could test it for you. And then it comes back to the engineer and it's here's a report of why we think this is working. We also give you a PR. Of course we still want some auditability and human in the loop. But that is like the way that a lot of people work now is you hand off tasks to an agent. It comes back with all this proof that it works and you just approve it.

Nancy Wang: That's amazing. In fact, my eyes just lit up because, as 1Password, we're deployed across all the client apps and desktops. And so we have windows, we have Android, obviously Mac, iOS, and so and of course, we're also in the browser as well across many different flavors. And so for us testing is always a very manual process. Right. And so the ability to essentially set up a test environment and come back to you with validation that, hey, this branch or this version is working. that's a huge unlock it.

Event-driven agents: The work nobody wanted

Jeff Wang: There's a thing happening with event-driven agents as well. So let's say there's a bug report for your Android app for 1Password. You ask everybody in the development team, hey, who wants to replicate this? Nobody's going to raise their hand. I can tell you from experience, nobody really wants to be the person to replicate bugs. But Devin's not going to complain. You can give it to Devin. The ticket comes in. Devin. We'll see if it is a bug, and then it could automatically fix it. So this whole like there's a whole paradigm shift right now. Event driven agents are roughly 40% of all of the workload right now, which is probably a shocking stat for people that are initiating these agents, like having dozens of agents. And it's still not the majority, or it is a slight majority of all the work. Right? So this is crazy stuff, like the whole world of like how we're interacting with code and how we fix problems and how we want to do net new stuff. Yeah, it's a very different world where just like six months ago, then.

Richard Liu: What are some specific types of examples that you would say that have been particularly valuable for these event-driven tasks. So things that come to mind are like long running tool migrations or database migrations, like the issue to PR being able to go from the initial bug report all the way to a pull request. like what are the most common or valuable type of scenarios or events that you've really seen in power? Your customers.

Jeff Wang: You go into any engineering organization, you don't want to take away the things that they want to do. You want to take away the things that people don't want to do. And one of the biggest ones is vulnerabilities. So if vulnerabilities come, you might have like 24 hours to address it. You might have like a week to address it. Nobody is like raising their hand either to drop what they're doing and fix the vulnerability. But for event-driven Devins and agents, this is like one of the biggest use cases. Like we have some like large banks that have like 70% of all their vulnerabilities are just automatically being fixed by Devin now. And they might have a backlog of vulnerabilities that you just throw the agents at, and it will just go start fixing them one by one as well. And I can tell you, nobody wants to take that work they they don't want to drop what they're doing to fix vulnerabilities. Another good example is like ci, CD. Like you want to merge your coding. There's some checks that happen and it breaks. Nobody wants it to break. Everyone wants to see the end result. Right. So then the cool thing is Devin can automatically be set to fix those as well. So these are some examples. It's really the toil that the things that people don't want to drop and work on that. This is really good for the event-driven use cases.

Beyond code: How agents change the workday

Nancy Wang: And so from your seat right. We're talking a lot about of course coding type of tasks for these cloud agents. I imagine, in your day to day you do a lot more than just code. And so how does a cloud agent or remote agent help you in your seat? What is your day to day look like with this?

Jeff Wang: Yeah, I'm spending a lot of my time, as like traveling the world, trying to open up new regions and offices and really understand what the customer pain is and trying to meet with the customers. So a lot of that is like go to market if you think about it. And I could ask Devin, well, first of all, you can plug Devin into a lot of things that are productive. So one of them is like a Salesforce connection. Another one could be we work with a company called Exa that like indexes the web and you could.

Nancy Wang: Search for a company.

Jeff Wang: Yeah,. And you can pull like LinkedIn profiles and, and you can ask Devin hey I'm going to go to like Google Cloud next, I want to meet with who's there. And how about you just find out who's there, get their email, write me some ghost notes. I'm going to send this to my reps. And then that's all. That's all I said. And then just like gets it done. I could also connect like an email like MCP server or something, but I haven't. I don't want to get that far. I still want like the reps to like really think about, is it worth the time to meet this person or not? But that's a good example. Another one is since it has access to the database like customer usage billing, you can ask a lot of questions of I'm about to go into this meeting with this customer, what is their usage like? And traditionally, maybe you have to like look at a dashboard and filter, but it is just something that I just like am naturally asking agents about. now. I just asked hey, what's going on with this customer? And it will probably be hey, usage is dropping. You probably want to fix this, right? but this is like just how all of us behave at the company. probably we should use more dashboards, but a lot of the behavior now is just like asking agents to, inform you of the latest. And it just creates you a. Net new dashboard a lot of times too.

Richard Liu: Now, I think this is pretty interesting. I've been hearing stats recently that for typical developers, workflow about like anywhere from 8 to 15% of their time during the day is actual hands-on-keyboard coding work, and the rest is exactly what you described. Like all kinds of other kinds of coordination, research and meeting with people like how do you think about like how to how to improve some of these additional tasks and some of the things that are coming up that are pretty interesting or like knowledge graphs or understanding, like how the relationships between people can happen, like for the remainder of the 80 to 90% of the time.

Building Devin: Playbooks and knowledge infrastructure

Jeff Wang: There's two transforming. Like right now. The. I guess the root of all this is because it takes so long to get one of these, prompts done. It could take hours to get done. And that means people are sending like another task and another task. And eventually soon you have like a dozen different boxes on your screen. Like go in at once. what is probably going to happen is a you move up a layer of abstraction. So instead of like the granular features you want the agent to develop, you might say, hey, we need to revamp this, the superset above. Here's all the features we want to have. We're we want to like get more revenue. We need we need $100 million more revenue this month. So how do I do that right. So that's a little maybe a couple layer of abstractions above. So you're telling this agent some high level thing and it breaks it down for you, the work it needs to do. And then it starts doing the work. So like we've been implementing features to create sub agents for dev and to spin up other Devins. And that's how we're preparing for when people start The transitioning to this next abstraction layer. And it could be to the point where someone needs to, start a company and says hey, I want to start a company. I want it to be worth a billion. All right, Devin, let's let's let's go. Right. that's probably a couple years down, but we are like moving towards that state.

Nancy Wang: And so maybe coming back to what it takes to build a Devin. Right. Maybe let's do a deeper dive. And so if you think about a coding agent, right, there's a couple of primitives that are pretty, common things like evals or, shared memory or things like tool calls or configs. So like what goes into, obviously as much as you can share, like building Devin. and, where do you think teams typically under invest in deploying or rolling out a Devin.

Jeff Wang: So at the very base layer and this is a problem is you do need like a codebase attached because you do need something for Devin to modify. And again this could be like an onboarding issue where like oh, I have to have a codebase. what if I just want to use, like a chatbot and said, right. Yeah. So we are we are working on that too. I just want to make a caveat there.

Nancy Wang: Thank you. I was going to say you took out my feature request before because for go to market teams, right. They may not have a codebase attached.

Jeff Wang: That's exactly it. That's right. And once you have a codebase attached then there's other tools that might be useful. Like maybe it's your Datadog and you want to investigate something that broke. Maybe it's your usage data. So you connect like a, like a database like BigQuery or Redshift or whatever. Right. once you have enough things connected that you think are useful for Devin, maybe the one thing that most people don't invest in the beginning is creating a playbook. So a playbook is a set of tasks that you can get a repeatable outcome from. So let's say you want to do like a data migration. You can put in the playbook. Here's database A here's database B, here's the outcomes. Here's where you need to check to like investigate or use. create your success criteria as well. And then what you want is like a very repeatable outcome or a very, deliberate outcome. And as long as you have, agents that know what they're supposed to do, the, the harness and the models are really smart. Now to go and use the available tools to, to get it done. And the cool thing about Devin is it can keep like using trial and error and if it gets it wrong the first time, it learns. So like I know there's like a lot of research going into like continual learning and you don't make the same mistake twice. we have shared knowledge in the, in the organization too. So if Devin messes up, you can correct it. And then Devin learns from that and it goes into the knowledge base as well. So everybody will be able to benefit from that learning.

Context management and agent failure modes

Nancy Wang: Let's maybe for our audience come up with like two different mental models, which is how does it work in Windsurf and how does it work in Devin? So let's start with, context, like how does each system decide that enough. They have enough to start working on.

Jeff Wang: Yeah. So with Windsurf, we used to build a local context. this is mainly for security reasons, because some companies, they never want their code to be exposed anywhere. And we had this like complex configuration where the moment you open the codebase, we generate embeddings immediately create a local vector db. We've since stopped doing that because it took a lot of resources. And now we have Devin, which when you plug in the codebase, it generates an index remotely. And that it also creates a whole documentation of your codebase as well. But the agent can use that to really deeply understand the codebase. But on Windsurf side you might have like just a random directory you open up and that doesn't have again a remote, vector DB. So instead we have a lot of other tools we do have like grep where you can like do keyword matching. You could also have the semantic search as well. So a semantic search is like you can use a smaller model, look for the string and go crawl through files very quickly. And we use very like complex info for that

too. For example, we do use like Cerberus to do a lot of the agent search, and it flies through like we could have like millions of lines of code and it could still fly through. so both that works on both Windsurf and Devin. But then, because Windsurf doesn't have all of the like. Indexing potentially if you're opening a random directory, that's how that's how it works.

Richard Liu: Let's talk about common failure scenarios like what's some of the most common failure modes or like either like editing the wrong tools or files that you've seen, like with Devin, that typically are most objected or customers are worried about.

Jeff Wang: Probably the most common, as if it doesn't know the specific output it's supposed to get right. So you have to be very clear what you're asking of the agent and what is what does success look like? So if you give a very ambiguous task and it doesn't know what you want, it might fail because it doesn't know what you want, right? And to avoid this, there again there's two things. One is like building this knowledge base of if it keeps failing, you probably keep creating knowledge of not to do that. But the second is creating more playbooks. So then if you create a playbook again, it's like a very, very specific list of instructions to accomplish the task. That is another way to avoid those common failure modes.

Nancy Wang: And, harkening back to something that you said previously, which is when you correct Devin, it understands it learns it, stores it somewhere in its memory and then comes back and learns, your preferences for the next time and does the task. I guess the question there is, how do you do that without having to turn back into prompt engineering?

Jeff Wang: Anthropic has discovered this too, which is like you separate the actual trajectory, like the whole conversation with the analysis of what's going on in the conversation. So as the conversation is going for like many, many hours, there's another agent that's examining the whole thing and then retrieving the right context when necessary. It could be codebase, it could be knowledge, it could be like the available MCP servers. But all of it has to do with how much information you give upfront and how much help, how much you plan with the agent. but Anthropic has discovered this to like having multiple like systems working together to avoid some of these, failure modes like long context and long horizon. context stuffing. Right? all these things can be mitigated with all these other methods.

Richard Liu: I think we've looked at also like just with a limited context, windows have gotten pretty large nowadays. Like we've hit a million. But as you get multiple long-running agents that are consuming a lot of that, like things like compaction or like with memory, and now we've really streaming, where like we can analyze some of these memory patterns and try to, derive some insights. definitely. Like we'd love to hear more about where you think some of this is going and how we can improve that continuous chain of thought as some of these flows get longer and longer.

Jeff Wang: Yeah, and I mentioned it earlier. One of them is having dev and create other daemons. Or when you generate like sub agents, the sub agents are like a tool call of another agent that has a specific outcome too. So all these things are creating net new trajectories that like link to the main agent or the agent spawned other agents. And it's like organizing the results there, too. And yeah, there's other methods of like compaction, which I think removes like the thinking traces or like different ways to summarize the actual message itself. Like there's like a lot of technology that goes into that as well. but all these things work together to keep track of, like what is the actual task at hand and where are we right now? And it can go on forever. I just talked about earlier about like a 29 day agent, which is insane to think about, but even after a few hours, you're definitely like out of your context window, right? There's so a lot of a lot of things happen to make you keep your eye on the ball and focus on the outcome.

Richard Liu: And we're improving a lot of these on the model layer as well, right. Whether it's context, windows or Core capabilities that we can harness on top of the core models. how are you thinking about like how much of this is truly model driven versus like building on top of the models and the primitives?

Jeff Wang: Well, whenever the model has a capability, we want to leverage it as much as possible. It's designed to do that, right. So, for example, I think Opus 4.6 was the first model that like took notes, like it would generate a Markdown file to like keep track of what's going on. Yeah, we want to leverage that because that's the that's what the model was trained for to use. Right. And we also provide stuff on top of it. If it does get a little bit off and we provide a lot more like again, like a lot of tools, a lot of data so that we are we just want to increase the probability that we can get things end-to-end done. Right. And a lot of the things we were doing, like nobody else is doing it like we talked about the UI testing and the, deploying Android or Windows, right? These things like nobody else is doing right now, but it does increase, like the amount of end-to-end agents that can get to the very end. Right? So we like to think of it as like it's a superset of all these other end-to-end agents where we're just getting a lot farther. And if it gets a lot farther, you could

probably deploy even more because you can just not think about interjecting and correcting and unblocking. You can always just keep hunching agents until your all the work is done, hopefully right. But again, there's a problem with that. Like when we go to like big enterprises, their work is typically their backlog and the net new stuff is like like they're not like prepared to do a lot faster on the net new stuff. Surprisingly, startups though is a different story. Startups is like there's a lot of tech debt. There's a there's unlimited things they need to do. Like their roadmap is unlimited. They're competing with everybody else. So startups is probably where you're seeing this like crazy like sprint on like token maxing or just moving really fast. so that has been a really interesting insight as well.

Success metrics, ROI, and token budgets

Nancy Wang: Yeah. In fact, this week I was at a OpenAI, frontiers forum. Right. And they were saying how, typically an engineer was maybe publishing five hours a week. And that was maybe you're a P70 engineer and now it's become 50 a week, right? And, I have to thank for your P99 engineer, that might be even more than that. And so coming back to, something that you mentioned around, Devin and be able to let's say, finish tasks like what is the state of being done look like for Devin. Like what success metrics do you look for that you're okay, this autonomous agent has finished this thing. I can now move on to something else.

Jeff Wang: Yeah, it's a great question. In fact, is there are multiple people working on a project. It's constrained by the person that is behind on the some like overarching feature. I'll tell you like an example, like for Windsurf 2.0, we were held back a week or two because one person was supposed to be doing off. Then everybody else had finished like all the other features, because they were moving fast. Yeah. and I can tell you another cool stat, which is we are moving roughly 700% faster in terms of the number of PRS that we're merging, in the past six months alone. And we've only increased the headcount by like 10%. So it is working like obviously this like this jump in token spend across the board. And Windsurf revenue is like spike. Like you can see it right in the revenue. and we haven't reported ours yet, but it's also a similar trend. But the that jump is because of like the number of things you can do end-to-end and the ability to just like keep stacking things and yeah, like CFOs are calling me now, they're hey, like we've already used our budget and you guys just signed last month, so and so this is happening now.

Nancy Wang: These this is probably the top question, right, for CFOs these days is, how do we even or how should they think about like for example, forecasting. Right. This might be less of a technical question, more on how do you, talk with CFOs about budgeting and headcount.

Jeff Wang: Yeah. Well, first of all, the thing we can do is optimize for costs today. you don't need to fly a private jet to from San Francisco to San Jose, right? Like you just don't have to. So like you should think about what's the best model for the job, what's the most efficient model for the job. And Devin is uniquely positioned where it can use multi models from different companies as well. We might be the last company that can do that right. and that gives us an advantage to like really think about is this a is this the most efficient model. And we can do different modes of like this is the cheapest, like a cheaper mode versus like the max mode, right. We're thinking about the branding around this. And then for open source for our own in-house models, for task specific models too, we are optimizing on cost too. So we've built our own agent search model. We built our own bug checking model as well. And then these are like very optimized, very cheap models to run that we can serve to the customers for free. And we have our own coding models as well. Again, we're serving those for free as well. I don't know how long that's going to last, but we want a viable alternative so that if again, CFOs are I got to turn this off, we can still be well, hey, we have some alternatives.

Richard Liu: So what are some of the success metrics that you really look at, like in terms of optimizing cost or latency for some of your customers? would you try to get faster iterations out there or just a better like end-to-end result for customers? how do you balance some of these things, in terms of cost management.

Jeff Wang: The main thing is what? How do they feel about ROI? Right. Is ROI a project that would have cost $100 million? And we just did it for a couple of million. Is that worth it to you? Because you're about to spend a couple million, like in this month, right? And that would have taken maybe three years. So like the time horizon, divided by the number of months, it's compressed to one month, but it still like was worth it. The ROI is definitely there. And then it also becomes what is your what is the value of time for you. Right. Because if you think about a project, it's like it's budget, it's headcount, and it's time. Right now you are in full control of the time. Now, you didn't have that before, but that's going to explode the budget and you could compress the headcount as well. But this is like the conversation is like how do you prove how do you make the decision you're okay with this or not? And most organizations are not okay with that. So there are other ROI metrics. There is like velocity of like the amount of PRS and merch PRS and tickets are burning

through a backlog that you're burning through. So that part is like stuff we're figuring out with as well. But we are seeing customers take projects and give it a token budget like this is never this is not a thing like just a few months ago, right? So they're saying, like for this project, here's the number of tokens we've allocated to it. And if you run, if you hit that number, you can't use like opus anymore or like GPT five, right? Like it's like you have to use the cheaper models after that. So that's like the it's like starting to happen now as well.

Richard Liu: And I think the question probably would be how do you demonstrate that ROI directly to customers. Like are you thinking more case studies or like real, real world like scenarios to demonstrate this or have AI like FDEs deployed like in these companies to show them using like pilots, how do you really build out that study so that the like CISO or the CFO was able to purchase this?

Jeff Wang: I think a lot of this is based off of what their budget was for the previous year, and how much they thought each of these projects were going to cost, because that's completely all reset. Now it's like going to a sprint team and being how many points is this like? That whole system doesn't make any sense either. Like story points. Right? So it's like when you talk to CFOs, you have to look at some benchmark that they had priced before. As one of the examples. So when you go into and the best thing to do with a like if you're going net new into a customer is saying what are the most expensive projects? Like what is the thing that's going to take the longest. And then the forward deployed engineers can go in and do it much faster and much cheaper. And that's a great way to show ROI, right? It is no longer like with us, it's like because it's so complex, you have to set up a bunch of tools and give it access to data. You do. You do need to like pick something that is like beefy. whereas like if you're like a Claude Code, it's oh, everyone's just gonna sell it. Okay, great. it's a very different, motion. We want to really get deeply involved with the customer and really get attached to their outcomes. So we want to say we sell outcomes. We don't really sell the dev tools, but that is truly the motion that we have.

Security, access scoping, and forward deployed engineers

Nancy Wang: That makes sense. And especially, from a CTO buyer perspective, that's really compelling. obviously now I'm thinking on the security side, right? with agents now being new actors within like a company system, how do you draw the boundaries, right? Is it like CI pipelines? Is it by like repos? Is it by jobs? Right. And I'm just thinking, you might want to apply different guardrails to each environment so teams can move fast. But also like your C CISO doesn't freak out.

Jeff Wang: Yeah. And this is where, having an isolated instance is really beneficial. in this isolated sandbox and VM, it can't really do anything if you don't give it access. So if you only give it access to write code on the repo, the only thing you can do is give you a PR, but then you might say, hey, I wanted to access, our production systems and I wanted to modify data. Then there's other, failure modes, right? There's like destructive modes. There's things that you can prevent agent from doing, but we don't recommend doing that. We do recommend for like testing purposes, you can like deploy the app in the environment and then provide it with data that is dummy data. But we would never say like try to log in to the actual system with admin and then do go and go home. Like that is not something we'd like to recommend to our customers when we help them set things up. And by the way, when people try to build this themselves, they forget that part. Like they might build their own VMs and sandboxes, and it's like their account and they log it like the agents logging in has access to what they have access to. And it could it could do something pretty bad. So like this is why like again, the isolated VM does seem to be like the right mode to scale, to like thousands of developers in an organization.

Richard Liu: I think the isolation is definitely important. But you talked about access as well. And so how do you make sure that as you start getting more and more autonomous agents, that the agent's task at hand is properly scope to what the actual purpose that agent is doing, because that's definitely another vector where they can start getting a lot of unauthorized or elevated access.

Jeff Wang: Yeah, for sure. And when again, when you connect like a database that say you don't want to give it like edit access to the database, right. You heard stories of people's databases getting wiped by the agent. Yeah. But what.

Nancy Wang: Happened with AWS.

Jeff Wang: Right. Yeah. And you, when you set up these, Devin's you are very specifically giving the right access to it. And again we help you set it up. We give you best practices about this too.

Nancy Wang: And inside the app itself or with FDEs.

Jeff Wang: The FDEs are the ones that usually go in and help the, the customers do this.

Nancy Wang: Yeah. And that's a trend we're seeing more often, especially across enterprise, customers who are deploying these autonomous agents or remote agents for the first time. So maybe tell us what, prompted you to bring in FDEs and also, how did these FDEs typically work with your customers?

Jeff Wang: Yeah, I can tell you the whole story. it's funny, I think Palantir was the first to come up with the concept. But for us when we were I think this was like three years ago now. Like we were only like 10 or 12 people on the team. we had to go into the customers to try to make them understand how to use these tools. This is like the first time AI is like even a thing. And we realized, if you just try to like let them use it, they're not going to use it. Like even if you gave them this magic solution to all your life's problems, only like 15 to 20% of the people would use it, and that's bad. so then we started discovering wait, all right, how about we just go through some training sessions, go through adoption metrics? We, we try to go through, some of the top developers and have them teach it to the other developers. we just went through all these methods and we saw, adoptions, like adoption rate just spike up and then we're okay, like this is this is something we need to repeat. So as we hire like a sales team, we were like saying, are these solution engineers, solution architects, sales engineers. And we said, wait, no, they're going in there. And they're like getting everybody On their feet and teaching them how to use it and making sure that's successful. And then we call them deployed engineers at the very, very beginning. So I think we might have been like the second company, at least after Palantir, to call them to forward deployed engineers. Someone's going to like right on my Twitter that I'm wrong. But that's that's at least that's how we felt like two and a half years ago or so.

The autonomy dial: Windsurf versus Devin

Nancy Wang: Gotcha. Well, certainly two and a half years ago, I don't even think anyone was really thinking about remote agents. Right. It was still very much I think someone was saying like tab autocomplete code, right. So it's just been crazy, how much we've changed. And so now moving away from, how the different like learning loops work for Devin. Maybe let's dive into some of the product design choices between Windsurf and Devin. So maybe starting off with, where do you set the autonomy dial for Devin. and how do most customers go about configuring that?

Jeff Wang: A lot of things. a lot of the outcomes of Devin is like a PR. so when you give a task to Devin, you expect to come back with something that's done. So what I mentioned before, like playbooks, is probably the number one mode for this, because you will have very common things that the organization has to do. And you can tell what is the outcome that you want from Devin. And then you can then scale that to the organization, let everybody know this is the playbook. Even simple things for like I mentioned like what is the what's going on with this customer. We do have like a customer playbook. Like you just run the playbook, say the customer name and it just like tells you everything you need to know. So even that is something that's very predictable, even though it's not a PR, but it's like something the agent knows what it's supposed to do. Now with Windsurf, it's a little different because you are interacting with the agent most of the time. And this is in the old days, I guess now that we have Windsurf 2.0, you're like running a lot of end-to-end tasks now. But in the old days, you're like going back and forth with the agent because you don't know or the agent doesn't know, for example, what design choices you prefer. You might be saying, I don't like that color change to this. Or I think that button's too big. there's all these things that you want to go back and forth with the agent or like even the experience is slow or something doesn't work. that is something you would work with agent, but it doesn't have like what Devin has, which is Devin will go and try and see if it works. It'll click around and if it doesn't work, you'll like try to fix it. So like Devin has more of this like self feedback verification loop fixing loop that Windsurf doesn't have because it's local, I see.

Nancy Wang: And so that begs the question when would you use Windsurf then? Because it sounds like Devin is a superset of all of Windsurf capabilities, including even Windsurf 2.0 with that self-learning loop.

Jeff Wang: So there are things that you will want to go back and forth on. Like maybe it is like trying out different ideas and the output, the outcome is not like a PR you're I need a plan. What we're going to do together. Break it down at different tickets and I want to see what things look like first. And then maybe you do hand that off to a Devin as well. So there are ways to use Windsurf locally and then hand it off to an agent in the cloud as well. So I think this like whole, this whole concept of using an IDE is going away there. There will probably be no IDs after this year. It will just be agent management systems and then.

Nancy Wang: It look like a terminal.

Jeff Wang: I think it'll it'll be what we're doing with Windsurf 2.0, which is like a Kanban board of agents and what state they're in and being able to manage them in the different states. So for example, if an agent gets blocked and it doesn't know what to do next, that is a state that you can just like focus on. Like you might put a bunch of agents in the morning, you go get lunch and you're oh, I'm gonna go and block some agents before I get lunch. And you just, open those agents up, help them out, and then go to lunch, go get lunch and come back. Right. So that's like what it's going to be like going forward.

Nancy Wang: Wow. That's a very powerful vision.

Trust, traceability, and legacy modernization

Richard Liu: I think moving more towards some of the product strategy areas, like if you were to balance things around reliability, traceability and autonomy, which are the ones that you would really balance for. Devin.

Jeff Wang: Yeah, when we go to talk to enterprises like Auditability and traceability is a huge thing because right now agents are going ham. Creating PR is everywhere. So it is important for an organization wide like admin slash being able to see where each agent got launched from, who launched it. So every PR that happens, where did it come from? What was the session that set it off because something could break downstream. And then you need to know why. And you need to know like who is who is at fault. Right. So all of that is in the way we've set up the Devin platform, everything is traceable to who launched agent and what happened in that session.

Nancy Wang: Yeah. Maybe piggybacking off of what Richard just said around reliability. I think the most important thing, especially for enterprises. Right. Let's take the UX fix loop, for example. that's a lot of trust because essentially then, if we were to, onboard Devin for all of our UX loop, issues, it would be out to our over 6 million consumer users across all of our different service areas. how do you earn trust with engineering? Right. Is it what metrics matter most? Is it plan quality? Is it, for example, the, narrative of intent like maybe walk us through like how do you convince engineers from I don't know about this because that's literally the conversation I'm probably going to have with my engineers when I tell them about Devin to, maybe or even like resounding, yes.

Jeff Wang: I think you got to set that success right. And maybe it's a very long list of what would this agent need to do that you would be comfortable with and then put it on that success criteria, and then the agent will just like go and prove it. And the other thing is like reviewing code right now. So as you have like 100 x more PRS, probably there's a lot of like bottlenecks now in reviewing. And we've rolled out a whole bunch of features about before you submit a PR or like for example, if you're working locally, we have this like bug checking model that you can run. And then after you get into a PR, we have a product called Devin Review. And Devin Review batches the features up to look at the file diffs versus like alphabetical file. and we also flag net new things that engineers should know. So if like if there's a very major architectural change. We flagged that in Devin Review. So as the engineers like not just like throwing slop in 100% of the time so that.

Nancy Wang: How does it get that context?

Jeff Wang: It runs through a, there's like a very deep analysis. And that's why it can be expensive as Anthropic has their own review system. But it is like a lot of stuff going behind the scenes to be very thorough and pass that to the models and coming back. And we use multiple models for review as well. But going back to your question about trust, right? Like eventually we're going to be in a world of we only have to review code. Perhaps maybe that's like only like nine months away. And that, that's that's a big leap of trust, of faithful trust. So getting to there is just demonstrating that you can catch all these failure modes and all these bugs and all these issues during the review process, but also the success criteria, if it's demonstrating its ability to complete all the success criteria and find all the bugs and, being able to like just like you just click and let things go through review, that's probably the world we're going to end up in where it just happens end-to-end. Even more to the end. And that's that's how again, the trust is built through that whole flow.

Richard Liu: So I think a lot of this you can really control on the Devin layer. But for customers that may have more legacy repos or they like have weaker testing and their environments overall just aren't as safe or secure. How would you treat those situations? Like would you still have Devin Go and focus on autonomy, or would you slow down and try to work around the environment?

Jeff Wang: This is a major, major, like bulk of the work that we do now is modernizing a lot of the very old enterprises out there. So we look at their codebase, we can increase the code coverage like just using Devin. So just like making it more

robust in the testing for how it's set up. And then when you modernize it, those same test cases should pass. If you convert it to a different language or you migrate only pieces of it out. Or if you're converting like a monolithic codebase to like a microservice like agent ready codebase. So all these things are part of the process. In the beginning, for these large enterprises, they have all these really, really old codebases. like mainframe and COBOL code is still very, very common. Yeah. Surprisingly, even to this day. And I'm wondering if, like in a year.

Nancy Wang: Financial services.

Jeff Wang: Exactly. really any company that's older than like 50 years, 30 to 50 years, there's, they're always, there's all this mainframe code they can't get rid of and they're all retiring to all those engineers. So they're like panicking right now I but my point is like this whole there's this, this whole effort to modernize code. And I think that's like step one. you don't really want to be holding on to that old code and not being able to be iterative and put agents on it and have I really, really benefit. It is all about getting that modernized first.

Nancy Wang: Yeah. I feel like you're saying a lot of things are certainly in top of mind for me, right? Which is, we're going through that ourselves right now. In fact, we recently published a blog, this time using cursor agents. But certainly, we start to go in kind of, set up like the test plan, right? Set up the success criteria ourselves manually. But I think, with what you're describing here could be a lot more automated.

Jeff Wang: That's right. And yeah, I think, this is why we also send our deployed engineers, a lot of these legacy companies, like they don't know how much that these tools can do. So we want we want to have people boots on the ground, plug things in and get them started with their playbooks. this is something they should. This is not something anyone wants to drop what they're doing and work on. So you want them to just, give all their toil and all the projects that, just need to get done and then throw it at agents and rip through. Right.

Richard Liu: And one of the things that I think still is generally like one of the long polls or blocking is PR reviews. Like for some of the outputs that a coding agent would create. And of course, like both Devin and also cloud we've released like review code review tools that customers can call. how is the how is the overall like adoption been around these things and how do you expect? I think something you said very interesting, which is long term, that, we may not need these PR reviews anymore. how do you get to that state from like where we are today, where a human might still want to go and do that or, and eventually like probably use one of these like automated PR tools like help me walk through like what our we've talked about a lot of these different things. But how do you see us getting toward that vision?

Jeff Wang: So I think step one is the agent has to prove the work is done. So right now, like Devin can send you a screen recording. It can show you all the, test cases past. It can even show you like here's some sample queries. The data is the same. that is the step one. The agent has to have some method to prove that all the work it did is correct. And sometimes you even have like a report, which is really cool. It's who wrote this whole thing? but then the second thing is going to be like maybe six months down the line from now. The number of PR reviews that didn't have any changes, I think the percentage, right. And that percentage might be I don't know, 60% today or something. And then in a month later it's like 70%. And then this is why I'm saying like nine months. It's like maybe nine months from now. It's like 90% of all PRS did not need any verification, any corrections at all. And again, when we get to some threshold, maybe it's 95, maybe it's 99%. Most likely people are just going to let things through and maybe they're doing it already. Right. But that's why we created Devin Review so that at least we can inform you of the important changes and the things we're spotting. And eventually, like maybe that's not important anymore either, right? Maybe it's maybe the code doesn't matter anymore in the future. You're just thinking about here's your product spec. I'm just modifying the product spec and you don't even know what's going on underneath. And that's what happened with assembly and other, layers of abstraction, right? We keep moving up. So who knows. This is where we're at.

Richard Liu: And you just see the business outcomes or the metrics that show that it's working. On less incident reports, less customer bugs, things like that.

Jeff Wang: That's right. But as you can see in the last couple of weeks, the last couple of months, you can use these tools to find all these vulnerabilities and security issues too. So maybe it's just going to be a constant battle. that means it's even more important to use AI to plug things in, right? Absolutely.

Customer stories and failure modes

Nancy Wang: And so maybe let's make this real, right. And let's talk about maybe two very different scenarios. One scenario in which it was a win scenario and then the other one maybe where it was a miss. Right. And so maybe starting with the win, tell me about maybe an example with a customer where Devin worked out way better than you had expected. What was like the hidden enabler again?

Jeff Wang: Oh, here's an example. Like in Brazil they changed their like citizen identifiers, like their Social Security equivalent. They made it alphanumeric. They changed the number of digits it was. And all these banks had to, convert all their identifiers into this new column. And they only had I think I believe it was like 15 months, 12 to 15 months to do it. And they were panicking like there's, there's there's no way they can get this done. This is like so many dependencies on all this data, right? But when you have these agents that can check for these dependencies and can check verify hey, like it did this column translate over. Is this tied to these other databases? When you set that part up, there is success because you can be comfortable with all the evidence that it worked. Right. And so we did I think that project only took like 6 to 8 weeks instead of a year. and that was just like proof that, okay, these agents are probably working. They're pretty good. in terms of failure modes. the funny thing is, when there is a failure, we usually build a new feature or add on some technology to, to like stop it. Like for example, like some of these codebases, can get really big. like you were thinking, probably gigabytes big. I think we've met the first company with, a terabyte size repo. so, these things are like things that break things. But then we in turn, then have to, create the technology to support it. So, so that any, anything like that we've usually gotten around to, to build.

Richard Liu: And I think for some of these failure modes, you can usually target it using better user experience or better tooling or better guardrails. how do you usually take a look at like what are some best ways to address these failure modes?

Jeff Wang: Oh, there's there's so many types of failure modes is the thing. Like maybe it is like long-running tasks. Like how do we fix that? And then we talked about like compaction and summarization. Or maybe the models are getting better at taking notes. I think there's just always going to be things that we run into, but eventually some technology comes out or something we develop, overcomes it. And then you'll probably see, like all these other managed sandbox, solutions come up with the same thing. Okay. Like one very clear example, which is oh, I want to create manage sandboxes too. I'm going to create it in this data center. And then whoops, I ran out of space. Like that's a that's a very common mode that a lot of our customers run into. so then for us, it's how do we make it, so elastic that you'll never run out of space? Right. So, that's like part of the, the work that we've done. We've done this for like two years now. Like everybody else is just in it for the last couple of months. Right. So we're like really in it deep within infrastructure apart.

Nancy Wang: Yeah. And that's some really hard stuff behind the scenes because then essentially you're going to have to gather capacity yourself and but present that seamlessly to or, showcase like the true elasticity of what the customer could have.

Jeff Wang: Yeah. And the state is like persistent too. So like if you have to jump back in, you can if somebody else wants to hop into the conversation as well. with that VM, you can. so there's and then you can spin things up and down like immediately. So it's I know everyone else is thinking about this stuff, but yeah, they're still playing catch up. I'd say Relative to what? Where Devin is today.

The Devin loop and the future of software development

Nancy Wang: So maybe to wrapping up this segment like walk us through the typical Devin loop, right. Starting with the PR, what is it doing? What is a human or if the human is even engaging?

Jeff Wang: Yeah, it starts with some task and the task can be very broad or very specific. ideally it's specific or ideally it's on a playbook which is specific. So you ask Devin to do something right. Maybe it's oh, it's a website change. Very simple. Or it's like our entire like database went down. Do some investigation. What's going on? Right. either way, it should have some outcome that it knows what to do. And it will run through multiple ways to, to investigate and go through the different tools it has. look at the data to see if anything is inconsistent. And it's very good at that right now. all these things that it can do to plan, investigate and really know how to execute is very good. Now it is. And then it has to execute. So what does execute mean? It could be writing code like fixing code. It could also be like making people. This is interesting. If it makes configurations on something else, that means it has some like admin control. So that's usually

not very common. But in theory, like you can give it access to something to, to fix some admin thing. Right. Or if I suggest the user to do that too. But whatever that thing is the end result is usually a PR. So if it's writing code, it fixes something. It spots it. It will try to give you evidence that is the reason why it broke and the before and after state. But that's what you get at the end of the day. It's here's a PR, here's the proof that it worked on the environment that we spun up in and go ahead and review it. And then we have Devin Review right there that you can go to next.

Nancy Wang: And if you could introduce one primitive to make all of this loop even more reliable, what would that be?

Jeff Wang: Great question. Hmm. probably the limitations now is when you have a to generalize of a task and if it's able to really nail down what you mean. And I think we talked a little bit about knowledge graphs before and being able to like look at a very, very large organization and getting to the very specific thing it needs. again, you can mitigate this by creating playbooks and pointing to it. But I think that's the biggest problem now is like very generalized task for across a lot of data, a lot of different sources and being able to figure out what to do. I think models need to get more smarter about that.

Richard Liu: So is that something that you're also building out along with the FDEs, to figure out where's the authoritative source of truth, like in the organization? Who has the authoritative voice, which direction that you should steer? All of these things that model reasoning typically struggles with.

Jeff Wang: Yeah. This is why we're training the organizations to build playbooks. And these knowledge entries is because, we want them to be very specific. Like if there's a test that they know is repeatable, it will it will always get done correctly. Right? That's what we want to get to. And if you don't do that work, meaning like you just plug in the data and your hands off. it'll work if it's like a startup, right? But if it's a, 50,000 developer org, well, boy, you gotta really you got to be very careful about like how you set it up. And there's again, other ways to mitigate it. You can create different orgs that have their own sets of data. You can create playbooks. You can create things that like even in deep wiki, which is the documentation that we provide. When you plug in the codebase, there are ways to link the codebases together. So if something if there's a dependency on something change that is like noted in deep wiki as well. So all these things like you want to just be very robust when you plug things in. And again, organizations are big. We had we had one of the banks had 300,000 repos that we documented with DeepWiki. So like the scale that we're operating in is crazy. Probably not very like well known for like maybe some startups or like newer companies because, 50 years, 100 years of, of stuff being created is it just lags or it just stays behind. Right. So that is a the type of customer we're dealing with.

Nancy Wang: Wow. That's incredible. From everything that you're mentioning from 300,000, right? Repos to monoliths to mainframes, even COBOL. this is now you're entering real engineering problems versus, I think, prototyping. And that's where it gets really exciting. so looking forward here right in the next 12 to 24 months, what part of the agentic coding stack do you think is just going to disappear?

Jeff Wang: I think reviewing. Yeah, I think or at least there will be so much trust in the review, that people will probably not look at it. I'll tell you an example. Like just seven months ago, people were still looking at the code, and I don't think anybody is looking at the code anymore. So like that's and I didn't think that was ever going to happen. But we are already at that stage today. So then the next I keep talking about layers of abstraction. But then next like level is like we just trust that the agent did the right thing. whether or not that increases the amount of security incidents or not. That's another question. But we will always provide tools. So of course, to make the customer feel more comfortable that things are done.

Nancy Wang: And so in that world, what do you think is going to become the new standard primitive?

Jeff Wang: Like what do you mean by that?

Nancy Wang: Yeah. So you mentioned, reviews are going to go away. So what's going to stay as part of the agent coding stack.

Jeff Wang: Probably you will have to have the ability to prove something is done. And there are tools today that do it. But there are probably better tools in the future to do it.

Richard Liu: Yeah, and I think we're seeing that as well even internally and Anthropic. Right. And so we're definitely heading toward that direction. I completely agree with that. And the hardest thing is tying that, that back to your original business outcome or what you were trying to get out of the PR. and I think that's definitely a challenge that a lot of people are still figuring out.

Nancy Wang: And maybe this one is a selfish question for me. If you're advising a CTO on what's the best rollout plan for, Devin and Devin class problems. What's the first 30 days look like?

Jeff Wang: It is identifying the best. Maybe like 3 to 5 use cases and then trying to do it together. for, for either all five or if they're too big, maybe 1 or 2. And then proving that out and then staging the, the instance to work with the next few use cases. once you really prove out that it can do these use cases, then the next step is like rolling out to the team. And in that month, you're like doing all those things like getting the team up today, how to use it, rolling out those use cases. You might add some use cases for everybody else. If they don't, if they're not part of that loop. but that is essentially is like we want to again deliver the outcomes like we the tool can we assume like you're rolling out agents, right? You're not you don't want to think like you're rolling out specific tools to do specific types of work. You're thinking about how can these agents be applied to a broad set of problems within the organization? And, and can I really show it to them by delivering those outcomes and solutions. So I think that's like the difference of how we're operating with like traditional SaaS. Because traditional SaaS, you're probably oh, does it have autocomplete? Great. like but no, it's like now it's like we're solving these major problems for you.

Closing: What would you build?

Nancy Wang: So that's one of my favorite closing questions for our builders here, which is if you could take the next two months off to build anything that you want to build, it doesn't have to be necessarily Devin or Windsurf. It could be, though. What would you build?

Jeff Wang: I would probably build something that is taking all the taking away all the, the, the not useless, but the tedious stuff of life. but that's like a I just need, like a, an e, like that's that could be it. But I don't think the reason why is because I don't know if the audience knows, but I'm traveling to, a new region almost every other week. And every time I go to the new region, then there's like a set of folks that are setting up the schedule and the meetings and the customer interactions. So like the EA is like a new a new person every week it seems. but back to your question of yeah, if I could build anything, I would hope it's, maybe the other most, busy thing in my life is like the kids. So if I could build something like that, can help them learn or, keep them. Yeah, like make them smarter somehow. Right. And maybe that is something that's software related or media related, or it could be like a game. that's probably something I would build over the next two months, because I do feel like that's like the one thing that's very challenging now is like keeping the kids entertained, but also like making them smarter and, yeah, being present with them. So hopefully it's like a cool thing I could build for that now you, motivate motivated me to try to build this prototype.

Nancy Wang: Do you even think it's going to exist in physical form?

Jeff Wang: . It could. I was thinking about that too. I was is this robotics thing? But that might be harder. I don't have a vibe robotics, coder thing. So, not yet. Maybe.

Nancy Wang: But there's llms out there like skilled AI that are, especially made for robots.

Richard Liu: Yeah. I think one of the things that I typically encounter talking to enterprise customers is that, the failure mode that we're talking about, these agents are very powerful, but a lot of our regulated customers and enterprises, they, like banks, have 3 to 6 month review cycles, like a lot of these security and regulatory requirements. I think building stuff outside of this, like artifacts that can help better navigate some of the organizational complexity and work with FDEs, but really like and harness the power of these agents in organizations that have a lot of the red tape and failure modes. I think that's definitely something that I'd love to look at and try to build.

Jeff Wang: Yeah. I think it's funny when you when you look globally, there's different levels of awareness. if you look at Bay area, it's like really up to date. exactly what happened in the last hour. Everybody knows what happened. But then you start traveling around, even like Midwest, East Coast. they probably don't even know the difference between, sonnet and opus. They're which one do I select? This one seems more expensive,. so I think, another thing is and that's. And then, by the way, contributes to what you're talking about with like the red tape and some places only went on-premises. Right. They won't even allow code to leave their, their region. Right. So these are all problems that we're dealing with as well. And yeah, and these are things that are top of mind to us to how do we like get around that and help make them successful.

Nancy Wang: Yeah. Wow. I feel like we covered so much in this episode. And definitely I think I'm going to go back to, to Devin for the UX loop that you just talked about. I'm going to try this afternoon.

Jeff Wang: Awesome.

Nancy Wang: Great. Yeah. Thanks so much for coming to our studio, Jeff. And thank you, Richard, for being our guest co-host today.

Richard Liu: Thank you.

Jeff Wang: Yeah. Thank you for having me.

Zero-Shot Learning is presented by 1Password.