(no title)

49 MIN · YOUTUBE · HTTPS://WWW.YOUTUBE.COM/WATCH?V=HA7LZD7ZK3M
https://www.youtube.com/watch?v=HA7lZd7zk3M

SUMMARY

Guillermo Rauch, founder of Vercel and creator of Next.js, discusses his journey from teaching himself to code in Argentina to building a multi-billion dollar infrastructure company. Vercel focuses on enhancing developer experience and leveraging open-source tools, enabling seamless deployment of applications, especially in the era of AI and coding agents.

- *Vercel is valued at $9.3 billion and has revolutionized developer infrastructure.*
- *Guillermo taught himself coding and English through software manuals, starting remote work at age 11.*
- *The company emphasizes open-source tools, making technology accessible to a broader audience.*
- *Vercel aims to simplify deployment for developers, allowing them to focus on creating user experiences.*
- *The rise of AI has dramatically expanded the market for software creation, with coding agents becoming essential.*
- *Vercel's infrastructure supports the deployment of AI applications and agents, shifting from traditional web pages to more dynamic, agent-driven interactions.*
- *The future of software is seen as increasingly agent-centric, with a focus on automation and efficiency.*
- *Vercel's growth is attributed to its ability to adapt and leverage existing technologies, creating a comprehensive ecosystem for developers.*

Well, tonight's guest needs no introduction. I'm sure a lot of you have shipped code on on Vercel. Um Guillermo is obviously the founder of Vercel, but also he created Next.js, the most popular React framework in the world. Vercel is now the most important developer infrastructure companies valued at 9.3 billion. This next part that I'm about to tell you, even I did not know about G. Um the best part about your story Yeah, go ahead. >> is um that you grew up in uh a suburb of Buenos Aires. You taught yourself to code as a kid.

Learned English by reading software manuals. Typically, it goes the other way. >> Yeah, I had to. There were no manuals in in Spanish at the time, or very few. >> And you were doing remote JavaScript contracting by the age of 11. >> Yeah. >> Uh you dropped out of high school, moved to San Francisco to pursue the startup world. My And this is this is it gets even more fun. So, you needed an O-1 visa to get here.

And so, you wrote a book. The book is the most popular book uh called Smashing Node.js uh um with Wiley in 2012. And you know, you lit I mean, you literally wrote a book to get the visa. >> Yeah. >> Um >> I did a lot of things in hard mode. Like, dropping out of high school is really bad for getting a visa in the United States, it turns out. So, some things it helped me a lot, and some things like, "Oh, it turns out people place a lot of value in getting a degree." So, don't drop out of your degree.

>> Good. >> Or maybe drop drop out, but there's pros and cons. >> Dog River is happy about that. >> [laughter] >> Um And so, this house high school dropout from Argentina is a is is is uh is the guest for today's class. Please welcome Guillermo Rauch. >> Yeah. >> [applause] >> Guillermo, thank you for joining us. I got your slides up here. I'd love to have you present them for us. >> Awesome. Okay. Thanks for the intro.

Thanks for having me. I'll try to make this as quick as possible so we can get into Q&A and a dynamic conversation. So, uh for those of you who don't know about Vercel, we've built a vast ecosystem of tools, especially open-source tools, that have now shaped a lot of the modern web and the modern internet. So, even if you don't use Vercel directly, um my most recent favorite example is if you order a Big Mac from McDonald's, uh you're using Vercel. If you're ordering a Porsche, you're using Vercel. Not that you're doing it today, but maybe after the SpaceX IPO. Um so, anytime you're using the internet, you might be interacting directly with Vercel through pages, but also increasingly, if you use systems like uh Open Evidence uh or Grok, you're interacting with agents hosted on Vercel. So, we've created a uh whole set of tools and frameworks that what's I think relevant to today's conversation is because we bet on open source, uh and that came from my background as a guy in Argentina that wanted access to free tools and free information in in teenage years. Um I wanted to make my technology accessible to as many people as possible. So, open source was a very easy decision for me. But then we built a remarkable business on top of open source, which is also kind of contrarian. There's been examples, Databricks I think spoke to you guys recently, but we're able to build a formidable uh infrastructure business that basically sits on top of these tools and allows you to deploy, secure, scale uh pages and agents, as I like to call it.

So, something that's super interesting about the journey of Vercel is that when I started the company, my obsession was developer experience. Uh, there were ways to deploy. There were AWS, Google Cloud, and Azure. But, they were kind of a pain They, uh, made it really hard for me, even a seasoned engineer who had been studying the blade for decades. I sat down before I started this company to deploy the website of my new startup at the time, and it took me weeks, and I realized it was kind of like the inventor epiphany. It can't be this hard to deploy a freaking website using the latest and greatest technologies of the time, uh, which were React JS and and Kubernetes.

And so, I decided that group, JavaScript developers, was a really large group that I decided to address directly. If you knew how to create a front-end project, so like the user-facing side of an application, I wanted to give you the superpower to deploy, and not just deploy like an okay thing. I wanted to deploy something that could scale to the entire size of the planet, that was fast everywhere. And so, from that little, uh, I could say set of people, we grew out into what today is Vercel. But, what's changed since then is that AI, by the way, the math that I used to do is there's maybe 20 million developers in the world that could basically fit this uh, what I thought was like pretty relaxed constraint.

I don't know if you remember, but at the time, something that was really, uh, hot in the Silicon Valley was, uh, coding boot camps. >> Right. >> Yeah, you could learn React in 3 months and get a job. Like, people would like tell you this. And it was true. Like actually the learning curve of modern JavaScript tooling was pretty low. And so it was pretty awesome that I could tell, "Hey, if you know how to create a compelling user experience, I will

take care of all of the infrastructure for you. And it will scale. You will never go down. You will never have to worry about load balancing and and low-level infrastructure." And and so Vercel will will do that. But what's happened since then that's been really exciting is AI and coding agents have created a massive expansion of the people that that could create software. In fact, if we go even further back in time, developers were like a very very tiny group of people that had access to the mainframe computers in universities. And then the story of programming and the story of technology is a story of expanding access to more and more people. So a lot of what we're going to be talking about today is that AI has created them the biggest expansion and revolution in the total addressable market of software and specifically the creation of software in the history of our times. And so concretely what's happened to Vercel is that since October last year, especially with the arrival of a very specific coding model, which is Opus 4.5, we started to see that more and more and more people were deploying to our platform. We became sort of the peanut butter and jelly of coding agents. The whole world is being plastered with peanut butter. That's the coding agent. And you need infrastructure to deploy the software.

So one of the insights that I had when I started the company is that writing code doesn't make you special. Deploying code and putting it in front of a customer does make you special. The learning really begins when a user is confronted with the running version of your code. In fact, I was kind of perplexed that GitHub and before GitHub SourceForge and if you go further back, there was something called fresh meat. Uh people got really excited about storing software in in lines of in the form of lines of code and you could go to any GitHub repo and it's actually kind of hard to run them.

We have the world is made up of piles of piles and piles of software that doesn't run. And so my thing was that people the people that will win the world are the ones that don't just write software, they deploy it and they run it. And so what's been exciting about the rise of coding agents is that coding agents don't succumb to this bias that we have sometimes as humans which is that the code works on my machine and I keep it in my machine where I feel safe with the code in my machine. It coding agents love to deploy and and so we're sort of this jelly layer of deployment of the software that the coding agents write.

And so the other thing that this has taught us cuz we're watching this world unfold in front of our eyes is that the very essence of the cloud is changing. You were joking about Elon web services. So I challenge that if Amazon started AWS today, I don't know that if they would call it Amazon web services because the entity that people are interested in creating and shipping today is an agent. So they'd actually call it AAS.

Amazon agent services. And so what we're doing today with the platform is we created a whole set of infrastructure primitives and frameworks for what's going to be the next entity or object that people want to deploy to the cloud which is AI applications and agents. And so some examples of this are, you know, we used to talk a lot about the wisdom of of Amazon that when you go to amazon.com which is a fantastic experience. Like I buy a ton of stuff there. One of the things that they they unlocked is that if they load pages really fast, people buy more.

What a concept. If you have really fast front ends, people buy more and convert at a higher rate. Not only did they observe that, their data shows that for each 100 milliseconds of slowdown, they dropped conversion by

1%. So, a lot of the internet was built on this idea of instantaneous request-response. We're now seeing with our agentic cloud, this is changing. You don't always get a response immediately. So, we need to create new infrastructure for this long-thinking streams, where the agent goes off into the world and does work for you. It started by taking multiple seconds, which is already pretty different from the kind of infrastructure we used to deploy. It then became minutes. It then became hours. And now we're seeing agents that could cook for an entire day before they get you your report, before they get you an analysis, before they build you software.

We're hosting platforms on Vercel today that are creating companies for people. And And they're deploying agents that maintain, advertise, grow, and scale these companies behind the scenes. And so, the there's an inversion of the old world of pages towards agents. And the kind of compute that we're running is changing quite dramatically. Uh another thing that's been fascinating is a lot of the cloud was built on this amazing compute product called EC2. EC2 popularized the concept of if you want a computer, all you need to do is put in your credit card and you get an instance of that computer. And if you need a million computers, you just need to pay for a million computers. It's elastic compute.

So, elastic compute was designed for human-written code. So, you can think of it from an economics perspective, how much compute I can sell to the world was bounded by how many programmers do we have? And so, I mentioned earlier, the number of people that can create software is expanding by maybe like 10, 20, 100 times. But now agents are writing software, are writing software, are repairing software, are securing software. We're seeing obviously the consequences of really smart models in cybersecurity.

Uh we have hacking agents. We have good agents, like bad cops and good cops out there in the world. And so we're seeing a massive demand for compute for agent-written code. Um and the other thing we're going to talk about a lot is that tokens are the new hot commodity. Right? It used to be that Vercel only allowed you sort of stream pixels. So what you get is is a UI or or some kind of software application. Now we're sort of streaming intelligence in the form of of of tokens, which is also changing the pricing models and the business models. You've probably heard a lot of conversation around the death of SaaS and the death of the seat-based pricing model and the rise of the token-based uh pricing model, which measures intelligence.

And by the way, this is not to say that it's a transition where we're over-rotating on one side of the equation or the other. I'm actually super bullish still on human-centric experiences. So you will go and visit brand experiences. You will have to see the result of what these agents cook in some kind of rich environment. In fact, I even have uh you know, sort of predicted the return of a more whimsical web and more whimsical internet. When I When I grew up, there was this really cool software Microsoft Encarta.

It was an encyclopedia that was alive. So you would you would select a topic and it was like this interactive rich experience. So if you were to study the pyramids of Giza, it was like this super cool experience. Now Now that we've made the cloud and all this stuff, you go to Wikipedia and you get this like wall of text and maybe you're you're dignified with a photograph on the right-hand side. So I actually predict that we're going to get the most immersive, craziest, uh pixel-based experiences through video generation, through 3D model generation.

There's a couple platforms on our side that are doing really cool just-in-time 3D rendering. So, human-centric experiences will matter a lot. Um but our new conception of the world is that we're making this transition towards what we call agentic infrastructure. So, agentic infrastructure has three sides of it uh in a beautiful triangle. So, number one, agentic infrastructure is the infrastructure that you need to give to your coding agent. That peanut butter uh and um and jelly metaphor that I gave you. So, if you're using cloud code, if you're using Codex, if you're using V 0, those software coding agents need to deploy somewhere. So, that somewhere we're calling it agentic infrastructure.

The other thing that's really important is you will be building your own agents. So, just today someone came to my office and pitched this amazing vision of what could be an AI-native school of the future. And a lot of what this person said resonated with me because they're rethinking education from an agent point of view. It's still involving the human, the teacher, the school, the student, but what they're going to be shipping is an agent, not a set of connected web pages.

And perhaps one of the most exciting things is this idea that it's going to be automated by agents. So, I'll cover the first principle quickly. Um the fact that you need infra for coding agents I know that you've asked about some of the numbers of the growth that we're seeing. One of the fascinating dynamics is when you deal with a coding agent, the coding agent has a preconception of the world. It has a world model sort of behind the scenes. And one of the things that's given us tremendous um um you know, tailwinds is that models have learned from the vast knowledge that exists on the internet about Next.js, React, our open-source software. In fact, this report that came out calls, for example, our UI engine, we call it shadcn, uh it has near monopoly status at 90.1% and when you decide to deploy React or Next.js, Vercel also has near monopoly, which I love that how they qualify that for regulators cuz it says also it's 100%, but I guess it's a near monopoly.

So, anyways, the the coding agents have sort of made up their minds in to some extent and this is not cemented by any means that these are really good tools to use when you give it a certain task. So, for example, if you're creating a SaaS app, it's very, very likely that it's going to use shadcn because we created infrastructure that the coding agents can use. And you might be asking the question, and I certainly have asked myself, why do agents even need to reuse software that already exists?

There's an economy of there's an efficiency bias here, right? In theory, a coding agent could reinvent the world every time you ask it something. In order to produce an apple, it has to reinvent the universe. It could write its own Linux kernel, it could write its own networking systems, it could write macOS. And so, betting on infrastructure as open has actually helped a lot because now the coding agents have a target to throw code on top.

>> It's like a LEGO block. >> It's exactly. And in fact, there's a really cool article by Mitchell Hashimoto, founder of HashiCorp, who's recently joined the Vercel board. He calls it the world that we're going into now is he calls it the block economy. >> Mhm. >> You need to be thinking about what are the building blocks that you'll be able to give the agents. If you want to participate in that, meaning, "Hey, how do I make it such that Claude Code or Codex or or or Grok CLI choose my technology?" You want to be thinking about blocks that fit agentic

ergonomics. So, that's one.

And the other one I was mentioning is the software of the future will be agents. I For example, at Vercel, we have created a support agent that now answers 93% of user inquiries. It's made our business massively efficient. It's improved customer experience. So, we measure if customers are happy talking to an agent versus talking to a human. 93% It It's also allowed us to give free support to a much larger number of people. And so, that thing that we shipped using Vercel was an agent, not a knowledge-based web of hyperlinks. Uh so, we've basically created the tools, the blocks, and the infrastructure to build agents.

So, I'm not going to get into the technical details, but I'll tell you something that's fascinating is there are a lot of metaphors with the old web. For example, whenever you go to production with a web system, you typically have a CDN in front of your system. The CDN market, the companies like Akamai and Fastly, they emerged to scale, accelerate, and secure the delivery of pages and pixels.

We're seeing the same happen with tokens. We created the AI gateway, which is a category-defining product. It's like a CDN for tokens. Tokens that you get from Claude, from Anthropic, from Gemini also need to be observed. They need to fail over. They need to be secured. And they need to be accelerated and cached in many cases. And even what's happening now that's super super exciting, we can load balance them. Uh you might have heard a meme of how expensive it is to ask to say to GPT 5.5, you say, "Thanks."

And you just activated like 300 GPUs to say, "You're welcome." And so, imagine a smart CDN that says, "Hmm, if they ask me, 'Thanks,' I have a smaller model behind the scenes that can just say, 'You're welcome.'" Right? Or a semantic cache. So, there are a lot of metaphors from the infrastructure we had to build for the first chapter of the internet to this agentic internet. And so I love the CDN example. Sandbox I already spoke about, which is like the EC2 or the fundamental compute unit of of agents.

To give you a very sort of non like basically technical explanation is agents get more powerful or you extract more IQ points from a model if you give it a computer. So models are being trained to use computers. In fact, in the post-training phase, models are handed Docker containers with play grants. So models sort of like cut their teeth in mini computers before they see the world. And so when you use a model you'll get better performance out of the model by giving it a computer. And this is kind of not unlike humans, right? The average knowledge worker that we hire, what is the first thing you do when you join a modern company? We give that person a computer. IT, here's the laptop. You can pre-install with your software, etc.

That's the same thing we're doing with agents. We say, "Agent, here's your computer. Knock yourself out. And I IT also pre-provision some software for you to be super effective." Now, much like the personal computing revolution, it came with viruses, it came with fishing, it came with Nigerian princes promising, you know, fortunes that you can claim with one click. And so agents can also fall for the trap of exfiltrating your data, leaking your data. So we're creating a whole set of security products to protect these sandboxes and computers.

So you're going to see the emergence of again entire categories of of products and companies that are tailoring

cybersecurity for agents in in in many other matters. Um, and the last and perhaps one of the things that I'm most excited about since we're talking about Tesla full-self-driving and self-driving cars and Waymo. Um the cloud itself will become like a self-driving car. If you have some experience running software at scale, you know that it's plagued with things like holding pagers.

Um it's a horrifying experience. So, I have an anecdote that I reuse from the founder of Stripe. >> Mhm. >> He says that anytime he hears ducks, his cortisol levels spike. Why? Because his pager duty ringtone used to be ducks. So, it meant that his Stripe was crashing. And so, anytime he hears ducks, he's like on edge. And so, the average experience of maintaining software and scaling it in the cloud is actually pretty dramatic. Like, you get paged in the middle of the night, a data center went down. And Vercel, to a great extent, automated away a lot of that pain, but pain is still exists. I'm not going to like sell you things that are not real. Like, you have to sit down and monitor things. You have to make sure that performance doesn't degrade over time. And so, this is my vision of this self-driving car of the cloud where everything configures itself, everything optimizes itself. We're all going to live a great life because the agent will be doing all of that work behind the scenes, and it's going to come to you and say, "I just optimized all your software. I made it twice as fast. Here's a PR." Or maybe I already shipped it, and I measured that it improved conversion for your customers.

So, this is a third leg of this agentic infrastructure thing. Uh so, some really quick customer examples. Meta had already built infrastructure for years and years and years. Uh many people that have, you know, studied in these halls have gone on to work at Meta and create incredible developer infrastructure, but they found that Vercel's infrastructure has had this edge. They need to move really fast. >> Mhm. >> Their engineers were using coding agents, and that's how Vercel got in and now runs a lot of Meta super intelligence labs. In fact, we've helped Meta move faster by not having to procure so much software.

>> Mhm. >> They they basically quote-unquote vibe coded tools that have allowed them to train models faster. They shipped meta.ai much faster than they would otherwise have thanks to agentic infrastructure. Um Notion is also investing heavily in their chapter two. Fantastic tool, I use it every day, but Notion itself is becoming agentic, which is that prediction that I mentioned, like all software of the future will be an agent. So, Notion is also using Vercel infrastructure to add these AI and agentic capabilities. So, for example, right now, we could have a Notion transcribing agent that is taking notes from this class, and that infrastructure is running on Vercel.

Um So, long story short, we're basically building what you could think of as the AWS of AI or agents, and it's a full-stack cloud from developer tools all the way to infrastructure. And uh now we can get to Q&A. >> Awesome. Thank you, Jay. The um >> [applause] >> software is dying. Vercel feels like the machete that is like chopping through the forest. This is what people are building vibe coding on, and you must hear hear about all these things people are vibe coding, all the software that people have stopped procuring.

Is there are there some examples um of let's say public software company products that have been vibe coded on V0 or Vercel that people customers have replaced their public uh software, the bigger the better. >> Yeah. One of the things that I always think about is the best software will always be the one that's more most tailored to

you. >> Mhm. >> The the kind of uh weird thing about SaaS is that on one hand, it took a lot of really, really intelligent people, product managers, designers, to get themselves in a room and say what is the common UI that'll make the biggest number of customers happy?

>> Mhm. Mhm. >> And then we'll give it to the world. And then they hope that they get this exponential adoption because I show you for example an expensing tool and say, "Yeah, I can fit my business to that expensing tool." And then I give it to another customer and say, "Oh yeah, maybe you only have to translate the labels to my language, but I can also sell it to that person." But that's a still lowest common denominator software.

And and it requires this sort of intense design process of what is the set of pixels that are going to make the number the highest number people happy. I think what's happening now, I wouldn't call it a death of software. It's I will say, if you use Vercel, software has never moved faster and been more alive. We've seen examples where I heard this recently from a startup customer. The lead engineer told me, "Man, your platform is dangerous. My CEO is shipping software. He replaced our parking lot management software with something he live coded in V0 and he saved us a ton of money."

>> Yeah. >> And so you have that example of like, yeah, I mean who I mean how many smart Palo Alto based companies are creating a parking software, right? Like not many. And so there's a whole category of software that we've been living in a world that's just not ideal. It's just not high quality software. And now you're one or two prompts away from being able to create a really good version tailored to your problem set.

Another thing that we see is the emergence of All right, the system of record or the database stays but I can create a presentation layer that is highly tuned for my business needs. So what something that might be surprising for people to hear is that we within Vercel, we kind of reinvented all all Salesforce. All of it. Like all of our sales reps >> Mhm. >> when they need to uh learn about a an account, >> Mhm.

>> an opportunity, when they need to read uh business intelligence about like how should I pitch, >> Mhm. >> all of this is being generated and it took a team of like two people >> Mhm. >> to create a new version of Salesforce, which is kind of crazy to think about. Like how big is like Salesforce like uh stock market-wise? Like it's hundreds of billions, tens of billions of dollars, right? >> Yeah. >> And now to think, oh, I can just generate a version of that that's like tailored to my business. But by the way, we're still using a lot of workflows and uh uh databases that Salesforce set up. So, it's not a and/or situation.

>> Mhm. >> It's you can incrementally make software a lot more plastic, malleable, and tuned to your needs. So, on on one level, I think people are underestimating just how many off-the-shelf applications will be thrown away and replaced with this incredible velocity that software generation is giving you. But on other levels, I think you're overestimating that creating the underlying system of record, the ACL system, and the access control layer, all of those things we can reuse. So, the companies that expose themselves with the right agentic interfaces, like MCP, like CLIs, and and other APIs, those are the SaaS companies that will fit really well into this world and will not be, I guess, killed by by the coding agent.

>> Yeah, fascinating. Well, we'll have to get a session of advice for software from from from the creator of Vercel. Um you know, your business you had a nice chart there that has changed quite a bit since Opus 45 last quarter Q4 25. Um maybe outside of the volume, are there other parts of your business that have changed as Yeah. Coding has gotten so good. Maybe maybe maybe you know, retention is the biggest thing people are talking about with with a lot of the vibe coding apps is like, hey, well, you're making such custom software. Um, do people still need it? Like, hey, it was a good thing to have on a Saturday morning, but do you still need it on a Wednesday morning?

>> Yeah. We're seeing all of it. Like, we're drinking from the firehouse, right? So, we've seen software that is useful for one customer call. And that's fascinating. We have a lot of customers that have purchased Vercel and V0 because they're sales engineers and they've accelerated pre-sales so dramatically. They can come into a customer conversation with a custom version of their software already built. Imagine the difference between I show you a slide deck of what my software company could do to a prospect or I show you living, breathing software.

And so, that is an example of throwaway because there's no illusion of like, three calls later we discover something else and we threw it all away. So, in that sense, you're right, software is dead or like throwaway or I like to say that software is basically now free. >> Yeah. >> And and so, there is a but that actually drives a lot of engagement because of what Toby Shopify has called the reflexivity of AI. Once you know that you're one prompt away from communicating with another human being with a high-fidelity piece of software, prototype, example, demo, you will never forego that. I really think I wrote I actually wrote an essay many years ago called, it's hard to forego efficiency.

This is giving companies such a high degree of efficiency. I have engineers lamenting, oh my god, I'll never be able to code the old way again. Like, my brain just doesn't work that way. And so, but also, let's, you know, also be clear. So, there's a lot of agentic engineered engineering that's still really hard. A lot of the infrastructure software that we build, really hard. Sometimes we have to uh, you know, summon a quorum of three agents plus a lot of smart humans like look at a single line of code and tell us what's true or not. And so there's a lot of software that continues to be extremely long-lived. And and so the really big difference has been the audience.

>> Yeah. >> I have customers reaching out for support sliding in my DMs on X saying, "I ran into this error. My friend, I don't even know what Vercel is. My agent took me here. I deployed. I was really happy, but now here we are." And then the way that I help them is by "Hey, can you introduce me to your agent?" >> Mhm. >> I ask them, "Can you show me the transcript so that they can bring it back to our engineers? We maybe we can turn it into evals and we can understand how did my customer's agent get into a bad spot?"

It used to be a much more direct conversation. I would get introduced to the engineer and the engineer would tell me, "Yeah, man, I went to the docs. You said this was the API. It was wrong. This is the error." >> Claude now. >> It's we're going to end up in a world where it's just agent-to-agent communication. The customer's agent files a feature request or bug report with my agent and my agent says, "Hm, how do I prioritize this among all of the tasks that we have with this token budget that we have with these deadlines and the bias of my CEO

that wants me to ship this other thing and software emerges."

>> Yeah. >> I'm going to pull up a slide you had here which was the peanut butter and jelly slide. I think that's a phenomenal framing. And the big achievement you have here is actually could you break this down for us? So 86 out of 86 times and let's say Claude picked the Vercel deployment >> Yeah. >> option and your UI component options that shadcn. >> Yeah. >> How did you I mean that seems like a slam dunk.

>> Yeah. >> How did you guys land up there with a 100% or 9200 90% market share on the UI components, 100% on deployment? Is this a Was this an enterprise deal with Cloud? Was this show good that the agent was like, "Hey, I got to do a meritocratic search?" Real talk. >> Well, there is that, fascinatingly enough. Yeah. >> W-What is it? >> There's all of the above. So, there is a >> All of the above including the deal or >> It might be a deal. Although, I can't confirm or deny the existence of deals in the direction of expanding access to Vercel. But, um what I'll say is on one hand, there's cause and effect.

>> Okay. >> As I mentioned, for many, many years, we created uh huge amounts of content and and frankly, really high-quality APIs that work well for humans and agents alike. >> Like SEO stuff. >> Uh no, no, even before SEO. So, I'll give you an example. That thing, Tailwind, that is underneath uh Vercel. >> Yep. CSS. >> I bet really hard on that. Uh I'm not going to say it was like super the earliest person to bet on it, but I this acknowledge was extremely controversial in the human developer ecosystem because it looked weird.

I don't have Has anyone used Tailwind before? Raise your hand. Okay. So, I'll explain it. Code has an aesthetic sense to it. You look at it and it can look symmetrical, pretty, nice, or it can look like a piece of junk. Tailwind kind of moved us in the direction of piece of junk. Like the lines got really long, which for people with extreme OCD like me, I had to overcome biases and bet on truth. So, what Tailwind gives you is a property called local reasoning.

So, when you when you design a component and what I mean by a component is Let's say that I'm designing uh like this UI, right? Like this uh slide UI. It has a bunch of components. We can call this the slide preview component. Yeah. Tailwind allows me to it in a way that is extremely future-proof. I can reason about its design in a way where like if I take that component and give it to you, you can even insert it in another part of Google.

And the component works perfectly. So, it created an economic scalability to the code. >> Mhm. >> And I bet hard on that. >> Mhm. >> And I overcame my own like sort of like gag reflex of how code looked. By the way, if if the Tailwind guy listens to this, like he knows. He's he's heard this feedback before. So, that's I think when we started moving from this human-centric uh code design to what actually scales better for humanity and organizations and economies design. And so, a lot of the technologies that I've designed, like Next.js, has this local reasoning property.

>> Mhm. >> React has it in spades. The The Facebook team arrived to the same conclusion. As they were hiring more and more and more engineers, they needed systems of scale to the code. And so, there is a lot of things that we've put into the design of the APIs that agents such as eaten up. >> Mhm. >> Uh for reasons of for example,

the context window. >> Mhm. >> You can't fit all of the code of humanity into the context window of an LLM.

>> Right. >> Today, we have 1 million. And so, this local reasoning property of the code became really really important. >> Mhm. >> Uh there is the content >> Mhm. >> for training and for grounding. So, when a coding agent kicks in, it Googles, right? >> Yeah. >> Um there is the the kind of thing that this agents create. I mentioned that you can create your own Salesforce on Vercel by still using Salesforce as an API.

>> Mhm. >> You can use Salesforce headless in Vercel to host the application side. And so, this technology set is also fitting really nicely into that world. So, there's basically a confluence of factors that have gotten us here. >> Very composable. >> Yeah, composability being is a great summary. Composability is a key prerequisite for this agentic scalability. >> Fascinating. Um another slide that got my fancy was how much you're doing. So, this is a lot of things. There's independent companies whose full-time jobs Yeah. is to do one of these things.

From sandbox to chat to workflow. And this would imply that you were competing with a series of companies across the stack. >> Yes. >> Down from maybe not quite bare metal, but like just above bare metal to to to all the way to the to the to the UI elements and obviously deployment. >> How did you decide to do it all? >> Yeah. >> Do it all in air quotes. What parts of it would you give yourself a grade of like A+ and maybe less than A+? And where where is like the most competition you're you're you're you're facing?

>> Yeah, I only brought you A+ products on this slide. I think >> [laughter] >> I think there is a lot of things that we frankly throw away or deprioritize over time. I think even hopefully before they see people's eyes. >> Yeah, but a bigger design choice to do the whole thing. >> Yeah. >> This is kind of the Steve Jobs you got to do the whole thing from the back of the >> I really think so. I really think so.

You have to do the whole thing for the thing that matters. Agents matter. Agents are probably the last class of software. And I'm building the tools and and infrastructure services to build agents. We can't not participate in the most exciting, fastest growing economy in the world. And there's a couple things. One is the Vercel product development philosophy. The Vercel product development philosophy is we build by dogfooding our own platform.

So, AI Gateway is this CDN of tokens. Mhm. built on Vercel. In fact, we kind of open source the recipe so that you can build your own competing uh token gateway. >> Like open router or something? >> Yeah, you could build another one. And there's companies that have built different ones that made different trade-offs or differently integrated into their services. So, it uses fluid compute, which is our compute platform. It uses the global vast network. >> Mhm.

>> It basically reuses a lot of our CDN, right? So, to your question about why would I go into that business? Well, I was able to reuse 95% of what the the rocket engine. >> Mhm. >> Right? It's the same rocket engine and same rocket fuel that was powering the CDN of Pixels or Pages. Um Sandbox, perfect example. So, I told you that the Vercel platform has seen this insane growth in popularity in a very short amount of time. And what typically happens when that happens to startups or even scale-ups is that everything breaks.

I'm happy to report that almost nothing broke in Vercel. Obviously, like no service is perfect. >> Mhm. >> But, our Sandbox reuses the same uh virtualization primitive that powers every deployment made on the platform. And so, we're able to leverage our operational expertise in scaling compute, which I I cited another metric. I mean, we three-Xed in a few short months. Uh we even doubled the number of daily deployments since January.

So, we're we're producing all of these ephemeral computers that then we destroy >> Mhm. >> every time a deploy happens. >> Makes sense. >> And so, Sandbox basically reused all of that compute expertise and virtualization expertise. And so, it was a very obvious bet for us to make as well. >> Fascinating. Um the question I'm about to ask you I'm actually very excited to ask you the question because of how much you've done underneath the product. And obviously, beautiful products being built. You know, one of the biggest questions that we ask in this class is where will value accrue?

>> Yeah. >> From chips to data centers to in infrastructure even below the model, the model, and the agents. >> Yeah. >> And frankly, it feels like at least in 2026, right this second, there's a lot of lot more value be accruing to the stuff below the model, chips, data centers, power, cooling, energy. Um above the model, you've got some value in in the coding models, you've got some value in in in customer support and and legal and others, but it's it's you know, it's it's very concentrated.

>> Yeah. >> Talk about that for a second. Where do you see value accruing based on everything you know that's being built on Vercel? >> Yeah. My sense is that the you you've seen the figures of like how much of the token economy is going to coding agents, right? >> Mhm. >> Coding agents seem to be one of the most promising paths to AGI or if they might just become the the path to AGI. And so what's great for us is that we built the infrastructure that, you know, again, coding agents need in order to actually do useful things.

>> Right. >> And I think there's going to be a lot of opportunities like that. Like at the end of the day, what determines whether we're in a gigantic bubble or in the most exciting chapter of humanity is whether we're delivering useful >> Mhm. >> services and products. And so it turns out that for the model to be useful, it needs a sandbox. >> Mhm. >> It needs a deployment platform. It needs a domain name. Actually, I was hearing from the Codex team the other day that uh one of the reasons that uh and this goes into this world that you're sort of alluding to of like, how do I understand what agents are doing? Or is like agent engine optimization.

>> Mhm. >> The Codex team was telling us that a lot of people want to name their creation. They want to give it a domain name. >> Mhm. >> And so they're actually finding Vercel first through a domain name, which is actually, you know, I I made an early bet on making it extremely fast and easy to purchase and configure because DNS is sort of hell. And I And my intuition was when a human has an idea, I I we're in a very entrepreneurial room here. A lot of us go and buy the domain name first. I have so many domain names that I don't even know what to do with them. Sometimes up on renewal, I feel bad. And so, I bet >> What's your What's your What's your budget? What's your annual budget on >> It's embarrassing. Like it's a lot. But it's I tell myself it's digital real estate, so it's worth it. Uh and uh and so, we we made that bet of like, "Oh, like when you have an idea, you're going to need this thing. You're going to need DNS infrastructure. You need domain names, etc." It turns

out to have, you know, returned very handsomely because whenever a new user of this software world has an idea, they're coming to Vercel. Our obsession is, can we become the front door to any emerging idea on the planet? I think every business that is going to help these ideas come into fruition and be useful is going to do really well. I think security is going to be a massive consideration of this world. You know, I need to stay focused and and we're we have a very uh uh promising product lineup. But I mean, there's just so much left to build. And and a lot of what I'm thinking about these days is not only how to make these agents useful, but how to govern them, how to give them guardrails, how to make them secure, how to secure the broader internet. So, a lot of our mission, I mean, we just can't stop hiring engineers. I think there's a lot of talk about like, is it over for engineers? And I can tell you, there's just so much more to build to to serve this amazing demand that's happening just from the coding agent uh use case itself.

>> Fascinating. Um rapid fire last couple of minutes. Um pick a business that you're long that you think are very you're up to very optimistic on, and a business that you're short on that you think based on everything you know might not it might not make it. >> Um short on you know, I predicted the downfall of I I mean this is an obvious one now, but I predicted it pretty early. Like any static data or content business is is kind of cooked, right? Um the classic example at the time was like a Stack Overflow, which was this aggregator of this database of like programming questions and answers.

>> Mhm. >> Uh I'll tell you a category don't want to like throw companies under the bus. A category is anyone that's thought that code was scarce or difficult to produce. There's a whole category of like drag and drop builders and >> Mhm. >> uh basically it's like coding with training wheels. >> Mhm. >> I always despised that category because I never liked to be looked down on. Like I always found that kind of software patronizing.

>> Yeah. >> Too opinionated. >> Opinionated, constraining. I'm like I'm like a little kid and like I need to be given this like little interface and code is scary and so there's a ton of companies that were built on that foundation and they're they're going to have to have some serious like pivoting conversations and whatnot. >> Also the companies that are not opening up. One amazing consequence of coding agents is that they want to access the raw signal.

They want to go straight to the data. >> Mhm. >> They don't want to like talk to sales and let's chat about your requirements for 3 months and what are your pain points? No, they just want go. Uh consumption-based pricing. >> Mhm. >> Instant sign up. Start getting tokens right into my blood stream. Like that kind of thing is going to win. Uh but a lot of the internet economy is predicated on oh, we don't really have a product. Uh we have this uh e-brochure uh and you have to talk to the enterprise rep for, you know, 3 years and then maybe some software will emerge.

You don't forget how much of the software ecosystem is still that. And so I'm very long on anyone that's moving at the speed of tokens. And so we're doing so much work just to meet the demand of uh so one one of the write-ups that I made internally the company recently is, you know, as an engineer you're taught to apply rate limiters. Rate limiters are very useful for the operational health of your system. So a good example would be uh and this goes back to that, you know, hypothetical Palo Alto room where smart people get together and say like,

"Hmm, no company will deploy more than 100 times per minute." So okay, rate limit, put it into the code, 100 times per minute.

Nowadays, uh no one can really know what demand or right quota is. You do need to be super worried about abuse and KYC and things like that because I'm backed by supercomputers. So I can't have anyone immediately like send them a bill for you just spent a trillion dollars of supercomputing power in 2 seconds. >> [laughter] >> Can you pay me, please? But I told the company, "No more rate limits, guys." As a provocation, of course, again, for operational health reasons you do need to have some things in there.

And that goes hand in hand with this pricing model of you pay for what you use. And so why would I rate limit you your ability to use more resources? And so right now, and this is a good problem to have, I have this massive agent platforms where a a YC company that didn't exist 3 months ago is producing so many deployments on the Vercel platform that they're surprising us about, "Oh, I don't remember we had that rate limit." We couldn't have imagined that there was going to be so much demand for a certain piece of infrastructure. So, bullish on the companies that can serve that kind of demand.

>> Final question before we wrap. If you were not building Vercel right now, what would you be building? Asking for a friend this class. >> Space tech. Obviously, like can we get into Mars as soon as possible, please? Yeah. >> Why? Why? Why the rush to get to Mars? >> I I I mean, my bias has always been I love the multi-planetary species aspiration. I've always thought about it as like I'm a high availability multi-AZ, multi-region, multiple layers of failover guy.

>> Love it. >> So, I live I chose my location in San Francisco based on like the highest structural safety and the best and most historic retaining wall that has never collapsed in any earthquake, whatever. Like I'm I'm an infrastructure guy, so I I have to think about these things. Uh so, but also I'm uh really excited about energy. >> Yeah. >> And honestly, what we're witnessing is the that the emergence of intelligence is this bidirectional flow of energy uh energy goes in, intelligence goes out. So, any breakthrough that we can have in fission, fusion, geothermal, what have you is really deeply exciting to me.

>> Awesome. Well, that's a wrap. Thank you so much for your time, G. >> Thank you. Appreciate it. >> [applause]