EPISODE 1317

[INTRODUCTION]

[00:00:00] ANNOUNCER: Latency is the time it takes to get from point A to point B. In programming, this might be the time from when a user selects their photo library to the time when pictures reach their computer screen from the database. Fly.io is a simple platform for running full stack apps and databases close to your users. Some available features include provisioning CPU, memory, and storage based on your apps demands. They also offer zero configuration private networking and global load balancing, metrics and alerting and managing SSL certs at any scale. Fly.io beta is launching highly available Postgres clusters with a single command and creating read replicas in different cities. In this episode, we talk with Kurt Mackey, Social Media Manager at fly.io.

[INTERVIEW]

[00:00:48] JM: Kurt, welcome to the show.

[00:00:50] KM: Nice to meet you. Thanks for having me.

[00:00:52] JM: Yes. So you run fly.io. And that is a layer two cloud provider, or I would maybe classify it as maybe a new cloud provider.

[00:01:04] KM: That's fair.

[00:01:06] JM: Is layer two cloud provider, is that an okay terminology for what you do?

[00:01:10] KM: Yeah, I think so. I think that a lot of what we do is sort of defined by how early we are more than where we want to go with it. But that's a pretty reasonable take right now.

[00:01:19] JM: So the category of layer two cloud providers, I think of it as the first one was Heroku, kind of. The first main was Heroku. And then you got Firebase. Firebase got acquired

by Google. Google pimped it out. Or can you use that term? Maybe you can't use that term? Do you want me to cut that? Or do you care?

[00:01:38] KM: No, I don't care. That's fine.

[00:01:39] JM: Yeah. Okay. So Google made Firebase great. And then there was, I guess, the Vercel Netlify family of layer two clouds. And there's also like the company Spot. What are the other big layer two cloud provider companies?

[00:01:54] KM: I'm probably think about them slightly differently than you, because we came at this from a CDN perspective, rather than starting a cloud on purpose. Does that makes sense?

[00:02:02] JM: So you're from a CDN background?

[00:02:05] KM: Yeah, we actually started the company specifically with the idea that CDN sort of suck for developers. And if you could build a CDN that developers really got a lot of power out of, you'd have something pretty valuable. And then, over time, we kind of landed on this idea that CDNs sucks because they're an entirely separate complicated layer that if you can deploy your apps right, you wouldn't really even need. And so we didn't really start thinking, "Hey, we're going to build like a Heroku like thing," as much as we got to where we were running small, arbitrary applications and appreciate the Heroku developer experience and felt like that was a good way to get people kind of onboard to our platform.

I tend to think of Vercel and Netlify as kind of in the same ballpark where they're responding to. And what they're doing is they're solving a lot of problems for kind of the frontend. And I think we're kind of complimentary to those folks being more backend fullstack stuff. I think the other people who are in the thing, so there's Render, render.com.

[00:03:03] JM: I'm an investor.

[00:03:05] KM: Okay, cool. And then I think Render's truly like Heroku too in some ways. And then I think there's like Railway is another other one I've seen. You see like DigitalOcean doing their own cast thing on top of DigitalOcean, which I think is interesting. But there's a fair amount

of that stuff. And I think everyone's still trying to –I mean, App Engine was kind of one of the original ones too.

[00:03:25] JM: It's such a wide open design space. It's really interesting to me. I have a few like cloud provider ideas I'd like to talk to you about. Do you want to go with the kind of cool one or the crazy cool one first? I got two ideas for you.

[00:03:40] KM: Let's go kind of – I took a bunch of improv classes and they taught me that to make an improv show funny, you need to be like grounded and real at first before you do the really zany zombie apocalypse stuff. So like let's just follow that.

[00:03:51] JM: Awesome. Okay, great. So we start with kind of cool, kind of cool, the kind of cool cloud. We can even call it that if you want. And .cloud is, I believe, a legal domain extension, right? So we can do kind of cool.cloud.

[00:04:02] KM: Yup. Yes. Kind of.

[00:04:03] JM: I know that because I bought Kanye.cloud recently. That's the cloud that's so cool we can't even talk about it on this episode.

[00:04:12] KM: Yeah.

[00:04:14] JM: Okay, so the basic one. Do you know anything about Knative?

[00:04:18] KM: A little bit. It's a Firecracker-based – Maybe not. Just tell me in two sentences Knative.

[00:04:24] JM: So Knative is like the next thing Google built after Kubernetes. So Kubernetes, then they did Istio. Istio is kind of a platform, but not really. Knative is a legit platform. Knative is like if you want to build the Heroku warming, like to infer warming stuff, like you want to do the pooling, like pooling of resource instances kind of thing. Did you have to build that by the way?

[00:04:44] KM: Yes, kind of. We use we use the hacked-up version of Nomad.

[00:04:49] JM: Yeah, you probably should use Knative. Maybe not. I mean Nomad is pretty cool. I would love to have that debate with you. I guess Nomad is totally underrated, right?

[00:04:58] KM: I think so. We like Nomad because we could keep the whole thing in our head and do what we needed to do with it, basically fork it. And so that was helpful for us when there was three people basically.

[00:05:08] JM: What I love about HashiCorp is they just say, “Yeah. You know what? We're not going to do this Kubernetes thing. We don't want to play in Kubernetes land. We're just going to go into Nomad land.” And I just love that attitude.

[00:05:19] KM: Yeah. Yeah. I think it was smart of them. They've inherited some of this stuff. They implemented like CSI and I think even CNI, but it's still very simple and easy to wrap your head around.

[00:05:29] JM: Well, it shows their ambition as a company, that they're willing to just say we're going to say no to Kubernetes. Even AWS said yes to Kubernetes.

[00:05:37] KM: Right. Kind of. AWS is a yes-ish to anything. It's very much – It’s like that seems like a thing we can make money off of. It’s not an opinion. It's an extraction.

[00:05:46] JM: What is the meme? The meme is like, “Everyone else**:** We're moving to Kubernetes. AWS**:** No comment.”

[00:05:54] KM: Pretty much. And then it's on like day two of Reinvent they finally talking about it basically.

[00:06:00] JM: Day two of Reinvent. We're introducing a bad Kubernetes system. It’s delineately bad.

[00:06:03] KM: Yeah, pretty much bad. But it's good enough for our customers to pay us more money.

[00:06:08] JM: Right. Oh man. So okay, my cloud provider idea is basically you – So do you know Google Cloud Run? Have you seen that product?

[00:06:16] KM: Yep. Yep.

[00:06:17] JM: Google Cloud Run is the Google-ized thing that is like it treats Knative as a reference implementation. So Knative is the reference implementation for Google Cloud Run.

[00:06:26] KM: Okay.

[00:06:27] JM: So Google does Cloud Run. Kind of cool.cloud does like a better cloud run. Like let's do cloud run, but not on GCP related flavors. And what you can do is you can actually build it as a company over Cloud Run. So you basically take Google Cloud – Because like GCP has the same problem that AWS has, right? The console is terrible, right? So you just abstract away the console. And you're like fresh. You're basically like Render. You're doing the same thing Render is doing, but you do it on top of Cloud Run. So Google manages all the infrastructure for you. And then you can do the Render trick later on if you want. Like if you want to later on move to your own server infrastructure, you can totally do that.

[00:07:04] KM: Yeah. that's interesting. I think you'd have to, by the way. I don't think you can build a good developer UX on top of public clouds right now. Mainly because they extract so much of the potential money you can make that you're like left with like a fraction of a percent of what companies are willing to spend. And then the rest goes to AWS or Google.

I tweeted about this the other day, and basically like you have to basically raise enough money to give all your money to AWS to then achieve AWS escape velocity and then run your own hardware, or you're never going to be able to build a very good business out of it.

[00:07:34] JM: Okay. So your ideas like deploying app servers close to users. You're a CDN person. The cool thing about CDNs right now is that they're doing this WebAssembly stuff, right? Or what was it? Like WebAssembly CloudFlare worker. CloudFlare workers can do WebAssembly, right? Like that's one of the interesting things about them. So when you enter a

world where you can do like, essentially, a headless browser-powered WebAssembly functions wherever you want them, like subject to the extent of CloudFront or Fastly's reach around the world, how do you build a cloud in that world?

[00:08:16] KM: So we actually started with something like CloudFlare workers. We're using VA, just like they were. Basically, the way that apps run is in just a thread within a process within VA slots. I think one interesting thing about what CloudFlare and Fastly have done is their architecture is sort of defined by their footprint. So like where they deployed servers and the type of servers they deployed. And one of the things we ran into when we're talking to customers is that we were too small to make them care about something like a custom JavaScript runtime. CloudFlare has millions of existing customers already, they can get somebody else to do it on top of. Like they can basically be like, "Look, we've got this product. You can write JavaScript on it now." And people are like, "Whoa! That's great." We didn't have a million existing customers. We had to actually convinced people we were really from scratch. And actually that's what took us away from that model, is we got to where we had so many people running to run code that we couldn't build API's for.

And actually, WebAssembly is interesting, because one of the big customer requests we got was doing things like transcoding video and always seem like WebAssembly would be an answer to that. But even the WebAssembly – Kind of the WebAssembly ready library world is very small. There's not like a WebAssembly compatible package for all of the things that people would want to do. And so part of what took us to what we're doing now, which is we've gone down the stack. So I actually started as a database person. We did hosted databases in my last company. It was called Compose, which we ultimately sold to IBM. But the first kind of step down that path was realizing that there's a bunch of stuff that people do where they need actually real CPU horsepower behind it, particularly for images and videos, and CloudFlare and Fastly aren't necessarily putting real CPU horsepower behind their scripts. They're more like scripting existing infrastructure rather than kind of giving people what I call like actual compute, if that makes sense. And so like I said, we kind of built a cloud by accident just because that's where customers led us. And I don't know that it would have made any sense from scratch if we just started this way, if that makes sense. It's like a product development got us here. And that's why we're here sort of thing. Does that answer your question?

[00:10:22] JM: It does. It's interesting. I'm looking at your dev.to post about building a CDN in five hours.

[00:10:31] KM: Yup. That was – It's basically an NGINX cluster that runs around the world. And so one of the things we end up giving people is primitives like they get on a cloud. So they get compute. They get desk. They get private networking between all of their little things that are running. And so it makes it relatively easy to actually ship something like a CDN, which is not really like our primary business, but I think it's sort of interesting to have a PaaS that would let you build a CDN. It's kind of a thing that doesn't exist yet. And I think a lot of devs, in particular, that have ever even thought about building a CDN have sort of looked at this unattainable thing to try and ship, because it's like the infrastructure underneath it seems big and hairy and complicated. Then it's just a difficult thing to wrap your head around how you turn something like that on some of the time. So we wrote that. Basically, it was like, "Look, you can actually do this. And you don't need a whole team to do it. It can be a one-person thing."

[00:11:19] JM: Can I tell you about the crazy cloud idea now?

[00:11:21] KM: Yes. Wait. Let's summarize the less crazy cloud idea, which is basically Cool Cloud – Or cool – Wait. What was it?

[00:11:28] JM: It's just cloud run. It's just cloud run.

[00:11:30] KM: Alright. Okay.

[00:11:31] JM: You just sell cloud run. It's like cloud run is your EC2. You're built in Heroku.

[00:11:36] KM: Right coolcloud.cloud. No. What was it? You had a name, a domain for it, which is available by the way.

[00:11:43] JM: Kindofcool.cloud.

[00:11:44] KM: Kindofcool.cloud, yeah.

[00:11:45] JM: Okay, don't buy it. Can I buy it right now? I'm just going to buy domains.

[00:11:48] KM: I'm not. Yeah, go for it.

[00:11:49] JM: What's your preferred domain purchasing system?

[00:11:52] KM: I just usually use DN Symbol because I think –

[00:11:55] JM: DN Symbol. I actually don't know what that is. Wait, so kindofcool. – Kinda or kind of cool?

[00:12:01] KM: I did kinda. You do both. You have to own both and redirect one.

[00:12:05] JM: Kindacool.cloud. Kindacool.cloud. $20 a year. Why is it $20 a year and not $12 a year?

[00:12:13] KM: I don't know. I assume this is Google making money. Who owns a .cloud?

[00:12:18] JM: Kindacool.cloud and kindofcool.cloud. Kindofcool – I got both of them. Boom!

[00:12:27] KM: There you go.

[00:12:28] JM: Yeah.

[00:12:29] KM: Man, a register is good business. I own so many domains I pay to renew that I don't use.

[00:12:33] JM: It's a stupid business. It's so stupid. It's so stupid and annoying. I hate it.

[00:12:37] KM: It is. Like they can literally stop functioning for a year at a time, and I would still pay them because I probably never notice.

[00:12:44] JM: You know, the funniest list I have on Google domains is the list of domains that I've hearted where it's like I want to buy it, but I actually just don't want to pay $20 a year for it. It's a lot of hearts.

[00:12:57] KM: We paid $7,000 for fly.io back in the day. That's the thing, is to go to Flippa or whatever and find yourself a domain.

[00:13:01] JM: I'm sorry, man. That just seems like – I mean, I guess fly.io is pretty good. I bought Software Daily. Software Daily is the most expensive purchase I've made.

[00:13:10] KM: We bought it at the last company. This was post acquisition by IBM. We bought compose.com for $250,000.

[00:13:15] JM: Oh, you were at Compose?

[00:13:16] KM: Yeah. That was my last company.

[00:13:18] JM: Did you start that company?

[00:13:19] KM: Yeah, kind of. Basically, we went to Y Combinator together and I swapped companies mid Y Combinator, which is apparently a pretty common thing to do.

[00:13:26] JM: So who had the more frustrating outcome, Compose or Parse?

[00:13:30] KM: I was frustrated about Compose? I don't know. We'd have to ask Ilya probably.

[00:13:35] JM: Oh, I've asked him.

[00:13:37] KM: So like I feel like Parse is a less frustrating outcome probably, because it didn't – So like Compose, we actually had a profitable $15 million a year business, and we sold. And I felt like we probably could have done a lot more than that. And so it was frustrating to like have this thing that you'd spent seven years on, six years on, go that way.

[00:13:57] JM: I'm sorry.

[00:13:57] KM: No, it's fine. It's actually like complete first-world. It's even worse than first-world problems. It's like we sold a company in IBM. I didn't enjoy that very much. It's kind of a silly – It's a silly take. But I feel like it might have been slightly more frustrating because it was more maybe at that point than it was like – I feel like Parse was like a year in. It was like an idea, and then went to Facebook. And I'm sure their Facebook stock doesn't make them sad either.

[00:14:20] JM: Oh, no, no. So I wrote this book about Facebook. Did you read that book? Or did you see it, by the way? It doesn't matter. If you're a podcast person, it's available for free as an audio book on the podcast. But I spent like two and a half years writing a book about Facebook. And we tell the Parse story in there. And it's super sad. I mean, basically what happened is like Parse was acquired by Facebook, and Facebook was doing a lot of things at the time. And Facebook figured out mobile advertising right after they acquired Parse. And they just said, let's put all of the woods behind the mobile advertising arrow. And they sacrificed Parse. I don't think it does make sense. How many people do you need to make parse like run at Facebook? Like I think this was like, basically, they have an Instagram on their hands and they like dropped the egg, and the egg shell cracked and the yolk just ran over everywhere.

[00:15:07] KM: Yeah. I guess there was that, Meteor and Firebase, all came from our same Y Combinator class. We were on Y Combinator together.

[00:15:13] JM: Oh my God! Are you serious? Wait, it was Compose?

[00:15:17] KM: Was MongoHQ at the time, same YC batch.

[00:15:20] JM: So Compose, Firebase, Meteor and Parse were all in the same Y Combinator class.

[00:15:27] KM: And Segment.

[00:15:28] JM: Oh my God! Meteor is another thing that makes me kind of sad, because I really liked Meteor. I had a lot of fun with Meteor.

[00:15:35] KM: Yeah. So I mentioned earlier that you can't build a Dev UX product on top of AWS or Google because they extract all the potential revenue. And I think that –

[00:15:43] JM: Do you actually believe that? Or are you just saying that?

[00:15:45] KM: No, I believe it's entirely true. Because basically, like when you run – We did this at Compose. And basically, to build a profitable bravo company, we had to move off of AWS and do our own hardware. Because what happens as people look at kind of an EC2 instance and think, “Hey, this cost me eight bucks a month. I'm not going to pay more than like, 8.40 a month on top of that to make my life easier.” And I think that one of the interesting things about, I think, the future of cloud stuff is the stuff like Parse, and Meteor, and Firebase even I think have a lot better chance to build viable independent companies if they can literally make money off of the hardware underneath. And so far, it's been very difficult to do that.

[00:16:24] JM: Okay, gotcha. Anyway, so all I was saying with whatever, like the kind of cool cloud thing is that it's just cloud run, whatever. Its cloud run, and then you on top of that, you do either Render or Heroku and see where the world takes you or whatever. Anyway, that's idea number one. You want to do idea number two?

[00:16:42] KM: Yes, crazy. Let's do the crazy one.

[00:16:43] JM: The crazy one? Okay. So making this games company, Super Compute, just wrapping up the first round of financing for it. And it's – Do you play Magic at all? Magic the Gathering?

[00:16:52] KM: No, I don’t. I have been tangentially affiliated, but I've never played seriously.

[00:16:57] JM: Tangentially affiliated? What does that mean?

[00:16:59] KM: I've had friends that played.

[00:17:01] JM: Okay. I am tangentially affiliated with the class president.

[00:17:09] KM: Yes, exactly. Right. I know his name. He doesn't know mine.

[00:17:15] JM: We go to the same things, same functions. No. So I'm building this game. Are there any card games or board games you play?

[00:17:22] KM: Not really, no.

[00:17:23] JM: Chess, checkers, poker, something?

[00:17:26] KM: No, nothing. I played Risk a lot when I was a teenager.

[00:17:28] JM: Risk. Okay, fine. Risk. Risk is fine. So imagine Risk. So you're building Risk in the browser? Let's say you're building like a Risk client and server system. So what's the best way to build that today? Let's say you want to get to market as soon as possible with Risk. Okay? You're building risk in the browser. Today's technology, how do you do it? Go.

[00:17:48] KM: How do I do it specifically?

[00:17:50] JM: Just out of curiosity. Like I'd love to know what's your model? And then I want to tell you like what we're thinking.

[00:17:56] KM: I'd basically build a Phoenix, an Elixir Phoenix application. Use LiveView probably with Postgres.

[00:17:59] JM: Great. What's LiveView?

[00:18:01] KM: LiveView, what it is is an HTML fragments over WebSockets? So it's sort of like Rails, only it lets you do a relatively dynamic thing. Like a turn-based game is perfect. We actually have a tutorial for building a turn-based game in Phoenix, because you don't end up having a – Anyway, I go way into this. But the idea is like it's a full stack thing that one person can effectively ship like a multiplayer kind of dynamic game with without having to be very good at frontend JavaScript and backend stuff and like WebSockets and all these things.

[00:18:32] JM: Gotcha. Here's my critique. You have to run your own Elixir server. Like I would not want to operate my Elixir server.

[00:18:39] KM: Or you just deploy it to Fly. But, yeah.

[00:18:41] JM: Ah, so you guys do Elixir well.

[00:18:43] KM: Yeah. In fact, one of the interesting things about running kind of a full stack close to people is that stuff like pushing from a WebSocket is very low-latency. And so you get a lot of that. Yes. No. Nobody should run an Elixir server. But you should deploy Elixir apps to Fly basically.

[00:19:00] JM: Interesting. Does that mean I'm still writing them in Erlang or – No, I'm writing something, writing them in Elixir. Elixir is the language, right?

[00:19:07] KM: Yeah. Aha.

[00:19:08] JM: Okay, here's my other critique with this. People don't know Elixir. Like am I literally saying I need to hire Elixir developers or have my employees learn Elixir?

[00:19:16] KM: Probably. I feel like learning Elixir is – Well, A, JavaScript is basically the hardest language on the planet. So anyone who knows JavaScript probably would be able to pick up Elixir pretty quickly. I feel like it's easier to learn the language than it is to figure out like how to model a database schema, for example. But yes, that is the downside, is you'd end up writing in Elixir, which if it's not a language you want to write stuffing in is bad.

[00:19:39] JM: Anyway, I think the Elixir approach is actually really interesting. You can actually build these Elixir companies where you say we're an Elixir shop and then you attract the Elixir people. I think that's a really cool approach and you get a really interesting employee base. Anyway, my architecture or the architecture that we're working on – I should say more accurately, the CTO, Nick, sort of had this idea. Basically we're just going to use Firestore and write everything directly from the client to Firestore, which I think is pretty interesting because,

then there's just no – You don't even need Cloud Functions or anything. We can just talk directly to Firestore.

[00:20:12] KM: Right. That's interesting. I know some people who've done that.

[00:20:17] JM: But here's what worries me, is if you talk directly to Firestore from the client, you're going to have a really variable latencies, right? Isn't it going to be like terrible latency variability?

[00:20:28] KM: Yes, it will.

[00:20:28] JM: So that's a bad experience. So then there's the question of if you want to be doing that, how do you do it? Because the advantage of that is, essentially, your entire infrastructure is a key value store, right?

[00:20:39] KM: Which is – Yeah.

[00:20:40] JM: Which is great. Ideally, it's like an in memory persistent key value store. Like both the in-memory and persistent. Available in-memory persisted to disk. And it's like it feels like that should be accessible today. There should be some system that allows you to do that. So like what's the best system that gives you essentially the in-memory key value store for your infrastructure that you can access directly from the client frontend is my question.

[00:21:04] KM: Does Supabase do the Firestore stuff yet?

[00:21:07] JM: Exactly. So my next thing is going to be like I'm going to go talk to Paul at Supabase and say, “Hey, we want to be a flagship customer. But we want to start with Firebase, because I don't really like – I'm not sure I want to bet everything on SupaBass quite yet. Like it's just not advanced enough. Firebase is kind of there. Firebase is at least at – Firebase, I just trust it, right? It's running on Google. It's fine. I trust it. I've built something on Firebase before, and the Cloud Functions were super slow. And we're kind of crippling. It was awful. It was actually really terrible.

[00:21:34] KM: I was going to say, we've had a number of customers use Firebase kind of in anger. This is true for S3. I've learned to trust those things less, I think, than I would have before.

[00:21:43] JM: Dude, tail latency. Tail latency is horrible.

[00:21:46] KM: Yeah, or weird. So the standard practice on S3 is to retry your requests if they don't work. So you end up having to build – Like you have to kind of work around this thing that's supposed to be magical, and then you end up writing almost as much code as you would have if you just use like an open source database and run your own kind of little app server library. And then you don't understand it as well.

So I know we actually have a couple of larger users that probably we wouldn't name them right this second moving off of Firebase because it really put a strain on their ability to kind of keep things reliable. It's really kind of an interesting thing, which I guess the big problem with Dev UX is it's like it's really easy to get started and then really hard to move later on once you're sort of committed to something like that. But I also think that like just having a cloud thing to put data in is super interesting. I'm just old and would just default to using Redis and writing a little bit of Go in front of it basically, or Node, or Deno. That's what I'd do. I do Deno. That'd be the hot interesting thing.

[00:22:43] JM: Deno?

[00:22:44] KM: Deno. Deno, the Jordeno.

[00:22:45] JM: Oh, Deno. Yeah, Deno is cool. Yeah. Okay, that's kind of interesting. But let's take it a step further. Like what's really the best way to do it? Like you're just told me the language, but like what's the deployment medium?

[00:22:56] KM: Why? So one of our goals with this stuff has been to make – So I feel like building an app that sits in front of Postgres is about as simple in my head as it gets. There's not a lot of complexity there. I sort of have an app that talks to Postgres, and then I have a database that I trust, because it's 25 years old at this point. So one of the ways – We've tackled this

differently, I think, than Firebase, because our goal has been to make kind of what people use from their work globally, rather than building something from scratch with an API. So we have a way to run Postgres globally, and even Redis. But like, I buy us, right? This is the way I think things should work. So this is what we built. This is not necessarily whatever we'll pick from scratch.

[00:23:34] JM: Well, I like it. And I like that you're a CDN person. It's interesting that we've been thinking about this, and then you, a CDN person, is coming to me and kind of thinking in the same direction. But let me just take it a step further. So basically, I think, first of all, the MVP. We're trying to ship in five days, basically, and we have most of the stuff done. We're trying to ship in five days. I was originally going to say let's do everything with these Cloud Run servers. I was a Cloud Run seems like the best abstraction on the Internet. Let's do everything in Cloud Run servers. And then Nick is like, "Well, why don't we just write everything directly to Firestore?" And I was like, Okay, that's a better idea. We should do that. That's definitely the MVP." So we have the clients directly talking to Firestore. I know the experience is going to be bad. Like we're going to have all of the state persisted to Firestore. There's no way this is going to work out well. It'll be decent, right? It'll be tolerable. It'll be a good MVP. And then we can start filling it in. The question is what do you fill it in with? I think you could potentially do – Tell me if this is like a terrible idea. I really want your honest opinion. You basically start to set up a situation where you're doing as much client side stuff as possible. Because it's a game, right? Ultimately, it's a game. Let's do everything in the client. Security is not really a big issue because it's a game. There're not even payments involved yet. It's a free to play game, okay?

So you do everything on the client that you can. You abuse chrome storage. We try to do web assembly. You may maybe like make basically a language-agnostic operating system in the browser based on WebAssembly and you do whatever language you want. And then if it gets too crowded in there, you just start federating stuff out to like CloudFlare workers, and you start doing a tiered computing model with CloudFlare workers. Is that terrible? Is that too complicated?

[00:25:21] KM: It sounds really complicated to me, I think. So I tend to start with the backend and go out. And I'm betting that a lot of JavaScript folks now like start with the frontend and go back. So there's a little bit of like with just different worlds stuff here. But one of the interesting

things, I think, for me is I have a much easier time with like a strongly consistent thing where I write to it and I don't have to like sing changes around later on. It's like easier for me to wrap my head around.

And one of the things about like doing so much in the browser or on the client for something that ultimately has to share state with another set of clients is you have to get very good at having like distributed data infrastructure very early. And it's a complicated problem. There's actually a guy named James Long, but he writes. He wrote a talk called CRDT's for mortals. So CRDT's reduce conflict.

[00:26:12] JM: Yeah, I know CRDTs.

[00:26:13] KM: And he did it because he's doing, I believe it's like a money management accounting package that's offline. So what he does is you do all this stuff with your money, and then it has to replicate. And he talks about how he basically created this distributed eventually consistent data problem that's difficult, it's hard. Like you really have to get good at things like CRDT's to make it work, which I think is kind of interesting. So the only reason this sounds complicated to me is because it starts with a distributed, eventually, consistent data model. I'm using air quotes if people are listening and not watching. And that's like, to me, that's a complicated thing to keep in my head. It's a thing that would make it slower for me to ship new features, for example, than writing to something even like SQLite that's strongly consistent. You know what the writes are doing, and you don't have to worry about syncing all this stuff sometime down the road.

Does Firestore have – I know S3 finally got consistent reads after writes where you can put something in S3 and then read immediately and you're guaranteed to get back what you just wrote? I'm assuming Firestore does that, but I'm not actually sure that's true now that I think about it.

[00:27:15] JM: I don't think you want to ever use consistent and fire in the same sentence when you're running Google infrastructure. I mean, it's great. I'm sorry. I don't mean to be too harsh. It's like sort of like a deliberately obscure system. It's like we make no guarantees about anything. We're Firebase. That's the tagline.

[00:27:37] KM: Which makes sense for Google and even Amazon, because –

[00:27:40] JM: They're like, "We're eventually eventual."

[00:27:43] KM: Yes. We're maybe consistent.

[00:27:46] JM: We're partially tolerant of some things that are fault-like.

[00:27:50] KM: What's so interesting to me about most of what Google and even Amazon produce is that I don't have Google scale problems. And so you end up working within this architecture that was built by people who do have Google scale problems. And I'll probably never have Google scale problems. It's not like any side project ever built is going to be replicated, need multiple master writes and do all of this craziness that Google has to deal with. And so it's always fascinating to watch the things they produce, inherit some of their organizational complexity, because it is stuff like it's much simpler if they do give you guarantees. And maybe it only runs on one server, but that's actually fine for so many people sometimes.

[00:28:28] JM: So I've heard this guy named Avi. He runs a company called QinetiQ. Have you heard of the company QinetiQ?

[00:28:34] KM: I have.

[00:28:35] JM: Do you use it, by the way?

[00:28:36] KM: We don't.

[00:28:36] JM: You're probably like a little too – You're like a little too new-ish, hipster-ish to use it, I guess?

[00:28:43] KM: Yeah, we also offload most of our networking to other companies. So we run our own physical servers, but generally we outsource networking in front of everything.

[00:28:51] JM: But you would still want network observability, right?

[00:28:54] KM: We will in a couple of years. I think when we actually absorbed – Most of the network observability issues that happen upstream of us, I would think, not necessarily between our hosts or anything. So we would someday, but we're not there yet. We have something like – I mean, our customers are in the thousands, not the hundreds of thousands. So it's like we're not as big as you would necessarily think someone like QinetiQ. Anyway, I met Avi before. I think I ran on him like 15 years ago.

[00:29:19] JM: He's awesome. He's a really good friend of mine. I mean, he's kind of becoming a mentor, because he kind of can tell me how infrastructure should be or something. I hung out with him at this – Like a place where there are physical servers. It was like I felt like a child. Like I'm seeing physical servers for the first time pretty much.

[00:29:36] KM: Yeah, like an Equinix or something?

[00:29:37] JM: It was an Equinix-like place. I mean, I'm a software engineer, but I've been in a physical server in one other time. It was when I was interviewing at SIG Susquehanna. It was like a trading firm. They have their own servers. Other than that, I've never been inside a server room. Don't even really know what a server looks like. But Avi showed me servers and he talked to me about his beliefs around servers. He's an on-prem – He calls himself a server hugger. He's an on-prem guy. He runs a network observability company, like a venture-backed network observability company. They've got great investors. This guy runs his own infrastructure. And I'm like, "Why do you do that? Like what are you doing? Seriously, man, just like embrace AWS. What are you doing?" And he calls it cloud jail. He just says, "I don't want to be in cloud jail. I just don't want to be in there."

And for the longest time, I just thought this whole Oracle lock-in, Microsoft lock-in, AWS lock-in, I thought it was just kind of a joke. Like people don't actually believe this, right? Like this is not actually a problem, this lock-in idea. But more recently, I start to see it everywhere. Like I start to see it, it's like an inherent facet of just like corporate – What the corporation is, is if a corporation has control over your infrastructure, they will abuse that control. It just has to happen.

[00:30:53] KM: Probably not even intentionally. It's not like they necessarily want to as much as – It's just an incentive mismatch all the time.

[00:30:58] JM: Right. And open source solves a lot of that I feel like. But that's a different conversation. Anyways, Avi runs all these infrastructure. And I'm just like, “Man, you're nuts. Like what are you doing running your own infrastructure?” But I kind of get at this point. Anyway, I don't know. What was my point there? I don’t really –

[00:31:15] KM: Oh, yeah. What were you talking about?

[00:31:16] KM: Well, I like cloud jails. So that was a good point, even if that wasn't the point. I was a good iterative point.

[00:31:21] JM: Yeah. Anyway, the only thing about with this like architecture is like, basically, what's – First of all, like what's the fastest to market architecture for a game that has like no money involved? It's just like let's get to a game. Let's do a fun experience really quickly. Like get risk in the browser basically. Let's set up like a persistent risk in the browser. If you close your browser, risk is still there, right? I really feel like you can do this sort of like all the processing is client-side and all the storage is in Firestore. Like what's wrong with that? And what's my next step?

[00:31:52] KM: Actually, have you looked at FaunaDB?

[00:31:56] JM: Oh, man. I've learned FaunaDB. I did a couple shows about FaunaDB. Is this what FaunaDB is trying to do?

[00:32:01] KM: So FaunaDB is probably a better – From like kind of zero to MVP thing than Firestore probably, because you don't – So it's sort of like CDN-ified by default. You don't have to worry so much about latency being bad across users, if that makes sense. However, I think it's – I mean, I would never have any problem with someone starting a project and just using Firestore. Because, basically, like if you're anything like us, all the code you write for the next three years is probably disposable anyway. It's all wrong from the beginning. And so whatever

gets in front of people fastest seems like the right choice regardless what that is. I'm pretty sure no company has ever died because they chose the wrong infrastructure at the beginning.

[00:32:42] JM: I mean, maybe like one of those companies that got hacked and lost their entire codebase.

[00:32:47] KM: Yeah, but did that actually ever actually killed a company?

[00:32:50] JM: I think that happened, that happen. There was a Hacker News post about that there was a top Hacker News post where somebody – I think they got hacked, or like they had a disgruntled ex-cofounder.

[00:33:00] KM: Oh, yes.

[00:33:01] JM: Control the company just nuked everything eventually.

[00:33:04] KM: We'll still have it on Firebase.

[00:33:07] JM: Well, I don't know. I feel like if there's one thing Firebase has is it's some kind of archive system where you can always go back.

[00:33:14] KM: Yes. Right. Yeah, yeah. I mean, it could – But anyway, that's also like most companies died for entirely different reasons, it's because they don't get something to people that's valuable. It's not actually anything before that.

[00:33:23] JM: No. But like talk about tail risk, right? Like I kind of like the idea that my backend system is going to have – I probably going to have some write ahead log thing where if anything happens to my company, if I have some maximally evil participant that just jumps into my company and just does shit for like a day or two, there's a complete rollback log. I like that idea. I don't know if Firebase actually allows that, but I feel like that's the kind of thing that Google would build.

[00:33:50] KM: Yeah, that's a good question actually.

[00:33:52] JM: Can you do that? Because probably Fly can't do that, right? Like you don't have like a reversion system.

[00:33:59] KM: No, we have backups and snapshots like anyone else. But I think what's interesting is like database backups are good. And then there's this entire question mark. And then it's like a whole like audit trail. I'm not sure anyone offers something like that, which is an interesting problem.

[00:34:14] JM: Yeah. I been sidetracked the whole conversation. We should talk a little bit more about Fly.io. Okay, tell me how applications – Like what's your vision for application deployment? Like what's your current like go to market? How's usage? Just tell me about your business, like how it looks today and how it's going to look in a few years.

[00:34:32] KM: So our special thing here, and I think one of this is like my opinions on Dev UX, which you've probably noticed I have. I feel like –

[00:34:40] JM: WX?

[00:34:41] KM: Dev UX, developer UX.

[00:34:41] JM: Dev UX. Got it. Got it.

[00:34:42] KM: Yeah. I feel like Dev UX is important for kind of getting people access to something that couldn't otherwise use, if that makes sense. So like back in the day, Heroku – Heroku made it so a developer could deploy a Rails app, and that's all they needed to do. It was their CLI, and it just worked and it ran so fast. Our goal right now is to basically make all backends run globally. Because it doesn't it doesn't actually make any sense to me that you would run an app in Virginia for users that aren't in Virginia. And something like 92% of the Earth is not within 100 milliseconds of Virginia. And so we're sort of a PaaS for making your typical backend or full stack app run all over the world. That's kind of our entire purpose of life.

And currently, the goal for that is just get in front of as many devs as possible. Unlike when we did Compose, we actually have to work hard to get people's attention, because there's a lot of stuff happening on the Internet these days. It used to be developers who try any new service that came around. And now developers are like, “I don't have time to try new things, because there're a thousand of them, and I already have my one, and I ultimately just want to ship my app.” So we're kind of working hard to kind of put this idea in front of people and make it as easy to try out as possible. And that's going to be basically the goal for the next two years, I think, until we get to hundreds of thousands of developers on top of the platform.

I think, ultimately, I don't understand why there's not. Ultimately, you look at like the scope of AWS, and it makes sense to me that you should be able to offer AWS level of power without the user experience being trashed. Like you should actually have a reasonably good self-service developer facing user experience for like all of the things AWS or GCP offer. And so I think that if we do what we want, we'll kind of get to that point where we're another sort of public cloud option. But you don't necessarily have to hire Kubernetes people to do anything useful with it.

[00:36:34] JM: So when you look out at the market of all the stuff that we've been talking about, Render, and Cloud Run, and AWS, all this stuff, like how do you divide it up? Or how do you – What’s your go to market? By the way, the way I think about this is so positive-sum. It's more like we're just rearranging computing. We're presenting people with different Lego blocks. As long as your Lego blocks are reasonably cool and they snap together with other Lego blocks, you've got a good business. Your cloud hosting is so high-margin. It grows with such interesting properties. The upsells are beautiful. Nobody ever churns. It's basically the best business ever. And it's kind of hard to lose.

[00:37:14] KM: Yeah, once you get over the hump, I think that's true. I'm very fond of everyone who's kind of – Almost everyone we mentioned who to me looks like an alternative to AWS in the future. Like I really don't want to live in a world where everybody always deploys things to AWS or Google. I mean, when I say AWS, I mean one of the giant tech Titans, right? I think that my life will be better in the future if I have a lot of interesting places to ship my work. And so I'm actually really fond of Render. I think like renders interesting, because we sort of compete a little bit at the very beginning just to get people to try us out. Like our pitch to some of these people is

like it's easy to deploy an app. But like we diverge very quickly after that for what our actual value is and what theirs is.

And so like I actually think that when we pitch to investors, we're mostly pitching against CloudFlare. I think that most of my belief about the world is that CloudFlare is not really the future of how we ship applications. It's a big CDN business, but it's sort of the last of the big CDN businesses. And kind of future infrastructure won't really have a distinct CDN component exactly. So that's the only company I would throw any shade at, because I just did when we were raising money, was specifically Cloudflare's take on how computing close to people should happen.

[00:38:30] JM: Okay. Wait. Do you live in San Francisco, by the way?

[00:38:32] KM: I live in Chicago.

[00:38:34] JM: Chicago.

[00:38:35] KM: Ironically, we moved here after we sold our company. And it was like the houses are too expensive out there. I would rather – We also have four kids. So we need a lot of bedrooms.

[00:38:42] JM: Got it. Chicago in the summer, can't beat it.

[00:38:46] KM: Right.

[00:38:47] JM: I moved out to Chicago to work in a company called PEAK6 like eight or nine years ago. You probably have a lot of friends in finance in Chicago, right?

[00:38:55] KM: It's weirdly insular. I end up not having that many friends in Chicago. They're mostly spread out throughout the US.

[00:39:01] JM: Oh, I gotcha.

[00:39:03] KM: I've run into finance people at meet-ups and things, but it's all – Chicago's tech scene is very strange. It's all very insular.

[00:39:09] JM: There's a lot to learn from the finance people. They do structure a little bit differently. But that's neither here nor there.

[00:39:16] KM: But Chicago in December sucks because it gets dark at 4:30. That's the thing I hate.

[00:39:21] JM: I mean, also, it just gets bitterly cold outside of summer. And it's kind of – I don't know. It's not for me. I can't live in Chicago. It's just too cold for me. I'm from Austin originally.

[00:39:30] KM: I get it.

[00:39:31] JM: I'm looking up your Crunchbase right now. What's the financing been like?

[00:39:35] KM: We basically raised a seed round almost on the strength of Compose, and then are in the process of closing our A round. It turns out I did some equity grants improperly. So we're having to go through the slog of cleaning up paperwork.

[00:39:48] JM: For advisors or something?

[00:39:50] KM: No. For – I'm trying to think of what actually held us up the most. So what happened is we started this company in 2016, 2017. And what we're doing now is nothing like how it started. But we have kind of four years of legacy to work through for like document trails and things. And so it's been a chore to go back and dig all that stuff up. But mostly for ex-employees, so people who kind of we hired very early and then switch what we were doing. Anyway, closing in A round right now, specifically, like I said, to get what we're doing in front of more people. We're in this fortunate position where when people see Fly they sign up and try it and deploy their stuff and for some of my works. And so all we actually have to do is get in front of more people than know we exist right now.

[00:40:33] JM: When Bedrock did the – Was it the series A of Vercel? They did the series A. Or was it the series B?

[00:40:40] KM: Yeah, maybe. That's a good question. It's been CRV that's been heavily – Yeah.

[00:40:45] JM: CRV that's in what?

[00:40:47] KM: CRV have been heavily involved in later rounds for them, but I don't –

[00:40:51] JM: For Vercel?

[00:40:52] KM: Aha.

[00:40:53] JM: I don't think that's right. You mean GV? It looks like GV did the B. Bedrock did the C. Bedrock led the C for Vercel.

[00:41:02] KM: Oh, okay. Well, I had that backwards. That was Reessen.

[00:41:04] JM: Yeah, that's Reessen. I don't think this is right. I don't think this data is right. I don't trust this data right now. But anyway, maybe it's directionally right. Oh, Bedrock, I think was involved with the series B, Vercel series B. I like Vercel a lot. I think for Vecel is a cool company.

[00:41:18] KM: Yeah, me too.

[00:41:19] JM: Yeah. Okay. So Bedrock was heavily involved in the series B. And then they led the series C. That's really cool. I like that. Actually, when – So I started talking to the Bedrock people around the time when they made their first investment in Vercel, because I was kind of interested in working with them, like doing venture stuff with them. And I remember talking to them about Vercel. And hearing their perspective was pretty interesting, because they're not actually like infra guys. They're not infra people, right? They don't know infrastructure. So they were sort of like looking at Guillermo and just betting on Guillermo. And I feel like Vercel is a company where you basically can bet at any stage on Guillermo, because he's so good.

[00:41:56] KM: Yeah. You know what's interesting? When we're pitching companies, I think sometimes the traditional infrastructure companies don't know what to make of – I think the word is product-led growth, like dev focus companies. They're so kind of caught up in how enterprise sales enterprise software historically works.

[00:42:13] JM: Oh yeah. This is the major failing of a lot of the major venture capitalists, and it's like conventional builders too, is they're like – And this is why Kubernetes sells to them. It's like Kubernetes smells like VMware. Like smells it on like VMware that they know how to bet on it.

[00:42:30] KM: I missed out on Mesosphere. So let's do Kubernetes. Yeah.

[00:42:33] JM: Yeah, kind of like that. So man, I say all this because I'm just I'm curious how you're pitching up an infrastructure company is going, because like I'm pitching an infrastructure company too. We're building this like fin tech company. And pitching is just like kind of a bleak exercise, because they just don't seem to really understand the economics of like an infrastructure company.

[00:42:52] KM: Yeah, it's interesting, because like it took me till this round to realize that I'm never going to be good at pitching to investors, because like you maybe get like six chances in your life, really. It's just like it's not a thing that I'm going to do 100 times.

[00:43:05] JM: What do you mean? What do you mean you only get six chances? What are you talking about?

[00:43:07] KM: Well, so like if you're building a successful startup, like if I go through six rounds of funding, and even like after the first two or three, a lot of the work falls on other people at that point. And so like even I think the most experienced like CEOs of companies don't really get that much experience pitching to investors. It's not a thing they've done enough to be experts at it. It's a thing that definitely like doing it three times is better than doing it zero times. But in general, it's like I'm not going to pitch your company to VCs enough to be really good at it ever, because just like I need the repetition. That's not going to happen, which actually relax me. But

this time, I realized that most of my job was to find VCs that probably already agreed with our take on the world and worry less about the points it sounds.

[00:43:48] JM: Yeah. But the problem with that is like a lot of those people who actually get it only have what? Like 10 million under management or something, right? So they can't fire as big a bullet. So you have to put together a round of a bunch of 500k or a million dollar checks, right?

[00:44:03] KM: No, I think that I feel like there's a lot more traditional like kind of half a billion dollar funds now that get it.

[00:44:10] JM: Yeah?

[00:44:11] KM: Yeah.

[00:44:11] JM: So are you not having any trouble? Are you like picking between different investors?

[00:44:15] KM: So we've already signed a term sheet, but we had several to choose from, several investors to choose from.

[00:44:19] JM: Okay. And how do you choose? What was your decision criteria?

[00:44:23] KM: We had bought – Disregarding the obvious of like – I mean, maybe it's not obvious, but disregarding these partners at this place seem really easy to work with and helpful in that way. My goal was to diversify the board and make sure that we don't have a board full of just white dudes from the beginning if we can help it. And so we ended up finding really good partners. In fact, we have multiple non-white dude partners to pick from, which I thought was really exciting.

[00:44:46] JM: That's great. That's very, very smart forward-looking sort of like 4D chess level –

[00:44:53] KM: Maybe 3.5.

[00:44:55] JM: Maybe a 3.5. So like the multidimensional chess there is like some people might think this is like a decision that's more about like “diversity”, like diversity. But that's not what it is. You actually really want – Like you legitimately want diverse opinions when you're building a cloud infrastructure company.

[00:45:15] KM: Yes, it's true. And like, even for hiring and how to target customers, like we need it, because like, we're going to end up getting customers from backgrounds that I can't even fathom. We have a tremendous amount of Brazilian customers. And like I know nothing about Brazil other than it sounds amazing and I should visit.

[00:45:30] JM: Well, that's the cool thing about building a cloud company, is your customer base is so diverse, and you essentially have really good insight. You have really good insights into what they want and what they need, because you can just go to each of your customers and say, “Hey, I'm Kurt. I basically run the electricity at your company. Is there any other electricity running devices that you need?” And they'll always say, “Yes. We need this. Like we're dying here.” Do you know how bad your product is? We need new product. Do you know how good your product is? Here's how it could be even better.

[00:46:02] KM: Yeah. Yeah, yeah. It's actually one of the things I really love about the developer-focused model, because if you can just survive long enough to get enough developers using it, you can start building those relationships with other companies really going under the radar in places you never would have guessed otherwise, because it's almost like the companies coming to you. You don't have to pick the companies you want, because this and that, and then you get people using it that you never would have identified otherwise. And so you have accurately identified why diversity is important for us, because like Midwest dudes in the US don't quite have enough world experience to get to the developers that we need to get to. Anyway, I feel like the big funds are much more – So, A, product-led growth now is a big thing that some funds actually make a big deal of.

And then there's been enough success. Like one of the things I kept telling them, I was like, “Look, the knock against CloudFlare was always they didn't make any money. And that was accurate until the day they started making money. And then they were wrong. You didn't invest

in CloudFlare. And don't you wish you would have? Like it's people sort of buy the revenue will come later if it's a valuable enough product and it's actually getting a good enough user base from developers now.

[00:47:08] JM: What should we close on? Are you coming to the Bay Area sometime?

[00:47:12] KM: I should be out there – When do I need to go out there? I've been putting off a trip. Probably when my kids are back in school. So probably late August, September, something like that.

[00:47:18] JM: Cool. I mean, I might be out for some of late August, but I'd love to like grab coffee or something.

[00:47:23] KM: Yeah, for sure. Anytime you want.

[00:47:25] JM: Yeah. Just let me know when you're coming to town. So why did you take an improv class?

[00:47:34] KM: Why did I take it? It was sort of always like, "That seems fun." Like I'm always like –

[00:47:37] JM: I think improv is so useful. It's really, really useful.

[00:47:41] KM: Yes. So it was valuable for – I don't know if this is why I took it. One, it was valuable to have a thing that made me just completely shut out work and never think about my company and like have a few hours of just peace kind of. And then the other one is actually I'm prone to like trying to make money off of everything I do. And improv is great, because like there's no chance of ever being a career. So I can sort of make it fun and just do it –

[00:48:05] JM: Why not? Why not? I mean –

[00:48:06] JM: Why would I not make an improve career?

[00:48:08] JM: No, seriously. Whose Line Is It Anyway? Like that probably made you know the careers of at least Wayne Brady, right? Like Wayne Brady basically made his career based on improv.

[00:48:18] KM: Meanwhile, there's 16 people per class I'm in in Chicago all trying to get on SNL.

[00:48:22] JM: Who cares? I mean, cloud providers are there, right? A bajillion?

[00:48:28] KM: Yeah, pretty much. I think the bar to creating a successful – Well, there's way less luck involved in creating cloud provider company, becoming like an entertainment person with no context.

[00:48:37] JM: Strong disagreement. Strong disagreement. I'm so serious. I totally disagree. Becoming a good entertainer is a pretty deterministic process.

[00:48:44] KM: Well, don't ruin it for me. I like not thinking I can make money off improve. It makes my life easier.

[00:48:49] JM: I hate to break it to you. I hate to break it to you. If you can build a cloud provider, you can probably do improv successfully. It's going to matter of how much time you allocate to it. I hate to break it to you. So your only solution is to practice improv while building a cloud provider.

[00:49:04] KM: Right Yes, yes, which I've been doing. It's fun. Anyway, it is pretty useful. It taught me how bad I am at listening. That seems to be a thing that's true for everybody that I didn't realize at first.

[00:49:14] JM: Can I ask you a totally random question as our closing questions?

[00:49:17] KM: Yes, hit me?

[00:49:19] JM: What should the terms of the daily.co series B be?

[00:49:25] KM: Alright, let me go see what they're at right now.

[00:49:27] JM: And I have no privileged information on the matter. I have no privileged information. You know what Daily is, right. You know what they do?

[00:49:36] KM: Yeah. Uh-huh.

[00:49:39] JM: Streaming video API. Streaming video API streaming. So like that's how you build telemedicine, basically.

[00:49:46] KM: Or for your own Zoom.

[00:49:47] JM: Or your own Zoom.

[00:49:49] KM: That is, I wouldn't say whatever the terms of the plaid series B were. That seems like the most interesting company.

[00:49:55] JM: Really? Finance series B.

[00:49:57] KM: Basically, like Daily is built kind of the – I think It's really fascinating to look at companies that build almost like a developer platform for an existing successful company. So daily.co is a developer platform, and Zoom is the existing successful company.

[00:50:13] JM: Dude, I would never want to build any application on top of Zoom. I'm sorry.

[00:50:17] KM: No. No. No. What I mean is Zoom was really successful as an application. Daily.co, Daily, is giving developers the tools to build a Zoom alternative.

[00:50:25] JM: I agree. I completely agree. I completely agree. I completely agree.

[00:50:28] JM: So what did Plaid do for series B?

[00:50:31] JM: I wonder if Plaid had moved beyond their like remote login system by the time they were a series B? Probably not, right? That was probably still what they were doing.

[00:50:39] KM: Probably. They're at $44 million series B. So that's probably what? A 250-ish post? Yeah. So I'd say $40 to $50 million series B for Daily.

[00:50:47] JM: The thing is I feel like Daily is less obvious than Plaid. Plaid was so obvious. This is like a really powerful idea. Daily is less obvious. People don't seem to grasp it. They're like, “Oh, so it's like Mux?” I'm like, “No, it's not like Mux. It's just a very different category.”

[00:51:02] KM: I feel like plaid was not obvious to people. It is now. Yeah, I feel like at the time Plaid was – There a lot of cynics about Plaid

[00:51:10] JM: Wait. Wait. Okay. Okay. So $44 million series B at what? At what valuation? Doesn't say the valuation, and I can't find the valuation.

[00:51:20] KM: I mean, I'm sure it was like $250, $300 million.

[00:51:22] JM: Right. Okay. And then it looks like by the time they get to series C, it looks like the hockey stick from series B to series C. Series C was 250 million at 2.65 billion. So they went from a $44 million round to a $250 million round. So it looks like the hockey stick from series B to series C, which could happen to Daily, but it seems unlikely. Like Daily doesn't seem to have that kind of – Well, maybe. Who knows?

[00:51:48] KM: The knock against Plaid was you can't get big companies to use this. They won't trust you. And I would guess that Daily is the exact same –

[00:51:55] JM: Probably correctly at the time.

[00:51:57] KM: Yeah, I think we have this problem a little bit. You sort of have to build something and outlast that argument. And then it flips. I mean, that's sort of Vercel did. Like this is not an uncommon pattern.

[00:52:08] JM: That's what Stripe has done. Amazon is on Stripe now, really? Amazon is on Stripe, which is on Amazon.

[00:52:17] KM: The best thing about the best thing about dev for customers is you can actually build a business and survive until that switch flips and the big companies understand. So because like I feel like Daily exists in this world where big companies now are going to have to figure out how to handle a remote workforce because they're going to do it whether they want to or not at this point, and like the plumbing they use –

[00:52:38] JM: But it's bigger than that. It's bigger than that, because even like in the metaverse, in the metaverse you want streaming video. Like you want streaming three dimensional video systems. Like Daily is basically a WebRTC API company. Everybody has been looking for the WebRTC bet? Here it is. Here it is. It's Daily.

[00:52:57] KM: Yeah. I think the biggest problem is like it's looks like Twilio to people sometimes, and it's not.

[00:53:03] JM: What's wrong with Twilio? What's wrong with Twilio? Oh, it looks like the Twilio streaming video system?

[00:53:08] KM: Yeah, to people making a bet. It's like, "Well, Twilio does that too," which is not super nuanced and probably incorrect. Yeah.

[00:53:13] JM: I've tried that product. I use that product. It was really pixelated.

[00:53:18] KM: Yeah. Yeah. It's a hard, hard problem. I think Daily is cool for that. It's like it's hard in different ways in Plaid works, but it's incredible – Like what Zoom actually makes work is pretty incredible. And if they can give what we're doing right now to devs as a thing that they could put in their own apps, like that's friggin amazing.

[00:53:36] JM: Anyway, interesting closing discussion.

[00:53:39] KM: You have to write that down and see if we're recording about series B.

[00:53:40] JM: Daily is really cool to me. Daily is a really cool company to me. I just feel like if you build a core competence in WebRTC, you're basically building a core competency in data movement. You're just building a core competency network data movement and pricing of that network data movement. And it just sounds really hard and really interesting. Mux is like batch data movement. Mux is like we can build you a codec ladder in batch, and then lets you serve your own Netflix on a codec ladder and give you a really, really good CDN experience and give you really, really good analytics around that. And Daily is something completely different. Daily is where we're real time data streaming over WebRTC. I mean, WebRTC is kind of like a TCP level protocol, if you think about it, right? It's just data movement.

[00:54:26] KM: Mux is more interesting that I think you just said. One of the companies we ran into early in Fly days was NASCAR. And they were trying to broadcast live events, which is like forget the protocol for a second. That's like a thing you'd go to Mux for. But one of the problems they were running into is they had to actually ensure, it's like they had a latency objective for all their clients. And the reason was that they had this problem where gamblers, if they had lower latency than other people watching, could actually front run the betting, which is probably you'd use WebRTC to solve that, but it's it problem and something like Zoom. It's like actually a big hairy issue of how you provide like live broadcast stuff to the world now that even gets close to what old school TV services used to do is a really fun problem. And incredibly valuable because, again, anytime gambling comes up and you need to prevent gamblers from having an unfair advantage, there's a lot of money there.

[00:55:23] JM: Do you think gambling taboos will like go away in our lifetime? Like do you think gambling taboos will track marijuana taboos?

[00:55:30] KM: That's a good question. I think what will happen is well – I mean, we see a little bit now with a lot of the games, even like Farmville, is like things that aren't called gambling, but act like gambling and where a lot of that attention goes.

[00:55:43] JM: Yeah. That's how you know there's like a societal cognitive dissonance, or just Instagram. Man, Instagram is gambling every time. Every time you go in, it just messes with

your emotions. Roblox is a casino for children. Let's just be honest about what we're doing. I mean, look –

[00:55:59] KM: It's the same brain chemical thing happening, yeah.

[00:56:01] JM: Yeah. Like it's a casino for children where you get daddy's credit card out and you charge it for Roblox. Look, I grew up playing poker under age. And it was a great formative experience. So like I'm not criticizing anyone. I'm like, "Let's have more gambling for children, right? Let's just be honest about what we're doing."

[00:56:21] KM: Yes. And like make it – It's interesting, because like I think any business who calls customers whales is probably doing some kind of gambling. If you have a customer that can spend so much money, you consider them to be whales. Like it's basically gambling.

[00:56:34] JM: Oh, yeah. I mean, I don't know, the whole moralizing around gambling is really annoying and boring.

[00:56:42] KM: I think it's like – What's interesting is how society should treat addiction and not necessarily moralizing about the activities that lead to addiction, if that makes sense. Like gambling addiction is a problem for people in the same way like heroin addiction is and like how we actually respond to that is important. But I think it's weird to tell people like that's wrong because you may get addicted someday, instead of like looking at it as like a top down society view of like we probably don't want people to have dependencies they can't break free of as a whole if we can help it. So how do we actually build a world where that might be true?

[00:57:17] JM: Yeah, but the whole thing is like everybody's a gambling addict. Everybody's a gambling addict. It's all about like how are you integrating that into your life. Like meditation is a variable reward system. So like you can sit down and meditate and you're actually gambling. You're just like hoping that the right –

[00:57:32] KM: Hoping I get that – Or running. That's the other fun one.

[00:57:35] JM: Yeah. Actually, running is like a positive expected value gambling experience. That's why I love running.

[00:57:42] KM: I think the interesting thing about the anything addictive is not necessarily what it's like for the individual, but who exploits that to gain power, or wealth, or whatever? That's the big scary question.

[00:57:52] JM: Oh, yeah. Well, I mean, look, I just wrote a book about Facebook.

[00:57:55] KM: Yeah, right. There you go.

[00:57:57] JM: I was like, “Huh, interesting.” So that's what's really going on at this company.

[00:58:01] KM: Yeah, yeah, pretty much.

[00:58:03] JM: It’s a great mission statement you have there, guys. You don't really like show the intent of your company through a mission statement. You show it through your actions.

[00:58:11] KM: Yeah. Google could afford to learn that at this point probably. First, do no evil or whatever it is. Anyway, yes, we should talk about addiction more. That’s a fun one.

[00:58:19] JM: The problem with do no evil is like you can't really systematize do no evil across 200,000 people or however many people Google has. If a subdivision of Google can do whatever they want to with like a drone software or whatever, like whatever that big controversy was about drones a while ago. If some sub-team can do that like just autonomously, which you have to do if you're Google, you have to allow for autonomy on teams. Larry Page is not going to have complete visibility into how malicious the drone systems are. He simply can't be – So you can't have this like two dimensional or one dimensional value judgment system where something is either evil or good. These things are like a matrix of values. And you can't really impose judgment across the 200,000 person organization at the level of don't be evil. It's the most coarse-grained, bad mission statement that you can possibly have.

[00:59:11] KM: Yes. It’s very churchy at that point, at which point you're inviting a bunch of –

[00:59:15] JM: I mean, it's like – I don't know. Why is your mission statement like don't do something? What?

[00:59:20] KM: Right. Yeah, exactly. I don't think that's ever been their mission statement, but sure they like that. Yes. I know. The idea of like we have a big group of people that somehow more moral than your big group of people is really interesting to –

[00:59:32] JM: Oh, no. This is my joke about Stripe. It's like a planned economy run by idealistic Irishman is still a planned economy.

[00:59:40] KM: Yeah, pretty much. It's true. And they're very nice idealistic Irish, and I like them a lot.

[00:59:48] JM: They're great. There's nobody I trust more than Stripe pretty much to run my planned economy. A planned economy with a publishing house is still a planned economy.

[01:00:01] KM: Yeah, yeah, exactly. Oh, Stripe.

[01:00:03] JM: A planned economy with a biased anti-fraud system is still a planned economy. Anyway, I don't mean to tear into them too much. It's just – Anyway, great talking. Fly seems really cool. What's the call to action? Like who should use Fly other than Elixir apps? We know Elixir apps.

[01:00:20] KM: We're actually focusing a lot on Elixir and Phoenix right now, mainly so we can make a really good dev UX for something focused. But basically, anyone building – I think anyone building a backend or a full stack application that thinks it might be useful to run it close to their users is someone who's tried Fly.io. It takes like two minutes. It's actually easier to use than Cloud Run. So if you're looking at Cloud Run, you should look at Fly instead.

[01:00:42] JM: So you've used Cloud Run. You've dog fooded this?

[01:00:45] KM: Yeah, actually, were very similar to Cloud Run and Fargate, just like the user experience does.

[01:00:50] JM: Wow! Okay, great. Then why doesn't your website say like spin up a container?

[01:00:55] KM: Well, okay, so the snarky answer to this is if you go to any blog post, it says it at the top, because that's what developers read. And investors read the homepage. And so the homepage needed to match my pitch to investors.

[01:01:06] JM: Oh, that's a sad commentary, man. Totally, I'm fine with it. Like I'm totally fine with it. It's just a really sad commentary. You have to format your website like a pitch deck.

[01:01:17] JM: While you're pitching, I believe that to be true, yeah. They will go look and – Yeah.

[01:01:21] JM: Live I've decided I'm saying no to pitch decks. Like I don't do pitch decks anymore. I do like Memos or Zoom calls, or like what's the video, Loom Systems or whatever? I'm not doing Pitchdecks anymore.

[01:01:31] KM: I am a fan of that. I did won this time. And I decided also like next time it's going to be a Memo and like a thousand appendix slides that are just like graphs and things. Long-form writing is such a powerful thing that – I think the fear has always been, "Well, why if investors don't have a good attention span and don't read it?" It's like, "Well, you probably don't want them to invest." They should get better at their jobs.

[01:01:52] JM: Yeah, that's exactly right. That's exactly right. Okay, so real quickly, like my fundraising process has been to act as ridiculous as possible, and basically just frustrate investors until they say no. And then the few who say yes are the ones who you actually want.

[01:02:09] KM: It probably works. It's like Silicon Valley season two, I think.

[01:02:13] JM: Yeah. Well, it's so fun. If you just think of like can you be as unpalatable to VCs as possible? Like you just play that game, you've basically got a good pitch. But as soon as they

get on the Zoom call, you're the most unpalatable person they can possibly be. So they get really conflicted.

[01:02:30] KM: Yes. I don't even think they get conflicted. I think they recognize the certain class of behavior that probably overlaps with just being a psychopath in some ways and think that's what people need to build a good company. It's like it's weird – Anyway.

[01:02:42] JM: Speaking which, I got to go to the bathroom before my next interview. But this was awesome.

[01:02:45] KM: Enjoy.

[01:02:47] JM: Talk to you soon, Kurt.

[01:02:46] KM: Alright, bye.

[END]