EPISODE 852

[INTRODUCTION]

[0:00:00.3] JM: Cloud computing was popularized in 2006 with the launch of Amazon Web Services. AWS allowed developers to use remote server infrastructure with a simple set of AAPI. Even with API, it was still not simple to deploy and manage a web application. In 2007, Heroku launched a platform built on top of AWS. Heroku focused on the developer experience by optimizing for users who were deploying Ruby on Rails applications.

Since then, Heroku has expanded into other forms of managed infrastructure, including other application frameworks like Node.js and databases like PostgreSQL. Heroku was the first popular layer 2 cloud provider. Layer 2, meaning it was built on top of a layer 1 cloud provider, like AWS.

12 years later, Heroku is still probably the most popular layer 2 cloud provider, but there have been many other layer 2 cloud providers, including Netllify, ZEIT, Spotinst and Firebase. Layer 1 cloud providers are Google cloud, AWS, Azure, DigitalOcean and other raw infrastructure providers. These companies provide a great service in their low-cost, their commodity infrastructure, but the layer 1 providers are not optimizing for developer experience. They need to cater to a broad set of developers. Some of those developers want to work at a low level.

For example, the AWS console exposes lots of low-level primitives you can use to build an application. A layer 2 cloud provider can build an opinionated solution that serves only a subset of the overall cloud market particularly well.

Render is a layer 2 cloud provider that optimizes for specific developer workflows, such as deploying a Node.js web server, or a static site, or a Docker container. Anurag Goel is the founder of Render and he joins the show to discuss the strategy and the economics of Render. Anurag was also one of the early employees at Stripe. He discusses his experience and learnings from working at the company.

A few updates from the Software Engineering Daily universe, FindCollabs is the company I'm building. FindCollabs is a place to find collaborators, or co-founders and build your projects. We are having an online hackathon with $2,500 in prizes. If you're working on a project or you're looking for other programmers to build a project or start a company with, check out FindCollabs.

Also, I've been interviewing people from some of these FindCollabs projects on the FindCollabs podcast. If you want to learn more about the community, you can download that podcast. We also have a new Software Daily app for iOS. Many of you are probably using it. If you're not, the Software Daily app includes 1,000 of our old episodes, as well as related links and greatest hits and topics. There's comments. There's connections with other members of the community. There's topics. You can find the things that you're interested in in the Software Engineering Daily catalog.

If you want to get ad-free episodes, you can become a paid subscriber at softwareengineeringdaily.com/subscribe. The Android version is coming soon. You Android users will get that notification from this podcast as soon as it's ready.

With that said, let's get on to today's show.

[SPONSOR MESSAGE]

[0:04:02.7] JM: You probably do not enjoy searching for a job. Engineers don't like sacrificing their time to do phone screens and we don't like doing whiteboard problems and working on tedious take-home projects. Everyone knows the software hiring process is not perfect, but what's the alternative? Triplebyte is the alternative. Triplebyte is a platform for finding a great software job faster.

Triplebyte works with 400-plus tech companies, including Dropbox, Adobe, Coursera and Cruise Automation. Triplebyte improves the hiring process by saving you time and fast-tracking you to final interviews. At triplebyte.com/sedaily, you can start your process by taking a quiz. After the quiz, you get interviewed by Triplebyte if you pass that quiz. If you pass that interview, you make it straight to multiple on-site interviews. If you take a job, you get an additional $1,000 signing bonus from Triplebyte, because you used the link triplebyte.com/sedaily.

That $1,000 is nice, but you might be making much more, since those multiple on-site interviews would put you in a great position to potentially get multiple offers. Then you could figure out what your salary actually should be.

Triplebyte does not look at candidates' backgrounds, like resumes and where they've worked and where they went to school. Triplebyte only cares about whether someone can code. I'm a huge fan of that aspect of their model. This means that they work with lots of people from non-traditional and unusual backgrounds.

To get started, just go to triplebyte.com/sedaily and take a quiz to get started. There's very little risk and you might find yourself in a great position getting multiple on-site interviews from just one quiz and a Triplebyte interview. Go to triplebyte.com/sedaily to try it out. Thank you to Triplebyte.

[INTERVIEW]

[0:06:22.0] JM: Anurag Goel, you are the Founder and CEO of Render. Welcome to Software Engineering Daily.

[0:06:28.0] AG: Thank you. It's really great to be here.

[0:06:30.5] JM: It's 2019. Cloud computing started getting traction about 12 years ago. How has the cloud advanced since the early days of AWS?

[0:06:41.2] AG: Sure. The first thing AWS offered back in I think it was 2006, or 2005 was S3, with simple storage service and S3 continues to exist today. Over time, people have added more and more services to the cloud, both Amazon and other large cloud providers. These days, it's almost impossible not to run your service in the cloud unless you're a legacy, non-technology company and you have existing data centers that you want to move over.

In terms of how the cloud has progressed, there are several different axis, right? The technology axis was just all right, well we start with virtual machines and then we give people access, SSH access to them and they can install things on them using chef, puppet, more recently terraform. Then more recently, just in the last maybe three years, we've seen the rise of Kubernetes. That is a paradigm shift. The reason it is one is because it completely abstract VMs away. No one has to care as much about VMs if their workloads are running on Kubernetes. Now I'll caveat that with the fact that even if you're running a managed Kubernetes clusters, there are a lot of things that the system administrators have to do to make sure your Kubernetes clusters are up and running, but the end user of the Kubernetes cluster does not have to log in, or SSH into a VM. In fact, that's an anti-pattern.

More and more software projects, more and more internal applications, production applications, SaaS applications are being packaged and deployed as containers. That's another huge change that has happened largely because of Docker and advancements in the Linux kernel. We're continuing to see progress in that area with things like Kata Containers and gVisor.

[0:08:48.1] JM: We have the level 1 cloud providers, like AWS and Google. Then there are the cloud resellers; you have Heroku, Netlify, Spotinst. I think Render falls into this category of level 2 cloud providers. How do the level 1 cloud providers differ from the cloud resellers?

[0:09:13.2] AG: Sure. Think of level 1 cloud providers as largely commodity providers who do not differentiate in terms of product offerings, to the extent possible, right? They give you VMs and you can do whatever you want with them. They give you building blocks like S3, or Google Cloud Storage. They have some notion of load balancing. They have some notion of persistent disks and obviously, some notion of internal networking. It's really on you to get everything up and running given those building blocks. They're just Legos and you need to understand. It's a lot harder than putting Legos together, but you need to understand how they work.

In a lot of cases, you need to have certifications from these guys to really build a robust system on some of these large clouds. Having said that, they have made it extremely easy for a lot of people to gain that experience over time. It takes time and effort. Then companies end up building out large DevOps teams to manage their infrastructure in these clouds. That's usually what ends up happening with level 1.

Now when it comes to something like Render, we are focused not on DevOps engineers, like level 1 cloud providers, but on developers themselves. I've been a developer pretty much all my life. A lot of my friends are developers and designers. When I looked at how they were using the cloud for their applications, how they were using Amazon and – I was using AWS and Google cloud. It was just extremely painful for every new app that I wanted to deploy.

For level 2, it's much more about adding value in terms of product and differentiating in terms of product. You mentioned Netlify. Now they differentiate purely in terms of deploying static sites. In fact, they're using AWS to support some notion of lambda functions for some level of backing execution. Their primary product is hey, give us a github repo, or any static site and we will make sure that we deploy it on our CDNs. They do the hard work of deploying everything on level 1 cloud provider.

Similarly, Render makes sure that you don't have to worry about Kubernetes. You don't have to worry about virtual machines. You don't even have to worry about where your servers are located, or how you SSH into them. We just give you a completely self-contained, simple, instant way to deploy your app. It's not intimidating. When you log into AWS, you see a hundred different services. It's not that different for Google or Azure.

With Render, you come in and you just see a few simple buttons that ask you what app you want to deploy. You click on one of them. You connect your github repo that you want to deploy. Most of the time, we detect the commands that you need to run to deploy that app and we fill them in. Usually, people know what the commands are to build their app and to start their app. That's all they need to give us. We take care of everything from there. That includes, obviously building their app, putting it up in production, giving them a render on render.com subdomain, but also allowing them to add custom subdomains, taking care of SSL for them, taking care of redirects for them, taking care of stuff like [inaudible 0:12:58.8] compression or HTTP 2. Now these are all things that you would otherwise have to configure on all DL and cloud providers.

We’re focusing on ease of use and minimizing cognitive overhead and the amount of time that developers have to spend on trying to get their apps up. Because that's not what they want to do, they just want to code and push things to production and get them live ASAP. With

something like Render, every time you push your github repo, you actually see that change deployed in real-time and it's up and running with zero downtime in just a few minutes, or even less if your deploy takes less time to build. That's what we're going for. We're focusing on making sure the developers only care about, or only have to care about code and they don't have to worry about how it's deployed and security patches and things like SSH'ing into their VM to set up terraform, or chef, or puppet. They don't have to learn any of those things.

Even for larger companies that have large teams of DevOps people, our goal is to make sure that the DevOps people there are focused on higher level things, which is what they want to do in the first place, and are working with product teams to focus on things like capacity planning and helping the product teams build apps that are more readily available and secure, as opposed to herding VMs, which is what they do today with the large clouds.

[0:14:31.0] JM: Whenever I talk to somebody about this layer 1 versus layer 2 cloud provider evolution that we're seeing, I think about the transition in the industry from spring and Java-based applications, Java-based frameworks to Ruby on Rails. With the Java-based framework world, with Spring, you had this system that was really good for enterprises for a period of time. You got typed language. It gives you unit testing frameworks. Spring helps you deploy your web application.

As an engineer coming out of college, I would sit down and try to learn Spring and then try to learn the application at the large organization I'll be working at and I would just get exhausted. Then I would go home and I would want to – I would still feel inspired and want to write some software, but I definitely did not want to use Spring, then I found Ruby on Rails and I found documentation for easily standing up something on Ruby on Rails was much more pleasurable.

Now that said, people still use Spring. Spring still has its place. People use Ruby on Rails. Ruby on Rails has its place. Ruby on Rails is used at gigantic organizations. Spring is used at gigantic organizations. Now taking that to the cloud world, that analogy, we have the layer 1 cloud providers, like AWS. AWS is still growing like a weed. AWS suits the needs of certain organizations. Layer 2 cloud providers also serve the needs of certain organizations, like Heroku. There are gigantic companies that are built on Heroku.

What you just said is that you are already thinking about both the individual developer market, as well as the large organization market. How do you see yourself in relation to a layer 1 cloud? What kinds of companies would still make sense to go with an AWS and what kinds of companies would make more sense to go with Render?

[0:16:53.8] AG: I think the answer to that question will change over time. As I said earlier, we're focused on developers right now. As we build out more functionality, as we scale out our systems, it's going to become increasingly possible for extremely large complex applications to be hosted on Render. In those cases, if you see companies that are using Kubernetes right now on AWS or GKE, those companies will be able to run their workloads on Render without having to learn Kubernetes and without having to build out large have DevOps teams to manage Kubernetes, because we'll offer the same functionality and we still offer a lot of Kubernetes functionality without you having to do anything, like private services are built in, VPCs are built in to Render, multi-port services are built in to Render.

The answer to that question, like I said, will change when we've gotten to the point where we can deploy an extremely large application stack on Render. That will probably happen over the next few months and years. At that point, I actually don't – I can't think of many cases where a large company would need to use something like AWS directly. Now having said that, Render might still be running on AWS behind the scenes and Render might still be using GKE behind the scenes. That's completely transparent to the end-user.

[0:18:27.2] JM: Right. You think, maybe we should have these layer 2 cloud providers that should be offering a better experience to people who are developing business logic. Right now, we have layer 2 cloud providers that are built on the layer 1 cloud providers. The layer 1 cloud providers are more like these big utilities. The developer experience for these big cloud providers is just not as high-level. It's not as fast to market as a layer 2 cloud provider.

I think one way of thinking about this is if I go to the AWS console, which I haven't done in a long time, so I could be outdated in my perspective for it. My recollection is it's this big all-you-can-eat buffet of just a huge variety of different services and you can pick and choose your different services and you can find documentation to work through it. It's just a cornucopia of different things.

Whereas if I go on render, I see a more narrow presentation of the different user stories that might suit my needs. I can develop a web app. I can deploy a static site. I can set up an API. I can throw a Docker file on there. I can stand up PostgreSQL. How did you select what user stories you wanted to focus on and how do these user stories fit into your process for architecting Render?

[0:19:54.1] AG: Sure. I was really the first user of what is now Render, because – I used to work at Stripe and I worked there from when it was just 10 people to when it was well over 400 and 5 billion dollars evaluation. I saw the company scale and the company's architecture scale. After I left, I realized that I was completely shielded from all the development deployment infrastructure, because we had a large systems team who were amazing and a lot of them are my friends.

The problem with that is developers, obviously when they get to the point where they have to deploy something online, they log into AWS and see exactly what you just described, which is just this humongously large list of services that don't even have comprehensible names. Someone has to make a website to define what each AWS acronym stands for and what it should actually be called.

The types of services that I wanted to build were actually reasonably complex applications. One of the things I built was a single-click Jupiter deploy backed by a GPU with a bunch of library thrown in. This was called Kressel. This was a company I created and was later sold to dot.ai. This was after I had left Stripe. Along with this, I built out a lot of other applications of varying degrees of complexity.

For each of those applications, deploying them was always a chore. This better application that I just mentioned, that is really when I discovered Kubernetes and Docker, because Stripe wasn't using Kubernetes or Docker back when I was there. That really changed my perspective on how applications will be and can be deployed going forward. Kubernetes has a tremendous learning curve. I went through it and I saw all my friends who were trying to get into Kubernetes but were extremely intimidated by it, because it frankly is quite a behemoth. You have to spend a lot of time coming up to speed. Even then, you run into [inaudible 0:22:20.4]. We've been working

with Kubernetes for years and we run into [inaudible 0:22:23.6] every now and then and we fix them, so our users don't have to worry about it.

The use cases were all okay, well, this is an app that I want to deploy and this is a fairly complex app, something like Elasticsearch, or something like Kafka, but also something as simple as an express API back-end, or a Django, Python app.

When you think about it from the point of view of developers and developer UX, you know the most popular frameworks and apps that they have. You have a lot of online surveys and Stack Overflow has a survey of technologies and there's the TOB index. There are some fairly well-known set of technologies that comprise perhaps 80% or 90% of what people are using today. That is how we define Render's use case is in the beginning. Now having said that, once we launch and once we had users were not us, at that point a lot of our choice of what to build next is governed by what users want. There's so much we want to build, but the priorities are defined by what users want.

[0:23:43.3] JM: You could have been the Andy Jesse of Stripe. Why didn't you just start this cloud provider at Stripe?

[0:23:50.9] AG: Because that is not what's Stripe is doing as a company. It's not their core competency. They're a payments company. They have a system steam, because they need to have a system steam to run the payments business. Now Stripe could have spun off a large cloud provider, but that was just not where Stripe was at in terms of its evolution. To me, at least it didn't make sense.

Also, I didn't happen upon the idea of Render, or I didn't decide to actually build Render until almost a year after I left Stripe, because I was trying out a lot of different domains to figure out what I wanted to do next. I started looking at healthcare, at real-time communications infrastructure, at AI and data science. I was building small apps, prototypes to test out what a good product might be in each of these domains. The chore of having to do that every single time led me to this notion of hey, this is way more complex than it needs to be. All my friends were complaining about it all the time. They were like, "Why do I have to do this over and over again? Why can't I just deploy my code by pushing to github?" I was like, "Yeah, why not?"

For a while, I didn't take it on, because it seemed too big, too ambitious of a task. I eventually decided, or the question before me was that I can either have a AWS build out this utopian developer-focus infrastructure, which seems extremely unlikely based on their history and their DNA. Or I could go build it myself. I think the choice was pretty clear. That's how Render came about. I wasn't even thinking about building something like Render back when I was at Stripe.

[0:25:48.9] JM: Well, I mean it's hilarious, because AWS is not the developer utopia. It is the Amazon e-commerce marketplace for developers. What they really need to start offering is we saw you bought PostgreSQL, you might like SageMaker or something like – it's like literally, the shopping cart experience.

[0:26:13.1] AG: Yeah, and it baffles me that the customer-centric focus that they have on the retail side doesn't really translate to AWS. I've been on support calls –

[0:26:23.4] JM: What? Are you kidding? They own the market. They have 75% of the market.

[0:26:28.1] AG: Well 23, or 30. Yes. I mean, they're just focused on a very different set of priorities when it comes to the cloud business. Having said that, I don't blame them. I mean, they're making so much money. I think if AWS were to be spun out, it would be what? A 200 or 300 billion dollar company? They're doing just fine with their current approach. Why would they change?

[0:26:54.7] JM: Y0ou're saying there is some subset of the market that they are not able to cater to, because they're spread thin, or what's your criticism there?

[0:27:05.7] AG: My criticism is that they don't focus on developer experience. They build for DevOps, they build for people who are systems infrastructure engineers, right? There's only so many of them, especially when you think about the number of engineering jobs that go completely –

[0:27:23.5] JM: What about what about Lightsail? Is Lightsail good? I've never tried that.

[0:27:27.7] AG: Lightsail was there. I suppose their attempt to copy VPS providers, like Linode or DigitalOcean. I mean, you can use it, sure, but a lot of people still use VPS providers, right? Lightsail is just I'll give you a $5 VM and that's it. You can SSH into it and install whatever you want, but that is your $5 VM.

They also have something like Heroku, called Elastic Beanstalk. I've used it and it is a pain. It is extremely annoying to have to deal with everything, because what they've done is taken their concepts of VMs and they've tried to put some lipstick on that peg and they've really failed the developer experience over there as well. Again, it comes from the company's DNA. That's how they started. It just does not seem likely to me that that will change, unless of course they try to buy a developer-focused company, which again, they've chosen not to do so far.

[0:28:36.9] JM: It sounds like, okay, you were head of risk at Stripe. You were there for four and a half years altogether. Then you left and you did a two-year sojourn through the world of startup ideas. Then that's your experience trying to stand up all these little hacks and prototypes and stuff. I'm sure that led to a bunch of infrastructure sprawl and annoying little headaches over the course of building those ideas. That was what led you to start Render. Why didn't Heroku get you there? I mean, Heroku is pretty good experience.

[0:29:15.5] AG: Yeah. The problem with Heroku is that it doesn't scale as your application grows, unless you move to there – also by the way, it's Salesforce Heroku. That's what I like to call it. The problem with salesforce Heroku is that once you want to get even 2 gigs of RAM, you know how much that costs on Heroku?$250 a month. That is completely insane. Why does getting just two gigs of RAM have to cost that much?

Eventually, what ends up happening with Heroku is people get started on it and then as their application grows, they either end up spending way more money than they like, or they run into limitations with the platform. Now let me list out the limitations. Heroku does not have built-in cron jobs. You can't run a private service on Heroku. They restart your dynos every 24 hours. There is no notion of zero downtime deploys, unless you keep two servers up and running and then you pay double. They don't have persistent storage. They'd still don't support HTTP 2. They don't have a notion of private services, which are not exposed to the internet.

There are all these limitations that modern apps run into when they try to scale on Heroku. That's why Heroku didn't work for me. The thing that I was describing earlier called Kressel, where I could just deploy a Jupyter Notebook with a click of a button and it was backed by a GPU, which by the way Heroku doesn't have. That was just not possible on Heroku.

[0:30:49.3] JM: They need to follow the Reddit playbook. They need to spin out Heroku, make it its own company, rework the incentives. I want Heroku back. I want Heroku – I mean, I shouldn't criticize, because I love Heroku. It is yeah. It's –

[0:31:06.5] AG: We've had a lot of users transfer over from Heroku, because they've run into these issues.

[0:31:11.7] JM: I know. It's on your landing page.

[0:31:13.0] AG: Yes, it is.

[0:31:14.7] JM: What about the next generation layer to cloud providers? The ZEITs and the Netlifys and the Spotinsts? Give me your diagnosis of the competitive landscape.

[0:31:26.4] AG: Sure. Netlify is focused on the front-end. They are focused on static sites and they've built additional extremely useful features on top of static sites, like forum handling and authentication. They've had some integrations with them like I mentioned earlier, but they don't really – you can't deploy a node server on Netlify. You can't deploy a simple Django, or rails app on Netlify. That's just not possible. It doesn't look like they're planning to do that. There was actually a Twitter thread where someone was asking them if they could deploy a node server and they were like, “Nope. We’re not really going to do that. You should look at these other options.” This guy was like, “Render does this, because they have both static sites and back-end servers.” Netlify was like, “Well, then maybe you should use Render.” That’s Netlify.

ZEIT has gone in this other direction of serverless, which is again, you can't keep a rails app up and running on ZEIT. You just can't do that with their current iteration, their V2 which was released a few months ago. That is the direction they’ve chosen to go in. Think of Render as the answer to the stagnation of Heroku and the complexity of AWS. What we're trying to do is give

people an extremely user-friendly deployment experience, but also the flexibility and the cost-effectiveness that they require when their applications grow.

[SPONSOR MESSAGE]

[0:33:10.0] JM: DigitalOcean is a simple, developer-friendly cloud platform. DigitalOcean is optimized to make managing and scaling applications easy, with an intuitive API, multiple storage options, integrated firewalls, load balancers and more. With predictable pricing and flexible configurations and world-class customer support, you'll get access to all the infrastructure services you need to grow.
DigitalOcean is simple. If you don't need the complexity of the complex cloud providers, try out DigitalOcean with their simple interface and their great customer support. Plus they've got 2,000 plus tutorials to help you stay up to date with the latest open source software and languages and frameworks.

You can get started on DigitalOcean for free at do.co/sedaily. One thing that makes DigitalOcean special is they're really interested in long-term developer productivity. I remember one particular example of this when I found a tutorial on DigitalOcean about how to get started on a different cloud provider. I thought that really stood for a sense of confidence and an attention to just getting developers off the ground faster. They've continued to do that with DigitalOcean today. All their services are easy to use and have simple interfaces.

Try it out at do.co/sedaily. That’s do.co/sedaily. You will get started for free, with some free credits. Thanks to DigitalOcean for being a sponsor of Software Engineering Daily.

[INTERVIEW CONTINUED]

[0:35:10.6] JM: Taking a step back, you're in the – deep in the startup landscape. There is this conventional wisdom that you should not talk about competitors. If you're in the infrastructure business, the competitive landscape is pretty – everybody's doing cost comparisons, everybody's doing the UX comparisons. Is it such that you have no choice but to essentially form your narrative in relation to your competitors?

[0:35:47.4] AG: I don't think it's as much about not having a choice, as it is about making it easier for developers to understand why and how this is different. AWS exists. A lot of companies use AWS. Why should I use Render? Or why should I use Heroku? Heroku works fine for me, why should I use Render?

This is why explaining what Render does in the context of what exists today becomes helpful for developers as they see okay, yeah, actually I don't think of that and I think I need that. Oftentimes, this is very true in all kinds of products, users often don't know what they need until they see it. I think that Render is doing a lot of that.

[0:36:41.7] JM: Okay, so let's say I have an app. I have a Node.js app. It's got a React front-end. Describe the workflow for getting my app created and deployed to Render. What is my developer experience and then what's going on in your infrastructure to get that thing stood up?

[0:37:03.3] AG: Sure. Let's talk about the experience first. I'm assuming that you have a github repo for your front-end React app and you have another repo for your Node.js back-end, or you could even have the same repo for both, it doesn't matter. The developer experience is you connect that repo, or both repos to Render and that connection involves just giving Render's github app permission to access your repos, so we can actually retrieve and build your code. That's one click. Then we bring you to a screen where we show you the defaults. We detect what application exists in that repo.

Most of the time, we're right, but we obviously can't get it right a 100% at the time, so we show the user what we think their repo deployment should look like in terms of what their build command should be and what their start command should be. Those are the only two things people have to worry about really. Usually, a build command is as simple in the case of a React app, we don't even need a start command, because it's a static site that we deploy over our CDN.

All you need to do is run yarn build, and that's already pre-populated. Really, you connect the repo and then you click another button and that's it. We start building your front-end and we give you a URL. That front-end is live on that URL as soon as your bill finishes, which is in – depending on your app of course, but usually it's the order of minutes. Then you have a live app

up and running. This is on a CDN. It's globally distributed. It's taken care of in terms of SSL. It's over HTTP 2. You have all these nice things out of the box for static sites.

Then for your node app, it's very similar. That's I think what's causing all these developers to really fall in love with the product, which is again, you connect your node repo, and in this case, you have a build command, which is typically NPM run build, or yarn build. Then you have a start command, which is typically NPN, run start or yarn start. Again, that's it. You fill in those two things, or they're prefilled and you make sure that they're what you want and you click a button.

Again, we give you a URL for your back-end app and it's live as soon as your build finishes. You can see the progress of the build. We show you the build logs as soon as you click the final button, which is actually just the second button. You can follow along. As soon as it's live, you see that it's live and you can go to your URL and that's it.

You don't even have to install anything. It can all be done on a regular browser of deployed sites from zero to production on my iPhone. You don't need the command line. Everything is extremely streamlined and you can do it in minutes.

[0:40:00.3] JM: Why would you make a deployment from your iPhone?

[0:40:02.4] AG: Because I want to.

[0:40:04.8] JM: Did you make a code change? Did you edit a file, or were you just messing around?

[0:40:11.2] AG: I was just showing how easy it is to someone. The fact that you didn't need to deploy – you don’t need to install some CLI, or make sure that you had all these fancy tools at your disposal to be able to deploy an app.

[0:40:27.0] JM: It is the way it should be. My little brother is always at the bleeding edge of doing weird things with his infrastructure. He for a long time, wanted – he had an Android phone. He had some strange CLI thing in his Android phone where he was SSH’ing into –

[0:40:47.1] AG: That's terrible.

[0:40:48.6] JM: It was really bad. I encouraged him, because he was in college and I was like, "Yeah, you know what? Just do your SSH terminal from your phone. All the more power to you."

[0:41:03.1] AG: This is like asking people to learn assembly. We don't do that anymore. That is where we are with the state of deployment. We're literally asking developers to learn the equivalent of assembly code just to deploy their apps.

[0:41:15.4] JM: What is going on in your back-end? You mentioned AWS and GKE. If I deploy my app, is that getting scheduled on to a Kubernetes container somewhere? What's going on there?

[0:41:30.0] AG: Yeah. Without obviously giving too much away behind the scenes.

[0:41:33.8] JM: Oh, come on. Nobody's going to steal your technology. Everybody knows you're just scheduling it onto a Kubernetes container, probably deploying it to AWS, maybe also GKE.

[0:41:43.9] AG: Yeah, yeah. I think you got all that. Do I need to say anything else?

[0:41:48.0] JM: Yeah, tell me more. I mean, give me a little bit of secret sauce.

[0:41:51.0] AG: Sure, sure, sure. I was just kidding. Okay, so what we do behind the scenes is again completely transparent to the user. They don't even know – they shouldn't have to know that we're building a container and we're running things behind the scenes, unless of course they wanted to put a Docker file, then they obviously know that we're building the container. Behind the scenes, what we're doing is we have this notion of converting code to deployable artifacts, which is essentially the build command, which when you run yarn build or NPM run build, we are running that command and taking all that output and then deploying it in a different container, which is completely segregated from all the other users and containers on Render.

That is the container that spins up your app that starts your app using the start command that you supplied. We're constantly monitoring these containers using controllers in the backend. As soon as we detect that your build succeeded, we start the deployment process. As soon as we detect that your deployed succeeded, we let you know in the front-end. Obviously, if there are any failures along the way, we let you know of those as well.

Behind the scenes, it's essentially containers on Kubernetes, but that is not to say that we don't plan to have other technologies to deploy the same things. Containers and Kubernetes are a means to an end. They're not the secret sauce, or the primary reason for Render to exist.

[0:43:31.6] JM: Completely agreed. This will allow you to talk more honestly about your perspective on this space. I just did a show about operators yesterday, the Kubernetes operators. This is a higher level pattern for deploying a distributed system on Kubernetes. You can deploy a PostgreSQL database, or an etcd, or a Kafka, something like that. I did a show in KubeCon recently with the – have you looked at the Crossplane project?

[0:44:01.5] AG: Yes.

[0:44:03.0] JM: Okay. What do you think of the cross-cloud, multi-cloud story in Kubernetes? Do you have a strong opinion on where we're going, or are you just waiting to see what happens?

[0:44:15.1] AG: I think hybrid cloud is actually more important to a lot of companies than cross-cloud. Usually, cross-cloud is one of those things that sounds great in practice.

[0:44:33.4] JM: Wait, but these are totally disjoint sets of customers. I mean, not totally disjoint, but mostly disjoint. I mean, hybrid cloud is you're a bank, you've got a data center, you want to merge with AWS and have scale out into AWS or whatever. Cross-cloud is your thumbtack and you have all your business logic in AWS and you want to run ETL jobs and BigQuery jobs in Google Cloud, so you need some cross-cloud stuff there.

I mean, these are different sets of problems. I am certainly interested in your hybrid cloud commentary, but I was more curious about the cross-cloud stuff here, because that's more in your purview.

[0:45:13.0] AG: Sure. The reason I brought up hybrid cloud is because I think that when people talk about cross-cloud, it isn't as common as the hybrid cloud situation from what I have seen. Because if you're an AWS, you built up all these years of expertise with your DevOps team in AWS and you want to put all that expertise to use. It takes a lot of time to get acquainted with another large cloud and learn the ins and outs. Obviously, even if you're running on Kubernetes, the nature of Kubernetes distributions in these clouds is very different. The versions that you want to run on are different. There are all these gotchas.

My take on cross-cloud is that it should be as easy to run your service on cross-clouds as saying, “Hey, I want to run the service on both AWS and Google and I'm going to click those two buttons in my dashboard.” That is what Render can provide, because we are the ones managing that cross cloud infrastructure and we have your code, we deploy containers for your code behind the scenes to multiple clouds. That's the easiest way for you to go cross-cloud. That is something that we intend to add in the future.

[0:46:39.0] JM: What about the CDN? The CDNs are getting more richly featured. You got Fastly and CloudFlare that are presenting a world in which we're going to want to run our applications at the edge. Maybe we want to do it in WebAssembly. Maybe we want to do it in containers or something. What do I want out of a CDN today and what am I going to want out of a CDN in the near future?

[0:47:05.6] AG: Sure. It depends on what company you are. For most companies that are not serving large files, like video and audio from a CDN, your needs are very different from say Netflix’s need from a CDN. If you're just a developer who's deploying a static site that might be extremely heavily trafficked.

The biggest difference is serving small files on a CDN is technically very different from serving streams of large content, right? Most of the users we have, or actually all the users we have right now for CDNs and this applies to most level two cloud providers, they all cater to the

developer market where you can serve content that is split up into small files very effectively over a global CDN with minimal latency.

Then you have when you talk about Fastly and CloudFlare and other CDNs that are offering edge functionality, I think that's again, that's a layer 1 CDN. Then you have your layer 2 CDN providers, which I guess is Render. It's the same analogy that we have in compute. I think that will continue to exist. You will continue to have something like CloudFlare and Fastly and stack path and key CDN.

What we'll see is again, people are going to build much more user-friendly experiences on top of these providers. For now, if you wanted to deploy your app on a CDN and you decided you wanted to use CloudFlare, you still have to have a back-end to deploy your app on and you just put CloudFlare in front, so that it can be served more efficiently, or specific resources like static site files, like CSS, JS, HTML can be served more efficiently.

Now there's this separate concept of running code at the edge. I think that that is extremely useful for certain situations. When it comes to most applications that exist today, and even for applications that might exist five years from now, it feels to me that running code at the edge will have its place, but it won't be the majority of compute in any meaningful way.

[0:49:40.4] JM: A bold statement. I think I agree with you though. I think it's going to be like machine learning workloads or something.

[0:49:46.3] AG: Right, exactly.

[0:49:47.4] JM: Something very niche.

[0:49:49.5] AG: Exactly. Again, I mean, these things will continue to exist and should continue to exist, because they serve the needs for people who have those needs, right? It's the same thing with something like serverless. There is obviously so much hype about serverless. I'll tell you this, none of our thousands of developers, Render users have asked for a serverless solution. That is –

[0:50:14.5] JM: The cold start. The cold start is such a killer.

[0:50:16.6] AG: It is. Then it's not just that. You have to architect your application in a way that is making you beholden to the serverless cloud provider. Sure, there are translation layers and you can deal with –

[0:50:32.5] JM: Knative.

[0:50:33.9] AG: Yeah, yeah, yeah. I mean, Knative we can –

[0:50:36.9] JM: Knative presented by Google.

[0:50:38.2] AG: Well, Knative is open source, cloud run is the Google equivalent. Well, they're using Knative under the hood, but yes.

[0:50:44.9] JM: Who owns the Knative repository? I actually don't know the answer to that question, but I'm not sure if it's been donated to the CNCF yet.

[0:50:51.8] AG: Yeah, that's a good question. I haven't looked recently, but I would be very surprised if not now, eventually it becomes part of the CNCF. It's still a lot of Google engineers working on it, so it is what it is. Yeah, serverless has its place and it is great for certain workloads that need to scale to arbitrary volume over spiky periods of time. For your average web app, or for your average server API, serverless just doesn't make sense, because we build up all these paradigms over the years to optimize apps, like connection pooling and caching and all of that goes out the window. The only thing that you have to say for serverless is oh, you only pay per request. Do I really care that much about paying per request if compute costs me $5 a month, which it does on Render?

At some point, the price becomes irrelevant because ease of use wins over. That's Render's answer to serverless. Having said that, I'm not going to say no to Knative, or exposing serverless in an equally user-friendly way if and when we have enough demand. Like I said, we just don't see that demand.

[0:52:14.1] JM: Yeah. Well okay, we'll go maybe deeper into serverless another time. You mentioned pricing. Looking at your pricing, it seems pretty cheap. I don't have the full comparative advantage to other cloud providers. Thinking about your depth of strategy and you come from Stripe, so I think what you're doing here is you are looking at the landscape of choice that developers want and you're saying, "Okay, I'm going to have a layer 2 cloud provider that stretches across all the layer 1 cloud providers and offers the developer experience that everybody wants out of the layer 2 cloud providers." Everybody's getting out of the layer 2 cloud providers. People love Netlify.

You want to offer the depth of choice that people get from the AWS deep services and the Google deep services, the BigQueries and the SageMakers and the Redshifts and these super cool services that actually cost a lot of money. You want to offer it in a more friendly manner. The cheapness of your services today, do you think of it as lead gen for services that you will build yourself in the future?

[0:53:44.6] AG: Not really. By the way, that description of Render was extremely accurate. Thank you for doing my work for me. Yes, we do want to offer the same flexibility and power that level 1 cloud providers as you call them, have. The pricing is really for us, it's more of a question of making things accessible to our end-users and at the same time, not losing money on compute. That is the last thing we want to do.

We're not losing money on compute with this pricing and it's not a marketing cost that we're going to write off in our S1. We're simply what we think is fair and it is some combination of what it costs us, but also the value that we provide to the end-user. We're not going to be and I don't think we are necessarily the cheapest cloud provider always, and that's not what we're going for. We're not going to compete on price. Competing with on price with AWS is insane, right? No one should do that. If they do, well good luck to them.

It's much more about giving people enough value where our prices make sense to them and they don't feel – and this was something that a user said to me recently, a Render user said to me, their Heroku bill started feeling like a second mortgage. Then they moved to render and they're extremely happy and they don't have to take out a second mortgage on their house. Render is not losing money on them, right? The pricing model is essentially look, here's what we

think is fair based on the value we're providing. What we've seen so far is people respond really well to it.

[0:55:37.9] JM: This is what is so crazy about the Heroku thing. It's like, what are they doing? Why is it so expensive? It's a rounding error for Salesforce. It's somehow –

[0:55:48.4] AG: Do you want to know?

[0:55:50.2] JM: Okay, tell me. Sure.

[0:55:53.8] AG: I think and I could be wrong. I've spoken to a lot of ex-Heroku people and some of their investors. I think that the perpetual free tier is what's causing a lot of this. In effect, if you're a paying Heroku customer, you are subsidizing all the people who abuse their feet here in a big way. That is unfair to you as a paying user and that is what we're avoiding at Render.

[0:56:25.3] JM: I guess, their free tier is pretty good.

[0:56:29.3] AG: Yeah, you can run a node app.

[0:56:32.1] JM: It just falls asleep is the thing.

[0:56:34.7] AG: Yeah. I mean, people have ways to get around that.

[0:56:36.2] JM: People paying it.

[0:56:37.2] AG: Yeah, exactly.

[0:56:37.9] JM: Yeah, people paying it.

[0:56:40.3] AG: People paying these add-ons, or just stuff you can do to ping it and super easy. Then they have a free PostgreSQL here. I just think that that takes a lot of compute capacity, which is never free. Computers much more expensive compared to memory, or bandwidth, or storage.

[0:57:00.3] JM: They should just do rate limiting though. Monthly rate limit.

[0:57:03.8] AG: They do have that. They have this notion of 730 hours or something like that. I think, the key thing to understand about what we do at Render with our pricing is we don't have this notion of all right, well you can just get compute for free, because it's not free for us. What we do is we give people a trial period during which they don't have to enter a credit card, they can try Render out and they can deploy back in services without entering any payment information. Then if they like it, they should enter a credit card and then they become a paying user.

[0:57:41.5] JM: I mean, are the unit economics profitable for all for all your services and your databases and your cron jobs?

[0:57:47.6] AG: Yes.

[0:57:49.5] JM: Okay. Interesting.

[0:57:50.6] AG: Yeah, so that is one thing that we have just said as a line in the sand that we're just not going to cross that line any time soon, unless something big changes in the macro landscape.

[0:58:04.9] JM: I heard AWS does that with cloud front. They offer cloud front at a loss, in order to build more of a market. They can do that, because they can look at the portfolio of things. You don't have enough volume yet, where I guess you could say – I mean, you are offering static sites for $0. That's what you're doing. Those are so cheap.

[0:58:25.1] AG: Yeah, cloud front bandwidth is actually one of the cheapest thing, especially when your AWS bandwidth is probably the cheapest you can find anywhere, right? Because they probably have contracts with all the different ISPs and these ISPs can't not give AWS good pricing. It's like Walmart and their vendors. Even without that bandwidth, that’s cheap. That is why we offer static sites for free as well. We will have paid tiers for static sites, which will have additional features, but that is our way of saying, “Okay. Well, this doesn't cost us that much.”

We have this portfolio and we're hoping that you like the static side offering so much that when it comes to deploying a back-end, or a cron job, or database, you'll use us.

[0:59:10.7] JM: Tell me something that you learned at Stripe that I wouldn't hear from anyone else.

[0:59:15.8] AG: I think people underestimate the value of appealing to a small segment of underserved developers. I think that that is something that I learned firsthand and that is applicable to Render as well. It seems like, “Oh, these are just developers with hobbyist projects.” You can't really build a big business off of them, but Stripe is a 22 and a half billion dollar company and guess how they started? All their initial customers, the first few hundred customers, they were all individual developers. We didn't really start getting large companies using us until well after launch.

Also, because no one wants to use a payment provider that is such a critical part of your business that just came out, right? This is why Render is also catering to developers and large companies are hesitant about us, because we're new and I get it. Having said that, the thing that people don't realize is that if you build up enough of a critical mass with developers, some of those developers are going to become Airbnb and Snapchat and Dropbox. You grow with them. That is how your company becomes successful.

[SPONSOR MESSAGE]

[1:00:48.4] JM: Deploying to the cloud should be simple. You shouldn't feel locked in and your cloud provider should offer you customer support 24 hours a day, seven days a week, because you might be up in the middle of the night trying to figure out why your application is having errors and your cloud provider’s support team should be there to help you.

Linode is a simple, efficient cloud provider with excellent customer support. Today you can get $20 in free credit by going to linode.com/sedaily and signing up with code SEDAILY2019. Linode has been offering hosting for 16 years and the roots of the company are in its name. Linode gives you Linux nodes at an affordable price, with security, high availability and customer service.

You can get $20 in free credit by going to linode.com/sedaily, signing up with code SEDAILY2019 and get your application deployed to Linode. Linode makes it easy to deploy and scale those applications with high-uptime. You've got features like backups and node balancers to give you additional tooling when you need it. Of course, you can get free credits of $20 by going to linode.com/sedaily and entering code SEDAILY2019.

Thanks for supporting Software Engineering Daily and thanks to Linode.

[INTERVIEW CONTINUED]

[1:02:25.7] JM: When you were at Stripe, you spent some time working on marketing and PR, but I mean, you were and you were a developer. I mean, you ended up the head of risk. Pretty interesting diversity there in terms of your work within Stripe. Now that you have your own developer tools business and you spend some time working on marketing at Stripe, what's your perspective on how developer tools businesses should think about marketing, in addition to this bottoms-up sentiment that you just expressed?

[1:03:01.4] AG: Sure. I think there's a difference when it comes to developer tools businesses and what we're doing, in the sense that there are developer tools businesses like crossplane that don't actually necessarily host things for you, right? There is a real cost to us when we host things for you. You can use whatever developer tools you're using. In the end when you host something that – it's essentially a platform for deploying your app as opposed to a tool that a developer is using in the course of their work, something like when you mentioned GraphQL, Prisma for example is a good example of a developer tool. Again, I wouldn't call Stripe a developer tool either. To go back to your question, marketing to developers, I think that is that the essence of your question, if I'm not mistaken.

[1:03:59.0] JM: Yeah, yeah.

[1:04:00.1] AG: I think that it's all about developer empathy and understanding that developers do not like to be sold to. I'm a developer, I don't like to be sold to. I hate ads. I block them everywhere. I'm skeptical of every new thing that comes out that claims that is the best thing

since the sliced bread. I need to test it out myself, or I need to learn from a trusted source that it is actually the best thing since sliced bread.

You can choose to run ads and of course, there's this whole notion of people use what they encounter. If you are running a lot of ads, or if you're doing a lot of PR, or marketing then obviously, more people would know about you and then consequently, more people will use you. In the end, it all comes down to the product because sure, you can get them into the funnel, but they're not going to stay if your product doesn't live up to its promises. That was the thing that Stripe did really well, continues to do really well. That is what render is doing, because when people sign up for Render, we see over and over again, people are extremely delighted by what they see. They're just like, “Oh, my God. This is so awesome. I had not expected it to be so easy.” That is why they stay. You can spend however much money you have or want on advertising, but your product has to live up to it.

[1:05:31.8] JM: Yeah. I mean, you're plucking my heartstrings with your words. We have four ads per show, four podcast ads and they're interruptive and some people don't like them. Some people will get exposed to new tools from them. Companies have had great results from our ads. They are getting leads that convert and make them good money. Me, as I'm a power a podcast listener. I listen back to many of my episodes to do a quality check on them. I do not skip my own ads, because I –

[1:06:11.7] AG: Fair enough.

[1:06:13.1] JM: I want to endure the listeners’ experience. At the same time, I can recognize that this may not be the end state of the content that I'm producing for people. Because ads, the interruptive quality of it, I mean, oftentimes when I am talking, or I'm trying to sell companies, cloud infrastructure providers on these ads they're like, “Yeah, I think we'd rather spend the budget on a conference.” I'm like, “Really? In a conference? Is that what you want to spend your money on?” I can't really fault them, because actually the more I think about it's a conference, it's not interruptive. People are there to walk through the expo hall as part of their journey at the conference.

Anyway, I hear what you're saying and I think it's interesting – it's an interesting niche to be marketing to, to the extent that marketing can be interesting. We both start as engineers. I don't think either of us had the intention of getting into a business where we would have to do marketing, but nonetheless, we find ourselves here. I think it's an interesting area to do marketing within.

[1:07:19.1] AG: Yeah. I think that there are other ways to make yourselves known to developers. A lot of those ways actually involve actively being helpful to developers, right? Being able to show them how to deploy a Redis server on Render – we were talking about content marketing now, but it's not just content marketing, it is actually useful – extremely useful content, because it allows people to achieve their goals. That is what we'd rather do than run ads. We'd much rather have a lot of content around being able to build and deploy applications that help our users and that help potential developers, not just our users, rather than run ads.

Guess what didn't work at Stripe when we were testing ads, well ads. We ran ads on Stack Overflow. We ran ads on Google and they just didn't work. I mean, I'm pretty sure Stripe doesn't run them right now, but I wouldn't know because I have all of them blocked. I'm pretty sure that ads just –

[1:08:24.9] JM: Stripe advertisers on 20-minute VC.

[1:08:27.2] AG: Well, they're going out for a different market now, right? They're going after the startup CEO market. Yeah. That's different.

[1:08:35.3] JM: You're talking about relative to banner ads.

[1:08:36.4] AG: Right, exactly. When I was at Stripe, back in 2012 I worked with Stack Overflow and we created banner ads. It was just a waste of money. Then we ran Google ads and that was also a waste of money. Then a few years later, we tried them again thinking that maybe the first time we didn't know what we were doing and it would work. They still don't work. This is why you just won't see Stripe advertising on Google ads and I don't think Render would either. It's much more about making things and helping people discover us by being useful to them.

[1:09:15.1] JM: I hear you. I know we're out of time. You got time for just one more question?

[1:09:18.2] AG: Of course.

[1:09:19.6] JM: Okay. You went to IIT in India. I've started to try to understand how developers in different places in the world other than the United States, how they differ in their strategies and their mentalities. I go into Shanghai for KubeCon last year. I just got back from Europe. I think Europe, I didn't really get a sense for regional developers there. Just like you have regional cuisine, you have regional art, regional music, there is a regional sense of how businesses come together, how software engineering takes place. What is unique about the way that engineers in India build software and build businesses?

[1:10:09.0] AG: I wish I could tell you. I moved to the states right after I finished college. I never really saw businesses being built in India, or software engineering being done professionally in India. My first job out of college was actually a startup in Boston. Having said that, I will say that there are – it's hard to put a label on a large set of developers from any segment or country, because it's a very large population. You're going to have a distribution. You have developers in India who are just as passionate and skilled and contributing to open source in ways that you'll see in the US. Then you also have developers in India who were just trying to pay the bills. It's just like you see here in the US. That's what I have seen just from the outside. I don't know if there is a deeper, more substantive difference between developers.

Now obviously, some of it comes down to education and your motivations. One thing I will say is in the US, a lot of developers end up – or at least this used to be the case. I think it's changing in the US too, but a lot of engineers became engineers because they wanted to become engineers. In India and in places like China where the economy isn't as plentiful as the US, or there are still large sections of the society that are not rich, you'll find that a lot of people see becoming an engineer, but also not just an engineer, like a doctor, lawyer, those things become a means to an end, as opposed to something that you enjoy doing. That might be one difference.

Like I said, I think that it's changing. Even in the US, you see places like lambda school, where people who were waiters decided that they want to learn coding. Then they go and find a tech

job, because it's paying them a lot more than they were making earlier. Again, it's like, you'll find people of all kinds, you'll find developers of all kinds in other countries. It's hard for me say, “Oh, look. India developers are different in this way.” Also, I don't have enough data for you unfortunately.

[1:12:40.9] JM: All right, well fair enough. Well, it's been really fun talking, Anurag.

[1:12:44.6] AG: Likewise.

[1:12:46.4] JM: Your company is really interesting. I am definitely going to figure out something to throw on there and to try it out.

[1:12:52.0] AG: Please do.

[1:12:54.0] JM: We should definitely do another show in the future, perhaps when you launch your high-margin data warehousing product.

[1:13:03.1] AG: Hopefully before that.

[1:13:04.2] JM: What's going to be the first time Argent's service, is it going to be data warehousing, managed Tensorflow? What are you thinking?

[1:13:10.3] AG: Oh. Well, so our users want managed Redis. We have managed PostgreSQL, but we don't have managed Redis yet. There are people who want managed MySQL.

[1:13:22.9] JM: I think Kafka. Actually, I guess you could do Kafka pretty easily now. Kafka is pretty easy. If you do use the operators, it's probably easier, right?

[1:13:30.4] AG: Yeah. Actually, we use operators for our managed PostgreSQL offering already. I think that operators are still in their infancy and –

[1:13:40.0] JM: Still early.

[1:13:41.1] AG: Very, very early. A lot of them have bugs and a lot of them are – they don't claim to be production-ready. That will obviously change and it's going to make things easier for people who are used to managing Kubernetes clusters and who can manage Kubernetes clusters. It won't make things easier for developers, because they're operators, like what? CRDs, what? Right? No one should have to understand all of that just to get a rails app up and running.

[1:14:07.8] JM: You should just do a data warehouse. I was talking to somebody about this. Indie Hackers are going to need a data warehouse and they have no idea how to operate a data warehouse.

[1:14:17.4] AG: I don't know. I can ask –

[1:14:18.5] JM: It's my own question.

[1:14:20.6] AG: I can ask Courtland. He's using Firebase right now for all this data.

[1:14:23.9] JM: Oh, right. I know.

[1:14:25.6] AG: Maybe he needs a data warehouse. I'll ask him today.

[1:14:28.9] JM: I don't know if he needs it today. I don't mean Indie Hackers specifically. I mean, Indie Hackers more generally.

[1:14:35.4] AG: Oh, I see. I see.

[1:14:36.9] JM: Right, right, right. That Indie Hackers. That's hilarious. No, but seriously, these people who they start a startup out of frustration or whatever, it's Indie Hacker business, it's a solo entrepreneur business. It takes off and then they don't scale up their hiring squad, so they're not getting data infrastructure people.

[1:14:57.0] AG: Yeah, because it's hard.

[1:14:59.4] JM: It's hard.

[1:14:59.4] AG: It is so hard to hire engineers and it's even harder to hire DevOps engineers.

[1:15:04.1] JM: Yeah, but I don't even – I actually don't even know how hard it is to – maybe there already is a data warehouse solution that Indie Hackers can operate. I'm totally misguided. Maybe I don't know the market.

[1:15:14.9] AG: Also, you need to have yeah, so I think data warehousing is definitely becoming more and more important. We'll see that as people build out more applications that can derive more insights from the data people already have without you having to manually write SQL queries. We're seeing some of these startups just coming up now. That's going to mean people will need data warehouses, just like they use Google Analytics and other analytic solutions today, data warehouses I think are going to become this lightweight thing that you can use to get really quick insights on what's going on with your data. Yeah, maybe we'll think about that. Like I said, a lot of it depends on our users and what their needs are and as we see more users wanting to build data warehouses, maybe we'll build it as a hosted solution.

[1:16:04.9] JM: Well, yeah. I mean, I think, yeah, you're probably right. Just sitting here, maintain optionality, your current vision, you got a lot of ground to cover. Man, we didn't even talk about – you mentioned gVisor. We have a lot to cover for the next show.

[1:16:18.4] AG: Yeah.

[1:16:19.0] JM: Okay.

[1:16:20.0] AG: Kata Containers. I actually think Kata is slightly better right now.

[1:16:24.2] JM: I thought – don't you need both of them?

[1:16:26.5] AG: No, no.

[1:16:27.4] JM: You don't need both?

[1:16:28.0] AG: No.

[1:16:28.4] JM: Okay. Because I thought gVisor is not actually a container, it's just like a shim that sits in front of your sys calls. It's not an actual container.

[1:16:38.7] AG: It's not. Yeah. I think that you can still use Docker containers with gVisor. Kata Containers is a completely different stack in that sense. It's micro-VMs in a way.

[1:16:50.7] JM: Right. Doesn't it still have the full syscall interface?

[1:16:54.9] AG: Kata Containers doesn't actually shield syscalls. It doesn't worry about what syscalls you can make or not. That's actually one of the reasons is it's much faster than gVisor.

[1:17:04.8] JM: No, but that's my point, is that these are disjoint solutions. I don't know understand what you're saying.

[1:17:08.9] AG: They're disjoint solutions, but if you want to run isolated workloads, then you use one or the other. You don't want to use both. There's no point in using both is what I'm saying.

[1:17:19.5] JM: I see.

[1:17:20.3] AG: Because Kata Containers isolate your workloads in your micro-VM. Then gVisor isolates your workloads in your Docker container. That's the main difference.

[1:17:32.1] JM: Okay. Interesting. Okay, well maybe this will be good for another show. I'm definitely out of my depth here. That sounds like, we can do a whole show on firecracker, micro-VMs, subjects that have nothing to do with Render.

[1:17:44.6] AG: Well, they do because we run isolated workloads, right? We're running multi-tenant clusters.

[1:17:52.8] JM: That would be a can of worms, but the firecracker thing I thought was such a curious open-sourcing by AWS. It's like, AWS, we don't do open source. Then all of a sudden, AWS, here's our open source VM thing?

[1:18:06.9] AG: Yeah. Yeah, I wonder what prompted that. I mean, it could also just be internal engineering pressure, where people were like, "Look, we have all of this and Google's open sourcing gVisor and Kata Containers is doing its own thing and we built this out. Why don't we open source it? It makes us look worse or whatever it is, right?" I don't know what the internal discussion was.

[1:18:31.0] JM: No, it makes them look better, right?

[1:18:32.4] AG: No, what I mean is if they didn't, then it would make them look worse. Yeah. It would make them look worse than GCP, or whatever. Yeah, so maybe that's why they did it. Probably, it's also because lambda is this black box to a lot of companies. Knowing that firecracker, the container runtime is open source might give some of their larger customers more confidence in being "locked into lambda."

[1:19:02.1] JM: Okay. Well Anurag, thanks for coming on the show. Been really fun talking to you.

[1:19:06.0] AG: Likewise. Thank you.

[1:19:07.5] JM: We will do this again in the future.

[1:19:08.7] AG: Absolutely. I'm really looking forward to it. It was a really fun conversation. Thank you for having me.

[END OF INTERVIEW]

[1:19:16.9] JM: GoCD is a continuous delivery tool created by ThoughtWorks. It's open source, it's free to use and GoCD recently launched a test-drive service that makes it easier than ever to try out GoCD. You can go to gocd.org/test-drive-gocd.

If you've been wondering about what continuous delivery tool you should use for your cloud native software, GoCD is worth checking out. Now it's easier than ever to just try it out and see if this looks like something that you would want. Just go to gocd.org/test-drive-gocd and find out how GoCD fits your workflow.

GoCD has support for Kubernetes and it was built with the learnings of the ThoughtWorks engineering team. If you want to try it out, go to gocd.org/test-drive-gocd.

[END]