EPISODE 1393

[INTRODUCTION]

[00:00:00] KP: As cloud providers enable greater levels of specificity and control, they empower compliance driven enterprise companies. This level of parameterization is downright inhospitable to a new software engineer and can be a cognitive barrier to entry for even a senior professional with a great idea, but limited time. Developers want to focus on their code, algorithms, front end, and user experience. These concerns have created a huge vacuum in the market that companies like Render are filling. Render is a unified cloud experience to build and run all of your apps and websites. It can auto deploy from Git and offer some really novel developer ergonomics.

In this episode, I interview Anurag Goel, founder and CEO of Render.com about their approach to delivering a developer centric, all in one cloud solution.

[INTERVIEW]

[00:00:51] KP: Anurag, welcome to Software Engineering Daily.

[00:00:53] AG: Thank you, Kyle, it's great to be here again.

[00:00:55] KP: Yeah, I guess I should say welcome back. For listeners who need to go into the archives and hear the previous episodes after this one, if they're tuning in first here, can you give us a sense of your background,

[00:01:07] AG: I am the founder and CEO at Render, which is a new cloud provider, not so new anymore. We're about four years old and we serve billions of requests every month and have tens of thousands of customers. And before this, I was the fifth engineer at Stripe, which I joined pre-launch in 2011 and helped grow to over 400 people by the time I left in 2016. I love developer products and building the best tools that exist in the industry. Obviously, did a lot of that at Stripe, and I'm very excited about doing that at Render.

[00:01:50] KP: When I think about cloud platforms, I think of, the baseline, your big three, Amazon Azure, GCP, and then on top of that, things like Heroku and all the way up the stack, maybe, Zapier and if this then that's at play. Do you guys conform to that sort of hierarchy? And if so, where do you fit in it?

[00:02:09] AG: It's an interesting space for us, because in some ways, we're similar to Heroku and we have a lot of former Heroku users who use Render because they like the features and the price point which is much more reasonable and scalable compared to Heroku. But in other ways, we have features that AWS has, but that require a lot of work to do to set up on AWS, like private networking, or persistent storage. So, you can run something like Elastic Search on Render very easily compared to Heroku, where you just can't run anything that is persistent. But then also, on the flip side, on AWS, you would have to set up a lot of stuff to be able to get to the point where you can run Elastic Search in a private network and then make it accessible to your services through a discoverable address.

[00:03:02] KP: So, when a developer first signs up for Render, is there a menu of services that they choose from? How do they typically get going?

[00:03:10] AG: Yeah, they can pick from a number of different service types. So, you can obviously create any kind of HTTP service, whether it's an API or a full stack Rails app. But you can also create a pure static site, which is served by a global CDN, and you can create other forms of services that typically, micro service-based architectures or even full stack architectures need. For example, we give you a managed database in Postgres, and we're going to launch our managed Redis offering very soon. We also allow you to run background workers and cron jobs. And the other service type we have is actually pretty different and unique. We call it a private service. And this is the equivalent of, if you set up console and VPC's and security groups and firewalls on AWS, all you need to do for Render is connect your repository, give us your build command and start command for the service, and we automatically give you an internal only URL along with whatever ports that it's opened on, that you can connect to from other services in your network. But it's not visible to anyone outside that.

[00:04:30] KP: Very interesting. Well, let's say I've already got a Rails app or something like that, and I'm looking for better options for how to host it. Maybe I'm looking for cost reduction or better services and features around it. What's the typical path onto the platform like?

[00:04:45] AG: It's actually fairly straightforward, especially for something like Rails or in other languages like Python, something like Django. We ask you to set up your Render account with us integration with your GitHub or GitLab. And once you do that, the repository that contains your Rails server and all the other components along with it, we have a guide on our website to deploy Ruby on Rails on Render. But even without that guide, you can actually click through, select your Rails repo on our dashboard, and just give us a build command, which we auto suggest, we detect that it's a Rails app, and we suggest it. You probably need to click a button to generate like a secret key or something, and then you can also create a database with a single click, and there you go, you have a Rails app up and running.

On our website, we actually have a whole tutorial on how to update your existing app for Render, which basically just means, making sure that you have set up all the correct database settings, and then you've also like made sure that you have the right number of workers and things like that. So, it's very straightforward and we consistently hear from our users that it's a lot easier, even compared to Heroku.

[00:06:13] KP: Once Render's built my app, and I can connect to it, and I've fixed any configuration errors I made. That's great. I've got the URL, and it's up. But what do I actually have is that a serverless version, or a single worker? What's the actual instance I'm connecting to?

[00:06:29] AG: Yes, you're connecting to your Rails server behind our network of load balancers. And it is not serverless per se, especially for something like Rails where you do need to make sure that you have a certain number of things in memory at all times. Rails is not built for a serverless execution model. You can get this thing up and running on a Render subdomain, and then every time you push to GitHub or GitLab, we automatically build your app and deploy the newest version and you can obviously select – you can choose to deploy it manually using some specific URLs and that is it. You will have a simple Render subdomain where you can connect your app from outside Render. And then you'll also have internal

addresses for all your apps that you can connect to. But most people end up adding a custom domain to it and there, we take care of all of SSL and certificate renewal.

We also allow wildcard certificates, again, as a differentiating point versus Heroku. We manage all of it for you. And that's it. I mean, really, for a lot of our users, when we ask them what they like the most about Render? The answer is the thing that I like the most is I never have to think about it, which I think is a great place for us to be.

[00:07:48] KP: Absolutely. I know, DNS configuration is a surprisingly painful process for a lot of developers when they think they've gotten their project done, because you highlight some of the ways in which you ease that final step to getting live.

[00:08:02] AG: Yeah, so we give you just very clear instructions that are essentially for each domain, there's a single step, we give you the exact record to add to your DNS provider. And we have instructions for specific providers, like Name Cheap and toddler on our website, and we will link to them from the dashboard as well. And in the future, I suspect that will make it even easier by maybe creating these records for you by asking you if you're okay with it, by asking you for your Cloudflare or Name Cheap API key. But even now, our users are able to set it up really quickly and get it working. And we issue certificates really quickly as well, so you're able to see your site on your custom domain, within minutes.

[00:08:48] KP: Well, I like the modern CI/CD aspect of this, that as soon as I merge a pull request it's going to build and that's all I guess I need to worry about. Until my site gets super popular and demand is high, how can I help scale up on Render?

[00:09:04] AG: Great question. So, we obviously allow you to create multiple instances of your application manually. But we also have what we call true resource-based auto scaling. And what that involves is you can just tell us what you want your target CPU percentage to be on average across all your instances, and the same thing you can do for your target memory percentage. And they're usually linked, you don't have to use both, you can just use one. You can also choose to use both targets. So, if I say, “I want my Rails instances across the board to be utilized at like 60% on average. I know that my application will perform well, when the average CPU utilization across all my instances is 60%.” That's all you have to do. And we obviously

show you metrics in the dashboard. We also allow you to connect to external metrics providers like Datadog, then we just take care of scaling your app up and down as demand hits.

In fact, we did this back in the day for Pete Buttigieg's campaign back in 2019, when he was running for president as a Democrat. His entire online presence was on Render. And every time there was a debate, we saw a huge spike in the number of people coming in to view his website, and we were able to generate a bunch of instances for them and make sure the site is up or was up throughout, which was really fun. But also, right now, I mean, we've obviously improved our auto scaling functionality since then, and our users find it to be more responsive and more accurate compared to Heroku. The native auto scaling there is built on request response times. But response times are only like a poor approximation of the actual load on your app, and we think that CPU and memory-based auto scaling is actually much nicer. Our customers feel that way as well.

[00:11:03] KP: That's interesting. Are there any use cases you're able to share where maybe someone learned that the hard way, or at least discovered that this was a more appropriate indicator?

[00:11:11] AG: Well, all we know of is the people who were using Heroku and have used Heroku. And so, we have a few former Heroku people at the company, including Heroku's lead developer advocate Chris Castle, who is now at Render and leads developer advocacy here. And one of the biggest issues when he spoke to Heroku users as a Heroku employee, one of the biggest issues they ran into was the auto scaling just did not perform as they expected it to. It either took too long, or it was too expensive. And when you think about the request latency-based model, it's not clear that your latency is going to be solved by creating new instances of your app, because it could be because of a bad network resource that your app is connecting to. It could just be your database that is suddenly responding really slowly. And in that case, creating new instances of your app actually is worse, because it creates the thundering herd problem.

So, this is why looking at actual usage, for CPU and memory, when it comes to auto scaling is much nicer. And you can obviously set an initial number of instances and a max number of instances. So, if you're worried about cost, we allow you to control how much you might pay at a

given time, so you don't end up with a bill that's thousands of dollars more than what you would expect.

[00:12:38] KP: Well, cost and flexibility are definitely going to be interesting features, you're going to see startups, small, medium enterprises need to have, maybe a large enterprise company, not as concerned with cost, although I could have that wrong. What about large enterprise companies makes them attracted to Render?

[00:12:55] AG: Well, large enterprise companies are obviously sensitive to cloud costs, because for them, it's a big part of their overall spend. But more importantly, it's the cost of hiring DevOps engineers and the time involved in hiring DevOps engineers, and especially in this market, where it's incredibly hard to hire any engineers. Render allows larger companies and teams within larger companies to spin up projects and get them into the hands of their customers a lot faster than they would otherwise. They don't need to get DevOps to intervene. They don't need to expand their DevOps team, assuming they have one.

And what we're hoping for, as we grow, we had companies that started out on Render, and we're just two or three people and are now 30%, 40%, 50% companies, and they don't have a single DevOps engineer. We want to continue to build Render to the point where the way DevOps engineers have to manage AWS right now, it just becomes a thing of the past. I think we're well on our way there.

So, for larger companies, it's all about our ability to minimize time to launch, to minimize features, and then we have some other products. Also, for example, we have this whole product around preview environments where for every pull request, we can spin up an entire copy of your production application. And when the pull request is updated, when people push new commits, we continuously update the copy of your production setup. So, this can include all the services you're running in production, plus the database, plus Redis. And people at larger companies who use Redner love this feature because it prevents them from having to rely on DevOps engineers to set up staging environments for each new feature. And then also they get real, shareable URLs which means they can share a product in development with potential sales prospects.

We actually have companies that were sales engineers log into Render, they can create a preview environment from the UI, and then they can go share and have a new environment for just a specific prospect, and show them features in progress. It's also used by QA teams. It's used by product and design teams, engineers and designers wanting to work closer together, obviously, product manager. So, being able to actually see your pull request live is a huge win for not just large companies, but everyone really, but that's where I think large companies find Render particularly helpful.

[00:15:37] KP: Yeah, this is a really interesting feature to me. The first use case I thought of was quality assurance engineering, as you said, like, how great to have an environment per pull request or pull feature, per branch, or whatever it is, we're packaging our code as. Have you seen QA teams change the way they operate given this tool?

[00:15:56] AG: Definitely, yeah. I think that it also really allows you to test everything in isolation. And yet, have a shared staging environment if you want to test everything together. The QA teams can operate on individual pull requests. But usually, we actually see these individual pull request environments as a replacement, often for developers who want to test everything that is related to their change, but they don't want to set up a whole thing on their development machine. And the pull request preview environment also has a way for you to seed your database once the pull request environment gets created. And this means they can work with real data or scrub data as the case may be, as opposed to, local development data. So, all of that is set up in a way that makes feature development a lot easier. And more often than not, we see developers using this as a way to continuously develop a feature on Render.

[00:17:00] KP: Do I need to do any maintenance to clean up those environments over time? I don't know if they're long lived or not. Is that something to keep an eye on?

[00:17:08] AG: You don't. So, as soon as your pull request is closed, either it's merged or it’s closed, we automatically clean those environments up. And then you can always set up a maximum amount of time in days that the pull request will be open. So, let's say you have a stale pull request, you can set the stale timer to seven days and after seven days, will automatically get rid of your existing pull request environment. And basically, all you need to do

to recreate that environment is either close the pull request and reopen it, or push a new commit and then Render will just bring it back up again. But even then, we will always get rid of all these resources automatically so you don't have to worry about any of it after your pull request is done.

[00:17:54] KP: One appealing aspect of Render to me is that I don't necessarily have to worry about all the servers and the DevOps, as you said. Although if I'm using some other service, maybe I'm attached to S3 that's in AWS or something like that. I have my data in place, I might need to worry about network. How much do users know or have to think about these sorts of things when working with Render?

[00:18:16] AG: They generally don't. So, obviously, they can use S3 or any other service. If you're on AWS, for example, and you want to use SageMaker, you can continue to do that with Render. We are also going to soon make it possible for you to run in the same geo region as we create new regions, we're probably going to launch at least five new regions over the next few months. So, you can select where your application lives and make sure that it's close to where other services live. But in general, Render has secrets management built in. So, you can just store your credentials, whether they're API keys, or JSON keys or PEM keys in a JSON format or in any format, really. You can store them in render, and we store them encrypted, and a per user encryption key, and you can connect to really any service you want.

[00:19:19] KP: Well, we talked about the path for deploying the Rails app and I see there's other environments as well. You'd mentioned Django, and things like that. So, I have a pretty good picture about how I would set up my application stack. But often, there are external things to that. Maybe I need some reporting jobs to run every midnight or I need some sort of background crawling task to manage. How does Render satisfy those needs?

[00:19:42] AG: Good question, because we satisfy those needs to a tee. We have cron jobs where you can just set up – again, , you connect your GitHub repo and then all you need to do is tell us the command you want to run and the frequency you want to run it at. We'll make sure your application gets built every time you push to Git and then we make sure that your code runs on your schedule. And it's always the version of the code that you want. So, if you're upset about our deploys, will always run the latest version of the code and you just don't have to worry

about it. We also make sure that you don't pay for the time you're not running. So, we build cron jobs by the second. A lot of our users find this to be super helpful.

We also have the ability for you to run one off jobs through our API. So, let's say you have some data munging jobs that need to be run in a specific environment, or you have the framework that requires just ad hoc jobs that need to be run, that's when our jobs on Render come in. And again, they're also prorated by the second. And you have a lot of different plans that you can select from in terms of CPU and memory capacity for those jobs. That covers most of our user's needs, and I can't think of a specific situation where what we have hasn't been able to cover the needs to run things like jobs and crawls.

[00:21:07] KP: Could you describe the use case for an app where I've patched it up in my own Docker image, and I really just want to keep it inside there.

[00:21:15] AG: Yeah, so if you have a Docker file, we automatically build it for you, and we make sure that every time you update a Docker file, or update even your source code, we rebuild your Docker image, and then we deploy it just as a standard container. So essentially, you don't need to do anything beyond create a Docker file. If it works locally, that's the beauty of Docker. Obviously, if it works locally, and you've tested all the dependencies, then it will work on Render as is because containers are supported as a first class concept on render.

[00:21:52] KP: If I contrast that with something like ECS, or Elastic Cloud Service on Amazon, there's a lot more steps. I've got to establish, I think it's an ECR repository and be pushing images and tagging images and all these sorts of things. I have to confess it feels a bit tedious to me, when I've done that sort of thing.

[00:22:11] AG: You're not alone.

[00:22:11] KPO: Am I on appreciating something? Is there a nuance that a security professional would tell me I need to follow? Or is it just complicated for complicated sake?

[00:22:19] AG: It's because the way AWS thinks about its products and its customers is very different from how we think about it. We focus on developers who typically are application

developers. We don't build Render for DevOps professionals who know how to run things on AWS. If you do, and you already have things running on AWS, and you're happy with where you're at, keep using it. We don't want you to move everything over to Render. But party because it's risky, right? Why change something that isn't broken. But if you don't care about that sort of thing, and most companies don't, and shouldn't, because it's not helping their core business in any way. They're just reinventing the wheel, the same kind of thing that's being reinvented across every company and that's really why I started the company.

So, our goal is to try to make it as effortless as possible and as automated as possible, but still give you escape hatches, if you want to do something custom. Just like we have these native environments, where you can just bring your code, you don't even have to understand Docker. So, you can use Ruby, Python, Node, Elixir, Java, Go, and Rust. You don't have to write any of your services using Docker. But at the same time, if you need, you know, a very special library that only your application needs, and it's not available in our native environments, then by all means, go use Docker. We want to make that as easy as possible, which is where we just asked for a Docker file, because that's the thing that you write anyway.

[00:23:58] KP: I noticed there's a really wide range of Quick Starts on the sites, even a couple of frameworks or languages I'm going to have to google after this, because I don't recognize all of them. Could you talk a little bit about the set of options there and how you guys maybe pick from the great list options I see?

[00:24:15] AG: It's really all about helping the Quick Starts we have. It's about helping the developers we think in our target market. And especially for something like Render, which compared to Heroku, which has been around for 16 years, we've been around for four. When we see or at least in the past, when we were even newer like we were one or two years old, the only people who use Render at the time were early adopters. And early adopters want new technology so this is why we ended up incorporating Rust because there is a big intersection between people who will try and let new language like Rust or Go, and these aren't really new languages anymore, but they were at certain point, and people will try a new hosting platform like Render.

So, that definitely helped us attract people from these communities. And then over time, and obviously we had Node and Python from the start, and we continue to think about other frameworks and other languages and we keep an eye on things. For example, Elixir is another place where we see a lot of people moving over from Heroku to Render, because Heroku restarts your app every 24 hours, and Elixir apps, especially Phoenix web apps, in Elixir, they require keeping things in memory, keeping state in memory. So, if you don't want to reload these things every time, then it's much better to use Render, and Heroku has some other limitations around Elixir.

So, that's kind of how we think about some of these things. And a lot of it is driven by user feedback, really. We have an open public feature request board, and we also list all the things we're working on right now and things that we have planned. So, if you go to feedback.render.com, you can see that list, and it's a great way for our community to give us feedback on what they would like to see next. It's a long list. It never stops growing, because people want us to build everything that AWS has. But it's great to see because there's a lot of user engagement, and people love what they see already, and they want more of it.

[00:26:34] KP: Have there been any particular celebrated releases, I don't know, a database or something like that, that's brought a lot of interest?

[00:26:42] AG: Yeah, so the first time we might have been the first cloud provider to do this, but we supported Postgres 14 on launch day. And going forward, I think we're going to support all new releases on launch day as well. But there are other things like launching a new region. So, we have a Frankfurt region. But there are a lot of people in the APAC part of the world where they want to region and Tokyo or Singapore or Sydney, and they've been asking for it for a while. I just want if you're listening, and you are one of those people, please know that we're launching those things soon. By soon, I mean, a matter of months, as opposed to a year from now.

[00:27:24] KP: Does that mean you're building out a physical data center or just setting up the right appropriate virtual but local instances?

[00:27:31] AG: We're not building our data centers. It's too early for us to do that. And we really want to focus on giving people access to the functionality that they need, and even functionality that they might not think they need, but they do. So, one example of this is every company kind of wants to have some notion of Canary deploys. It's something Render can offer relatively easily given how we built out our infrastructure. So, obviously, we are a small team. We're growing quickly. But our bandwidth is limited, and we'd rather spend that bandwidth on giving people more ways to run their apps and more features in the platform itself, or more integrations with other things they're using, compared to trying to optimize our costs by running on data centers, because I use us don't really care about that. They just want their apps to be up and running and scalable and secure at all times.

[00:28:29] KP: I noticed from reading some things on the website, there's a lot of examples in how you can get much better pricing on Render compared to a site like Heroku, even as much as 50% reduction, which actually seems like a lot. That's not just, oh, it's Salesforce, that can gouge everybody. That's a big margin. Is there a technology advancement that you have that they don't have that makes that possible?

[00:28:53] AG: So definitely, I think there is something to being able to pack machines more efficiently that I think Roku, at least until recently, perhaps was not doing or they were but maybe not as efficiently and they have a lot of historical baggage. And I think that they also, from what I've heard, again, given the Heroku employees who migrated over to us or who have joined Render from what I'm hearing, I think there's just isn't a lot of motivation to keep Heroku competitive or to add new features. Everyone in the industry sort of knows that Heroku is stagnant and kind of on a slow path down. That's where I think it's not just about pricing for render. It's also about continuing to add new features and giving you the best of the advances that we're seeing in infrastructure and exposing them to you in a way that doesn't require you to have a PhD and managing AWS.

[00:29:59] KP: Are there any of those features tours on the roadmap you're able to talk about?

[00:30:03] AG: Yeah, so like I said, our roadmap is actually public, but new regions are coming. We just launched a free tier, which is something that was perhaps the biggest thing that was missing on Render, compared to all of the cloud providers. I realized that render has not had a

free tier, while every other type of provider has had one of some sort, for the last three years, and so we just launched it, and it's been really well received by our users.

But in terms of new features, that we're integrating, I think server less support is another big one, where we are running workloads kind of already in a serverless manner for our free tier, and we expect to roll that functionality out to more applications where you can sort of scale to zero, or you can run a lot of threads in parallel, or where each request spins up a new small micro container or micro VM, and that's how you're charged. That's something that I just don't see Heroku doing anytime soon, given that they haven't even like focused on improving their existing features. But Render is very seriously, thinking about how we can do it like Functions as a Service offering that is a lot easier to use than something like lambda.

[00:31:22] KP: Yeah, that makes sense and fits, I think, in line with what I see when I look through the features and services of Render. It's really strong set of offerings for an application developer, and adding Functions as a Service fits right in with that. I'm curious if you have ambitions beyond that, surely there's a big enough market to service application developers. But there's also computer vision stuff coming online. Where do you see Render in 5 to 10 years?

[00:31:47] AG: Everywhere. It's a short answer. I think it really depends on how quickly we can grow the team, the business. And there has to be, if you play video games, especially real-time strategy games, you have to do things in a certain order in order to get to where you want, and that's really what we're doing right now, where we're focusing on this application developer market, because it has A, it's already huge, and it is underserved. I know that, similarly, there are data analyst markets that are underserved. There are machine learning markets that don't have to do with data analysis. And obviously, we're all going to live in the metaverse at some point.

So, all of those things, I think we're keeping an eye on but we're not really focused on them and we're really focused on improving the current feature set and improving observability day to operations, and adding things like compliance, sock to compliance, which a lot of our customers, as they get bigger have asked for, and will be especially helpful for larger teams want to use Render to run everything. So, these are all things that we think about daily.

Now, obviously, in 5 to 10 years, it could look very different. In fact, a friend of mine recently asked me if Render is ever going to offer Windows instances powered by GPUs, because he does not have a Windows PC that can play a specific game he wants to play. It's not completely out of our wheelhouse. But it's not something that I ever think about doing anytime soon.

[00:33:28] KP: Right. Makes sense. Yeah. Well, when you think about maybe a new startup coming online. Do you envision them adopting Render as the only platform they need, or their application layer, and they'll sample from other services that integrate?

[00:33:43] AG: It really depends on their needs. We find that most new projects can use just Render and not have to worry about it. But one of the things that, like I was saying, people want us to build a replacement for S3, and this is all our users who are currently using Render and are probably using S3 as well. But they know that if we do it, it's going to be definitely a better UX than what they get on AWS, and they'll often don't – a lot of people just never want to log into AWS, because it's so insanely complicated. And by the way, I've been programming for a long time and I've dealt with things all the way from bare metal to the highest reaches of the stack. I don't want to log into AWS and this is not my render bias talking. I just know that it's not a very pleasurable experience.

So, I think that we're going to build some of these other key things that people need in like 80% of applications out there. So, you don't have to worry about creating another account. You don't have to worry about, dealing with cloud providers or third-party services. You don't want to do it. But at the same time, I also acknowledge that we're not going to build out the best hosted MongoDB platform, and we're not going to build out the best hosted Elastic Search platform.

Our goal with those things is to give you an option to run those things on Render, with containers, with private networking, with storage, and we have templates that you can use to set up those things with a single click on Render. But if you really want to get serious about your Mongo, or your Elastic Search, or you know, even things like Planet Scale, which are newer, and are proving to be really popular, we're going to make it easy for you to integrate with those providers and with those data platforms by working directly with these companies. Just like these companies have ways to spin up their product on a specific cloud provider and specific

region, but that only applies to Google or AWS, or Azure right now, our goal is to make render a fourth option when you go to their website.

[00:35:55] KP: Earlier, you'd mentioned that you'd launched I think it was Postgres 14 on its actual release date. And from a, I guess, an engineering developer user's point of view, oh, great, that's a wonderful feature, I can have the upgrade as soon as it's available. And it sounds really easy once you've done it. But if it was easy, everyone would be doing it and they're not. I'm curious if there are any noteworthy engineering challenges that you're fighting under the surface or if solved under the surface, so that users don't have to worry about it, but still may be worth a mention.

[00:36:25] AG: Everything is that really. I think there is a lot of complexity under the hood, especially around dealing with Kubernetes, which powers at least a big part of our underlying infrastructure. I wouldn't wish it on anyone. And we found so many bugs in Kubernetes, or even the manage versions, especially the managed versions of Kubernetes. Less so in pure Kubernetes itself. So, I know for a fact that GKE has had a lot of issues, let's just say that don't get reported online that we run into and that I'm sure a lot of other users also run into. But I think that's where working around these challenges, and then also just managing large, really large clusters of applications, multitenancy. Obviously, a single user doesn't have to worry about that. But it's a really big challenge to make sure that that is happening.

The other big thing is DDoS attacks that might bring your site down. Again, by the time this airs, I think we would have enhanced it. But we currently give everyone running a website on vendor free DDoS protection through Cloudflare and it's just built into the platform. We haven't announced it as of today, but we will very soon. It's one of those things that you never think about until you are attacked. It isn't a straight up like we're not just throwing Cloudflare in front of your app and calling it a day. It's actually a fairly involved integration that took several months to build, and we work really closely with the Cloudflare team and actually found improvements that would have helped other people using Cloudflare.

So, it's it's a constant struggle and I'm really glad that we are in this space, because these are some really interesting problems to solve for any engineer working at Render. And this is also a pitch for pitch for me, for people who are interested in this sort of thing, or just building great

tools for developers to come join us. We're hiring, we're growing really quickly, both in terms of customers and revenue, and we'd love to grow really quickly in terms of people, given our most recent funding.

[00:38:46] KP: Oh, yeah, let's do the pitch, since there's a lot of software engineers listening. Tell me about the funding and what you hope to do with it.

[00:38:53] AG: Yeah, we raised $20 million recently, and the round was led by a relatively new venture firm called Addition. But the person behind the Addition, Lead Pixel is definitely not new to venture capital and investing. He is or has been on the boards of companies like Peloton and Warby Parker, and Spotify. He's also a big investor in Stripe where I was the eighth employee. And we've been really lucky to be able to work with them. But this money really helps us grow the team much faster, which in turn helps us deliver new features and new products to our customers faster.

[00:39:35] KP: So, it's a team investment then. Tell me a little bit about some of the roles you're looking to fill for.

[00:39:40] AG: Well, we're always hiring in engineering. I think every company is sort of, every software company at least is doing that. But I think we're looking to hire at varying levels, but generally sort of senior engineers who are very comfortable with the ins and outs of infrastructure and who we're building out an SRE team, we're building out a security team, we'd love to build out a team of experienced Postgres DBAs. We are also hiring people for more full stack role, where they build application, well, basically features that users look see and feel across the front end and the back end, although we have a very talented team of UX engineers. But everything on the engineering side, but then we're also looking for people who can help us manage the business itself as it grows. And that involves roles in operations, and in design and product marketing, and product management.

So, if you are interested in what you just heard, but you don't see a role on our website, please send me an email. I will be sure to get back to you my first name @render.com. That's unranked anurag@render.com. So, even if you don't see a role that jumps out at you as a perfect fit, we've definitely hired people, because they're just really good at what they do and they come

and help out. In a growing company, especially when that's going as quickly as we are, there's always a lot of stuff to do that you need great people for.

[00:41:26] KP: And are you remote on site or hybrid?

[00:41:28] AG: We are hiring everywhere in North America. We have a full-time employee in Canada, who's going to start soon. But also, we have a full-time employee in London, and especially as we launched newer regions in Asia Pacific, I think we'll want people there. But we also have an office in San Francisco that at least for the people who want to go into an office and are in the Bay Area, it serves as sort of the home base.

[00:41:59] KP: Well, I've seen products like render definitely pushing the meaning of what it is to develop software, making it easier and easier. Or at least I don't have to deal with the chore parts of it, I can focus on my core application development. I assume we're not going to hit any sort of ceiling on that. With that in mind, I'm curious if you have a vision for what software developers are doing in the future and how tools like Render will affect that job.

[00:42:24] AG: Yeah, so I see Render as sort of the next level of abstraction for other people to build no code tools, or for other people to build platforms like Shopify or Squarespace. Shopify ended up spending a lot of time dealing with AWS when or other providers when they started, because there was no other option available to them. I think that when you make something at the level of abstraction Render operates at right now, when you make it generally accessible and easy to use and affordable, then you just see entirely new kinds of things being built by people who would otherwise just not even build them. We have some really interesting potential pieces that folks will build with Render because our API will be public.

Again, by the time you listen to this, it will be public. And we're really looking forward to what people will build. It's hard to say. It's very similar to sort of what we saw at Stripe, what I saw at Stripe being really early there. I had the company launched back in 2011, and was there to early 2016. We saw people build all kinds of applications and accept payments in those apps only because Stripe made it incredibly easy. I already see that happening with Render and I expect that to continue.

[00:43:52] KP: Make sense. Well, to windup, maybe we could stick with that. Are there any interesting lessons you learned at Stripe that you're able to put in execution at Render?

[00:44:01] AG: Oh, every day, I come across things that at Stripe, we did really well, but also maybe not so well. I apply those lessons almost on a daily basis. But one of the things that continues to be a strength of Stripe is just the focus on product quality, and on having really high standards when it comes to hiring, but also when it comes to the product that is eventually delivered to our users. I think that's perhaps, and having those high standards with a very strong focus on developers, and I think that sort of has become part of my DNA, and is also becoming part of Render's DNA as a result where we want to build the best products in the industry for a developer audience that is typically underserved by existing offerings, and we want to let them go wild with these new tools.

[00:45:00] KP: Well, my next greenfield project is going to be a Render project so I can get my feet wet and learn all about it. Anurag, thank you so much for taking the time to come on Software Engineering Daily and share your experiences.

[00:45:03] AG: Thank you for having me. It was a pleasure.

[END]