EPISODE 381

[INTRODUCTION]

[0:00:00.2] JM: Building a startup requires constant evaluation of trade-offs. At the earliest stage, the founders evaluate different ideas. Once and ideas settled on, the company develop strategies for finding early customers and growing. As the company develops traction, the operators consider ways to scale further or partner with an acquirer.

Joseph Jacks and Greg Koberger are to founders who have both been on the show previously on separate episodes. Joseph started Kismatic which was the earliest company completely focused on Kubernetes. Kismatic was acquired by Apprenda Greg Koberger runs ReadMe.io which is a company that provides beautiful documentation as a service. He was on in one of the earliest episodes of Software Engineering Daily in a really good episode that we did. This is one of the good earlier episode and I credit Greg to just being a great guest and a great conversationalist. I recommend people check back both of those episodes, the episodes that I've done with Joseph and the episode I've done with Greg.

In this episode, Greg and Joe share their thoughts on running and scaling startups. We discussed engineering concerns, scaling strategies and discussions of what to build and what to buy.

[SPONSOR MESSAGE]

[0:01:31.1] JM: Hosting this podcast is my full-time job, but I love to build software. I'm constantly writing down ideas for products; the user experience designs, the software architecture, and even the pricing models. Of course, someone needs to write the actual code for these products that I think about. For building and scaling my software products I use Toptal.

Toptal is the best place to find reasonably priced, extremely talented software engineers to build your projects from scratch or to skill your workforce. Get started today and receive a free pair of Apple AirPods after your first 20 hours of work by signing up at toptal.com/sedaily. There is none of the frustration of interviewing freelancers who aren't suitable for the job. 90% of Toptal clients hire the first expert that they're introduced to. The average time to getting matched with the top developer is less than 24 hours, so you can get your project started quickly. Go to toptal.com/sedaily to start working with an engineer who can build your project or who can add to the workforce of the company that you work at.

I've used Toptal to build three separate engineering projects and I genuinely love the product. Also, as a special offer to Software Engineering Daily listeners, Toptal will be sending out a pair of Apple AirPods to everyone who signs up through our link and engages in a minimum of 20 work hours. To check it out and start putting your ideas into action, go to toptal.com/sedaily.

If you're an engineer looking for freelance work, I also recommend Toptal. All of the developers I've worked with have been very happy with the platform. Thanks to Toptal for being a sponsor of Software Engineering Daily.

[INTERVIEW]

[0:03:33.4] JM: Greg Koberger is the CEO of ReadMe.io and Joseph Jacks was a founder of Kismatic. He's and expert in the Kubernetes and cloud native space, and today we're going to talk about a variety of topics. It's a topic Roundtable episode which we do occasionally on Software Engineering Daily. Guys, welcome to the show.

[0:03:55.5] GK: Thank you for having us.

[0:03:56.9] JJ: Thanks, Jeff.

[0:03:57.3] JM: Greg, you started ReadMe three years ago. How has your perspective on running a company changed over that period of time? Maybe you could talk a little bit about what ReadMe is for people who haven't heard that episode.

[0:04:10.8] GK: Sure, ReadMe is we do documentation for APIs, code libraries, other technical stuff. We kind of call them developer hubs. If you're familiar with developer.anything.com, I'm sure if you're a developer you've seen some sort of like developer hub. We do that. We do documentation. We do reference guides. We actually play around with the API in line. We have suggested edits so people can help you out with your docs. We have support forms. Pretty much everything. That's what ReadMe does.

As far as how running a company, I've changed. It's all kind of sad stuff in the sense that I think that I got into it very idealistic and I still am of course, but I've kind of – I wouldn't say – There's a lot of things over the past three years that I've had kind of lose my stance on or changed my opinion on. It's always hard to decide what's something you can change our opinion on and what's not.

What I mean by that is I like product stuff. Product is kind of my passion as supposed to, for examples, sales. I think everyone when they start a company they think it's going to be amazing enough that they will not have to do anything, it's kind of like a build it and they will come type thing. At least when you really care about the product. You think you don't need marketing, because word of mouth is going to be amazing. You don't need sales because people are going to see the value instantly. You don't need to do anything. You don't need to build features. All your features can be on the cheapest plan because everyone's going to be using it and seven billion people times a small amount of money, you're going to be a trillionaire pretty quickly.

I don't want to imply that I've given up on my morals and stuff like that, because I haven't. It's just little things like try and having to build features that are only in the thousand dollar a month plan, not because you – Me, three years ago, that cripples the product, it hurts the users. I still don't not think that, but I've had to think a lot more about making money because when I started the company I thought that making money felt greedy. Now I feel it's more of a way to keep the company going, keep people – It allows me to hire more people which allows us to do cooler stuff, things like that.

I think there's a lot of differences that have changed. I think that's kind of the biggest thing is I shift from really caring for the product, which I still do, to caring about the product but having to care about making money as a way to keep the product going and make it bigger and better.

[0:06:32.8] JM: I think a lot of people have that transformation. I know even with Software Engineering Daily, initially I was like, "Oh! This will be so easy. People will be calling in for the podcast ads as soon as I start selling them," but you learn you've got to do some outbound sales. You learn, “Oh! This is why there are so many books written about sales. This is why sales is a thing.” It’s not like, “Oh, we’re not in real estate and we’re not in finance. We don’t have to do sales.” You have to learn sales in most parts of software, like 98% of software business is you have to figure out how to do sales.

[0:07:15.2] GK: Yup, even the best companies still need – Whether you call it sales or you call it customer success or customer support, or you don’t have sales, but just think about making money. I’m yet to meet a company that hasn’t had to kind of really throw their way behind actually thinking about making money, caring about making money because it’s a company at the end of the day.

[0:07:34.4] JM: Joe, when you started Kismatic, that was the first company around Kubernetes. You eventually sold it. Did you learn anything about sales during that process when you build Kismatic?

[0:07:46.8] JJ: Yeah. Full transparency, that’s kind of my core career background is sales and sales engineering. I’m like a self-taught programmer and computer science sort of person. I didn’t go through the traditional routes that I guess technical people go through. I don’t know if I had any – Through the Kismatic experience, I don’t know if I had any real sort of transformative insights or new learnings around just sales in general, but I would say that my – Just like the broader question of the sort of startup dynamics and things to focus on. Particularly, the kind of technology that I’m interested in which is open source software.

There’s so many differences of opinion about the sales energy required to go and generate a necessary output to sustain a business, i.e., revenue, or some sustainable source of momentum to keep a company at a growth rate where you can continue to raise investment or you can continue to fund salaries or pay contractors or what have you. In that regard, I think the opinions I’ve had and the observations I’ve made have changed quite a bit over the last three or four years.

I guess, in particular, lots of open source startups tend to focus a lot on just initially just building the project, investing a lot of energy around that and then focusing on sort of users and engaging with people who are actually deploying the software and running in different environments as a sort of afterthought or sort of second order thing maybe after the project has reached a certain stage. I think that's often times a mistake. Even more so in the really early days of doing a company around an open source project or projects which I think increasingly is sort of the primary way to build software companies of any kind. That's also a broader observation that I think has caused me to change my beliefs in some ways.

Sales I think manifests in much different ways and it means more than just convincing investors to give you money or selling a potential customer on a product that they'll in return pay for in one lump sum or a recurring fee or some kind. It think sales is super multidimensional. When you're a founder of a company, that's essentially all you're doing all the time. It's just selling. Regardless if you're a CTO or a CEO. I really deeply belief that. A CEO is obviously just constantly selling from a recruiting and vision and strategy perspective.

A CTO I think is constantly selling on the architectural dimensions, like the convictions around what the product roadmap should be, getting the engineering team excited about the specific feature on the roadmap that's getting implemented in the next week or so or two weeks or month. It's constantly selling around the sort of broader macro technology trends that he or she is seeing as CTO of that company and constantly needing to keep both the internal technical people in the company excited about that as well as the industry and the analysts and everybody else.

I think sales is pretty crucial regardless of any kind of company maybe almost in all cases, and I think there's some exceptions, like there're companies like [inaudible 0:11:25.9] that just figured out how to build premium or enterprise versions of products that don't require a ton of sales energy or sales investment upfront. I think that's ultimately the exception, and then later on they end up having to hire lots of sales people as they reach a certain amount of scale and stuff. Yeah, that's some general thoughts on the sales part of it. It's a pretty interesting topic.

[0:11:53.1] JM: Yeah. You mentioned kind of the default to open source in this cloud native space and maybe you figure out the business model later, or if you're wise, you figure out the business model upfront. In the cloud native space, it's like all these companies built around Kubernetes and there are companies that are selling something around container networking or service proxying or all of these little niche things in the Kubernetes space. Are these viable business? These companies that build around an open source and then they sell? I guess, typically, they're selling either like an enterprise edition or a nice UI, because you have to build some sort of open source project in order to gain traction in your vertical within the cloud native space, and then you have to have some sort of up-sell, whether it's consulting or – I mean the Cloudera model is typically like consulting plus enterprise edition. What are the viable business models in that cloud native, highly vertical open source technology space?

[0:13:11.2] JJ: That's a really big question. I have a lot of personal opinions on this. I think that might be why you're asking the question. There's far less pervasive knowledge, like distributed on the internet or at least from people's varying experiences around which business models or monetization models in particular might work as it relates to open source projects in company building.

I think there's a few that work well and some that work better than others in different cases. Just the umbrella of commercial open source sometimes means companies that just sell support and services and training around an open source project. I think there's a ceiling to the kind of businesses you can build with that. I would put a company like Mirantis in that case where open stack started to get lots of traction as an open source project and Mirantis had a fairly large data center consulting, integration and sort of services management business. They basically decided to double down an open stack and it put a hundred or so people on open stack development and implementation and integration projects with enterprises and they gradually grew that business. I think they've been able to grow a fairly large company with some transitions and courts corrections along the way.

I think there's other proof points of just companies selling support services or subscriptions based on 100% open source offerings. I think Hortonworks is another example of a company that's basically done that where they've taken a collection of the Hadoop ecosystem projects and also built some of their own. I think Hortonworks was the company that came out with Yarn along with a couple of other companies which is a sort of distributed scheduling execution runtime for MapReduce andSpark and all these other data processing tasks in Hadoop environments.

They've built kind of a distribution and packaged that but they don't have any proprietary software. They're like 100% open source, yet they were still able to go public and grow their business pretty meaningfully. I think that's one model.

Another model is there's this sort of – You hinted at it, this combination of proprietary and open source code in the product development sort of strategy and those kinds of companies are commonly known as open core. Open core means there's the vast majority of the code or the software that company is shipping is open source, but there's a small veneer or wrapper or proprietary codes sort of layered on top of that open core sort of nucleus or foundation of software the company is sort of built on. That's actually what they go and sell for commercial licenses.

That model I think is a little bit more widely implemented by commercial open source companies because I think there's just a greater propensity to capture value in that model mostly because you have these sort of enterprise edition, sort of community edition dynamics and this is something that Docker did recently where there's community edition, open source, top to bottom Apache licensed, GPL licensed. Then the enterprise edition is basically some combination of proprietary and open source and those companies will sell a license around that. I think that's the second fairly common model.

There's a third model which I think is pretty interesting and not as widely discussed, and that's basically open source distributed through a SAS distribution model where – I'll take an example of a company like Fastly. They're fundamentally on some open source libraries and projects that you could basically take coupled together and build your own sort of local version of what Fastly is offering on your own infrastructure, but they basically handle all that for you and they offer it up through an API and they give you the DSL and sort of interface and API consistency that you would see from the open source varnish and HTO and these other systems that they're built on top of, but they basically service their customers 100% through an API and a SAS platform, which is kind of interesting.

I would also consider those companies to open source, but it's like a totally different model. It's basically 100% proprietary essentially and you're paying for like access to an API and API keys and that's a third model. There's a fourth one that I haven't really fully internalized yet. I think it's kind of interesting, but it's way out there. It's like blockchain stuff.

[SPONSOR MESSAGE]

[0:17:59.9] JM: Simplify continuous delivery GoCD, the on-premise open-source continuous delivery tool by ThoughtWorks. With GoCD, you can easily model complex deployment workflows using pipelines and you can visualize them end-to-end with its value stream map. You get complete visibility into and control of your company's deployments.

At gocd.io/sedaily, you can find out how to bring continuous delivery to your teams. Say goodbye to deployment panic and hello to consistent, predictable deliveries. Visit gocd.io/sedaily to learn more about GoCD. Commercial support and enterprise add-ons, including disaster recovery, are available.

Thank you to GoCD and thank you to ThoughtWorks. I'm a huge fan of ThoughtWorks and their products including GoCD, and we're fans of continuous delivery. Check out gocd.io/sedaily.

[INTERVIEW CONTINUED]

[0:19:00.9] JM: Greg, does this stuff impact you, because I think ReadMe is a really interesting business because it almost seems like you're mostly focused on user experience and frontend stuff, because you're building developer documentation which is not necessarily high bandwidth for any individual instance, but the user experience, the design is really emphasized. Also, it's a developer product, so you probably see how – At least whether or not you see the first order effects of Kubernetes, like companies migrating their infrastructure to Kubernetes.

You certainly see the second order effects. It's becoming much cheaper for companies to run, to scale infrastructure. It's becoming much more – It's a much more orderly process for companies who want to scale on this containerized architecture. One of the second order effects is you see companies building more APIs and opening up their API because it's easier to build services and externalize those services. These changes that JJ just described, most of these are essentially backend changes. How do you view these changes from an engineering standpoint at ReadMe?

[0:20:26.2] GK: Yeah. In APIs, Kubernetes definitely, but also Lambda is something that shown up in the past year or two and gotten really popular. That is more specific to API. For people who don't know, Lambda is kind of new. It's been around for a while but this new concept from Amazon, some other companies. Google has what are called endpoints. That's basically just a way to kind of a run a function in the cloud and stateless function that runs and returns something and it's basically kind of like an API. It's perfect for APIs amongst other things.

Some other things are internal micro-services started to get big. There's a lot of technology trends that, yeah, like you said, don't affect us directly because we don't really care if you're at Kubernetes or if you're not in Kubernetes. It's an API, so it doesn't really matter.

As far as how it affects us and how it matters to us, I've always thought the most interesting to me, the most interesting part of kind of the heavy tech is taking it and making it palpable – Whether it'd be the average developer or even developers who aren't through CS majors, who have been doing this for 20 years. I think there's going to be a huge shift in what we see as someone who programs. There's always going to be computer science people. I have a CS degree. I do enjoy the heavier tech stuff, but I think that we, people like that are going to be quickly dwarfed. Even the people who listen to this podcast are going to be quickly dwarfed by the people who kind of are medaling around with JavaScript on a weekend or something like that.

I think there's a huge desire for people to make stuff, not developers, just people in general and code is becoming easier and easier and people are becoming more technically literate because they grew up using Facebook. They grew up with a MySpace page where they can mess around CSS, stuff like that. I think it's going to be a lot harder to find people who don't haven't written a few lines of JavaScript or written – Okay, we'll use JavaScript, I guess. It's going to be the main language for a while for better or for worse. That's kind of where we see our position kind of in between that. Kind of bridging the gap between these people, these many, many, many people who just want to write a little bit of code to make a personal website or simplify their workflow at work every day. Maybe they're VC and they wanted to build something that kind of helps them qualify leads a little bit better, something like that, by pulling in two or three APIs.

That's kind of where we see our sweet spot. It doesn't really answer your question, but I think that the technology has grown so much both on the backend, on the consumer side, just pretty much everywhere, that that's where we see our sweet spot is being right between kind of the really hard stuff and let some people work on the really hard stuff and let some people just consume it in an easy way. We see it as our job to kind of make it easy for those non-technical people to build stuff.

[0:23:20.0] JM: When you say people like us will get dwarfed, is that because we're too in the weeds, like we can't see the forest for the trees. We can't see how high level the APIs have gotten because we're just so steeped in the technology?

[0:23:37.2] GK: That's not what I meant at all. I do think that's true. We can talk about that separately. No, I don't mean dwarfed in anything other than. It's not like computer science people are going to be worse or make less money or be less valuable. I just kind of see a more of like –

[0:23:51.3] JM: Dwarfed in volume.

[0:23:52.9] GK: Precisely. Yeah. It's not because there's going to be fewer people going through CS. It's going to be way more. I just mean that percentage is going to be dwarfed, because it's like math, for example. Percentage-wise, everyone uses math. Not many people do math for a living or have mastered in it. Just because math has become so ubiquitous to everyone's job that you need it for no matter what you're doing. That's kind of what I meant.

[0:24:21.3] JM: Do you see the ReadMe business changing as those integrations get easier and the APIs become more robust and there's more APIs, because I can imagine – There's been some different attempts at maybe an API market place, things like that. Do you see that as a potential point of expansion?

[0:24:47.1] GK: Yeah. I think that the API space is littered with a bunch of things like marketplaces and stuff like that that just didn't catch on for whatever reason.

[0:24:57.4] JM: Why hasn't that happened? Why hasn't there been the marketplace for APIs that magically stitched together easily with drag and drop tools?

[0:25:08.1] GK: It sounds great, doesn't it? I think the reason why it hasn't caught on is because there's really – Most APIs tend to be for a company, meaning that Lyft or Uber will have an API and it's not an API you can to use out by itself. It's an API that is very tied to their brand. If you look at any listing of the most popular APIs, you've heard of the company and they're not an API company with the exception of Stripe and stuff like that.

I think that the hard part is that whether these companies don't want their APIs on a marketplace because it loots their value. They'd rather be very core to their business. They'd rather have it look like their site, be like their site, control the entire user experience. Most of these marketplaces are just apps and APIs that don't have a brand behind them and they're just one-off APIs. It comes off as as a bunch of like useless APIs, I think.

There are some good ones, like currency converters and IP lookups and stuff like that, but I think for the most part there hasn't really been a great market – There hasn't been a great – I think you have a hard time coming up with 10 really great APIs you could think of that aren't tied to a brand or aren't tied to a company. That's my opinion at least.

[0:26:21.7] JM: Yes. You have this API conference recently. You put on a conference about APIs, or just a developer conference. What were the goals in putting on that conference and what were the major takeaways from your experience at it?

[0:26:36.5] GK: Sure. My goal – There are obviously a lot of goals. Back to the whole selling my soul to make money type thing. Of course, some of the goals where it's like meet customers to introduce new people to ReadMe and stuff like that, but that wasn't the real goal. The real goal for me was I think basically what we just talked about. I think that when I go to API conferences, I hear a lot of stuff about Swagger, the open API initiative. I hear a lot about Kubernetes. I hear a lot about a lot of things, and I think that to a certain extent a lot of people who are building APIs lose sight of one thing, which is that the only point if you're building API is to get to people to use it. Maybe it's an internal micro-service. You still want people to use it.

I think that I wanted to hear more talks that were more focused on the end user as supposed to the development experience, not that I don't think that there's a gigantic place for development practices or for using different technologies or talking about what technologies to use. At API conferences, there is a lot of talk about things like a hypermedia APIs and stuff like that which are interesting concepts if you sat down and talked about them. Not to waste too much time talking about it, but hypermedia APIs are basically self-describing APIs that when you get a request, you know what to do next and it knows all about a lot of – It tells you information and self-documents.

The concept actually sounds cool, but in practice, they're impossible to figure out and to use and everything. The point of the conference was, basically, I wanted the message to be APIs are going to be so stupidly powerful, but only if we make them simple for the average person to use. When I say the average person, I mean not people — I can figure out any API given enough time, because I have CS degree. I've been programming for a longtime. I'll figure it out.

Not everyone has that ability. I think there's a lot of things that we take for granted with APIs that regular people, people who aren't programmers but still want to make something, we lose them at that point. Anything from just like headers and query strings and the difference between them and the difference between a put and a post and URL and coding things properly, and JDBT tokens. There's a gigantic list of things that aren't actually complicated to me, because I've been hearing about them for years, for example, but someone new to an API is never ever going to be able to figure it out.

That's kind of the message of the conference. There's a lot of really cool stuff out there, like GraphicQL I think is making a little bit easier to understand how to use APIs, stuff like that, but that was the message, was I wanted people to think about that.

We have three sections, so most API conferences talk a lot about the middle of our three sections. The first for us was designing your API, the middle is building, and the later was marketing. I was really happy with the beginning and the end. I was happy with all of the talks, but I was happy at the beginning and the end talks because all the feedback we got was, “Wow! We’ve never seen a talk like that in an API conference.” They weren’t very technical at all in the sense that we didn’t talk about programming, but they are very in-depth and technical about how to market your API or how to sell your API or how to price your API. Things like that that don’t always come up at API conferences.

[0:29:47.7] JM: There are so many conferences these days. Do you guys think that we’re at peak conference?

[0:29:55.6] GK: Oh, God! Yeah. There’s way too many conferences, and we probably should not have had a conference.

[0:30:00.3] JJ: Although, there are useful conferences. I actually don’t go to that many conferences. I started the Kubernetes Conference and I go to all of those. There’s only two a year, but I might go to two or three other ones per year in addition to that.

[0:30:15.8] JM: There’s going to be many more than two Kubernetes conferences before long.

[0:30:19.6] JJ: I think you’re right.

[0:30:21.5] JM: Yeah. Sorry, go ahead. I interrupted you.

[0:30:24.8] JJ: No. That’s pretty much all. I don’t go to so many conferences that I have this sort of super well-framed point of view, but I think there are too many conferences mostly because — As an example, I’ll see — I’m not going to name names at all. I should probably be even more abstract about this. People that are like I know are doing really serious and heavy investment initiatives around specific projects, and I just see them at conferences and they’re at every single conference. I’m like at some point, “What is the value beyond the networking and maybe the recruiting aspect? If that’s the case, you shouldn’t have all of your core technologists at those conferences. They should actually be working with customers or building the product or engaging with their internal teams to actually get the new thing adopted rather than spending their time at conferences. From that standpoint, I think we might be approaching peak conference, or like there maybe too many conferences.

[0:31:27.7] JM: Yeah. Conferences are great for networking and recruiting and lead gen, but you have to have a strong basis that you're compounding off of at those outreach efforts, because that's what it is, it's outreach. You have to have something that's underpinning the outreach. Greg, did you say that you had some conference fatigue or conference regret, like maybe you shouldn't? Was the ROI not there in retrospect?

[0:32:00.9] GK: No. That was definitely more of a not wanting to be a cliché and not getting the ROI back. I think one thing I want to make sure if we had one strong perspective we are pushing. I didn't want to just do it for mistake of having a conference, and I also want to make sure we had three or four things that felt unique and memorable. We followed the sale format that every conference does and we didn't do anything like completely random or like, "Okay, once we get there, put headphones on and listen to the speakers alone in silence." We didn't do anything weird like that. It was still just a conference, but I wanted to try to do my best to make sure that there was something special about it that was at least slightly memorable, because I don't think there's actually — I do agree that there's too many conferences, but there's too many everything. There's too many startups. There's too many podcasts. There's too many everything. That's not a bad thing. I wasn't going to say — That doesn't mean don't do a podcast or don't do something else. It just means if you're going to do it, you have to make sure you put time and effort into doing it right and make it so that it stands out and it's memorable.

There's so many podcasts, of course, but that doesn't mean don't do it. It just means you need to be better at it. You need to be whatever that means for the podcast. Maybe it means be more rigorous. Maybe it means be shorter or be longer or ask more interesting questions or have a higher production quality, stuff like that.

[SPONSOR MESSAGE]

[0:33:26.8] JM: Your application sits on layers of dynamic infrastructure and supporting services. Datadog brings you visibility into every part of your infrastructure, plus, APM for monitoring your application's performance. Dashboarding, collaboration tools, and alerts let you develop your own workflow for observability and incident response. Datadog integrates seamlessly with all of your apps and systems; from Slack, to Amazon web services, so you can get visibility in minutes.

Go to softwareengineeringdaily.com/datadog to get started with Datadog and get a free t-shirt. With observability, distributed tracing, and customizable visualizations, Datadog is loved and trusted by thousands of enterprises including Salesforce, PagerDuty, and Zendesk. If you haven't tried Datadog at your company or on your side project, go to softwareengineeringdaily.com/datadog to support Software Engineering Daily and get a free t-shirt.

Our deepest thanks to Datadog for being a new sponsor of Software Engineering Daily, it is only with the help of sponsors like you that this show is successful. Thanks again.

[INTERVIEW CONTINUED]

[0:34:53.5] JM: Speaking of things for which there are too many, I want to ask both of you about the state of cloud providers, because I feel like both of you have unique perspectives on cloud providers, and this is just a subject I like to cover and like to ask people their ongoing thoughts. We already talked about Lambda and Kubernetes which are two things that are impacting the growth of cloud providers and the perspective of cloud providers.

We have Azure, AWS, Google, and then we've got like IBM, DigitalOcean. There are some cloud providers. I want to talk about the major players, but first is there room for the other cloud providers, like the IBM and the DigitalOcean or perhaps others, or is this going to be a winner's take all environment?

[0:35:51.1] GK: I don't really have a strong opinion other than I think that the actual cloud providing tends to be a race to the bottom, and I think that because this is what I like is taking the raw tech and kind of building on top of it. I don't really see any incentive. Yes, there's Microsoft, Amazon and Google all have products in this space. I don't see a huge incentive for a new company to come along and try to steal away some of these pennies for — Whatever, businesses.

I do see a lot of people though coming along and building businesses on top of these, like for example Lambda. Lambda is probably a really cool technology, and I say probably because it's also really hard to get started, really hard to use, really hard to manage. Amazon does all the powerful stuff, but I think there really needs for Lambda to ever really take off and be used by people who aren't just technologists. I think that Lambda needs a Heroku on top of it. I think that is where we're going to see — I don't know if it's going to be that companies that make a ton of money, because you could argue whether or not Heroku is successful. They're very successful in a sense that we all know about them, we all use them. They seem to be pretty successful, but they're also not Amazon. I do think that we're going to see a lot more of those, companies that build on top of Azure, AWS, all the Google stuff. I don't know if financially they're going to be best off, but that's where I tend to see a little more — I'm a little more interested.

[0:37:15.3] JM: Joseph, any perspectives on this?

[0:37:17.1] JJ: Yeah. I would tend to agree with what Greg said around the race to the bottom. There's just this mad rush to offer every possible menu compute instance type and size permutation of CPU, memory, disk, RAM offering, and I think there's so much competition that people are just going to be forced to offer the lowest possible price point and it will just get commoditized down close to zero, but there's also lots of interesting "platform services" that are getting shift kind of trying to change the way people deploy software and integrate it with other middleware offerings. I think that this is where there's going to be lots of interesting developments and innovations in the cloud provider area.

I'm not sure, back to your question, Jeff, around the little guys. Is there a space for the Packets and the DigitalOceans of the world? I think there is in the sort of midterm or median between now and cloud provider sort of compute locality ubiquity and maybe in the future hundreds of datacenters servicing the latency needs of people running in Ireland serving a few million telco customers where they're still running in CoLOS and they're still having to run their own datacenters for speed and performance reasons and latency reasons.

Yeah, I'd say that I tend to think that there's going to be a concentration of three or four dominant cloud providers and all of the other smaller ones will just get sort of consolidated and aggregated in through sort of acquisition or through just getting absorbed and to just running on all those core major three or four platforms. Like Greg was mentioning, just the sort of higher level compute offerings that people are just going to build on top of cloud providers.

The Lambda stuff is very interesting for sure as well. Yeah, I think people want to really only pay for the compute cycles their applications consume and I think that the sort of function execution sort of model lends itself best to that, but I think we're very far away from solving for the usability and deployment sort of ease of use that people are used to now with things like Docker and Kubernetes and these other frameworks that run container-based apps.

Yeah, I think that all of these is still really early on. One of the things that I remind people off when they go, "Oh, it's all going to the cloud and datacenters are completely going away," is the IT industry overall is like maybe $4 trillion, and this is I think documented by Gartner and other analysts. If you add up all of the sort of numbers from public cloud and also including SAS vendors, like every single — Like Salesforce, Workday, it's only about 400 billion including all of those providers.

We've only scratched the surface of about 10% of like basically all the core infrastructure IT functionality enterprises in the sort of as a service model, which basically means — We're about 15 years into that transition, which means we have at least another 15 or 20 years before there's a material transition overall. I think that we're still super early days, which is pretty cool. It's exciting. I don't think anyone really knows what's going to happen.

[0:40:56.7] JM: Obviously, the Lambda stuff, the functions as a service, what that does is it raises the efficiency with which these large cloud providers can allocate their compute so they can more effectively utilize their servers because they're pairing their small amounts of compute with small requests for compute and so that lets them lower the prices on those. As they get better economies of scale there, then that further squeezes the smaller players who have less volume demanded of their services, like IBM or DigitalOcean, although IBM has a function as a service too. Maybe they get the same utilization, or maybe they get something close to the same utilization, or maybe the margins are good enough so that they don't need as good utilization as an AWS. They just need something that's similar.

Like you said, there's the deployment and interfacing that there's a lot of room for improvement. The UX stuff. If you built a nice UX or a nice platform as a service for something specific, maybe it's like self-driving car infrastructure or something like that, maybe you can make a DigitalOcean or an IBM work in the future, then there's also services. Rackspace went to services and that worked for Rackspace. I think Rackspace, the biggest part of their business ironically today is servicing AWS clients, which is interesting.

[0:42:43.8] GK: Wow!

[0:42:45.4] JM: Yeah, I heard about that. I'm not sure if it's true. Do you guys have a sense for whether the major cloud providers, like Azure, AWS, and Google, are they starting to differentiate or do you feel like they're pretty much mirroring what each other does?

[0:43:02.0] GK: I think mirroring. There's definitely differences where if you're writing a .NET application, you're probably going to want to use Azure. Off the top of my head, I can't think of anything great, but there's definitely reasons to use Google over other ones. I like AWS because when our site goes down everyone else is down, so no one notices or cares.

Yeah, I think that just kind of every time anyone comes up with anything good, the rest of them do. I have not seen a huge amount of — There definitely is differentiation. I do think it seems to me that — Maybe I'm wrong about this, because I don't really use it. It feels like Google is trying to go a little more high level where Amazon is trying to go a little lower level. When I say high level I don't necessarily mean like Heroku high level, but it does feel like Google is trying to go a little slightly more that route. Yeah, I don't know. I don't know why they differentiate. Probably simply because — They do, and then the other three or two copy them or right back to not being differentiated.

[0:44:03.3] JJ: Yeah, I wouldn't disagree with much of that. I think, historically, Amazon just had the sort of time to market advantage and they've been there the longest and they have the most kind of aggregate sort of service portfolio offering and hundreds and hundreds of services. I learn about new services almost like every month or so that have been around for a year plus. We have so many services. We've got this huge disaggregated buffet of stuff. Then Google is like the newest one and has been really focused on their sort of open source strategy and kind of releasing these projects that sort of reflect the interface and the API that you'd be consuming directly from them, and the configuration format and the sort of core runtime as well so you can take those projects and run them anywhere. I think that's a very unique kind of thing to Google, is sort of this kind of open cloud, open source based service offering.

Azure is sort of historically been very enterprise. Windows server and kind of office portfolio focused, and they've got like a huge advantage over both, I think both Google and Amazon in terms of their customer relationships and all of the historical customer engagements and revenue that they've done just through being Microsoft. I think there are differences, but I'd agree with Greg that there's lots of commonalities in terms of how things end up normalizing as far as like compute, choice and the sort of platform services that end up getting released.

Yeah, that's generally how I sort of look at that. I think Google is kind of in the most kind of situation because they have this sort of very long term investment around wanting to get open source projects to a certain level of ubiquity that they either created or are heavily endorsing and then making the sort of industry sort of believe and buy-in to the fact that they have the best sort of managed services or hosted versions of those open source projects which I think is just a radically different go-to-market and sort of product strategy as compared to Amazon or Microsoft. To Greg's point, lots of these other cloud providers are now starting to kind of sort of do the same thing. We'll see how it all plays out.

[0:46:18.7] JM: Okay. I want to begin a wrap up by talking a little bit more about running a company, because I'm kind of starting to — I've been working on Adforprize and just learning some stuff around running a software company that's been interesting to me, and I'd love to hear your lessons. What are the biggest myths around starting a company? Like stuff that you read in books and blog posts and you hear in podcasts that maybe has not born out to be as true in your respective experiences as it's advertised in the company lore.

[0:47:02.9] JJ: My quick one is a lot of people say product market fit sort of problems cause startups to fail the most in terms of never being able to reach that sort of perfect equilibrium of what you're selling is exactly meeting some sort of sufficient demand profile such that you can build a business. I think that's like a total sort of misnomer. The much more common reason that startups end up failing, the biggest thing to kind of harp in on if you want to really mitigate a lot of the risk is founder-market fit or founder fit in general and if you don't really kind of scrutinize that one the most, I think you'll either proceed forward and kind of sort of be in denial about a lot of the potential issues and conflict that might come up down the line or the company will be really limited in terms of potential because the founders aren't really on the same page about pretty much everything critical in terms of the business and the motivations and the reasons why people are doing the thing and everything like that. I think that's actually, in reality, the vastly more common reasons startups fail is the sort of founder- market fit or founder-founder kind of conflicts. I have some experience in that area, but I think that's something to kind of pay attention to. If you're doing a company or if you're evaluating like, "Do I have the right team on the sort of founding team side?" It's super, super critical to solve that and make sure that everyone is sort of super aligned the vision and the opportunity and the sort of excitement and the conviction, passion people have beyond anything else.

[0:48:47.2] GK: I 100% agree with that. I think that founder-market fit is like I phrase I use all the time because I think that — For example, neither our companies — Not to put your company own, JJ, but neither of our companies are ever — It's not as good of an idea is, for example, Airbnb. When I say it's good of an idea, I mean that, obviously — Or Uber, for example. There's no way I could have ever made a company like Uber successful ever. I think it's so important to find the company that fits your ability and your passion because ability is important because it's kind of like being in a basketball game. You can't stop, think about it every time. You need to just kind of like rely on your gut, rely on your reflexes, stuff like that.

For passion, it's not fun starting a company. I love it. It's the best thing I ever did, but day-to-day, you just take a beating. If you're not passionate about it. If you don't really believe what you're working on, it's not going to end well.

I think, yeah, I 100% agree that product — More important than anything else, is not the size of your market, not the size of anything like that, it's just how closely aligned are you to the thing you're building.

[0:50:01.8] JM: All right, guys. It's been great talking to you, and I want to thank you both for coming on Software Engineering Daily.

[0:50:07.4] GK: Thank you very much for having us.

[0:50:08.3] JJ: Thanks, Jeff. Nice to meet you, Greg.

[0:50:09.3] GK: You too

[END OF INTERVIEW]

[0:50:13.6] JM: Spring is a season of growth and change. Have you been thinking you'd be happier at a new job? If you're dreaming about a new job and have been waiting for the right time to make a move, go to hire.com/sedaily today. Hired makes finding work enjoyable. Hired uses an algorithmic job-matching tool in combination with a talent advocate who will walk you through the process of finding a better job. Maybe you want more flexible hours, or more money, or remote work.

Maybe you work at Zillow, or Squarespace, or Postmates, or some of the other top technology companies that are desperately looking for engineers on Hired. You and your skills are in high demand. You listen to a software engineering podcast in your spare time, so you're clearly passionate about technology. Check out hired.com/sedaily to get a special offer for Software Engineering Daily listeners. A $600 signing bonus from Hired when you find that great job that gives you the respect and the salary that you deserve as a talented engineer. I love Hired because it puts you in charge.

Go to hired.com/sedaily, and thanks to Hired for being a continued long-running sponsor of Software Engineering Daily.

[END]