EPISODE 745

[INTRODUCTION]

[00:00:00] JM: Amazon Web Services changed how software engineers work. Before AWS, it was common for startups to purchase their own physical servers. AWS made server resources as accessible as an API request, and AWS has gone on to create higher-level abstractions for building applications. For the first few years of AWS, the abstractions were familiar. S3 provided distributed reliable objects storage. Elastic MapReduce provided a managed cloud Hadoop system. Kinesis provided a scalable queuing system. Amazon was providing developers with managed alternatives to complicated open source software.

More recently, AWS has started to release products that are completely novel. They're unlike anything else. A perfect example is AWS Lambda, the first function as a service platform. Other newer AWS products include Ground Station, which is a service for processing satellite data; and AWS DeepRacer, a miniature race car for developers to build and test machine learning algorithms on.

As AWS has grown into new categories, the blog announcements for new services and features have started coming so frequently that is hard to keep track of it all. Corey Quinn is the author of Last Week in AWS, a popular newsletter about what is changing across Amazon Web Services. Corey joins the show today to give his perspective on the growing shifting behemoth that is Amazon Web Services as well as the other major cloud providers that have risen to prominence.

Corey is the host of the Screaming in the Cloud podcast, which you should check out if you like this episode. I should also mention that we have our own newsletter. You can go to softwareengineeringdaily.com/newsletter to check it out and sign up, and we also are looking for sponsors for Q1. If you're interested in reaching over 50,000 developers, you can go to softwareengineeringdaily.com/sponsor.

[SPONSOR MESSAGE]

[00:02:29] JM: I have been going to the O'Reilly Software Conferences for the last four years ever since I started Software Engineering Daily. O'Reilly conferences are always a great way to learn about new technologies, to network with people, to eat some great food, and the 2019 O'Reilly Software Architecture Conference is for anyone interested in designing and engineering software more intelligently. There are actually three software architecture conferences. One is in New York, February 3rd through 6th; in San Jose, June 10th through 13th; and in Berlin, November 4th through 7th. So if you're in New York, or San Jose, or Berlin, or you're interested in travelling to one of those places, take a look at software architecture. You can get a 20% discount on your ticket by going to softwarearchitecturecon.com/sedaily and entering discount code SE20.

Software architecture is a great place to learn about microservices, domain-driven design, software frameworks and management. There are lots of great networking opportunities to get better at your current job or to meet people or to find a new job altogether. I have met great people at every O'Reilly software conference I have gone to, because people who love software flock to the O'Reilly conferences. It's just a great way to have conversations about software.

You can go to softwarearchitecturecon.com/sedaily and find out more about the software architecture conference. That's a long URL, so you can also go to the website for this podcast and find that URL. The software architecture conference is highly educational and your company that you work for will probably pay for it, but if they don't, you can get 20% off by going to softwarearchitecturecon.com/sedaily and use promo code SE20.

Thanks to O'Reilly for supporting us since the beginning of Software Engineering Daily with passes to your conferences, with media exposure to different guests that we had early on. Thanks for producing so much great material about software engineering and for being a great partner with us as we have grown. Thanks to O'Reilly.

[INTERVIEW]

[00:05:11] JM: Corey Quinn, you are the author of the Last Week in AWS Newsletter. Welcome to Software Engineering Daily.

[00:05:17] CQ: Thank you. It's a pleasure to be here.

[00:05:19] JM: When did you first start working with Amazon Web Services?

[00:05:23] CQ: Great question. There's a little bit of revision as to history that goes into it. I started back in 2008, give or take, of, "Oh! What is this thing? You mean the online bookstore? I'll start clicking around in the console," or what passed for the console at the time, and it turned out it was super complicated and hard and I very quickly started looking at things like RightScale to make sense of it. I've sort of dabbled with it ever since through a variety of jobs, consulting engagements, but I didn't get really focused on AWS until a bit over two years ago as a primary area of interest.

[00:05:57] JM: You've been in software for a while. How would you describe the world of software before and after AWS?

[00:06:04] CQ: That is a terrific question. I'd say beforehand, it wound up having a much higher barrier to entry. You had to deal with a bunch of different VPS providers, all of them terrible in various ways, or you'd have to wind up building out your own data center. Having spent entirely too many years doing exactly that, it focuses doing all of the wrong behaviors.

[00:06:26] JM: In your background, were you like a person that was in a data center or were you just working with engineers that were in a data center? Give me a little bit of background on what you've been doing engineering-wise, IT-wise in the past.

[00:06:39] CQ: Sure. I started my career as a grumpy UNIX systems administrator, which you can also state's UNIX administrator, and from there it turned into an awareness of what the rest of the world was doing. It was pretty clear that my initial area of emphasis, which was large-scale email systems, was not the wave of the future unless I wanted to work for one of maybe four companies. So I started pivoting. I focused increasingly on automation and data centers using things like puppets. I was one of the early developers behind SaltStack, which really it should not be as much of a condemnation as it probably is if you've ever seen my code. I love the technology. My code is terrible, but at the time there was no one else to do some of the things that needed to get done.

There was another shift, where configuration management was no longer the way of the future. Immutable infrastructure as a concept started rising very rapidly. So if not getting environments to correct for drift, if that was not going to be what I focused on next, what was? It seemed to me that when I left my last job and was trying to figure out what's next, becoming a consultant focused around a very specific problem was the right answer. Specifically, the AWS bill was too high. What problems I ran into early was that Amazon releases an awful lot of stuff constantly. All of them affect the economics of what you're doing in your environment. So how do you wind up getting from understanding what's important and what moves that without getting lost in the minutia?

So I started collecting a list of everything that happened every week to keep myself up to speed, and it didn't take much for me to realize that maybe I wasn't the only person who would benefit from this. I put together a few drafts of a newsletter. Told people I would be starting to send it out in a couple weeks. I figured I'd get a couple people who signed up. Charity Majors tweeted about it, and the first issue went to exactly 550 people. Okay. I guess I'm doing this for a while, and that was at the time of this recording almost 90 weeks ago.

[00:08:40] JM: So your newsletter is the Last Week in AWS. Explain the goals of your newsletter.

[00:08:46] CQ: Sure. It has a few different goals depending upon who you are. For me, it forces me to keep up with what's going on and not drift and ignore it. It also winds up gathering all of the news from Amazon's cloud ecosystem, both the announcements that they release internally as well as what people are doing in the community.

The problem is, is because they’re a giant multinational company, they're not allowed to have a sense of humor of which they are themselves aware. My first language is sarcasm and snark. So I make fun of the announcements that come out and I include in a way that is hopefully not punching down, not calling out individual people that focuses on snark and sarcasm and being uplifting rather than putting everything down and crapping on it all the time.

That sort of keeps me engaged, because otherwise I'm more or less gathering a bunch of press releases, and midway through the second issue I would fall asleep and give up and go do something fun. So it keeps me engaged. It lets me express creativity and it forces me to keep up with a very rapidly evolving ecosystem.

As an added bonus early on in the process, I received email from someone. I think the first sponsor was Datadog, asking, "We love what you're doing. Can we give you money to mention our product in it?" To which my response was, "Can you give me money? Well, of course you can give me money. How much money are we talking about?" and it very rapidly emerged that I had built a strange revenue model. I expected this to be a labor of love, but it turned into something that's nontrivially – That's throwing off nontrivial amounts of revenue. So that became something I didn't expect.

[00:10:25] JM: Not only that. The advantage of doing developer-focused media is you build a really strong familiarity with what is going on in technology. So in some sense, you cap your downside risk, because even if this whole newsletter thing doesn't work out. Well, you've built a really good understanding of what's going on in AWS.

[00:10:51] CQ: Exactly, and if we have to take a slightly more cynical approach, again, my entire business is built around fixing AWS bills for large environments. That's lucrative, but it's also not something I can necessarily see myself doing for the next 50 years. I don't know what I'm going to do next. I know I don't hold still very well, but whatever I wind up doing in five years, I'm going to need a base of people to tell about it. So the time to build an audience from that perspective is now.

[00:11:19] JM: Indeed. Well, let's get into a discussion of AWS. Describe your mental model of Amazon Web Services.

[00:11:29] CQ: I view a company that operates in such a way that compared to any other company in the planet is it may as well be an alien organism. They're effectively microservices driven, which means they have very small teams each working on individual projects. Any time they wind up having to work on something that works cross functionally across all of the service teams, it turns into – I don't want to use the word disasters. So pretend I didn't. You see that in

manifest in things like the console, the bill, CloudWatch, which has aggregate metrics across everything.

But the individual services that launch as developed by small teams is fascinating. They have a product strategy of yes, which also means that they're at some point going to wind up competing with basically everyone, and that does give some people pause. But today, if I take a look at the entire cloud ecosystem, they are for most use cases. The vendor it makes the most sense to work with.

I'm not a partisan as far as these things go. The reason I focus on AWS bills is because that's where the customers are. If people were instead all on Azure, we'd be having a very different conversation. I don't know if I'd be writing Last Week in Azure, but I think that there would be a need for something like that in that ecosystem.

[00:12:47] JM: Do you think of AWS as a set of products specifically for engineers or do you think there are aims to move up the stack into – I don't know, design tools, or things for – I don't know, sending emails or things like that?

[00:13:03] CQ: They've already got that to an extent. As of the time of this recording, they just released a whole email deliverability dashboard for Pinpoint, which is itself a service built on top of SES. With over 150 services now, even Amazon employees aren't always sure whether I'm talking about a service that exists or something I made up just to mess with them.

At Reinvent this year, they announced AWS Ground Station, which provides telemetry services for satellites passing overhead, and even after mentioning that in a couple of talks in the Q&A section, I've gotten, yeah, follow-up, "Are you making that up just to mess with us?" "No. It's real. It's called Ground Station. Here's what it does." "Cool. Follow-up. No, seriously, are you messing with us?"

So if you're asking me you, "Is there anything that I would say definitively that AWS would never move into as a market?" Really, the only one I can think of is my own. I can't see AWS self-mockery launching anytime soon.

[00:14:03] JM: So AWS is this sprawling set of services. What's your sense of how the engineering org is laid out to foster that kind of sprawl or to sustain that kind of sprawl?

[00:14:18] CQ: From the conversations with people who've worked in that environment, some of whom were extreme champions of Amazon, others of whom would not cease the profanity, it seems to me that the – The feeling I guess is closest to a bunch of internal startups that are competing for funding, for mind share, and they go through iterative rounds until something winds up getting released. At the end of an entire laborious process of iteration going through series of fundings, their "exit" is when someone at AWS gives the service a stupid name and launches it to the public.

That means that a few different things emerge. A lot of things get built that never see the light of day, and occasionally you'll see services launch that appear to directly compete with one another, and that becomes in some ways a fascinating story.

[00:15:07] JM: I was doing a lot of coverage of the container orchestration system wars. There was this period of time where Docker had reached market acceptance and people were looking for what is the best solution for managing all of my Docker containers. Is it going to be Mesosphere? Is it going to be Docker Swarm? Is it going to be HashiCorp Nomad? Then eventually Kubernetes came out and it reached enough saturation or market share or was marketed well enough that people said, "Okay, this is what we're settling on," and AWS had placed a strong bet on any of the particular orchestrators. They had placed a bet on their own proprietary orchestrator, the ECS one. Then when Kubernetes got accepted, there was a sense that it impacted the strategy of AWS. How did Kubernetes impact the strategy of AWS?

[00:16:03] CQ: From my mind, I think you nailed it and that there were a lot of different competing standards, more or less, and when Kubernetes emerged as a more or less de facto across the board, it feels – Again, I have no inside information on this, that it sort of caught AWS flat-footed. They wound up releasing EKS, or the AWS Elastic Container Service for Kubernetes. Apparently, they get paid by the syllable for that one. It was very much a 1.0 product when it launched. It took over 15 minutes to provision a cluster. It was not at all clear what permissions it needed. So you needed to grant it roles that were incredibly broadly scoped. It didn't wind up doing logging appropriately. The horizontal pod auto-scaler didn't get released

until well after initial launch, and the initial guidance around that was, “Well, it's not the first to market. It's not the best-of-breed. I'm not entirely sure why I would use this today.”

That said, in typical Amazon fashion, improved rapidly since the time of launch to the point where, now, looking at it, it's not at all a bad option. That said, I think the larger ecosystem story here is that Kubernetes or how we orchestrate containers is not going to be an area of focus for too terribly long. I mean, when’s the last time you had a deep dive discussion with someone about which Linux distribution they should use? It becomes plumbing. It starts slipping below the surface and stops mattering to most people. I think that container orchestration is absolutely going to follow soon.

[00:17:37] JM: Yeah, and what's interesting about that is what has gone from the serverless idea, the Lambda, AWS Lambda, has gone from a fringe kind of cool idea to looking like that's going to be the modality that people are building their applications in from day one, more and more going forward. Although it's hard to see exactly what the serverless stack will look like, because today I guess the most serverless stack that you can build would be something where you're using a bunch of managed services, like queuing database, some kind of database system. Then you glue together these big managed services with the glue code that runs in AWS Lambda, but that could certainly change. You could certainly have further developments of things on top of Lambda. What's your sense of the adoption of serverless?

[00:18:36] CQ: I think that serverless is one of those interesting technologies that when it launched, it started to look like an awful lot like a toy, where it's the sort of thing that you see that more or less manifests itself in a way that, “Okay, this is neat for this one use case, but never for my use case. It only supports some certain languages and it only winds up working for five minutes.” As capabilities continue to expand and it starts to move up the stack, it begins to be something that starts to look a lot more realistic.

The, I think, thing hidden secret behind serverless that powers it is the event model, where an event happens in your environment and it automatically invokes a reaction to it that is highly parallelizable and it almost completely removes the need for a company to look at infrastructure. It's also priced incredibly competitively to the point where there is virtually no company on the planet that is spending a huge about of money on Lambda.

So it winds up shifting the attention in a few different ways. The problem is, is that as we look at serverless, the term come to mean a lot of things to a lot of people and it's not helped by people chiming in uselessly to proclaim, “Serverless still runs on servers as if it were this revelation that had not occurred to anyone until that person chimed in.” “Great. Terrific. Thank you. Do you have anything else to add other than the same hackneyed comment that everyone makes all the time?” That's not helpful and you're not advancing the discourse in any meaningful way.

[00:20:06] JM: Although there are fair criticisms of Lambda. It doesn't do everything these days.

[00:20:12] CQ: Oh, I have a list.

[00:20:13] JM: Oh! Tell me your list. What are the shortcomings of using Lambda these days?

[00:20:18] CQ: The initial problem of course is shifting the way you think about software into this new paradigm. If you have a 20-year-old monolithic application that has been doing something of business value, for example, running the ATM networks or this is what make sure the traffic lights don't all turn green at the same time, then shifting that to Lambda is likely not going to be beneficial in any strategic sense. Whereas if you're looking at this from the perspective of rebuilding something Greenfield and this is the way we think about things, maybe that ends up adding significantly more value.

The biggest danger from my perspective is people who see this new tool and see it strictly as a set of checkbox capabilities but don't shift the way they envision architecture, where they still try and shove an entire monolith into a Lambda function. Where they wind up trying to store state inside of Lambda. You can build an awful lot of terrible things with terrible anti-patterns using something like this technology, but that doesn't mean that that's necessarily how you should be doing a thing.

I mean, I gave a talk I called Terrible Ideas in Lambdas. Terrible Ideas in Serverless, Silence of the Lambdas, which showed exactly how not to do these things. Like anti-patterns, for example, it turns out most databases fall over and catch fire when you have 10,000 Lambda functions concurrently try to talk to it at the same time. When you have a whole bunch of Lambda

functions scanning the entire internet to find open Elasticsearch clusters, it turns out Amazon would like a word, because their premier serverless platform isn't something they like you using to attack the larger internet. So there are ways of doing things well and there are ways of doing things terribly just as with any tool. No tool survives first contact with other people's terrible architecture ideas.

[SPONSOR MESSAGE]

[00:22:18] JM: This episode of Software Engineering Daily is sponsored by Datadog. Datadog integrates seamlessly with container technologies like Docker and Kubernetes so you can monitor your entire container cluster in real-time. See across all of your servers, containers, apps and services in one place with powerful visualizations, sophisticated alerting, distributed tracing and APM.

Now Datadog has application performance monitoring for Java. Start monitoring your microservices today with a free trial, and as a bonus, Datadog will send you a free t-shirt. You can get both of those things by going to softwareengineeringdaily.com/datadog. That's softwareengineeringdaily.com/data.

Thank you, Datadog.

[INTERVIEW CONTINUED]

[00:23:12] JM: AWS uses the term serverless in the context of managed databases sometimes these days. So there is a serverless Aurora, and I haven't really delved into what that means. Aurora is I think their PostgreS or their managed relational database service, but I don't know why they use the term serverless, because like S3 is in a sense serverless, or DynamoDB is a serverless. What's different about serverless Aurora?

[00:23:43] CQ: Terrific. I would consider S3 serverless, and they might've even called what it that when it launched 10 years ago if they'd thought of it in advance. So you would have serverless simple storage service, so within the S4.

[00:23:54] JM: S4. There you go.

[00:23:55] CQ: Exactly. I think there are a few things that define something as serverless. One of them is, sure, well you pay for data that lives in the system, it scales down to zero. So when it's under load and processing work, it costs you money. When it's not, it doesn't. That seems to be one of the key distinctions. Historically, Dynamo was the subject of some debate, because you could only scale it down to one read and write capacity unit. So it's still costing you a few bucks a month. Not that that's massive when you have 1,500 of them sitting in various dev environments and it starts to add up.

So the ability for it to scale down to nothing other than what data you're storing in that and then on demand spin up and begin processing things from a compute perspective seems to be a fundamental tenet of serverless these days. They released, for example, DynamoDB on demand capacity at Reinvent recently, and specifically nailed that objection right there. Whether or not something is serverless or not is again something that I leave more for the philosophers and people arguing on the internet while the rest of us go about our jobs. But I do get the sense that serverless Aurora, which now I believe has announced both MySQL and PostgreS flavors where you have this thing that lives there. It's a relational database. You don't need to think in a MySQL context. But now whenever your application gets traffic, they can talk to a database as it would traditionally, but then that database turns off when you're done. You're not getting billed for. It doesn't need to sit around doing nothing. Given some of the new data models, it also starts to support much better concurrency. So you avoid the 10,000 Lambdas all talking to your MySQL [inaudible 00:25:30] problem.

[00:25:31] JM: Well, if that's the promise of serverless, then I really like where this is going, because I don't know about you, but I have a bunch of applications, like stupid experiments that I've run, but I refuse to turn them off because someday I'm going to come back to them and do something with them and they're costing me $25 or $12 a month and I just – Whenever I look in QuickBooks or my bank statement, it's like, "Oh, $12 here, $25 there," and it starts to add up.

Hearing you talk, it makes me think like this is going to be a relic of the past. Eventually these things are just going to shut down while they're not being used. It doesn't make sense to have a cloud infrastructure system that just like stays up all the time while it's not being used.

[00:26:18] CQ: Absolutely. Yeah, I played with AWS once. That's why I pay Amazon $0.22 a month and will until the earth crashes into the sun.

[00:26:25] JM: Yeah, exactly. So there are these companies that have huge amounts of infrastructure running on AWS. I'm sure you've met many of them. What kinds of challenges does a company run into once they have a really large AWS deployment?

[00:26:42] CQ: Visibility, first and foremost. For example, my bill last month was $16 on AWS. The month before that, because I did a demo for someone and then left some things running inadvertently, it was 50 bucks. So when your bill more than doubles, you notice that. When you're at significant scale and you're spending – I don't know, $120 million a year on AWS, it turns out that even big mistakes are easy to lose in the noise. You do some digging around and you realize that someone in your data science team wound up copying an extra few petabytes into S3 and left it there after they left your company three years ago. You start to see things like that start to accumulate waste and croft. It all comes down to a certain lack of visibility and control.

Large companies are generally used to the historical data center model, where you would wind up building things on a capital expense basis. You would plan out your data center build outs, and it's super hard for a single engineer to accidentally order $6 million worth of hardware without getting fired or arrested. The new model though is that someone can inadvertently spin up that level of resource and not only not be aware of it, but no one is aware of that, for in some cases, years at a time. It is not at all transparent what's happening in your environment.

I keep going back to the bill not just because that is what I do for a living, but it's also the only place in your entire AWS account where you can see on one screen all of the resources you have running across all of the regions in your account or linked accounts. There's no inventory service. It's just the bill.

[00:28:18] JM: What are some other ways that teams that are running on AWS end up overspending?

[00:28:24] CQ: A great example of this is if you take a look globally, something like 60% of all spend is on EC2. If you add four more services, you get to the 85% market, then there's a long tail of other things. So you see people overspend by not purchasing reserved instances, because they're convinced they're going to turn that cluster off next week and then months and months and months go by and that never gets turned off. In fact, it expands.

Yu also see effectively misunderstandings. If you're new to AWS and you want to spin up at T3 instance or a P3 instance, that's one letter that sounds the same difference, but you can go anywhere for spending half a penny an hour to 40 some dollars an hour based upon that single letter, and that winds up being something that is a tremendous shock to people who've never played with us.

The counter challenge that you see in these large companies is how do you govern that intelligently? You can act as a gatekeeper that says, "You're absolutely not going to be allowed to spin things up without a three-week approval process." Well, that's where cloud came from in the first place. When you have a corporate credit card that has a $500,000 spending limit, you can spin up a new cloud account and you're good to go until it's too big and serving production traffic and then it gets noticed and accounted for. If you block people, they're going to start doing that process again or they're going to go work somewhere else.

You also don't want to turn a human into the, "I'm going to nag people about what they've spun up in Amazon all the time." So I'm looking at the bill by user and who spun what up, and dear Lord, I don't know who this Jenkins person is, but they're spinning up all the resources in production, and that leads the biggest problem that you see. The person getting the bill in finance and the engineer spinning up the resources that impact the bill are five organizational levels apart in most.

Person in engineering spins up some resources, and the end of the month the person and finance sees an enormous Amazon bill and wonders how many books engineering is buying. They don't see them reading all that much.

[00:30:24] JM: A company he like Netflix has a massive AWS budget. Is there a department within Netflix that does cost negotiation and cost analysis? How do you think the dialogue between Netflix and AWS goes?

[00:30:42] CQ: Without speaking to specific companies, I have a number of clients in or around that general level of scale, and there's an evolution. In some companies, especially those that are generally either born in the cloud or have adopted a cloud native approach and that permeates everything, if you wind up going from the idea that an engineer can do this part-time, perhaps partnered with someone in the finance group to building out a cloud cost optimization team, to effectively having a dedicated usually small team of folks whose entire purpose is to go and optimize things for cloud spend. Those people are not inexpensive, but at certain points of scale they can save $10 million a day depending upon what they're focusing on. So the economics begin to make an awful lot of sense.

The challenging part for me and one of the great inefficiencies that I see in cloud is hiring those people when you're spending tens or hundreds of millions of dollars a year makes sense. If you haven't done that, you're probably doing something wrong. Whereas if you're spending – I don't know, $40,000 a month on your AWS bill, even hiring one person to do that will cost you more than any savings they could possibly arrive at. So there are inflection points as you go from zero to small, medium, large, “Holy crap! Is that a phone number?” level of AWS bills.

[00:32:04] JM: Yeah. I mean, this is one of these areas that people would've never expected to be a business 10 years ago. The cloud cost optimization specialist business. I've met probably four or five different companies that have built really big businesses off of this, and that's your day job. That's what you do as your core business. You help people fix their AWS bills. So what is that look like? What are the methods for cost controls that you have found to be most effective?

[00:32:37] CQ: Great question. I'm not going to talk smack about any of the vendors in this space. There are over a dozen at this point now. But they all tend to tie back to the same model. They come into an environment. They drop a bunch of analytics on to an AWS account. They charge a percentage of your AWS bill every month, which incidentally finance hates, and then

they wind up saying, “Here, we’re pointing out cost savings opportunities for you to go ahead and implement,” and in practice almost no one does.

I start to see this entire space as something that doesn't respond nearly as well to a platform as a service offering. Yes, you need analytic visibility into it, but most of what I do takes a much more advisory tone. It's not about tooling. It's about getting the person in finance and engineering to sit down and talk to one another. It's about building good governance processes. I don't sit there and write custom code for my clients. I have conversations, and that distilled down to do these five things and you'll knock 22% off of your bill on average as an initial assessment. Then we'll get into deeper discussions around strategy around how you want to govern this environment, how you want to start handling reporting of this and how you can start shifting how the business views this.

If it makes sense, past a certain point, cost savings no longer matters to a company. You’re not going to optimize to your next business milestone. You’re just going to be a responsible steward of your money or improve the picture of your unit economics. It's one of those areas where at some point your innovation is much better spent elsewhere.

The challenge of course is that this is a problem every company has and they all tend to solve it themselves internally like it's a bespoke unicorn. This is the business equivalent of we’re going to build our own version of container orchestration. No. Use Kubernetes. The trouble is, is today there is no Kubernetes for this problem.

[00:34:27] JM: Let's talk about legacy enterprises. There are large legacy enterprises that have been increasingly eager to adopt the cloud, whether it's banks or insurance companies or agriculture companies. There was a point in time where they were either resistant or were just shrugging their shoulders and say, “Wait, we’ve already got our own data centers. I'm not sure we need that kind of thing,” but now they realize that there is value. What is their path to adoption?

[00:34:59] CQ: Rocky is probably the best answer I've got for you in a nutshell. It requires a fundamental rethinking of your engagement with technology. It requires an understanding of aligning these things strategically. It requires accepting you're probably not going to save any

money for the first three years that you do it, and it requires an awareness on the company's part of things such as, “Well, you’re a 200-year-old company and you’re used to spending capex on your expenses.” If you start shifting to cloud, almost all of that could theoretically wind up being reflected as opex. There are ways around that. That'll affect earnings-per-share. That will affect the visibility of how the market views your business and it could theoretically lead to problems for you. Make sure you're aware of what's happening before you get there.

There's regulatory risk of people doing a click through agreement before signing an enterprise agreement directly with a cloud vendor. There are a bunch of nuances that apply to large established companies. I live in San Francisco. It's somewhat natural based upon the environment I’m in to think of companies as this thing someone started in their garage three years ago and, “It will be sloppy now, and fix it later,” but with a lot of companies migrating to the cloud, the “legacy businesses”, there's a lot more at stake. There’s a lot more risk and they have to be moved more deliberately.

This is not incidentally in any way intended to be a condemnation of those companies. It's just different. Making sure that they address this from a perspective not just economically, but from a business point of view and have very clear outcomes and ways of measuring that as they go through that process is critical. People like to make fun of the lift and shift idea of to just take exactly what you have and you put it in the cloud and then space two is migrate that to take advantage of cloud primitives.

The problem is, is the other approach of re-architect everything is you go, “Okay, now it's in the cloud. It doesn't work the same way.” You've introduced a bunch of regressions and no one has any idea why. Is it your code? Is it the changes to the code? Is it the environment? I mean, there's no silver bullet here, and the answer to almost everything in this space as there are with any complex problem is it depends.

[00:37:02] JM: Amazon has products that cater specifically to these kinds of companies that are – They’ve been around for 50 years, or a hundred years. One example is the virtual tape library or the tape gateway which allows people to move their tape-based backups to the cloud. I thought this was an amusing example that I found in your newsletter at one point. What are some other ways that AWS services are tailored to legacy enterprises?

[00:37:32] CQ: Great question. I put in a feature request back in May of this year, 2018, and they announced it two weeks ago that they now support it. Namely, what I wanted was the ability to upload a file to S3 via SFTP. When I tweeted that at them, I of course wound up with a whole bunch of people yelling at me," FTP is of the past. Just use the S3 API." Great. I appreciate where you're coming from on this. The problem is, is that large banks, the de facto communication pattern that they have between them is generally FTP or SFTP, of GPD encrypted files for transaction logs, and three different jobs now, when I worked in finance, I had to build an instance that waited for FTP files to show up, validate once that was done, that it was not still uploading. Package the entire thing and put it into S3, because you're never going to be able to teach a large bank how to use S3. If you start trying to talk to a banking partner about how S3 works, congratulations, you've opened Pandora's box of compliance and legal requirements, because now they have a lot more questions that you can't back out of. It's much healthier for those environments to meet people where they are.

Now, this service costs a couple hundred bucks a month for an endpoint, and people are screaming about that, but it's not for consumers. It's for large banking institutions like that where engineering time to build out and maintain an EC2 instance running this is an order of magnitude more expensive than just enabling this endpoint. Now, I saw that launch. I said it was terrific, and then two hours later I had a follow-up request, "Okay, now give it a static IP, because otherwise we're going to wind up in a situation where you have to have companies that only update the firewall rules after six weeks of cab approvals now have to do that constantly, and that becomes awful. There are workarounds for it, but I'd like something a bit more out of the box.

[00:39:32] JM: So as AWS is accepted philosophically more and more by these legacy enterprises, like banks and now they can – The banks can FTP the files to S3 and they can get their tape backups moved under the cloud. The legacy enterprises are going to be more and more willing to adopt things, and at Reinvent this year, AWS announced Outposts, which allow for custom AWS hardware. I guess it's basically you order a box from AWS with specific services on it and they give it to you on-prem so you can kind of get AWS functionality out of on-prem devices that Amazon sends you. This seems like a pretty big development. Explain what the implication of Outposts are.

[00:40:21] CQ: Absolutely, but first I want to clarify something you just said, and that we talk about legacy companies like banks, for example. If you take a look at Capital One and their transformation story of being a bank that has gone for purely on-prem to being entirely cloud-driven, they have radically transformed their entire organization.

If you're an on-prem company looking at a digital transformation, you could do a lot worse than to model your transformation after Capital One, and I highly doubt you can do better. They've nailed this. So saying someone is a bank, therefore they're legacy boring and crappy and slow is in many cases not accurate. I don't think that's what you were saying. I just want to make sure that that winds up, clarified.

[00:41:03] JM: Yeah. I need to stop using the term legacy, because I mean it in a categorically nonjudgmental way. It's more like a lovingly phrased agnostic. Just saying like this has been around for a while company, but I need a better word for it. I should just say something different. But, yeah. Okay. So go on.

[00:41:24] CQ: So the question was around AWS Outpost, where they ship you sealed racks starting next year that contain AWS hardware and run AWS services on-prem. I think that this approach is, to be very direct, brilliant. It's a great example of what drives Amazon in the sense of being focused on the needs of their customers. They're meeting customers where they are, and they are effectively extending their APIs and their model of doing business into the on-prem data centers.

I've seen some fairly poor takes on this not just on Twitter, but things like headlines and business insider saying this is, "Amazon's tacit admission that cloud isn't for everyone." My counterargument to that is, they launched a downlink station service for satellites in orbit. If people with satellites in orbit is a large enough addressable market for them to focus on, I promise, people who are intimidated by public cloud absolutely is. I think that that product also goes a long way towards saying without saying, one of the biggest problems you'll see in on-prem environments is that you cheap out on buying hardware, you don't manage it properly and your effective systems management approach is awful.

By dropping these things sealed on-prem into your environment, we get rid of most of that. Now all you're really responsible for here is power and cooling, and we can't do anything about that until next Reinvent where we'll announce AWS Power and Cooling or something. I've no idea if they do that. That is a complete guess speculation. The fact I feel the need to disclaim I have no knowledge of any plans to do such things really should give you a clue as to how far I think Amazon will go towards solving some of these problems.

[00:43:07] JM: They do have windfarms, right?

[00:43:09] CQ: That's a good question. I know they have some in – They say that they have some data centers and regions powered mostly or completely by renewable energy. I don't know offhand if they own the windfarms themselves, if they wind up leasing them from someone else, or they just put a turbine in front of Larry Ellison when he starts mouthing off about things that are not true about AWS and then just generate power from that. It really tends to come down to a few different ways that could be implemented. Their corporate structure is fascinating to me, but I try and stay in my lane with respect to AWS more than amazon.com as a whole.

[00:43:42] JM: So Outposts you could say are a form – Well, this probably wouldn't – They're kind of a form of edge computing. I guess it's computing that is done outside of the cloud, but I think that term edge computing more generally refers to IoT devices or some kind of smart security camera that's sitting outside of the cloud. Maybe it's sitting on a Wi-Fi network at a shipping yard.

[00:44:09] CQ: Or cloud front edge locations, which can now themselves run Lambda as well. Yeah, we're starting to see a bit of an exodus, where not just storage, but compute moving out of the regions it into the edge. I'm curious to see down the road if they wind up doing the same thing with state. If you can wind up not having to go all the way back to a region to store and update state, that begins to be something fascinating as well. We see parts of that now with certain implementations of things like Appsync, but these are still early days for a lot of this.

You're right. The idea of everything that Amazon offers now lives in Virginia, Oregon, and a burning fire somewhere in other parts of Virginia are great, but we're starting to see that rapidly

expanding not just with new regions and availability zone spinning up, but by not – By taking these things and deploying them now into customer sites, there's a great story there.

[00:45:01] JM: Well, tell me more about that. So you mentioned Appsync. That's like a mobile computing thing that they had, or just tell me more about what AWS is doing on the edge.

[00:45:10] CQ: You're more of a software person than I am, I suspect, by a long stretch, but there is the idea that you can now have a mobile app that if you break the network connection, you can still update things in that app, and once it rejoins the network, it will automatically wind up sinking any changes that wind up happening. That sounds like a minor implementation detail, but it starts to wind up pointing to a whole bunch of things. You wind up the reconciliation. You wind up with syncing ability. You wind up with now not having to trust the network, and you're starting to see things that are remote and far-flung going away from effectively dumb clients and into something that has more and more intelligence where you are. That has the effect of reducing latency. It reduces the reliance of the network bottleneck. It has the advantage of, in some cases, all of the processing of your sensitive data winds up happening in your environment, not in theirs.

For things that are regulatory sensitive, for example, stripping out credit card numbers from log data is a quick and easy example. That's something that by not ever having that leave your facility or the device this stuff happens on, you wind up with a much neater story for not just being able to do the right things from a security perspective or compliance, but as compliance always matters, demonstrating that and being able to prove to auditors that this is how this works. So it's addressing an awful lot of not just capability stories from a technology perspective, but as well as addressing higher-level business needs.

[SPONSOR MESSAGE]

[00:46:47] JM: HPE OneView is a foundation for building a software-defined data center. HPE OneView integrates compute, storage and networking resources across your data center and leverages a unified API to enable IT to manage infrastructure as code. Deploy infrastructure faster. Simplify lifecycle maintenance for your servers. Give IT the ability to deliver infrastructure to developers as a service, like the public cloud.

Go to softwareengineeringdaily.com/HPE to learn about how HPE OneView can improve your infrastructure operations. HPE OneView has easy integrations with Terraform, Kubernetes, Docker and more than 30 other infrastructure management tools. HPE OneView was recently named as CRN's Enterprise Software Product of the Year. To learn more about how HPE OneView can help you simplify your hybrid operations, go to softwareengineering daily.com/HPE to learn more and support Software Engineering Daily.

Thanks to HPE for being a sponsor of Software Engineering Daily. We appreciate the support.

[INTERVIEW CONTINUED]

[00:48:10] JM: There is a preponderance of AWS services, and if you go to the dashboard, you will see all these services and newer developers can get intimidated by the amount of options on AWS. So there has been this rise of the cloud providers that are simpler and have easier on boarding. This was early on and you saw this with Heroku, and there're things like Firebase, and now Netlify is getting quite popular. Does AWS have a strategy for appealing to these kinds of users that want a simpler experience?

[00:48:46] CQ: I want to say yes, but I'm not sure how well it's being implemented. When I first started using AWS, I logged into the console and was overwhelmed by the sheer number of services. I'm never going to learn all of these. What do I focus on first? This is incredibly confusing and I have no idea how to go about solving my problem.

There were 12 services. Now there're over 150 and that problem hasn't gotten better. I'm considered to be something of an expert in AWS, largely because two years ago I said I was, and then it turns into a scenario where I am very rapidly continuing to find myself incredibly overwhelmed.

You take a look at services that meet people where they are. Elastic Beanstalk early on was a decent example of this. A better one now is Lightsaid, where you wind up getting an instance, it winds up having load balancers. You could have databases do it in disks, but it's fixed fee. There aren't five dimensions you get build on. It's 5 bucks a month, 10 bucks a month.

Whatever steer you pick. At Reinvent this year, they also announced a transition process where you can take that and convert it into some of the higher-level services like EC2 when you hit that point of evolution. But that is probably one of the best examples of easy onboarding for humans without having to spend six weeks in cloud school first that I could point to.

We're also, and this is going to be somewhat controversial, I suspect, we're seeing that with Lambda. When you understand as a developer what your constraints are on the code you're writing and how it has to behave and you don't have to worry about things like failover, durability, anything of that sort and all that is managed for you, there's an entire class of problem that largely goes away and you just have to worry about writing code. Now there are still constraints and how that code winds up manifesting is still the subject of some debate, but that's the future. That's where we're heading to.

[00:50:35] JM: How does AWS compare to Google Cloud these days?

[00:50:40] CQ: There are a couple of ways to answer that. I would start by saying that Google Cloud is arguably 3 to 5 years ahead of AWS in terms of pure technology. The problem is, is that it is not at all clear to me that Google has ever learned to speak to business. AWS exemplifies meeting customers where they are. Google tends to not understand a few key things.

First, when a customer tries to move to Google and it doesn't work very well, Google takes a look at what they're doing and more or less says, "Okay, the problem here is that your code is written like crap. You should instead write code the way we write it at Google." It turns out that being incredibly condescending to the people who you're hoping to get money from is not a winning sales strategy. There is also the concern, and Google people do yell at me when I bring this up, but we all remember Google Reader, where a widely beloved service was suddenly turned off.

As of the day that we're recording this, which is December 6th, we wound up seeing that they turned off Allo this morning or yesterday. It made the headlines. Google has a history of turning off services that people grow to depend on, and while they do that less in the enterprise space, they all have the word Google in front of them. So people are leery about building their business on the backs of a service that may very well be turned off. With over 150 services that AWS has

offered, going back to 2006, they have never turned off service that had active users. That's something that winds up resonating with serious companies who take their business seriously. When a migration takes three years to execute, you don't want to have to do that again.

So there's something to be said for being able to speak the language of business, and the more I talk to companies that are not tech darlings in San Francisco, the more I realize the second choice for cloud after AWS is going to be Azure, not GCP. Not until Google fundamentally changes how they approach this. I don't know how they do that, because it requires complete shattering and reaffirmation of the culture in some ways, and that is almost impossible. But until that happens, I don't see them growing outside of a very specific customer profile with a few exceptions.

Now, most of the startups are using Google's productivity suite, Gmail and Google Docs and so on.

[00:52:56] CQ: I'm using it myself.

[00:52:58] JM: So does this give Google cloud any advantage or are the cloud productivity tools, do you see them as totally disjoint from the cloud infrastructure?

[00:53:06] CQ: To my understanding, the fact that I pay them five bucks per user per month for this does get lumped into the bucket that contains Google Cloud revenue. So there's that argument to be made. But at this point, it's from a collaboration perspective. It's kind of neat, but I have never yet built anything where I'm working on a data store or I'm working with S3 or whatnot, and, "Ooh! Now I need integrate it with my office suite." That's not really how I tend to operate.

If you start looking at serious businesses as well where they have built entire complex applications that tie into spreadsheets, they're doing that in Excel. They're not doing that in Google Sheets. To some extent, you might see Office 365 having a story here that ties largely into Azure. But I don't see that level of integration. On the G Suite are Google Apps for domains or whatever it is they're calling it this month.

[00:53:55] JM: So you mentioned your take on Azure. It sounds like your belief is that that is going to have the second biggest market share at least in the near future. I think it already does. I guess because of a similar willingness to meet customers where they're at I guess as well as their channel advantage with the pre-existing Microsoft services. Is there anything else that you see as differentiating with the Azure world?

[00:54:24] CQ: I do. There are two things. First, and I do not in any way mean this is an insult. Microsoft has over 40 years of experience apologizing for software failures. They speak the language of business fluently, and you need that ability in the cloud, because that's what things in the cloud do. They fail. As much as you try to build things that are completely bulletproof, nothing ever is. Everything breaks eventually, and being able to explain that in a realistic way without, first, read the SRE book about error budgets and then talk to us is an incredibly valuable skill.

The second thing that I find that Microsoft is doing that's going to absolutely change the landscape and has changed the landscape is, I mentioned a minute or two ago, that culture change is almost impossible. Microsoft has done it. They've gone from a company that I despised in the 90s to one that I deeply admire, and I don't know – People say, "Oh, what's the secret to that?" and I've asked people and they say half and just, "Oh! They fired their CEO and then replaced it with someone good. Cool.! There's more to it than that. I don't believe that one person can drive this. There has to be a collective cultural reckoning, and I can't believe I'm saying this. In 2018, Microsoft is a bit of a darling of the open source world. That is a statement that angry 20-year-old version of me a couple of decades ago would be gassed at and wonder what had happened to me." The world changed. An awareness of that change is something that has absolutely catapulted Microsoft to one of the most admired companies in the world right now.

[00:55:59] JM: Another sizable player in the space is DigitalOcean, and DigitalOcean is, full disclosure, a sponsor. I think they also sponsor of you, but –

[00:56:08] CQ: They are.

[00:56:08] JM: But I will say I think DigitalOcean is a sort of sleeping giant, because what I like about them is they take the alternative path to the other cloud providers in the sense that they're super selective about the services that they reveal or the services that they deploy that they make available to the developers. So it makes for – It does make for this kind of constrained experience that I think AWS might be trying to do with Lightsail. What do you think is the long-term strategy of DigitalOcean?

[00:56:42] CQ: I think that whatever their long-term strategy is, it's very clearly working. I occasionally see Azure in my client accounts. I occasionally see GCP in my client accounts. Again, all of my client accounts have AWS, that that's sort of a bit of selection bias there. But I see DigitalOcean frequently enough that it rounds to all of my clients. There is always something running there, be it a marketing side, a status page, a blog or something else, where they wind up spending something ancillary to the core product.

I don't tend to see a lot of this is the core application that makes us money living in DigitalOcean, but I see an awful lot of other stuff. The reason that I find when I dig into that almost invariably is that it is extremely approachable. You can get up and running within minutes and a matter of clicks. There's no back-and-forth of, "First, set up these 12 foundational services, like IAM and all the rest in order to get using it." It's click, click, done, and you're up and running and being able to build something.

They've been extremely selective in the services they support. They have a block store. They have a managed database. They have a load balancer and they of course have the VM or EC2 equivalent. They recently announced that they're in the process of launching a Kubernetes cluster service, which is cool, good for them. I don't have a problem it looks like that right now, but I'm curious to see what they do with it. But you're right, they've very selective. The product strategy is not yes. They're not launching a bunch of high-end machine learning services to the best of my knowledge. They're not out there building incredible data lake architectures on how to wind up doing incredibly complex queries upon unstructured data that lives in the exabyte range. That's never been what they been about. Its more or less a cloud for human beings.

A lot of their constituents tend to be business side users, not engineering side. So it's easy for those of us who've spent a decade ensconced in the deep dive technical architecture work to sit

here and say, “Yeah, that's what we’re going to wind up. We’re going to just ignore that and instead focus on the bigger, more exciting, flashier things.” You're right, they’re a sleeping giant. The quiet, but they're everywhere. Every time I have dealt with them from a support perspective, from a business perspective or from the perspective of just popping in and seeing what they're up to, it's been a wonderful experience. Again, they are a sponsor of some of the things that I do, but they pay me to include their links, not to say nice things about them personally. This is me being genuine. This is not me being paid for this.

[00:59:10] JM: One subject that we discussed in recent episodes is the idea of open source companies that are competing with AWS. If companies like Elastic that competes with Amazon's Elasticsearch product or you have Redis Labs that compete with Amazon's Hosted Redis, what does an open source project have to do in order to succeed as a product company that might be competing with Amazon's much cheaper, easier to sell hosted product?

[00:59:40] CQ: I would say give up, because an open source project is not a business model. It's a means of development. It's a means of community engagement. It's a way of solving technical challenges, but there's an enormous difference between that and having a viable, functional, healthy business.

A good example is might very well be Elasticsearch on AWS. It's an awesome service. Click, click and you receive it, and it works super well until you try to do anything even slightly off book or complex, at which point it turns into a screaming fire. At that point, you're generally reaching out to Elastic to work with them, and that winds up being sort of the narrative of how the rest of this works.

I think that if your company is solely built around his open source project that you’ve built and your value-add is either a pretty dashboard for it or a consulting assistance series around that, you don't have much of a business to begin with. Without casting aspersions with them, I think on some level, Docker suffered from this. Their entire company was built around a transformative repackaging and branding around an idea, containers, whose time had largely come.

But the best part of what Docker did was given away for free as part of open source. There was no narrative. When I was deep into the Docker world of step seven, now I cut Docker a large check. There was no upside outcome from my position in the universe, and maybe I'm wrong on that, but it felt like they wound up articulating different attempts at business models periodically. They weren't really sure what they were going to do next, and now similar to what we'll see happening to Kubernetes at some point, what container system you use has slipped beneath the waves as well. That's no longer the interesting part of the story. Now it's a question how you orchestrate them.

But last year at reinvent, Dr. Werner Vogels got on stage and had a great slide, "In the future, what does the future look like? The only code you will write his business logic," that I of course Photoshop the crap out of that and made them say all kinds of ridiculous things, like, "What does the future look like? Cats will love Amazon Prime Meow." But his point was well taken, and that people don't want to think about this at a business level.

As you move up the stack with things like Lambda, your code now handles business logic and becomes valuable and important to people making strategic decisions. When you're also paying per invocation in a large microservices environment like Lambda, you can also trace to a high degree of accuracy exactly what it is costing you money where. You trace the capital flow throughout your organization, as Simon Wardley puts it. That is transformative once you get more than trivial amounts of money being spent on Lambda.

[01:02:20] JM: So we're nearing the end of our time. I just want to say you make some really good developer content. So whether it's your newsletter, you also host the Screaming in the Cloud podcast and you have a distinctive voice, which I think is something that's pretty important for at least the content that I want to consume these days, because most of the content I consume these days is written by somebody who I respect and I know what the voice is. I know who I'm listening to, or I'm listening to a podcast of somebody who who's opinion I respect and who's personality is something I can tolerate, or I look forward to tolerating. What's been your experience producing content that developers want to consume?

[01:03:05] CQ: It's sort of a byproduct of the confluence of two happy accidents. The first is that sarcasm is my first language. It's how I grew up. We spoke it at home, and I wind up seeing the

world through a snarky, sarcastic lens. So being able to speak with that voice is refreshing, because not many people do it, and the reason for that is tied to the second, which is because I work on AWS bills, the opportunity for conflicts of interest to arise is massive. I have no partnerships. I am not an AWS partner. I have no partnerships with any vendors in this space and I have no – I'm not one of those consultants who has a single large client that drives most of my business.

I'm no one company is more than 20% of revenue by design. So as a direct result, I am not beholden to anyone else. I'm not one awkward meeting with HR away from not having a job anymore. My personality and my voice were tremendous liabilities to me personally when I was an employee. Now that I'm independent and really don't have a corporate overseer, I become free in a way that I never was before, and I'm taking advantage of that to say what I think. I don't always get it right, but I do occasionally tend to hit the nail on the head. It's one of those areas where that voice is hard to find, but I wouldn't give it up now for anything.

[01:04:26] JM: Corey Quinn, thanks for coming on Software Engineering Daily.

[01:04:29] CQ: Thank you for having me. It's been a pleasure.

[END OF INTERVIEW]

[01:04:35] JM: GoCD is a continuous delivery tool created by ThoughtWorks. It's open source and free to use, and GoCD has all the features you need for continuous delivery. Model your deployment pipelines without installing any plug-ins. Use the value stream map to visualize your end-to-end workflow, and if you use Kubernetes, GoCD is a natural fit to add continuous delivery to your project.

With GoCD running on Kubernetes, you define your build workflow and let GoCD provision and scale your infrastructure on-the-fly. GoCD agents use Kubernetes to scale as needed. Check out gocd.org/sedaily and learn about how you can get started. GoCD was built with the learnings of the ThoughtWorks engineering team who have talked about building the product in previous episodes of Software Engineering Daily, and it's great to see the continued progress on

GoCD with the new Kubernetes integrations. You can check it out for yourself at gocd.org/sedaily.

Thank you so much to ThoughtWorks for being a longtime sponsor of Software Engineering Daily. We are proud to have ThoughtWorks and GoCD as sponsors of the show.

[END]