EPISODE 1273

[INTRODUCTION]

[00:01:49] CQ: Hi there. I'm Corey Quinn, Chief Cloud Economist at The Duckbill Group. I also host two podcasts, The AWS Morning Brief and Screaming in the Cloud, as well as write the last week in AWS newsletter. I'm taking over hosting duties for Software Engineering Daily for this entire week, to take you on a tour of the cloud.

Today, we're starting with AWS. My guest today is currently the Chief Product Officer at Allma, but before this was my colleague here at The Duckbill Group. Pete Cheslock, thank you for joining me.

[00:02:27] PS: Thanks for having me. It is always fun to take some time out of my day, just to chat with you. Who knows that we could have just hit record and just share it with the world?

[00:02:36] CQ: Exactly. one of these days, we'll have to do this with one of our coffee meetups or something and see how that unfolds. You were the answer to an interesting problem I was faced with. If I want to go and talk about an overview of AWS, where do you even start with something like that? It's massive. There are over 200 distinct services. It feels like at this point, it's one of those if you knew about it, you know about it. If you're not in that ecosystem, give up, because you can never get into it at all. Who do you talk to? Do you grab some random AWS employee to talk about what they see from where they're sitting? Do you grab someone from one of their big customers? Instead, I thought I'd go a little bit more unique. That was in the form of you. What were you doing before you joined The Duckbill Group?

[00:03:21] PS: Yeah. That is a good question. I always consider that to be almost, the most amazing shade. I'm going to find someone unique, like you, Pete. I am a star in the sky. Prior to Duckbill Group, I had actually just moved on from a startup, I was at Chaos Search, and was taking some time to figure out what was next.

I got connected with via, actually some Amazon employees. They were looking for someone who had some experience in Amazon, but didn't work for Amazon, or work for really anyone would be ideal, because it turns out that around that time, so this was the fall of 2019, the AWS

and Microsoft have been fighting over a 10 billion dollar contract, that's known as JDI, which stands for joint defense, something or other than I completely forget the name of. They were looking for someone who could basically, help their lawyers understand cloud, which is comical, if you really think about it.

[00:04:24] CQ: It's hard enough to teach engineers how to understand cloud. Now attorneys, you have to explain this in very precise language, where there are terms of art that neither side understands. It sounds incredibly interesting, incredibly difficult to, I guess, communicate through that barrier at almost certainly, deeply NDA'd.

[00:04:42] PS: Yeah. It's very deeply NDA'd. I absolutely have an NDA with Amazon, as you would expect. Actually, there's another level that it's super interesting to talk about is the next level up from there, because having an NDA with AWS As with Amazon in general, that makes sense, because I'm going to be looking at the procurement documentation, all of the research all of the proposals that went to the DOD that the DOD would have reviewed to decide who to go with.

As part of this engagement with these law firms, I would also need to look at Microsoft's proposals and what they submitted to the DOD as part of this. There's another level above a standard non-disclosure agreement with a single company. There's something called a protective order, which is through the federal courts, I believe. This specific court cases through the court, federal court of claims, where they actually – it's a course designed specifically for these contractual disagreements in the procurement process.

I was actually put under and still am, under a protective order, which basically says that I'm one of a pretty small number of people that has access to the documents that are under this protective order, that are related to this procurement process. Of course, I'm obviously, unable to disclose any of those documents for extremely obvious reasons.

[00:06:11] CQ: I will not ask you to disclose any of them. It gave you a very interesting perspective, in that you see it at a very large scale from also a juxtaposition with another hyper-scale competitor in ways that most people generally don't. After that, you came to The Duckbill Group, and frankly, learned a lot about how AWS services fit together, which you've been doing

before, the same way that I do, which is I start with the bill, because if it's not on the bill, it isn't real.

That's the only place in all of AWS, that you can see all of the expensive resources you're running in one place. Otherwise, you are clicking through every different region, every different service. Again, there are over 200 services in something around 20 regions at the moment. It's an awful lot of clicking, and there's no great, just tell me everything that I have running.

[00:07:01] PS: You are absolutely right. The cost and usage report is the only single source of truth that exists. I mean, it's probably more accurate than the console even, is for what I have and where it's at. It's all in one place, too. It's right there. I mean, you have to turn it on first. You're right.

I think what was most fascinating about working on this engagement, it actually – it's still going. There's still legal documents being fired back and forth. My involvement in it is massively reduced. I spent a lot of time in January, February and March of 2020, very heavily working with a couple of law firms that were both on the Amazon side of things, helping them understand the process, understand what is cloud? What is this service? What does it mean, and how does it work? I mean, all those things. It was super fascinating.

I got to tell you, some of these lawyers know more about cloud technology than cloud technologists I've worked with. I mean, very impressive to work with these folks. Then, you would imagine that someone who has this level of knowledge. Again, knowledge that I literally can't share with anyone other than other people under this protective order. You would imagine going into The Duckbill Group, and doing more work with Amazon and cost stuff. Like, "Oh. Well, Pete must know so much already." There's just too much. There's too much, even with this insane level of insight into the inner workings of these cloud vendors, there is still just so much information that exists out there that requires just dedicated ongoing research.

[00:08:37] CQ: It's one of those problems, where my newsletter came from was, I'm trying to track absolutely everything that's going on in the world of AWS, that has some economic impact. It turns out, there's no one blog feed that does that. There are over 40 distinct RSS feeds that AWS publishes, for updates that come out. They come out with thousands of them a year now.

Which ones of those matter, which ones don't? It was basically, straining raw sewage with my teeth. By the time I wrap my head around all of that and got to a point of being able to contextualize this and understand it in a reasonable way, I was 80% of the way there of, well, why don't I just throw some jokes in to keep it interesting? Then on top of that, go ahead and just send it out to some people? Maybe a few people who aren't blood relatives would sign up for this and find it useful.

Four years later, I now have over 26,000 people who wind up getting that newsletter every Monday, where I just make fun of Amazon, because of deep-seated personality problems that I have. Only some of those are a direct result of working with AWS for the past decade, give or take.

[00:09:41] PS: Yeah. I mean, before you started that newsletter, you're right, where would you go for information? It was more piecemeal, out in the market. I mean, obviously, the content exists. You're not necessarily creating content. There's a lot of folks out there that do that already. It's distilled down the firehose of data that's coming. I would almost be curious to see almost – I'm sure this has been done, like a time series graph of just blog posts and content creation coming out of AWS from around the time you started last week in AWS, all the way through now, I would just imagine it has gone up by just magnitudes, orders of magnitude, right? Because of to your point, the services coming out, then additional blog posts around that, and they do so much education in the post more so than before, that it just adds on top of this.

How does one just coming into cloud and just trying to understand some of the basics, like where does one go? There are too many places. I think, yeah, to your point, by creating this newsletter, and thank God, there are the jokes and the snark in it, because otherwise, it would be supremely boring. You got to throw in a little bit of shade as you talk about the next iteration of AWS systems manager, systems manager Redux.

[00:11:05] CQ: Oh, yeah. They love putting repeated words in product names and the rest. The jokes are for me to keep it interesting, because my God, would this be boring otherwise. I still remember though, the first time I fired up AWS and looked at their console, and was just

overwhelmed by the sheer number of services. There were less than 15. There are now over 200, and that problem has never gotten any better.

There are other problems that are getting started with it. If you start with a basic task, I want to spin up a virtual machine, or an instance as AWS calls. They're very picky about their names for things. Okay, I'll do that. Well, I have to configure the networking correctly, which requires a whole bunch of knowledge and learning. I need to make sure that their DNS service, route 53 that I insist on misusing as a database, is configured to point to that thing. You have to worry about how to upload your SSH key through some other additional panel, you have to get the account set up and get your billing information entered. You have to wind up firing up IAM, if you're going to do this even halfway responsibly, so you're not doing everything as the root user. By the time you get up and running and log into that instance for the first time, you've already picked up about a dozen services. You look at the rest of these things and figure, “Oh, my God. That must all be like this, so it's impossible to learn all of it.” No, you can pick up an awful lot of these services in an afternoon, at least to a topical level enough to stand something up and kick the tires on it. You don't need to spend years of your life learning the ins and outs of every service, because no one uses all of these services. They're like, Pokemon. You've got to catch them all somehow. Some people seem to interpret it that way and say, “Oh, yeah. We're running over 80 AWS services ourselves,” like it's some badge of honor, rather than the sign of something deeply and profoundly wrong with what they’ve built.

[00:12:50] PS: Yeah. When I got started in with AWS, it was back in 2009 timeframe. I couldn't tell you how my services ran at the time. The ones we used was basically S3 and EC2. I guess, if you would count EBS as a service, that was really it.

[00:13:07] CQ: Moving data back and forth, and you basically hit all the salient notes of where most of the revenue still comes from today. That may have a bunch of nonsense that sometimes makes sense and sometimes doesn't.

[00:13:17] PS: Yeah, right. I will say, I appreciate how they build interconnectivity between services is a lot easier than I feel it used to be, where, especially around the data infrastructure that exists nowadays. You don't need to actually be an expert, or even really an intermediate

data engineer type person, to use a slew of Amazon services with no code, with really just clicking around a console and get insights from your data.

I mean, you can shove some data at S3 from the console, upload some data. You can go into glue, data brew, explorer and visually see your data and run transformations, again, with no code, and get insights and query that data. I mean, you can do so much now without needing to know how to write code. It really opens up, I feel like, a lot of cloud services to people that traditionally wouldn't really use them. I remember talking with the QuickSight team, while I was at Duckbill. We were testing out QuickSight. QuickSight is almost frustratingly bad.

[00:14:26] CQ: Yeah. For those who are unaware, QuickSight is very similar to Tableau, or Power BI. It's more or less Excel on steroids. It's a data visualization tool. Now, you could be forgiven for not understanding that, because AWS is a company built of a bunch of small teams, but they do have some things in common and one of them is that they are absolutely terrible at telling stories about anything that isn't a baseline building block of computer data center infrastructure. Anything higher up the stack, and they don't know who to talk to, or how to articulate a message that resonates with those people.

[00:14:57] PS: Yeah. You actually said the best line, which they should cut you a check, or you should go trademark this, which is it's Excel on steroids. That's what it is. Their user for Excel is not me. I'm not their ideal user. I know how to use tableau. I know how to use data visualization tools. In the technical operations world, we've been doing it for a long time, visualizing data in various forms.

What QuickSight is built for is the Excel power user that is reaching fundamental limits of Excel, or where they're an Excel power user whose company has moved all of their back-end data into Amazon. Now they need something that is like Excel to point at. Because once you actually understand that that's how QuickSight really is, almost like a powerful visualization layer on Excel functions, then everything changes and you're like, "Oh, I've been using it wrong, and that's why I hate it so much."

[00:15:54] CQ: Part of the problem, too. I also want to be very clear that every service we are talking about here is absolutely real. We're not making services up to have fun at your expense.

Although, let me be very clear. When I do play those games, I make up AWS services that aren't real and don't get called out on … by AWS employees. No one can hold this all in their head anymore. When you look at the list of services and realize you don't know what most of them do, congratulations. There's a support group for that. It's called everyone, and we meet at the bar.

[00:16:28] PS: Honestly, when it comes to these new Amazon services, so I've been in the startup world for too long. I'm now back in the startup world again, at an early stage startup with Allma. At each of these startups I've been at, whenever you go through the process of fundraising, until at least two of my previous startups went through this process on fundraising. The question you will always inevitably get from some VC, less so now, but this was really prevalent five, six years ago, is why can't Amazon do this? You're building a thing. Like, I want to build the next whatever. You'll get the question from an investor saying, “Well, why can't Amazon do this?”
The joke in almost every situation now is that they have already done it, no one uses it, no one knows it's there. Could they create these things? They're so bad at telling the story about who should use it and why and marketing it. They're great builders, but I feel that's where it usually ends.

[00:17:22] CQ: It also feels like, AWS absolutely iterate super quickly. They release things in early stages of development, where other companies would hold it back a while and put more polish on it. That has advantages, because they bring these things out to the world, people start using them, and they learn from their customers, how people are going to use the service. It would not have occurred to them, for example, that I would use text records in their DNS service as a database, because no one sensible would ever do such a thing. But I do.

Oh, they see things like that and they learn more about it and they keep iterating forward in that direction. That's great. I think that's awesome. The problem is, is when they launch a lot of these things, they feel unfinished and more than a little crappy. What people learn something, or they kick the tires on something new and it's bad, they don't go back and revisit that. They don't go and reevaluate those things. Once you learn something, you don't need to go back and relearn it. I think that's a mistake that an awful lot of folks make. We're all subject to it.

It's one of those problems where yeah, a lot of these services that are out there today, EFS, NFS as a service, file system that shows up in their environments, you can attach it to instances, you can attach it to containers, you can attach it to lambda functions for some godforsaken reason. It's pretty good now. When it launched, it was a piece of crap. It was the only way that you could get throughput was by how much data was stored on it.

It was a recommendation that if performance wasn't good enough, just put some big empty files onto the file system and just keep them there. I'm looking at this going, “What kind of clown shoes answer is this?” That got fixed and a lot of things got fixed. It went from laughable joke to an incredibly solid service that I recommend for some use cases. I used it a couple times myself. Things evolve, and they don't tend to get worse with time.

[00:19:14] PS: Yeah. It's honestly something I'm really impressed with Amazon still able to do today, by continually shipping at the velocity they do, at the scale that they operate at. I mean, that is really fascinating to continue to see. To your point, yeah, I mean, they come out with these features as a way of getting feedback. The sad thing is, is that when a lot of these features come out, their audience is larger than any startups audience likely will ever become. They come out day one with a huge amount of people using it and running into problems and edge cases and issues.

They call these paper cut type improvements, where you want to focus on improving paper cuts, because those are the things that grind on your current users of the product, so you want to really resolve those.

Now, most companies should ship products faster than they do. Actually, this is something I see a lot in the marketplace. It's honestly, mostly from software engineers who start companies, they feel they need to have the whole product complete and ready to go before they can start their business. You can't have a business, unless you've got the whole product ready to go from end to end. It's just not the case. I mean, you need to get an iteration out there. You need to collect feedback, because you don't understand your user yet. Just because you're building a thing that you used, or you think you would use, great, you've solved for yourself. Now you got to get out there and get that feedback.

I think that's where Amazon does do a really good job. I don't know, on the feedback side of things. I mean, in all my time as an Amazon user, the only time I really had an opportunity to share feedback of a service I was using was at my time at Duckbill. I don't know what that says for the fact that I was using Amazon for 10 years prior to Duckbill, and had plenty of feedback I would have been happy to give, but very rarely had a, I guess, a vehicle to do that. It's almost, things are being lost in translation. I'm not sure how or why.

[00:21:10] CQ: I mostly found that just complaining about services on Twitter, and being hilarious in the process is a good way to get attention. Not always good kinds of attention, but it gets attention, nonetheless. That, on some level, I think formed the basis of some of the relationships that we've built with AWS over the four years that we've been in business. I also want to point out that despite having a bunch of services, most of which have terrible names, and being shipped in very limited form before they evolve into something that a human being might consider using, one other thing makes AWS borderline unique in the tech space, and that is that they do not deprecate things ever.

I know that sounds a really weird thing. Every service they have ever launched so far in AWS is still available. They don't talk about them very much. SimpleDB is not a service that most people should be using. They don't advertise it at all, but it's still there if you want it. If you're going for a NoSQL database, their approaches go with DynamoDB in most cases, because it improves on SimpleDB in almost every way. Great.

Assuming you have a workload that you built, targeting SimpleDB, you can still run it. They view APIs as promises. That can mean some good things, and also some terrible things. You still have to work in some cases with weird XML stuff in soap calls for some of the more legacy S3 endpoints, but they're still there. All of the bugs that were horrible back then are still there as well.

[00:22:40] PS: Yeah. I really wonder sometimes, I wonder if Amazon needs to add product managers to the account teams. Like, how can they start hearing more of this? To your point, yelling loudly on Twitter with some good jokes, it's a way to do it, sure.

[00:23:01] CQ: It has some scaling problems.

[00:23:03] PS: Yeah. I mean, I've worked at companies spending millions of dollars on Amazon, I usually would have an account manager. That poor account manager usually had, I don't know, hundreds of other accounts. They were unbelievably overworked.

[00:23:15] CQ: It felt like, every six months like clockwork, they would introduce you to your new account manager and they would rotate through. It was either they get promoted to work with bigger accounts, or they don't work out super well and go somewhere else. The turnover in account managers from a customer perspective has been borderline ridiculous.

[00:23:32] PS: Yeah. The other challenge, too, yeah, around that, which is you're trying to create a relationship, a good line of communication with your account manager, so that you can share feedback. By the time you do, they're flipped out to another one and it gets lost there. Amazon optimizes for everything being hyper-optimized. I'm sure that they would love for you to send your feedback into a support ticket, or a forum, or a void where you shove your thoughts and feelings. Again, the user experience of that feedback doesn't feel that great.

It's like, you're trying to send feedback, because it's like, “Listen, am I using it wrong? Do I not understand something? Is there a document?” If I am having some problem, there's a breakdown somewhere on the Amazon side. Either there's a document that's not easy to find, there's a document that doesn't exist, there's a bug that I found. I mean, there's so many things that are probably need to be resolved that by shouting into the void doesn't maybe – that doesn't really solve my problem, other than maybe feel good at the moment.

[00:24:36] CQ: Yeah. It's a challenge, because how do you scale communication when I feel increasingly a disturbingly large portion of my job is introducing people who work at Amazon to one another. I know it sounds weird, but we're here to talk about the technology, but I think it's either impossible or irresponsible to work extensively with any cloud provider at all, where you don't understand their culture. Because companies ship their culture, whether they want to or not. Amazon is famous for its two pizza teams, where you have small teams doing independent things. That makes perfect sense when you look at the console and realize that some service consoles are completely different than other services consoles. It's like, they aren't allowed to talk to each other.

Someone took the NDA to mean, “Oh, don't tell the cloud formation team that this is coming out, because otherwise, they might be able to support it at launch.” It’s very inconsistent. I find myself constantly tripping over seams between different service teams. So much of my time is introducing Amazon employees to one another.

I wish that weren't true, but it is. Everywhere you have these problems that are holistic throughout the entire system, the experience is terrible. You look at things that have to touch everything, like their tagging system, or the console itself, or the billing system, or the status page. Anything that has to touch all of the different groups is generally pretty haphazard, because small, independent teams, almost a microservices culture is an advice there. They don't have a culture of czars, of someone who can be some SVP of nonsense, who goes around and says, “Nope. I am saying we are fixing this this quarter. The end. Full stop.” Oh, you're going to whine and complain, great. You can either fix it or work somewhere else. That's not their culture at all. It shows.

Now, if that were their culture, Amazon would almost certainly be a very different company. The thing that makes you rock, also inherently, makes you suck. Your greatest strength is very often, your greatest weakness. This is a great microcosm of that.

[00:26:32] PS: Yeah. I mean, their culture is what allows them to operate at this scale and to do it in this way. Now again, they didn't get here overnight. I mean, the formation of what is now AWS, started in 2004, probably earlier. I mean, the Dynamo paper around there. You've got the work in South Africa, around EC2. This has been going for a really long time. I think, the one adage they do – I mean, they’re so data driven. They're data driven on all of their decisions. I always feel like, the problem with some of those decisions are, are they misinterpreting that data?

If you're a product manager for some product, and your next promotion is going to be based on a series of 10 metrics that have to do with user adoption, you're probably not going to work on paper cut type bugs, if you can just shove a feature down the pipe and get more users to use it, to get your promotion ,to then move on somewhere else. I don't know. I've never worked at a company the size of Amazon. I don't know how the internal politics work. I would imagine, it's

probably a little bit true there. There's probably a little bit of truth in that. Gaming the metrics right.

[00:27:41] CQ: Yeah, there's a mess. I don't know what it takes to build a company at that scale, or at that size. I really don't. I think that it becomes something that is inherently, I guess, all about trade-offs and constraints in different ways. They're a 1.3 million-person company now. At that point, you have no idea who's working there, what orgs there are, no one has even the org chart in their head, let alone anything more involved than that. It's just a different scale. I am sympathetic to it. They're also creeping up on a 2 trillion-dollar market cap. I am sympathetic, but only to a point.

Cool. I pay them in many cases, far more money than I would like, as does everyone who uses AWS. I think that's something that gets lost is well, we can't fix this thing because it's hard. No, I get that. That's fine. I'm not paying you a phone number every month, just so you can only fix the easy problems. Get back to work. I know that sounds a little harsh and unkind, but it is accurate.

[00:28:38] PS: I mean, if you had a type of workload that had, I don't know, portability to it, and maybe your scale wasn't enough yet. I mean, even if you had a large workload on Amazon, and you feel you're not being treated well by them, it would be probably supremely stupid to actually migrate to a different vendor. There are other vendors that will happily take your data. They'll happily bring you over at any time. Not to say again, that that's a good move and it should be more than just like, my account manager pissed me off last time we negotiate our EDP, so I'm going to move over to Oracle. They're known for their kind and friendly renewal processes, right?

[00:29:20] CQ: Absolutely. It's a hard problem. I don't know what the answer is. I don't know how to fix a lot of these organizational dynamic issues. I just know that when I see that they've launched a new service, it's time for me to go and kick the tires and see what happens. Then, let's see what happens. It breaks. It's, oh, this is a bad user experience. I talk about it and people show up. Tell me more. They're genuinely interested. This is not just trying to get me to shut up and go away, because honestly, I won't do that anyway and they know that by now.

Instead, it's a, we hadn't considered this use case. Who would use a DNS service as a database? That's monstrous. Tell me more about what problems you're having when you try to do this monstrous thing. Maybe there's a better answer, like literally anything else. Okay. Then we start diving into it. In many cases, you have the problem, especially endemic to small teams, where they don't talk externally enough to customers, in that when they're building a service, they are too close to the service itself.

One of the last things that gets done is and now we build the console interface for it, because a lot of folks are going to talk to it just by API, as you "should." Okay, but the people who built the console and talk about this service are way too close to it. They don't understand what it's like to see this thing as it is now, without having the context of how it came to be, what the service does and the rest. A lot of that is left unsaid. For a while, that was okay for some services, but not anymore.

[00:30:46] PS: You mentioned this before about deprecation and removing old services and how they just really don't do it. I mean, I'm sure you we can count on probably one hand, the number of things that they've probably stopped doing. They're usually some sort of like, "We're going to not support this SSL library anymore." Or something that is more egregiously, probably security-focused.

[00:31:05] CQ: We're no longer going to let you deploy new lambda functions with an end of life version of the language that we set the runtime to, and we're enforcing this change two years after the language was end of life.

[00:31:16] PS: Yeah. There is really small number of these times to do it, again, from this, again, from the company, that you could still run an M1 medium. The first instance type that existed, those are still running out there. Think about that for a second. I think, that's actually more fascinating than SimpleDB. Because SimpleDB is just an API that you could run. You could run, I don't know, rebuild the back-end. I'm sure they've rebuilt that back-end a bunch of times.

M1. That M1, I think it was the M1 medium, I'm thinking of is right, was that first instance. That maps to a box. I don't know if it's under Verner's desk or something, like a Dell 2950 or

whatever. That's a approaching, what is this now? 10-plus 12-year-old “cloud resource.” Some poor person has to manage that. There's got to be technical debt associated with it. The question that I really have, as Amazon is so fast to come up with a lot of new stuff, they do fix things over time, sure. Maybe not at the speed we really want. We all know that technical debt can really have long-term implications to company's growth. If you're continually dealing with technical debt, how has Amazon not hit that yet?

I mean, are they just throwing bodies at the problem? How is it possible, they are able to both continually ship at this insanely high velocity new features, while continually maintaining things that have been deployed now for – I mean, we're approaching almost 20 years that some of these things have been deployed. Am I doing that math right?

[00:33:05] CQ: 15, I believe. Yeah, you're close enough to it.

[00:33:08] PS: Yeah. It's like, we're dealing with things that have been around for this long. How do they keep from the weight of technical debt, crushing them? I think, that's probably the most fascinating part of their no deprecation story is again, how they stay ahead of that, because that's a challenge, right? Everyone talks about technical debt being eventually, hopefully not a balloon payment that you have to pay and hopefully, you can pay it down over time. Maybe they're just doing that and maybe they're just being very diligent on those payments. That's the most fascinating part of the whole no deprecation item for me.

[00:33:39] CQ: One story that has made it into the public sphere is back in 2017, they took an outage of the US East one region's S3 installation for hours. It was the S3 apocalypse and half the internet was broken that day. Okay, great. There was a whole series of root cause analyses that are interesting fodder for other conversations. What they did is a couple years later, they got on stage and talked about it, how they had completely rebuilt S3 from the ground up. It was now over 230 microservices that were powering this thing. This entire transformation was done in a user invisible way. At no point during that transformation to complete refactor and rewrite, did the service go on available, did people see problems with data integrity and the rest. There was none of that.

[00:34:30] PS: Yeah, the S3 story. There's actually one even earlier than that. When I was at startup in 2009-2010 timeframe called Sonian. No one's really ever heard of it. It's fine. We were very early user. At one point, we were definitely one of the largest S3 users. We're definitely one of the largest EBS users. We're just that early. The S3 story is fascinating, because when we were using S3 for basic back-end for email archiving, back in 2009 timeframe, we were storing a lot of data there, we were finding a lot of edge cases.

I mean, I remember back then, you really did have to create UUIDs for your prefixes, if you wanted to get any scale because of how that architected it. That's long gone. You don't need that anymore. They've abstracted that away. I remember talking to S3 engineers, we had a really good direct line of communication with them at that point. It was probably the closest I've ever been to an Amazon team and probably the last time I was ever that close, was when we were that early. They told a similar story, again in that 2010-ish timeframe, that they had rebuilt major components of S3 four times a year, I think was something they had said, If I'm even remembering that right. It was something to that effect.

Multiple times a year, they had done complete rebuild of the infrastructure underlying the APIs. I remember nowadays, you hear that it's not as interesting, because obviously, it's been done. 10 years, 11 years ago, hearing that for the first time, when maybe – my experience prior to that was running exchange servers on sands and being blown away that you could lose network interfaces, and you could lose power supplies and stuff and the sand is going to keep running. Here, there's this whole massive infrastructure behind this API that they are completely rip and replacing, for lack of a better term. Just fascinating to think about, even back then.

[00:36:21] CQ: What I think is overlooked, especially for those of us who started our careers in data center environments, is just how much of that work no longer exists. There are no 3 a.m. panic drives to the data center, because something has broken. I haven't had to think about hard drives failing in years. In a cloud environment, a hard drive failure, at worst, looks like a very brief latency spike on what is something that you're working on. That's it. You have to be looking for it and know what it is. That's all that it is. It's just purely under the hood, API-driven magic, for lack of a better term.

[00:36:54] PS: No, let's talk about that real quick. That is one of the most, I guess, I don't know if people even think about this, but EBS is by far, one of the most amazing services that I really don't know if people give credit for. Maybe it's just because of our legacy thought process around what a hard drive is.

Rewind 10 years ago, or so, they had EBS. They would say, attach a hard drive to your virtual machine. Those were the terminology that we're using. Sure, it made sense. You're like, sure, I've got – I coming from the VMware world, and I have a hard drive on the host, I'm going to connect to this guest, or a virtual hard drive on the host, I'm going to connect to the guest, whatever. That was a similar thought process. There's always ways around, I think, where you could spin up a volume and you had to write to the volume to get full access to the IOs. I think you mentioned that earlier.

Fast forward to now. I've got some EBS attached to my server. There's all kinds of EBS now. There's GP2, GP3, IO volumes. I mean, there's an EBS volume for probably almost every workload, dare I say every workload. What's amazing is how you now have this ability to dynamically change the underlying of your server without downtime, without restarting services, restarting servers anymore.

You can say, “Wow, this volume is over-provisioned. I'm going to downsize it to another one.” Or, “Oh, God. I didn't give my EBS volume. I'm at 99% on my Cassandra cluster, and I can't move this data off of it and the data won't stop coming in.” You just add more, extend the volume. I mean, that is such a unbelievable thought that we can do that now. Back to your point of the data center world of how would we do it back then? We would have to, I don't know, shove more hard drives in the sand and hope there was room or something. I mean, it's just quickly move the data off, delete stuff, hoping that you can do it. It's so fascinating that yeah, click button, make bigger.

[00:39:00] CQ: Taking the object store story as well. Everyone likes to argue with me on this one, but I'm going to play this game. It has infinite storage.

[00:39:07] PS: They can buy hard drives faster than you can put data on it.

[00:39:12] CQ: I'm also willing to bet that they can buy more hard drives than I can afford to fill. That is just a general direction I'm taking things in. I'm probably right on that point.

[00:39:24] PS: Look, I mean, they're dealing at a level of scale with S3 that has not occurred before. I mean, it's so mind-boggling. I'd love to spend a day with the S3 team in a room with no recording, where they can just talk about that scale, because it must be unfathomable. I would say, when S3 first came out, which was – and I think there's a lot of debate out there, right, Corey? What was the first service, was it S3? Was it SQS? I always thought it was S3, but I feel that goes back and forth.

[00:39:58] CQ: It depends on who you're talking to. The first service in beta was SQS, simple queuing service. The first service in general availability was S3. If you get the leaders of both of those teams in a room, there will be an argument and you get to watch the tennis match, go back and forth and just sit there and enjoy the moment if that ever happens to you.

[00:40:16] PS: Oh, I got to remember that one. That's a good one for the future, if I ever run into that situation. I remember what S3 came out for the first time, hearing about it, and I didn't get it. It just didn't make sense to me. I guess, my brain couldn't compute that term.

[00:40:34] CQ: It's not a file. It's not a blog. What the hell is an object store and why would I want one of those?

[00:40:39] PS: I had a sand, I had file stores, right? I had a file server for my files. It was a file share. I had lunz on my sand that I would connect to exchange servers. I had block store. I understood that concept files. Object, I could not wrap my brain around it for the longest time. It was again, to your point, I think, you said earlier, they're really bad at telling stories. That I think, was their first scenario of them being bad at telling stories, because the people who really got it, like this founder, friend of mine, who started Sonian, he saw S3 as like, “Oh, wow. This is a game-changer thing. Here's all the reasons why.” I know a lot of other people out there did as well. Obviously, a lot of those people went on to build some pretty awesome things. Yeah, it's so funny looking back. I still remember that to this day, hearing about it and going like, “What is an object? What does that mean?” Yeah, it's fascinating to see how far we've come.

[00:41:37] CQ: It really is. I can forgive you and other small companies looking at this thing. I don't get it. What's the point? That's fine. The thing that surprises me, and the reason that we're talking about AWS as the 800-pound gorilla in the space, is because for almost five full years after they launched, there was no other competition. People made fun of it. Microsoft was basically asleep at the wheel. Google was too busy building and then turning off Google Reader, whatever it is they're up to. That's all that we really saw.

By the time that people realized, “Hey, there's money in them var clouds,” Amazon had gotten so far down the path of being the default/only choice that everyone else was having to relearn the lessons. They'd spent five years learning operationally by getting it wrong and learning from those mistakes. Everyone else was coming from behind, had an awful challenge to beat. It turns out, the cloud services are super sticky.

[00:42:36] PS: Yeah, absolutely. Once you're in, moving off is not something that you're really likely to do. Moving data. It's just too expensive. Even if you could move workloads, sure. There was the dream of Kubernetes, multi-cloud, single-API, workloads across multiple clouds. No one's really doing that yet. If you are, please do a conference talk on it.I think everyone would want to see it.

The fact that people don't even do conference talks about that solution proves to me no one's even really doing it, because that's a hard problem. Even if you could easily do that, easily run a workload in any cloud anywhere, you still have a data problem. You have a data and a network problem. That's why places like AWS, they make the data so easy, the data in S3 as that central world around all these other services, because the more you put into S3, the less likely you're going to leave AWS. I almost believe that that's one of their signals of knowing people are locked in there. It’s like, the more data you have, are you going to move petabytes, are you going to move exabytes out of S3? Unlikely. You're just going to leave your workloads there.

[00:43:48] CQ: Some people do, and they can, but it's atypical. Usually, it's one of those, that is where data goes, because we need it. Most of it is never accessed again. Think of how many pictures people take and store in various services that are using S3 as their storage for that, and then never actually access those things again. How many pictures of receipts have you taken, where you take the picture, you access it once and then it sits there forever and rots? Because

you don't need to know what your dinner receipt was back in 2014. They can't get rid of it, because you actually might need to present that at some point in case of an audit or something. How do you solve for those problems? Well, you wind up finding ways to do this at scale massively. So much of that is almost certainly deadstock/data that will never be touched again. God help them if one day it goes on available.

[00:44:34] PS: Yeah. I mean, that was the dream of the startup I was that with Sonian, email archiving. You're never going to probably need that email again, until you need it. When you need it, it's probably because you have lawyers asking you for it, right? Having a place that you can put an object, like an email or a bundle of emails and not have to think about it for a really long time is so compelling. Who wouldn't want something like that? Just dump your data here and maybe I'll do something with it later. Maybe some product manager comes along, he's like, "Oh, God. We could we could do some analytics on all this data we have." Because it's all there, and now you can do stuff with it.

[00:45:12] CQ: I'm really looking forward to seeing what the future cloud holds. The challenge is though, is that there are over 50 billion dollars a year now in run rate. That is an awful lot of revenue pouring in there. They got there by reaching out specifically to developers and infrastructure people to wind up driving cloud adoption. The next 50 billion is not going to come from those folks. It's going to come from a lot of the blue-chip enterprise style companies that are migrating in various directions. That's a challenge that historically, AWS has done a relatively poor job of, learning to speak to those businesses in ways that resonate. Talking to the IT ops person, who is not going to learn cloud formation. They're going to do things in the console, for example, is a bit of a stretch for them in many cultural respects.

You can't do bottom-up adoption the same way anymore, because it turns out that neither developers, nor operations folks in IT are empowered to sign a 50 million-dollar cloud deal the second time, because the first time they did it, they were fired and prosecuted.

[00:46:13] PS: Yeah. I mean, Amazon, they're not known for that. They're not known for being able to speak to what we'll call the old guard enterprise. I mean, there's a lot of data centers still out there. There's a lot of workloads that are still in those data centers. You have companies, like Oracle and Microsoft, that are honestly, just way better at telling that story to those people.

[00:46:38] CQ: Everyone thinks I'm kidding when I say this about Microsoft, but the honest truth is, is that they have 40 years of experience apologizing for software failures. That is exactly what you need to do in the cloud, because things break all the time and at scale. That is what happens, and you never get away from that. You'd best be able to articulate that in a language, in a context that your angry customers understand and appreciate. Microsoft is the undisputed expert in that space.

[00:47:05] PS: Yeah. I mean, Microsoft has the advantage in the enterprise. They've been in the enterprise for 30-plus years. It's so easy for them. They're already there with exchange server and all the other components that tie into it with Active Directory, they know how to talk to those IT folks, and they're already involved in those engagements. Whereas, Amazon is not. Amazon is not selling an email service to these companies. They're not selling domain controller services and exchange services and file services and SharePoint.

Microsoft is already there. They already have their foot in the door. I feel like, Amazon has a big hill to climb in order to even meet them at their level. They have to go way above to get to those old guard IT ops folks that are dealing with exchange servers, and SharePoint servers and file shares and things like that. Sure, Amazon has a whole slew of things that you can use, that can replicate that service. If I'm an exchange admin, I want hosted exchange. I'm going to go to Microsoft to get my hosted exchange. I wouldn't even think of Amazon for that service, even though they could probably provide a pretty similar offering.

[00:48:14] CQ: One of the biggest problems now is there are 15 different services that provide access to that exact model, which is the best. You wind up with analysis paralysis. Honestly, the one that you pick doesn't usually matter all that much. Pick one and move forward with it. Worst case, you can always back out and try again.

[00:48:31] PS: Yeah. I mean, I think if anything, like most enterprises, they're just going to use them all. You talk to an enterprise like, “Oh, what do you use for monitoring?” Everything. That's what they do. It's far more likely that these large enterprises will truly be multi-cloud. Maybe not single workload across multiple clouds, but they will probably consume the best of each service from each vendor. It's just the nature of enterprise. They will just eventually use everything, even

to come back to what we started on with the JDI contract. JDI was to consolidate the offering, because the DOD currently uses everything.

Of course, the DOD already uses Amazon stuff. Of course, they already use Microsoft stuff. These companies have been around forever. This whole project was to consolidate and they can't even get that through. What hope does the mere mortal enterprise have?

[00:49:23] CQ: Yeah. We don't know. It's one of those areas where there's always something you should have known, but weren't aware of, that solves a problem for you. Even now, I learn about intricate capability stories of common services all the time. Where wow, I really could have benefited from knowing that yesterday. You always find out about these things right after you really could have used them. Ever notice that?

[00:49:46] PS: Oh, yeah. That's the worst. Go off and build a thing, because it doesn't exist currently. Then you get done and someone's like, “Hey, isn't that just like this such and such service?” You're like, “Son of a.”

[00:49:58] CQ: There's no easy way around it. Pete, thanks for taking the time to I guess, reminisce about the joy of AWS and its associated nonsense. I'll be here all week. If people want to hear more about you, or angrily argue with you about what we've just talked about, where can they find you?

[00:50:15] PS: Yeah. You can find me on Twitter, @PeteCheslock. You can check out what I'm building over at Allma at allma.io. A-L-L-M-A. As one of my friends said, all my friends love Allma.

[00:50:29] CQ: Fantastic. This end today's tour of the cloud, but I'll be here all week. If you enjoyed this podcast, follow me @quinnypig on Twitter. Head over to last week in the aws.com and subscribe. To hear more from me on my podcasts, AWS Morning Brief and Screaming in the Cloud, as well as my newsletter, Last Week in AWS.

[END]