Brian Gracely (00:01.39)
Good morning, good evening, wherever you are and welcome back to the Cloudcast. We are coming to you live from the massive Cloudcast studios here in Raleigh, North Carolina, getting close to the end of June, 2024, another weekend perspective. Hope everybody is doing well. And something I've been thinking about quite a bit lately is, you know, kind of this, this idea that Benair and I have been talking about a little bit, which is the idea that, you know, we are kind of getting to the end of, you know, what I'll call sort of the, the cloud 1 .0 era, right? So the era in which,

we are talking about cloud all the time, we're watching the formation of public cloud, we're watching the formation of all these technologies that made up cloud computing. And it's not that we're at the end of something as if we tend to do in technology where we go, well, it's the end of something, so it just stops. It's more along the lines of something Aaron and I always talk about, which is nothing really ever ends in the IT realm, we just sort of add onto it. And so as we're thinking about...

different eras we're now starting to talk about, you know, kind of moving into this, this more AI centric era. And we'll see if that actually pans out or not. But, you know, I've been thinking a lot about, you know, what we learned from that space and specifically kind of thinking about, you know, all the technologies that have been sort of centered around the CNC after the cloud native computing foundation. So the idea of all of these open source technologies that have been, you know, driving a lot of this, you know, sort of cloud cloud one .o era.

as well as a lot of things that have been driving, you know, what's happening in the CNCF in terms of, you know, what originated with Kubernetes and evolved with containers, but just, you know, literally hundreds of projects which now are in there. And it kind of got me thinking about this phenomenon that we had in which everything becomes, in order for so many things to become successful that have kind of emerged out of the cloud 1 .0 era, you know, there's been this sort of distinct line between the things that,

solved a very specific problem or were in a very specific domain. So it was in compute or storage or networking or security or whatever it might be. And then the sort of second tier of things or, you know, overlay tier of things that all in order for them to be successful required some sort of overlap of stuff. So the DevOps, DevSecOps, FinOps, PlatformOps, MLOps, DevSec Platform, FinOps, you know, all these things that in order for some technology to be successful,

Brian Gracely (02:27.47)
it was going to require somebody to learn multiple domains or do things across multiple domains, or, you know, kind of force this team of people or multiple sets of people to kind of collaborate together in a way that, you know, didn't necessarily jump out as being, you know, driven or motivated by, by those people. It wasn't something that they would have naturally wanted to do, but this, this technology came along and in order for me to successful, it kind of required, you know,

these multiple types of teams and groups that hadn't necessarily worked together. So what I want to kind of dive into in today's show, and we'll do it after the break, is kind of dive into where are we with the success or lack of success with these sort of DevSec, FinOps, PlatformOps, MLOps, SecOps, you know, technologies, and which ones kind of over the cloud era really were successful because they allowed individual teams to say, well, that's what I want. Just give me that thing.

And I don't really want to think about that. And maybe somehow something else sort of took care of it. What I kind of want to dig into is, can we use any of those learnings from the Cloud 1 .0 era to sort of give us some sense as new things come out over the next two years, three years, five years, whatever it might be, to determine whether we should be chasing after those things the same way that we chased after everything the last 10 years, or if we will be sort of smart enough and learn some lessons to say, you know,

we've tried to do that before, that pattern doesn't necessarily work itself out because of maybe more people things than technology things. So we'll dive into that after the break.

Brian Gracely (00:01.71)
Welcome back to the Cloudcast. I'm your host, Brian Gracely. And as we mentioned at the top of the show, we are going to sort of dive into what, as I think about how successful cloud has been over the last 10 years or so, let's call it, and the things that have been successful and the things that have emerged but haven't necessarily gotten as much traction as we initially thought when we purely looked at them from a technology perspective. And so I want to dive into that.

somewhat from a historical perspective. I don't want to make this all about, you have to always think about the past. But there are things that the past will teach us about the future or potentially could teach us about the future, both from a good perspective and a challenging perspective. And so I thought what we'd do is sort of dive into that a little bit. So as we think about what's been successful with cloud over the last decade or so, to a certain extent,

A lot of the success of cloud was originally because IT organizations had been incentivized, had been organized around stability, around things being highly available, 99 .999 % uptime. We measured everything in nines, but we didn't necessarily measure the ability to be.

agile, the ability to do new things, the ability to work environments that change, we really kind of measured everything in terms of stability. And so when we look back at cloud, you know, the initial, you know, capabilities of cloud and the initial aura of cloud was number one, here are these distinct, kind of distinct, discrete building blocks, if you will. So here's compute, here's types of storage, here's

You know, you know, the ability to network those things, maybe a DNS, you know, whatever it might be, but, you know, sort of individual distinct things so that, you know, the people that were frustrated with it could say, look, this is the things that I want to build. I don't necessarily want all

of this other corporate stuff, you know, things that might hold me back. but I'd like to be able to sort of go and kind of pick and choose the things I want. And I want to be able to do that on a, on an on -demand basis. So that was the first piece of cloud that was interesting.

Brian Gracely (02:25.102)
The second piece was the idea that the billing for it would be, you know, the cost model for it was going to be, you know, something that didn't require huge, huge capital investment, right? It allowed you to, you know, get started right away with some payment method, whatever it might be. It allowed you to do it for an hour or a day or a month or a year, whatever you might want to do without requiring you to plan three years in advance, five years in advance, those types of things.

And so the initial sort of things that came out of cloud were the idea that, you know, I, I can go and get the things that I want. I can pay for them right away. And I can pay for them in a way that aligns to whatever my project might be. My project might be short -term, might be long -term, might be about stability, but it might also be about experimentation. And so we offered, you know, kind of all this optionality into a process that had very much been.

you know, very, very linear and very distinct, right? You had to know what you were going to do three to five years in advance, you know, whether you're buying storage, compute, networking, software, whatever it might be. And you typically had to do some sort of outlay, you know, capital outlay, if you will, oftentimes upfront, you know, regardless of what you were trying to do. And if new things came along, you...

might ask for them, but they hadn't been through the approval process. So that was kind of the original reason Cloud came along and was successful. And so it allowed individuals or small groups to sort of go, this is what I want. This is what I'm willing to do. This is the thing that I'm looking to trade off for to allow the Cloud behind the scenes to do some things for me. And over time, that early success led to new things coming along. So how do I manage?

a whole bunch of compute at scale, maybe I need automation. How am I going to deal with known or unknown failures within the cloud? And so you started getting into people experimenting with how they did CI, CD, and how they started saying, OK, how do we provide monitoring back to individuals as opposed to always having to go through a trouble ticket? All those types of things kind of came along. And so if you think about the things that were successful with cloud, they

Brian Gracely (04:45.142)
enabled individuals or individual tasks or individual functions to be somewhat isolated. And then over time, what we started doing was we started saying, well, in order for this stuff to work at scale, in order for it to work in bigger things, we started saying that, you know, well, this team and that team or this distinct function and that distinct function have to sort of blur together. They have to work together. They can't be.

sort of different and isolated and by themselves. So I use this, we often have here sort of swim lane analogies. If you went back two decades, 15 years ago, kind of pre -cloud, if you will, there's a reason why you had distinct companies and technologies that were for databases, for networking, for storage, for security, for operating systems.

for runtimes and middleware, you had all these distinct things because companies had distinct individuals and people with distinct skill sets who managed those things. And again, that was a fine process for when the organization was essentially measured and incentivized around primarily sort of stability, not thinking about the number of deployments per day or even the number of deployments per.

weaker month, but like we're going to deploy once a year, we will coordinate all the things that have to happen for that one, you know, weekend or whatever it was. And we start to come along and we, we, we go, okay, well, the world is, is moving faster. the demands on technology are changing. we need to start building some technologies that enable, you know, things to move faster and things to change more frequently things to.

to deal with failures that won't take out the entire system. So all good thought processes, right? Things around, how do we design around failure? How do we test in production, right? The sort of charity majors, observability approach to things. How do we deploy more than one time in a month or in a day or in a week? Kind of continuous integration, continuous deployment. All those things sort of came along.

Brian Gracely (07:01.998)
And if there's one thing that sort of sticks out as to where some of those things have struggled, it's where the technology that came along and said, this is the band aid for this. This is the new approach to this. This is the new tool that's needed. And in some cases, it was a distinct tool. In some cases, it was a band aid. In some cases, it was some combination of those things. They all seem to make an assumption that...

In some cases, multiple teams would work together and maybe those work out. But there was also many times an assumption that like one individual team or one individual person was going to have to, you know, kind of have skills and responsibility that straddled multiple groups, right? So, you know, if we think about the sort of DevOps idea, the idea of, you know, you have groups of people in which the line between

you own the application and you sort of build applications and you are the group that's responsible for deploying those applications and operating those applications. You know, those were kind of the initial teams that, you know, you had a lot of variability in there, right? You had groups like Amazon and Netflix and others saying, well, if you build it, you have to operate it. And so they really didn't create separation of teams. They sort of just enabled individuals to have enough capabilities.

to do those tasks because they said, if the right mindset is in place and the right incentives are in place, if you know that you have to operate your software and you have to be wearing the pager and you have to think about the hours that are required, you will think more specifically, you will think more effectively, hopefully, about writing software and thinking about testing scenarios and failure scenarios in which you won't just sort of.

hand something off to some deployment team and say, good luck, I hope it works. And, you know, you fix it. Right. But beyond that sort of absolute, right, that that far end of the spectrum, we have really struggled as a, as a community, as a, as a, you know, and as an industry, whenever there were requirements in which teams that had previously been distinct, were forced to work together. And there's been a lot of things that have happened that seem very positive.

Brian Gracely (09:24.59)
that have been working to making that somewhat better, right? So we see situations in which, you know, like monitoring visibility, you know, kind of observability tools are available and the same tools are useful for monitoring and observing infrastructure and the same types of tools, the same tools are there for application teams. Okay, so that's progress, right? We've seen this with lots of things, you know, Datadog and Prometheus and all those sort of things. We've seen scenarios in which...

the idea of things like source control and version control and those types of things are becoming more commonplace in which kind of the idea of sort of everything is code. So whether it's infrastructure is code or security is code or whatever the next thing is, CI, CD is code. Those concepts have made their way into the mainstream and they are moving in the right direction.

But I think we still struggle in general whenever we kind of require or expect that in order for a technology to be successful, it's going to require the teams to sort of cross domains. I'll give you a great examples, right? Or a couple of examples. I don't know if they're great or not. But a couple of examples. So things like Service Mesh. Service Mesh is a technology that if we go back, Kubernetes is now celebrating its 10 -year birthday.

here this past week or so. Service Mesh's things like Istio and Linkerd literally came along a year after that. And as widespread as Kubernetes is, Service Mesh really hasn't grown nearly as fast as it. And it was really entirely designed to be an adjunct add -on technology, a one plus one equals three with Kubernetes. And I think part of that is,

You know, you could argue, well, service mesh solves maybe a harder problem than Kubernetes. Kubernetes is just scheduling containers, but it's like, no, it solves a very, very hard problem, which is resource management, resource scheduling, understanding all sorts of different parameters, like different underlying hardware, different types of scheduling, whether it's a long lasting thing or a batch job or run ephemeral and get rid of when it's gone.

Brian Gracely (11:49.358)

I think the reason we've seen some of these technologies like a service mesh, for example, struggle is that while Kubernetes, you could say this distinct team owns this, and it's more or less infrastructure. So yes, it is the blending together of sort of compute storage and networking. We already had 10 plus years of experience of having done that. If you've been in the virtualization space or if you've been in sort of the VMware space or any of those sort of spaces, there was a decade plus.

of organizations who had built around the idea that compute storage and networking are really kind of one distinct thing, right? Whereas service mesh comes along and it's sort of this idea of like, well, it's networking, but it's also security, but it's also not really networking for, you know, sort of at the IP level layer. It's really sort of networking at the application layer. And so it becomes this sort of perfect example of, well,

You know, in theory, yes, it sounds like interesting technology. I should have the ability to, to dynamically be able to adapt my security profile and my routing profile for networking, based on, you know, certain things that we learn about, about the application and all this sort of stuff. But in order to make it successful, you needed to have, you know, people with application awareness, security awareness and networking awareness. And if you think about where those live in the stack.

They're all over the place, right? They're at the top of the stack, they're at the middle of the stack, they're at the bottom of the stack, and the stack being sort of like the OSI layers of where things would be. And it's sort of the perfect example of we convinced ourselves that the goal was, you know, we want to be able to allow applications to deploy more frequently, we want to make sure they're secure, make sure they don't break. Therefore, here's this technology. And we kind of didn't think through...

in order for that to work, the technology is going to require this massive, massive change of the people that do it. And I think what happened was, or what we still see today, because in essence, Service Mesh is sort of the perfect example of kind of like, well, that's essentially sort of a DevSecOps type of technology. It touches applications, it touches security, it touches ops, i .e. networking.

Brian Gracely (14:09.902)
And it's really struggled to gain success, right? We look at other things out there, like, for example, like FinOps, right? FinOps is one of these weird things where, yeah, it sounds interesting in theory, right? Like we should have more visibility to the tools that are out there, to what the costs of things are and to be able to project things and all this kind of stuff. Financial forecasting and finance teams have had spreadsheets and so forth.

engineering teams, technology teams have never really necessarily wanted to think about what things cost. They are always sort of fascinated with the technology. And, but all of a sudden, because we're doing things differently with sort of on demand computing, especially with cloud computing, as opposed to the way we used to do it, where we would buy things for three years

and five years and amortize them. And you could sort of just use regular accounting principles. We're now going, well, in order for this FinOps thing to be successful,

The engineers need to learn accounting terminology and concepts and engine in accounting needs to learn engineering concepts again Totally foreign concepts, right? Like I would never expect anybody with sort of an accounting background to be able to listen to five minutes of this podcast and have any idea we're talking about and I sort of know that for a fact because I have plenty of friends who go you run a podcast. I listened for a few minutes. I have no idea what you're talking about But it's just another example of one of those things where we go. well in order for this thing to be successful

We're going to create a technology and then on the back end, hope that these two distinct groups who really kind of don't have any idea what the other one does, they don't understand their domain, they don't have expertise in their domain. We're just going to kind of give you some tools and we're going to hope that you guys are going to figure each other out. And again, could they figure each other out? Sure. They could get in a room and I'm sure they could, you know, they're smart people. They could learn these things, but they're not necessarily incentivized.

to necessarily want to learn those things. It doesn't really help their career per se from the long run. It's not something that they are going to be excited about necessarily learning. And so we wonder why haven't those things necessarily taken off, right? And so those are just a couple of examples. I think we could look at, there is a whole category of packaging capabilities that we've talked about. We've talked about,

Brian Gracely (16:32.206)
You know, get ops and service mesh and DevOps and DevSecOps and all these areas where we have these sort of, you know, this plus this plus something, you know, and I think as a whole, we've, we've really struggled in those spaces. And so, you know, as, as I'm thinking about, well, where are we going to see this evolve as we get into, you know, AI, because, because in theory, AI has many of the same principles that we were kind of hoping for with these, these new.

kind of cloud -based applications in which we expect them to change more frequently. We expect us to kind of build feedback loops and things like that into both the business logic, but also the technology logic. And I think we have to think and maybe think hard about what will be the things that are the fundamental building blocks similar to the sort of compute network storage database.

firewall types of things that we were able to get more benefit from in the cloud world than we were in this sort of data center world. And, and what are those things that are going to be the service mesh slash fin ops slash dev sec ops of the AI world that, you know, some technology is going to come along. We're going to say, that sounds, that sounds fantastic. Cause that's that, you know, that that's going to bring all the groups together that have to kind of work together. And, and where are we assuming.

that there is individual level accountability for those technologies versus in order for them to be successful, the groups have to bring people together that don't know anything about those things, that don't think about those things, that aren't incentivized for those things. And in order for them to be successful, they would need to follow some variation of the sort of Malcolm Gladwell 10 ,000 hours rule in order to be an expert at it. Maybe they're...

10 ,000 hours light, maybe they need 2 ,000 hours, but that's still a year's worth of somebody doing it every single day. And so I think we've learned enough over the last 10 years that we can kind of highlight a number of these technologies. And again, I pointed out just a couple of them. We probably could debate several others if we wanted to. Some of the platform tools, PaaS versus infrastructure as a service versus some of the kind of in -between variations of that. But I think...

Brian Gracely (18:55.502)
as we start to think about this new era, at the end of the day, we're still going to, no matter what fantastic things an AI can do, it's still going to be some combination of it's an application. It's an application that has a data model behind it. In this case, that data model is now called an LLM or some model. And we're going to want to be able to make changes to the application. We're going to want to be able to test it through various iterations and experiments and so forth.

And we're going to want to be able to manage the sort of model, the data side of it, for stability as well as change and versioning and all those other types of things that we've done with applications and databases for the last 20 plus years. So be very curious what folks are thinking in terms of what lessons have we learned from the DevSec, FIN, Platform, ML Ops kind of mashing together over the last decade plus. Has been what successful?

Which ones haven't necessarily been successful and what are the patterns that we've seen for when they were successful, which I would argue is more, we allowed teams to kind of get specifically what they wanted with the current skills they had and they were able to offload other things to somebody else, to some entity, whatever that thing might be, versus the things that in order to get what you wanted, you had to sort of learn skill plus skill to get three.

Right. And those I think have been less successful and have sort of proven over time to be less adopted and so forth. So anyways, curious what people's take is on that. But again, looking back on where we've been over the last decade, where we've gotten excited about things and maybe they haven't panned out, I feel like that's sort of the dividing line to a certain extent. And again, trying to take some of those lessons learned or that historical context and sort of think forward about that.

in terms of, will we make similar mistakes? What are the building blocks gonna be? And where are there gonna be places where we can sort of learn from our past mistakes and optimize things for the future? So anyways, that'll wrap it up. Thank you all for listening. Hopefully your June's going well. We're getting into July. We're getting almost into the second half of the year.

Hard to believe the year's gone by so fast. And we're kind of out of the first half of trade show season and all the things that have happened.

Brian Gracely (21:19.246)
Folks are going to be going on vacation. We're going to keep cranking out shows. I think we'll mostly have shows every week. We may have a week in there when Aaron and I are on vacation, but we expect to crank out shows kind of on a weekly basis like we always do. So anyways, with that, thank you all for listening. Thanks for sending in your feedback. Show at thecloudcast .net. We really appreciate that. I know we've got a few to catch up on, so I apologize if we haven't gotten to yours yet. But with that, I'll wrap it up. Thank you all for listening, and we'll talk to you next week.