Brian Gracely (00:01.326)
Good morning, good evening, wherever you are. Welcome back to Cloudcast. We are coming to you live from the massive Cloudcast studios here in Raleigh, North Carolina. Hope everybody is doing well. Another weekend perspective as we get towards the end of July, 2024. And it's starting to feel like, not at the end of summer, it's still obviously unbearably hot in many parts of the world and this part here in North Carolina especially. But it does start to feel like we are getting to where people are starting to make plans for what the fall might look like. Kids are starting

think about going back to school. know for Aaron and I, we've got kids who are thinking about going back to college. you know, so, you know, it feels like definitely a transitional time in terms of where we are in the year, not tons and tons of net new technology news. We've had a very, very busy last couple of months and covered that both in the, you know, end of, know, kind of the monthly show as well as the mailbag we just did. So covered a lot of that. And what I wanted to cover today was kind of something I've been thinking about quite a bit, which

We've had so much technology over the last couple of years. We've had so much change in terms of the types of applications that are being built. We're building more modern applications. We're building applications that change very rapidly. We have all sorts of new interactions in terms of how applications are interacting with APIs, interacting with other applications, and so forth. And so we've had this evolution of applications. And hence, we've also had

a lot of thought about, you know, how do you, how do you build the platforms? How do you build the things underneath the applications that allow them to evolve quickly, allow the teams that have to support them to work very well, whether it's with the application teams and the operations teams where security fits in, where in this new AI world where things like a MLOps or data scientists fit in all those different types of things. And in many cases, we, we think about the way to solve those problems as

platform problems, right? If we have the right platforms in place, if we have the right underlying things in place, if they are available as self -service and developers can not have to think about too much or certain teams don't have to think about knowing everything, the systems underneath it will sort of take care of them for them, i .e. the platforms, things should be better. the theory is if you apply the right sort of self -service in the right places,

Brian Gracely (02:19.454)
If you deliver a level of consistency, you should be able to reduce costs because you've got consistency. You're able to sort of train your teams around one type of environment. And I think to a certain extent, when we look at the numbers in terms of, you know, adoption and we talked to, you know, different companies around the world in terms of how efficient they've become with these new things. think there's still a gap, still a lacking in terms of making these platforms do what we expect them to do.

We're now, you know, we've kind of lived through a decade or so of kind of DevOps and what that means, or actually I think we've lived through about 15 years now of DevOps. We're beginning to live through, you know, kind of the idea of platform engineering and, more more folks are on these platforms. And so what I want to do is kind of walk through some of the thinking I've had, and I don't know that I'm necessarily going to have a bunch of answers about this because again, you know, platforms as we'll get into are complicated and what people want from them is not always the same.

But I want to kind of dive into, you know, kind of the hurdles that people go through, that companies go through, that groups go through with working with platforms, why they think about using them, why they struggle with them, and maybe some ideas about, you know, how we can see greater adoption when it makes sense for companies and for application teams and development teams and infrastructure teams and all those sort of things. So we'll get into that after the break.

Brian Gracely (00:02.092)
Welcome back to the Cloudcast. I'm your host, Brian Gracely. And on this weekend's perspective, as I talked about at the top of the show, we are going to dive into kind of overcoming the hurdles, the platform hurdles that companies have to deal with. you know, again, let's put this back in context. We've seen a lot of changes in how applications are built in the types of applications. We've seen a lot of positive things that have come out of these new modern types of applications, whether they are microservices, cloud native, mobile, you know, working with.

a backend as a service, frontend as a service. We're seeing things with AI being integrated with all sorts of things. So we're seeing a huge expansion of how applications are built and how they're, you we're applying technologies, new technologies and so forth to be more adaptive to the business, to provide greater value to businesses. But to a certain extent, you know, how those go about being done is still a little bit of a black art. It's less science and

you know, kind of magic in terms of, you know, which companies have figured out how to do it well, which companies struggle, you know, is it on an application by application basis is on a team by team basis. And I want to kind of dive into some of that. So let me start with a little bit of context, right? So, you know, why do people think about platforms as ways to provide that foundation under the covers that allow developers to just focus on their applications, allow them

not have to know about all the underlying things under the covers, whether it's security or networking or how to mount storage or how to troubleshoot things, allow them to focus on the application, focus on what the business needs that application to do. And we've been through this for quite a while, right? mean, this problem goes back 15 years, as far as I can think. mean, it's been going back as long as we've been doing the Cloudcast. So that's nearly 12, 13, 14 years. And it

kind of originally when we first started talking about sort of platform as a service. And again, the whole idea of it was we want to reduce the cognitive load on application teams. We want to enable self -service as much as possible where it makes sense for those application teams. And I keep saying application teams, like this could probably be applicable to other types of things as well, but we're going to center it around application teams. We want to be able to deliver consistent services, whether that's

Brian Gracely (02:22.382)
you know, one team that's making rapid iterations and frequent changes to an application or set of applications or to a number of teams, you know, hopefully with, with a lot of efficient operations under the covers. And then hopefully if, if it all works out, we reduce cost, we reduce the number of snowflake environments, the number of silos that we have to build to begin with, and then maintain an ongoing basis. So there's all sorts of reasons why people have been enamored with this idea of platforms for, like I said, 15 years.

let's call it that, right? All the way back to sort of the, you know, why can't the world be like Heroku? And I think the more and more we've learned is that as you have to apply the idea of platforms to organizations, large organizations, small organizations, there's a set of hurdles that kind of have to be understood and have to be overcome in order to be able to make that thing successful. And I think a lot of times we don't necessarily

understand all the hurdles. We don't necessarily rationalize the ways that things have to get from idea to implementation to out into production, all that sort of stuff. And I think we sort of miss the boat sometimes in understanding just how much people fight against, especially certain technologists will fight against standardization because they feel like everywhere there's a standardization is one

less level of freedom, one less level of creativity that they can bring to the application, they can bring to their environment, they feel like there's a burden upon them. So we've had this challenge for the last 15 years or so. We will continue to have it as we continue to evolve our applications. But I thought I would talk about some of the biggest hurdles that I see and at least theorize maybe some of the things that are possible.

to kind of get over some of those hurdles. And again, I'm gonna be the first to say like, look, if I knew all the answers to this, we wouldn't, not me in particular, but we would have been talking about this over and over again in the show. And I think it would have gotten mainstream. I think as an industry, we still struggle with how do we get over these hurdles? Because like I said, we've been through various iterations of paths, various iterations of the naming of things. We've

Brian Gracely (04:46.734)
you know, serverless and we've seen platforms as a service and we've seen, you know, extensions of infrastructure as a service. And we've seen all sorts of things, um, come along and, and as a whole, it's gotten better. It's gotten incrementally better. And maybe, you know, if we really look back on it over 15 years, it's gotten enough incrementally better. That, you know,

we shouldn't be so frustrated with it, but I think, you know, oftentimes when we, when we frame this thing up as, you know, if you build platforms, right. And you organize your team correctly.

you really should be increasing velocity, reducing frustration, reducing cost, creating greater stability, creating greater operational excellence, not frustrating your teams, all these sort of wonderful things that should come out of doing it right. But we just haven't figured out the right way to sort of make that happen every single time. So let's step back for a second. We know for sure

If we deliver sort of lowest common denominator capabilities, i .e. compute storage and networking as primitives, we know that those types of platforms can be tremendously successful. And we just have to look at VMware with vSphere, or we have to look at AWS with EC2 and S3 and so forth. If you deliver the most basic primitives, and you do it in a way that is pretty easy to deploy, pretty easy to manage,

you know, kind of has wide applicability could be used for database bases or middleware types of applications or high performance applications, or just, you know, kind of plotting along websites and so forth. That can be a very, very successful platform. Now it mostly is, is targeting sort of infrastructure delivery, self -service infrastructure delivery. and you know, that's a very, very solid foundation to build upon. And in many cases we confuse.

what we call infrastructure as a service and what could be platform as a service. And we think about those as like two very, very different things. And the reality is, you if you wanted to deliver an infrastructure service and you wanted to deliver a platform as a service, about 80 % of those things are going to be the same because the fundamental pieces you have under the covers in terms of, you know, how you deliver the primitive, how you automate the things around it, how

Brian Gracely (07:05.73)
built sort of observability and all the tooling that needs to go on around that, and then a certain amount of self -service, whether you're delivering a virtual machine and a network connection and an S3 bucket, or whether you're delivering those things with a Java stack inside of it or a Python stack or something, aren't really fundamentally different. We struggle, for some reason, when we do put a little more in there, because we're introducing another team

you know, may or may not have wanted exactly what we put in there, but they're not fundamentally different. But when we think about the hurdles of this, there's a number of them that jump out at us, right? The first one is even at just purely a scaling perspective, how do we get from the team that started by building something, they had a business mandate or a technology mandate to go build something, and they went about doing that themselves and they really didn't have a need for a platform with that first application because they didn't necessarily know how big it was going to

And they didn't know what the end result of it was going to be. And so they just built something, something that worked in many cases, the things that they didn't really have great in -depth

knowledge about. could have been, you know, complex networking or some security aspect. They just sort of made do with what they could find available. They might've, you know, talked to a friend and said, Hey, how do you secure some stuff or, you know, how do you do a single sign on or, know, what's the simplest way to do networking? So I don't have to be a Cisco CECIE to make this work, whatever it might

and many, many projects start that way. And then many times that team wants to take it into production or they were recognized for doing something interesting. And the organization says, I want more of that, or I want that to scale. And so now you get into situations in which the application team that built it says, well, I didn't want to be in the operations business to begin with. Let's let somebody else take care of this for us. And you start to get into a scenario

People that are tasked with have taken care of it. You let's call it operations for now. Maybe it'll become platform in the road. Basically start looking for the things that they tend to look for, which is where can I standardize things? Does this align to what we already do today? Can I fit it into current best practices or am I going to have to change a whole lot of things? And that first phase of figuring out like, does this look like a standard type of thing that we do, or are we going to have

Brian Gracely (09:32.578)
you know, make this into sort of a unique snowflake is, is sort of hurdle number one, right? Like can we create an environment that can onboard things that don't look exactly like something we're looking for. And that oftentimes is sort of the first hurdle we have to get over is, you know, if you had an existing platform in place, how flexible is it, or, know, how forced to be standardized is it? And that's a, that's a huge hurdle. The second hurdle that typically comes from that scenario

platforms by their nature tend to be bigger in scale. They tend to be looking for a lot of capabilities because they're oftentimes serving multiple needs of whatever they're doing. And so you start to get into some really basic things like, well, how much is that platform going to cost? Whether it's cost in terms of acquiring it through a vendor or something like that, or the cost of saying, we're moving

kind of individual little silos within our organization to some team that has to build and manage this thing. Maybe you built it purely from open source, but it's going to take you X number of individuals to build it and then X plus some other number of individuals to maintain it and to troubleshoot it all that sort of stuff. you know, the first sort of hurdle is can I bring something that deviates from standard into the platform? That's often a challenge, right? Sort of the snowflakiness of it. The second part

The second hurdle is, what's the cost of getting from individual sort of scenario to platform level something that you can start to go, OK, I can deliver services, multiple services that have consistency, have flexibility, that have security, that have all the things that you want. Now, the

next type of hurdle that you run into is, let's suppose you have a team that builds out this application that works.

People want it to scale. want more of it. And then let's say you have another team that did the same sort of thing, right? Built their thing, built a sort of homegrown set of tools, scripts, whatever, to kind of keep their thing up and running. And now you get into a scenario in which you go, okay, who's gonna run the platform for this and who's gonna pay for it? And what you run into sort of the third sort of hurdle is that the buying centers,

Brian Gracely (11:56.076)
you know, that application number one and group number one and application number two and group number two may have their own budgets, but there's not necessarily a way for group one and group two to kind of combine budgets to, you know, both kind of pay for the platform to support their thing. Or you may run into a scenario in which they could pay for it, but collectively they can't coordinate themselves or they can't figure out amongst themselves, you know, which priorities, which capabilities.

would go into this centralized platform because they probably both want everything they had before and they might to those two things might be very diametrically opposed. So now you start to get into not just the cost of the system but sort of organizationalizing of the system, compartmentalizing of the system. And in some cases, organizations can work this out because they've got a very strong centralized culture.

they're able to say, we can expand upon something we do, or we can budget ourselves such that everybody chips into a certain level and we know how to do kind of internal billing, if you will, whether it's actual billing or just kind of paper kind of billing, but they can build centralized things. But in some cases, you get into scenarios in which every team wants to control their thing and they want to control their budget, and they're just not going to allow that platform thing to

Now, the next sort of hurdle that you run into is, okay, in some cases, the scenario that I laid out in which the application team built something for the business and it makes logical sense to move to a platform, that happens. But there are other scenarios in which you've got smart, clever, curious infrastructure or operations teams who are looking at the marketplace. They're looking at how they're interacting with their application teams today.

And they say, you know what, we know what the business goals are going to be maybe this year, maybe next year. and we know the direction in which the application teams are moving, the technology teams, the CTOs office, whatever it might be is moving. And so we want to proactively start thinking about what a platform would look like in order to be able to take advantage of these new capabilities in order to be able to say, Hey, we are going to be responsive to the business. We know that you're going to want containers or self or.

Brian Gracely (14:18.656)

serverless or database as a service or storage as a service, whatever those things might be, we're going to build that and have that ready for you so that when you come to us, we can just start saying yes, right? So they can kind of get ahead of the, or the department of no, we're going to start saying yes. Now that's another fairly common scenario. Then the challenge becomes you built it. How do you get the application teams, the people who are ultimately going to be your customers to buy into what you've built?

Right, and this is again where you really kind of get into the communication that goes on between teams of as you were building it, were you taking application team input? What would you guys want if we built this? Like where would you want self -service? Where would you want standardization of say t -shirt sizes for compute resources versus standardization for application stacks versus where do you want flexibility? Where do you want to be able to say,

We don't use that version of Java. We don't use that database. So too often teams will build a platform sort of, you you've heard the old saying, know, field of dreams, build it they will come, they will build a platform and then not necessarily know how to go out and essentially internally advertise, you will internally market that they have these capabilities that they could, you know, help better service the application teams, better service the business.

by making people aware of these things. So we oftentimes see that where these platforms get acquired or they get built and they aren't in sync in the right ways or they've created the wrong motivations or they've created the wrong just sort of pain points and un -pain points for the application teams and an adoption doesn't happen. It doesn't happen as fast as it needed to. It doesn't happen to help sort

burn down the cost of the platform, those sort of things. The next thing that comes up a lot is how do you, once the platform is in place, no matter how the platform got put in place, once applications are now running on there, how do you sort of figure out the balance between prioritizing things, features, capabilities that are driving innovation, things that have to go fast, new innovations, new technologies, and the things

Brian Gracely (16:44.61)
you know, applications or teams that really would want to prioritize sort of stability, right? I don't want things to change a whole lot. I don't want to, I don't want you to upgrade the platform every three months because then I have to recertify my application. I wish you would do it once a year. I wish you would do it every six to nine months. you cannot do it between November and February because you know, we're in the retail business and that's sort of the black Friday season and that's when we close our books. So any changes you make, you can't be doing it during that timeframe.

And so a lot of teams struggle with, you know, once they, get a platform in place and they start onboarding applications and onboarding teams is how do you prioritize who should get the biggest say what, you know, what should be the velocity of change? What should be the

priorities of stability? You know, how do you manage security issues as they come along? Right? Like how do you, how do you manage this sort of, you know, broad set of multiple use.

variable things running on the platform. And a lot of teams struggle with it. They struggle with how do you troubleshoot that? They struggle with, you know, how do you go about doing this? And a lot of these things, you know, can be boiled down to, you know, thinking about upfront as you're building these platforms, you know, how will you deal with those scenarios, right? So sort of doing some scenario planning and thinking about can the system inherently deal with slow and fast at the same time? Can it deal with, you know, where can it put self -service? Where

you know, where will maintenance happen? Where will all those sort of things go on in the platform? And a lot of times those aren't necessarily thought through upfront because you're trying to service that initial customer or those first couple of customers that maybe look the same. And as you get into more and more things, you know, you get this explosion of, you know, kind of variety that happens on top of the platform. So those, those are all sort of hurdles that the teams have to overcome and that, you know, the people

using the platform have to overcome the team, delivering the platform have to overcome. and a lot of them, I think the more people are aware that those kinds of problems are going to happen, the more they can start looking into them or they can start planning for them. the more they are coordinating, if you are the platform owner, coordinating with your stakeholders about what do you need now? What are you going to need in the future? and, showing an effort to sort of build around being responsive to

Brian Gracely (19:12.408)
But then also having open and kind of transparent conversations about the fact that if you've moved to a platform approach as opposed to sort of a siloed approach, these are the sort of shared challenges that you're gonna have to work through in terms of maintenance, security, troubleshooting, noisy neighbors, all that sort of stuff. So anyways, I kind of wanted to highlight the challenges that we see with people.

I'd be really interested for those of you that are building platforms that are operating platforms today, how are you overcoming some of these things? I know from my day job, how we go through certain things, whether it's building things like multi -tenancy into the platform, educating people about, again, just kind of how to best operate them both from an operations perspective, communication and collaboration perspective. But I know every company tends to have about 85 % the

perspective or definition or structure for their platforms. But the other 15 % is the part that is different from company to company to company. It creates snowflakes. Some do it better than others. Some really struggle with it. So we'd love to hear your feedback. If you are operating a platform, you're thinking about operating a platform, how do you go about doing that? Where do you sort of draw the line between where you have kind of standardized primitives versus where you create variability? Where do you kind

try to introduce self -service versus creating scenarios in which people can have some variability versus standard cookie cutter, t -shirt size availability of whatever you're offering, an application stack, an AI stack, access to data, those types of things. But we'd really be interested in hearing people's feedback. So you can hit us up, show at thecloudcast .net, hit us up on any of the places we're doing Slack, any of the social media channels. But I feel

You know, we've been doing this for a while. We've gotten better in some regards. We've struggled in some regards. And I feel like maybe it's time to open up a new conversation that maybe isn't necessarily happening in the platform engineering domain, because it feels like those folks are trying really hard to get you to buy into the idea of platform engineering, not necessarily the realities of platform engineering, i .e. used to be DevOps, all those sorts of things. So anyways, we'd love to hear your feedback. We'd love to hear your success stories. We'd love to hear your struggles.

Brian Gracely (21:39.358)
you know, your failures and maybe try and share those with some people that can maybe help make that sub work. But so it's something that's been on my mind for a while and kind of wanted to sort of talk through at least where the challenges are, maybe what we can do in future shows is kind of take each of those challenges one by one and kind of do, you know, some interviews with folks that are, you know, making it work, and, try and, bring some, bring some help to the hurdles that are oftentimes getting in the way of people building, building platforms.

With that, gonna wrap it up. everybody's doing well. Another weekend perspective. Like I said, we're getting towards the end of July. I think we're going to do shows through the end of July. We may skip one. I know Erin and I have got some travel and some vacation stuff coming up. So this might be the last show of July. I gotta kind of work through some stuff. I mean, there'll probably be one more. I may not do the next Sunday, but I gotta take a look at the calendar. anyways, if the next Sunday doesn't show up, that's why Erin and I, like I said, are trying to work through

some end of year stuff with the kids, as well as working through some vacation stuff that we've been planning for a while. So anyways, with that, thanks for listening. Thanks for telling a friend. Thanks for helping us grow the community, and we'll talk to you next week.