Aaron Delp (00:01.004)
Good morning, good evening, if you are and welcome back to the Cloudcast. We're coming to you live from our massive Cloudcast studios here in Raleigh, North Carolina. And it is just Aaron this week, no Brian. But we have a really fascinating topic today of something that's Brian and I have been kind of kicking around in our heads on the sides and kind of having signed conversations of this whole concept of, DevOps has been around for a while and then platform engineering turns into something for a while. But then what's next?

And also how do they interact and why we even had problems and needed some of those next steps, right? And so we're going to kind of explore that space a little bit today. In order to do that, we have Mark Friedle, co-CEO and co-founder at Kodiak. So first of all, welcome to the show, Mark, and give everyone a quick introduction, please.

Mark (00:51.552)
Yeah, thanks, Aaron. Glad to be here.

Like you just mentioned, I'm the co-CEO and co-founder of Kodiak. We're a DevOps platform, at least that's the easiest way to say it, but really it was designed out of the, it was really the brainchild of software engineers and architects that were working toward, know, through that space that got more and more complicated, that got more and more snowballing. Then we ran into the fact that there were many things happening in the industry that all kind of came together like,

for instance, infrastructure as code, like containerization, Kubernetes, big data, cloud, all those things are sitting there saying, streaming for, why are we doing a lot of this stuff by hand? And thus came Kodiak. And anybody that's gotten into even the question of, hey, should we use a platform for this? Probably knows exactly what I'm talking about. At some point, people turn around and they say, enough's enough. And that's where we were when we started this thing. We said, look, this is crazy.

We've got to figure out a better way to get through this stuff.

Aaron Delp (01:52.643)
Yeah, yeah. And maybe let's start there then too, because I definitely want to dig into the tech a little bit, but let's kind of start with the problem. Because the way I've kind of always phrased this, mean, just like anything else in tech, there's always like your classic S curve, right? Like it's a very bespoke solution and you're trying to figure it out. And then you kind of hit that commoditization and where the platforms and the standardization comes in. And then you got full end, full blown commoditization at the top.

Mark (02:09.382)
you

Aaron Delp (02:20.758)

Like we've like I said, we've been seeing this problem for a while. But I mean, there's been lots of labels all for kind of sort of the same space or slightly adjacent spaces, whether it's infrastructure is code, whether it's platform engineering, whether it's DevOps, and then you're a big proponent of SDLC. So the know, software lifecycle and how does that kind of fit in to all of this? And what's the problem you're ultimately trying to solve here?

Mark (02:28.675)
Well, SCLC was a great place to start with that because we believe that's the common language that all these guys need to speak. So.

I've got folks on one side of the fence putting together infrastructure and writing infrastructure as code, authoring that. However brilliantly or not so brilliantly it's being done, it's being done. But we have on the other side a bunch of people doing application development and application design. If you've ever been an application developer or architect who is speaking with your deployment system or your SRE, there's a lot of conversations that need to be had there. In fact,

There's usually deployment meetings to try to coordinate that kind of stuff. And then on top of that, you have the cloud providers, which when it comes to commoditization, obviously Azure and AWS wouldn't know anything about commodity anything. Of course, they're entirely unique from server to server and system to system. But the reality is that from the perspective of I've got software to run.

There are, there is a way to commoditize that and be able to say, look, ultimately I need to know where my stuff is. need memory, need CPU, need, I need response time and I need geography and I need to be able to stand it up and put it down. So with those three things alone, not to mention all the other people that are involved in the approval process and the release process and everything. If everybody's speaking SDLC,

you've got a fighting chance for everybody communicating the same language. And so that was a really great place for great places to start with this thing and kind of centralize a platform. And the other thing I'll add to that, which is being in the industry, like I mentioned, I'm a software person, a couple of decades of software architecture and that kind of stuff. And I've also been up and down through management and entrepreneurship here and there. What we find though is that

Mark (04:35.711)
throughout these decades, any software person that you speak to that is worth their salt will chime in about the notion of separation of concerns and the notion of, hey, that's business logic versus that's implementation, that's engineering versus that's business. That separation and decoupling, that word is a big one, I mean, that is a source of pride and differentiation and distinction in the software community.

However, when it comes to the business of software itself, somehow that gets dropped out. And somehow there are blurred lines from the business of managing software to the implementation of managing software. And so if you were to say, hey, back in the 60s, we figured out how to write software and deploy software in many different ways, but let's just speak about NASA. They had enormous ways to move forward, ways to manage it.

errors and all kinds of other things, they were bound to all of their implementation. suddenly, this was only just a few years ago, we finally have the ability to say, we have infrastructure as code and we have cloud providers and we've decoupled ourselves from who's running these things. And so that ability to say, well, these different pieces suddenly allow us the ability to say, right now we could separate the business of software from

the implementation of software, the business of software, which has not changed. I mean, what's the requirement? I got to run my stuff. I want this version running over here. I need this much resources underneath it. And it needs to stand up and stay stood up. And if there's something wrong with it, tell me about it. The implementation of it, are you running on cloud? Are you running on your own services? Are running in the back room? What version of Apache windows, et cetera?

All that stuff is implementation. And suddenly today we have the ability to say, well, maybe that can be a commodity. If we're speaking the same language everywhere, that's really where we're coming from. think that's where, at least from my perspective, it's one way of describing the emphasis we've seen going into platform engineering recently.

Aaron Delp (06:51.244)
Yeah, that makes sense. And also, a quick side note. I want to dig into this further, but you have a really interesting origin story with your co-CEO as well when it comes to starting the company. so tell everyone a little bit about how passionately you believe this and how far you went. I'll just phrase it that way. With starting the company.

Mark (07:03.828)
Yes.

Yeah, well, we went through a lot of trap, you know, initially, then Ben and I had been Gazi was my co CEO co founder. He and I had worked together several times in the oil and gas community. And we started doing a couple projects together. But we both had this bug, you know, and it was both, you know, what I'd mentioned earlier that this is getting to be so bad. And some of it's so difficult to do the things we want to do. If you've ever had a project that you want to do on your own.

you realize like, well, I can start the concept a little bit, but I know that I'm going to spend all my time doing the stuff around the idea and not the idea itself. So anyway, we realized the opportunity. I got into Kubernetes and started thinking, my God, the promise is enormous. In fact, don't think this is one of those times. I don't think I'm ever going to go back. So we talked

about it. We talked about building a company. I found a customer and this guy was willing to put it up and pay us for this contract. And so I said, here it is. Here's the vehicle.

Ben calls me up, we'd thrown these ideas around, but Ben calls me up about a couple weeks later and says, all right, I got a buyer on my house. And I thought, well, yeah, I was talking about doing this for a long time, but Jesus, we better get, we actually have to do this. This is real. So we bought RVs, moved out to Texas right out in the middle of nowhere and went dark for a couple of years. And we realized like, I don't know when we're going to get done with this thing. I don't know when.

exactly we're going to have that MVP, it was, you know, initially it's really nice to be able to say, well, we're on our own steam, we're doing this thing the way we know it needs to be done, we're redesigning and some of the discussions we had were, it was just fun and, you know, exciting to be on the, really chasing that intellectual thing. But there we were, I mean, we're running on our own money, our own steam, and then we finally chased down some investors and got some seed funding and, you know, kind of way it went, very slowly.

Mark (09:06.357)
Two guys trying to build a platform is especially, you know, something that's changing some of the, the O for the general business of software. It's, it's a tough thing to move right out of the gate and it's a confidence game. So it's been, it's been a challenge.

Aaron Delp (09:21.144)
Yeah, yeah. And further clarifying question though, this was separate RVs, not the same RV, correct?

Mark (09:28.545)
Yeah, Ben moved his family into an RV. He's got two kids and a wife and a dog. I've just got the dog. And so I put my place in, I live in Tampa, I put the condo up for rent, bought the RV, moved out to Houston. And we had our separate little work offices there. Mine, obviously, one guy and a dog, perfect for an RV. And I recommend it. Anybody's got the opportunity to do that. cut the budget right down. You give yourself an enormous amount of opportunity to kind of make the moves you want to.

Aaron Delp (09:46.113)
Nice.

Mark (09:58.26)
Wife, two kids and a dog, you different story. You got to really want it.

Aaron Delp (10:02.92)
Absolutely. That is a different level of commitment for sure. So let's get back to these trends though because

I see, yeah, like it's interesting. feel like this whole, I'm just going to use this idea of friction, friction, whether it's at the application layer, friction at the operations layer has just been around for so long. Everyone is bought in to the concepts of infrastructure as code and platform engineering, and then just Kubernetes at its most fundamental layer. But everyone kind of always steps back and kind of goes, well,

but it's pretty complex, right? And then you had microservices, you know, in this loose coupling of all the services, you know, that doesn't make it any easier as well, right? And so when it comes to development of this, especially if you're trying to recreate an environment, well, I mean, like, what if you have 20, 30, 40, 50 microservices that all combine together to build the app? Well, how do you build that?

Mark (10:48.027)
Yeah.

Aaron Delp (11:06.868)
Right. And so like, especially when I'm thinking about like the build pipeline for something like this, like, am I going down the right path with something like this? When you're talking about like, there's, there's like the central project, which is like, let's say your microservice, but then there's all the other stuff. Right. And is that what you're trying to solve for here?

Mark (11:16.963)
yeah. Yeah, absolutely. Well, I think, I think what.

You know, one way of looking at it is we were talking a little bit before about bespoke versus customizable versus configurable. That's really the question, especially to anybody building a platform saying, well, approach are you taking? And a lot of people that set out on that path, I think there's a very subtle differentiator that's really important. One is I'm going to go bespoke, which basically is saying,

I've got one size fits all. It's my script. It's my stuff. It's a black box to you. You hit the go button and boom, it just stands itself up and I hope you trust me on that. And most people I've talked to, myself included, are going to hear that and say, no. There's the occasional time where that works. And if you're an internal, if you're running an internal platform, maybe that is great because you've got a platform team that's constantly managing that and nobody else in the company wants to know anything about that.

Aaron Delp (12:09.344)
Yes.

Mark (12:20.865)
But if your target market are actual engineers and architects that, if nothing else, have a lot of say or need a lot of say in how it's configured, the bespoke model isn't really the greatest. The other model, is open configuration, is another one, which is sort of the other end of the

spectrum. It says, all right, I can do whatever I want. You're giving me a thin little wrapper over all of the tech. And I'll just go in there and I can tweak and adjust and play everything I want to.

And somewhere in between is there's a sweet spot in there that says, I'm going to make the very simple, the common stuff simple, and I'm going to make the complicated or rare stuff possible. You get an enormous amount of mileage out of that. If you can say, I'm not designing a bespoke model, and I'm not designing an open, just open,

configuration framework, but instead I'm developing a tool set with the goal being how much can I propel you? How quickly can we get the things that we know you need done done so that you can move on to the next thing? It's one thing to say, hey, we saved a little bit of time building this. It's another thing to say, wow, when the difference between doing this was 18 phone calls and three meetings and a month and a half of work, and now it's five minutes of bam.

I can do things that I didn't think were, I may not have had the other discussion about. Like one of the examples I bring up is the, you know, the, if it took a yaml file to here, take your, take your glass and bend your elbow and take a sip, there would be no such thing as cocktail parties. There would be a lot of people saying, well, I got to drink a drink of liquid and I can do that. And that's great. I know how to do that. And I've got a team of people that make sure that that happens.

Aaron Delp (13:54.946)
Heheheheh...

Mark (14:07.007)
Say, all right, fine. But once it's intuitive and it's part of your makeup, like nobody I know is wasting their time trying to figure out how to balance that glass in their hand, that's done. So they're having a conversation. Now we're meeting people, we're talking about business, we're setting up events. It's a whole different evolution that happens when that stuff is second nature. And that's what we're working toward. Like how quick can you get...

conception into reality. when it's intuitive and when it's smooth, you know, I've mentioned this before and some people like it or don't, but I like the notion of an elegant cyborg. You know, we've got tech all around us. It happens to be attached to our phones and our PCs. But when it's the brain moving through the arm and the arm does the thing that the brain's thinking, new things happen and they're exciting. And you can't look at tech today with all the stuff that's happening and just think, God, there's so much cool crap out there.

and there's so much I want to do, but I'm stuck because it takes me four and a half hours just to get started on that stuff. Screw that. Let's put an opinionated framework together that acts as a toolset and not as a bespoke model. So anyway, with that toolset, there's obviously a lot of opinions and you get a lot of mileage out of the 80 % and there's other things, there are most likely will probably be, and we haven't encountered it yet, but there'll probably be some folks that say,

I can't or that may not fit that model. again, I'm just going on a little bit longer. give me one more item on this, that the ability to say, have a system, it does this thing, it allows you to customize it. It also allows extensibility so that you can say, OK, I can customize this whole stuff. And it really gets me a lot of mileage. And then that thing that I'm doing and I really am Snowflake unique on this one point.

Aaron Delp (15:37.965)
All good.

Mark (15:59.294)
I need to add a little extension to it. And it's our own custom extension that we had to do. And that's healthy. When that's possible to extend it like that. And you have to ask. You've gone through the gauntlet of, you really need to write that? Because mean, really, which part of this is really unique for you? And you can realize, wow, well, if we zig here instead of zag, it all fits into the model. Hey, that's great. I think that's really the ideal position there.

Aaron Delp (16:25.708)
No, that makes, it makes perfect sense. And toss or just kind of add to that. especially when it comes to SREs and the work SES are doing, especially supporting build pipelines. you know, I've kind of always had this idea that, Kubernetes is a blessing and a curse. it's an amazing application platform. And like, there's always that running joke. It's a, it's a fantastic platform for building platforms. Right. But, but,

As long as you're not the one that's doing the care and feeding of the platform, it's great. And, but at the same time, like, is it really like, when you're talking, like, when we talk about reduction of friction, is it top to bottom in the stack? Is it more targeted at specifically at Kubernetes because it has such a perception of being such a bear in the industry? What is your thoughts of like, okay,

If you're tactically attacking it and you're talking to an SRE, like where are you specifically removing the friction in the stack?

Mark (17:28.99)
Okay, the first of all, Kubernetes, we chose as a platform because it offers all those great capabilities. The promise of Kubernetes is phenomenal. It's a modern platform. think the notion that you take, for instance, auto scaling, Kubernetes doesn't own auto scaling. There's plenty of different ways to do that. But back to the business versus the implementation side, if you look at it and say,

Today one who deploys software must a get logs get telemetry Auto scale as needed set memory footprints and that kind of thing whether it's kubernetes or not It doesn't matter the the so we chose it because yeah, it's a great platform for developing things. It's however If you've

ever used it it's somewhat cumbersome because there's just a lot to take a lot to keep in mind and and the approach that we took was hey if we

If we look at it sort of like a guy who walks in, know, take your favorite deli. I don't know if you have one or not, but pretend you have your favorite deli. walk in every day. Eventually people recognize you and they remember you. And the first day you walk in and you say, okay, look, it's going to be the roast beef with asiago and cheddar. And you lay out the whole sandwich order, right? After a while, that person remembers your sandwich order. And wouldn't it be nice if they just remembered it the first time?

And so that's kind of the thing that Kubernetes isn't necessarily going to do for you. Kubernetes is going to ask you for that whole order every time. And you're like, Jesus, man, I told you about this yesterday. So when you can say, all right, well, I've got an in-between. I've got this platform in between me and the ultimate platform. And the ultimate platform, let's just speak about it like it's an assembly compiler.

I don't want to write assembly code. In fact, I really don't want to repeat myself. You you know the sandwich I like, give me the usual and give it to me over here. In fact, give me the usual, but cut the crust off. You know, great. Very easy, very quick, very intuitive. And so I think that's really the, you know, the, the, the challenge that I've run into a bunch of this conversation a lot. I tried deploying the Kubernetes. It wasn't too bad. I set up a service and I was able to get it out there and I said, yeah, that's not too bad at all.

Mark (19:39.518)
When you have one service going to one environment with one bit of config and one version, I mean, it's really, it's not hard to walk through the demo. The challenge is the operational side. And again, back to the friction, you know, like, well, we got to make a change. okay. I got to get that config out of the secret store. I got to change that secret. I got to bring that secret store in and then I got to go back out to this and I got to have to make config change in this file, this file, this file, this file. And then I got to make sure that this lines up with that. And you're doing a lot of.

I mean, can be when it's the thing to do, it can be really exhilarating puzzle solving, but, and maybe even a bit of a game to try to make sure I got it this time, you know, but, you know, business-wise, operationally, when you can just say, yeah, make that change and send it out. And if that change had an impact of, you know, 15 different configuration changes that needed to happen there, well, that's not your concern if you've got a good tool behind you. You know, your concern is make this change, send it out, make it happen.

And that's really, you know, again, that's really the elegant approach to it that offers so many more opportunities there. anyway, I go on and on about it, but that, yeah, it's that evolution.

Aaron Delp (20:48.63)

No, I love it. I love it. Let me ask you this, though. if we kind of so we talked about tech stack, if you will. But the other half of that is always the people. Right. And. Cultures and politics and relationships and org charts and all these other things. Right. What if you're talking about reduction of friction? Sometimes some folks are absolutely on board and that makes perfect business sense. But then there's other folks that might be like, well, hey,

Mark (21:05.557)
Well.

Aaron Delp (21:18.232)
That's my job or that complexity literally is my job definition. Right. And so how does the culture and organization and relationships all sometimes either have to change or not, or what's things to look out for when we're talking about, you know, this kind of consolidation or optimization, if I use that term as well.

Mark (21:44.865)
I think there's an opportunity in especially for SREs where

you let's say with and without a really good friction removing system for instance right whatever that system may be the sre has a lot to do a lot to keep in mind and and some meticulous detail to keep up with not only that a lot of sres live in the

no news is good news world where look I don't care about you until it's broken and then when it's broken you're the center of attention and it's not good you know and so the ability to say like well I've got a better discussion to have with the team apart from is it running or not what anybody really wants to know especially at the you know the communication between the business and engineering that that proverbial gap that that lives there

Sometimes it's language, sometimes it's just simply, I'm going to you asking you for more money for another server. Here come the tech guys asking me for more compute jargon crap that I don't have the budget for and screw them. But when an SRE can come in and say, hey, I've been watching your stuff because I had time to set up auto scaling. I had time to watch and monitor utilization. Like a lot of people, little aside, a lot of people I speak to.

Spend a lot of time like I met a guy working for a pretty big company said I just spent all my time shutting things down Because I don't know what's in use and not we have an enormous number of servers that are just Running and we're not even sure whether anybody's paying attention to them or even using them. So utilization Alone can be a big deal. We have a feature called zombie mode that Allows you to schedule your stuff to shut down and so we take our dev environments and we started shutting them down on the weekends Well, that's two-seventh of the burn right there, you know

Mark (23:36.822)

We started shutting them down at night too. He said, well, you know, if you're working with something, we'll keep it on, but we started shutting it down at night. we were, I mean, we were cutting like 40 % off our costs just right then and there by shutting down stuff we weren't paying attention to. So when the SRE can walk in the door and say, I'm paying attention to utilization, I'm paying attention to auto scaling, and I'm paying attention to things like cost per user and cost per transaction, which is a normalized number that you can speak to the business about and say, hey,

Last month, our cost per transaction is this and our cost per user is this. If you allow us to make this change, we can drop that cost per user significantly. That's a real conversation to have. Also, I can see where the bottlenecks are. Because I have time and ability to, I spent less time setting it up and more time actually optimizing and paying attention to things. Now I can see we have a bottleneck right here.

and it's costing us. I mean, sometimes it's one query that's just loading it up. So I found some really great opportunities so that SRE can actually have a conversation in a sprint planning meeting to say, this one right here will save us an enormous amount of time. Let's prioritize it. When you're working with an SRE that can have conversations with you like that,

optimizing your systems, cutting down costs, and actually speaking at a business level about this whole operation of software that you're running instead of just, hey, John, is it working or not? That's an enormous opportunity, both for the business and for those people that are monitoring it.

Aaron Delp (25:12.674)
That's fantastic. Yeah, I really love that. And I think that's a actually a great place for us to pause, Mark. So let me ask you this final question. If anyone is interested out there and wants to dig in more, what's the best way for folks out there to get started, Mark?

Mark (25:14.465)
Kodiak.io or reach out to me on LinkedIn. Just mark Fridel or look up Kodiak.io on LinkedIn. We're happy to get back to you. We've got a beta test right now that we're running, finding, looking for people to help us out on to run local clusters.

on your local machine and do some self-hosting of some AI workloads. So love to get some feedback on that too.

Aaron Delp (25:50.168)
Fantastic. Awesome. Well, Mark, thank you very much for your time today. And on behalf of Brian and myself, everyone out there, thank you very much for listening this week. And if you enjoyed the podcast, please tell a friend, please leave us a review if you can, wherever you get your podcasts. And as always, send us feedback. Show at the cloudcast.net. Thank you, everyone, for listening this week. And we will talk to everyone next week.

Mark (26:13.15)
Next slide, folks.