Architecting the Cloud, part of the On Cloud Podcast

Mike Kavis, Managing Director, Deloitte Consulting LLP

Title: **DevOps and SRE—people are the secret to success**

Description: DevOps and SRE certainly help bolster digital transformation efforts, but only if they're implemented well. Unfortunately, many companies just look at the technology component and not at—perhaps—the most critical component, which is the people. In this episode of the podcast, Mike Kavis and guest, Red Hat's Matt Stratton discuss how fostering collaboration, properly incenting success, and promoting what Matt calls psychological safety are keys to successful implementation of DevOps and SRE. Matt also argues that having a "product" mindset and building resilient systems will produce better results than trying to anticipate every crisis that could happen.

Duration: **00:29:48**

Operator:
This podcast is produced by Deloitte. The views and opinions expressed by podcast speakers and guests are solely their own and do not reflect the opinions of Deloitte. This podcast provides general information only and is not intended to constitute advice or services of any kind. For additional information about Deloitte, go to Deloitte.com/About. Welcome to Architecting the Cloud, part of the On Cloud Podcast, where we get real about Cloud Technology what works, what doesn't and why. Now here is your host Mike Kavis.

Mike Kavis:
Hey everyone, welcome back to the Architecting the Cloud Podcast where we get real about cloud technology. We discuss what's new in the cloud, what's hot, why you want to use it, but most importantly we talk to the people in the field who are doing the work. I'm your host Mike Kavis, Chief Cloud Architect over at Deloitte, and today I'm joined with an old friend, probably the second or third time I've had him on, but it's been a couple years – shame on me – but it's Matty Stratton. And Matt is a transformational or transformation specialist – man, that was hard. Matt is a transformation specialist at Red Hat and a longtime member of the global DevOps community.

Back in the day – this is a trivia question, right? Back in the day his license plate actually said DevOps, and of course back then no one knew what that was. And what else we got here? I'm just going to let you introduce the rest of yourself, but I do want to plug in – Matt has a great podcast as well called Arrested DevOps, and he's also a global organizer of DevOps Days. If you've ever been to DevOps Days you've probably seen Matt. So, Matt, welcome to the show. Sorry it's been so long. Tell us a little bit – you've got a new gig. Tell us a little bit about your DevOps journey over time and what you're doing now.

Matt Stratton:
Yeah, absolutely. So the next trivia is that my license plate, while it doesn't say DevOps now, it says Cube Cuddle, and that's a whole other story. We can talk about that on Twitter, about how that happened. But before I was kind of even involved in DevOps I've got a couple decades' background a SysAdmin, so kind of coming from the ops background, but doing DevOps stuff for quite some time. As you might know I worked with Chef. I worked with PagerDuty most recently, and now I'm h ere at Red Hat. And the thing I get to focus on now in the new gig, which is maybe about two months in, although it feels like it was just yesterday – there's a lot going on.

So I'm in something called the NAPS transformation office, which I always love to say because it sounds like I'm transforming how we sleep and take naps, and I'd like to be a subject matter expert in napping, but really NAPS is North America Public Sector. So I'm working to help organizations in the public sector through their transformation, whether – we all know digital transformation's kind of just this vague word, but really thinking about how these agencies and organizations can deliver solutions to their stakeholders, which are people like you and me, much more effectively. And there's a lot of that really kind of topical given the kind of pandemic we're in right now. There's a lot of adaptability that has to happen. So I love to be back with you, Mike. It's good to have this chat. I think we've got a lot of fun stuff to talk about.

Mike Kavis:
We do, and before we started recording, we were talking a little bit about some of the things we're working on. We have a lot of synergies. And one topic that came up was SRE and helping companies adapt it. And the one point I want to make while you were doing the intro was yourself, myself, and all the other great folks that joined Red Hat – John Willis, Andrew Shafer, all those people, we all have deep, deep technical backgrounds but none of us are focusing on tech anymore, right? We're focusing on the soft side, and that's got to say something. I keep saying something to everyone every week – they're probably tired of it, but technology's easy; people are hard. We've all been on this journey for ten-plus years, this DevOps journey, and we're all strictly focused on transformation now. So why do you think – I know the answer, but why do you think that is?

Matt Stratton:
Well, and it's interesting, I'm going to kind of put a little corollary to that in a second, but it's because these systems that we're talking about are complex systems and they are inherently sociotechnical systems. And it's important – they're sociotechnical. So like you said, we're not really focused on the technology, but we're not ignoring it either, right? The technology is a key part of the enablement as these sociotechnical systems. And as Adam Jacob of Chef has said before, it's like tools influence culture and culture influences tools.

So what I like to think about when I'm looking at the outcomes that we're trying to accomplish is what are the ways that we can use the technology as an enabler for those socioeconomic, but also sociotechnical, systems we're trying to facilitate, right? Because to be honest, it's not enough just to talk about how people need to be different, right? Because we can sit there – and that's also a way that transformation doesn't work, if you have senior leadership that's just like, "Now, we're going to suddenly be more collaborative. Ready, steady, go." That tends to not work as well. So I've tried to adopt and adapt my conversations to be what's the outcome, and the outcome is almost always around people. And then how can we help express those outcomes? And how can tools help us do that? And the important part of it is how can tools help us do it, not how can the tools do it, right?

Mike Kavis:
Right.

Matt Stratton:
Like, so the things are very intertwined. But I think the reason you see – it feels like almost that we're doubling down on the transformation and the people part of it is because, like you said, that's the part that's harder, and frankly it's the part that to a lot of – especially in engineering seems less fun. Like, we don't have to work to get engineers to play with more toys. They've got that, right? It's cool. So we can understand that that's going to happen, so we enhance, and we emphasize the human side of it, but we want to make sure we don't do that at the - being dismissive of the technology, because those things are really a partnership. They're enabling each other and it's a feedback loop.

Mike Kavis:
Yeah, and one of the things you talked about there is that ready, set, go – the how. And I was watching one of your talks at – I think it was a 24-Hour AllTheTalks thing. One of those was 24 hours.

Matt Stratton:
Yeah. [Laughter]

Mike Kavis:
So I was able to watch you, and I happened to catch you talking, and the point was that we've been told for so long that we have to collaborate better, but no one ever told us how, to kind of sum it up. So talk about that.

Matt Stratton:
Yeah, so that's the thing, and I think it goes back to that, where it almost felt like we were over-rotating, but I think it was necessary. So if you look at – so DevOps Days is an example. It's a little over ten years going on, and it feels like the first half of DevOps Days it was – all you could do was hear talks about why empathy mattered, and it seemed like a lot. And you're like – and now we've kind of gotten to the point that is, okay, I get it. I understand why this is important. But again, I would like to actually now go do this, please? And [laughter] a friend of mine, Dave Shackleford, gave a great talk actually years ago called Operationalizing Empathy, and that was kind of my inspiration for this talk in the background, was how do we take this and be tactical and practical, and not even necessarily in the perspective of – and I especially gear it towards folks that feel like they don't have the authority within an organization,

because we feel like the way that we enact these things is by being the CIO and dictating direction for the company. And the reality is that it happens throughout the whole organization.

So I try to focus on what are some of the tactical things we can do if we know the outcomes we want to have around collaboration, and my focus in the talk was around some of the key ideas that I think are important around the collaboration, and a big key one around that is the idea of psychological safety, and how do we make psychological safety happen? And the little bit of background on that, so that really – the idea around psychological safety came from – at least our awareness of it in the industry came from a research project Google did called Project Aristotle where they said, "We're going to look at our high-performing software teams and see what we can learn from them." And the thing that they found that was most key was the teams that were high performing were teams that had a high level of psychological safety.

So what is that, right? So Amy Edmondson of Harvard Business School summarizes it really nicely, which is it's a sense of confidence that the rest of the team will not embarrass, reject, or punish someone for speaking up. And the thing that's interesting about that is we all probably are like, "Oh, yeah, well, we totally have that," right? "We have a friendly team. We get along. We're all buddies." That doesn't mean that you have a high sense of psychological safety.

And the last thing that I thought was super-duper interesting about this – we've been hearing about Project Aristotle for several years, but it was always like, "Well, but Google is Google." And, Mike, how many times have you heard, like, "Well, that's great. That's fine for Netflix, but I do real work here," right? Or I – "Google has different problems so I can't apply that." So in the 2019 Accelerate State of DevOps Report they wondered the same thing, and they saw a correlation between the elite performers and having a high level of psychological safety. So the point is it's not some Google magic sauce. This was consistently true across the industry.

And there are a lot of things you can do to help instill that psychological safety. And I don't want to necessarily give it all away – you should go watch my talk, but – yeah, a big part of it is one of the things, especially if you're a leader – and a leader doesn't necessarily mean a manager. I've been in this position a lot as well, as just like a senior person on the team, is modeling your behavior in certain ways. And modeling vulnerability is one way to do that, and it doesn't mean that you talk about like, "This is what I did at happy hour last night," or whatever, but just sharing parts of yourself. You don't have to show your whole self. You don't have to be a complete open book. But little bits of personal things, they show a vulnerability, which then we see that that comes into play and is effective.

And the other piece that I think is really interesting and it goes back to a lot of ways we think about DevOps as well, is the idea of replacing blame with curiosity, right? So when we're having something happening within our team, we want to as engineers get to this so-called root cause, and what caused this, and what is to blame, whether it's a person or a system? And if you reframe this to say, "I'm an investigator. I am curious. I want to know more." Because you don't know everything, you don't have all the facts, and your team can help build that. What that means is if you have that mindset, when someone brings something up if you're looking to "blame" or find cause, it's very quick to be like, "No, no, no, Mike, that's wrong. No, no, no, that doesn't matter. I don't care, whatever," right in so many words. But if it's curiosity, my framing is, "I didn't think about that; tell me more?" And that lets people feel more comfortable speaking up.

And why does this matter at the end of the day? I've always said this before. If people are afraid of being punished for their mistakes, it does not cause them to make fewer mistakes. It makes them become a subject matter expert in hiding their mistakes. And now you're really in trouble because you do not know what is going on in your environment. So people need to feel safe in bringing up concerns, bringing up things that occur to them, and it just helps the organization and the team be better.

Mike Kavis:
Yeah, I was at a company a long time ago where we would have all the teams do their status report, and if your thing wasn't in green it wasn't going to be a good meeting. So one of the experts at, SMEs, at reporting, used to use this phrase: "It's trending green." Which to me meant, "It's flames, on fire, burning red, but we'll call it trending green and I can get through this status report without getting destroyed." So yeah. So early days you were saying there was so much talk about empathy and it kind of drove me nuts, not because I don't believe in empathy but empathy's just a piece. The other piece that I see when I'm trying to help people whether it's DevOps, SRE, or cloud is alignment, goals and objectives and incentives, right?

So you have an ops team that's incented to – everything runs smooth, and a dev team getting features out the door, and a security team – nothing bad will ever happen, and a governance team – I think nothing will ever happen, or whatever it is. And there's not a lot of shared goals. So they can all have empathy for each other, but if their paycheck is based on them performing roles that contradict, I think this is why the operating model is so important. You can still have those centers of excellence, but how do we create these full-stack teams? And all this shift-left stuff we're talking about – people think they're losing their jobs, but really it's about full-stack teams getting – that's really how you collaborate, when people are driving towards the same mission, when they're aligning towards a product or a service instead of a technical discipline. Then their goals and objectives align, that to me then they almost naturally would tend to work a little closer. But I just wanted to get your opinion on that.

Matt Stratton:
So you're absolutely right, right? And you change culture by changing behavior; you change behavior by changing incentives. And folks will say, "What's the most important DevOps book I should read?" And I said, "The book you need to read is Freakonomics. Go learn about incentives and you'll understand DevOps." That being said, we also do tend to go to financial incentives first because it's a thing that is the most easy to reason about in an organization and it's a great place to start, right? So I'll just sort of illustrate – my classic example is I had a colleague years ago who was a TechOps director for an online real estate website. And her annual bonus was a percentage of uptime. Like they gave her a target that your target is ten grand. We're going to multiply that by site uptime; that's your bonus. So how, again, incented do you think that this person would be to let anything ever change on that website? That is literal money out of her pocket.

So we hear that, and we chuckle about it. We're like, "Oh, I'd never do a thing" – and I feel like we've kind of gotten our minds around the financial incentives because those are easy targets, right? We're like, "Oh, okay, great, well, we don't do things like give bug bounties to shipping features or whatever," and everybody's kind of on these things. But there's a lot of other ways that incentives happen and there's a lot of unintended consequences. And I think some of it is just even how we share and how we celebrate success, so to speak. And this is why – it's one of the things, again, kind of having that

SysAdmin hat – being a system administrator or an ops person is a lot like being a corporate lawyer, right? Nobody really knows about your job except for when you do it poorly. think about a corporate lawyer. Like, you don't hear about all the times you've kept the company from being sued, but everybody knows when you missed and you got sued, right? The same thing for ops. So it's thinking about ways to drive towards that. So incenting is not just about how someone is paid.

And the other thing is, remember, if we kind of think about – Andrew Clay Shafer talks about the Nash-Pareto equilibrium, which boils down to that people will work to the metrics you give them even at the detriment of your entire organization, is kind of my summarization of that. No matter what they are – and so this is reflected – you talked about trending green. So even in how we measure success of a system, not even as people, because we become emotionally tied to the systems that we build, right? And so it's really important to measure and quantify and understand metrics around our services but understand how they reflect. So an example is if we are measuring the effectiveness or the reliability of a service based on the number of SEV1 incidents it has, we become really focused – our practitioners become focused—on a metric that I call mean time to innocence, which means we get really good at just pointing out that it wasn't a SEV1, right?

Mike Kavis:
Right.

Matt Stratton:
We're going to game it but none of that actually matters because not having SEV1s is not a business outcome, right? So we can measure that but it's never a way – so the first place to think is, like, never tie somebody's comp to those numbers for sure. But I feel like now we're all nodding our heads and going, "Oh, yeah, well, we totally don't do that anymore." But then look at the unintended network effects of just how you do that. If you've got – the CIO has their big dashboard they talk about in the quarterly meeting and they throw up the numbers and are showing all that. That is a reflection upon those individuals. So you've got to think about what those effects are. And again, Freakonomics is a great tome to read to understand about unintended consequences of incentives.

Mike Kavis:
Yeah, and I try to push it a little further, because usually it's a Devs and Ops thing, but I think it's a product owner thing, right? So now we're moving to SRE and the SRE team's got all these SLOs and all this stuff, but if the product owner still is influenced by features, your error budgets don't matter, right? So we need to get past Dev and Ops, right? It's like, what are the golden objectives of this product? And then everyone else within their domain expertise should align objectives to help keep that product, and usually the number one thing is customer satisfaction, right?

Matt Stratton:
Yeah.

Mike Kavis:
So I've actually seen a company – I don't remember who it was. It was up on one of the DevOps conferences, where they were doing – they were doing customer satisfaction to IT people. That was their – it was the net promoter score. That's one of the key metrics. And really what it's all about – usually it's like, "We need to sell this thing, it needs to work, and customers need to be happy," and then all of our metrics should be driven off that. But that requires us all to be aligned around a product as opposed to being around a technology.

Matt Stratton:
That product mindset is absolutely key and critical. And that's the big shift, is like shifting from project to product, and then thinking about how is that product related. So like you said, you have your individual kind of subject expertise areas around, but they're all building up into that metric that is around the product. And the way that you measure success of that project may be NPS scores. It may be revenue. It may be – there's various ways that you understand how that product – but the thing that's amazing is, like, all of those things that these different groups do, those are all fundamentally to meet user needs and happiness, right? And this is the thing – I have a talk around the idea of full-service ownership, but one of the things I talk about to product owners is your customers have requirements, but they don't put them in those words, right?

Like, your customer is not necessarily – unless you're providing an IT service—is not going to speak in terms of reliability or latency or things like that? But they want that – or security, right? Again, if your product is very focused on that, you may have users that are like, "I have this security requirement." But a lot of consumer-facing stuff, they don't verbalize that, but they verbalize it in their usage. And those things all tie back into – and they all build – everything that we're doing within our teams and our domain expertise around operations and infrastructure and all of that, they all service – they all exist to directly or indirectly service the needs of our user and, fundamentally, , right? Like, and that's where your SLOs go. Your SLOs are – Dr. Jennifer Petoff who's the editor of the SRE book, and I accidentally created a term we call the "hadness point". And that's the point where customer happiness is just about to become customer sadness, and that's where you set your SLO, right?

Mike Kavis:
Right.

Matt Stratton:
And so we tend to go back with service levels. We always – because we're engineers, we always think about having a certain amount of capacity or things like that, and those are all interesting ways to reason about it. But if you don't have over-encompassing SLOs that are connected to the success of the product, then we're not thinking about it in everything we do. You may have heard me before – like, if I'm in a group full of engineers giving a DevOps talk I say, "Do you know how your company makes money? If you don't, go find out. I'll wait." Because it's always going back to these – and it might not be just about making money, but what are the business outcomes? What are those key business outcomes? And that's a hard thing to distill down when we're used to being held accountable to very activity-oriented metrics, right, to very, like, "I did this thing, so I'm a good developer," right, versus outcome-oriented measures.

Mike Kavis:

Yeah, and I think the point of view of the customer – you can do things through architecture that can overcome outages or poor SLOs. So for example, let's say you have a website and say you're selling something and you're bringing up an image, right? Well, you can have stuff cached, right? So if your call goes out and fails, you can go to a stock image stuff like that. One example I was using the other day is, let's say you're shipping a product, right, and someone wants to track where your product is. Well, you can store the last good location of where that image is and say, "As of," right? So even though your SLO for that service may fail, you could architect it so it's totally relevant to the consumer, and that's the system thinking, and that's – true SRE is those people who have that mindset sitting with the developers and saying, "Well, if this fails, how can we collectively architect around it?" as opposed to, "That's a SEV1 and it's" – you know….

Matt Stratton:
So taking that to the next level, so if you ever noticed that SRE is called site reliability engineering and I don't know if this was on purpose or not, but it's not site resiliency engineering, because resiliency is adaptive capacity for the things we could not think of, right? So like, again, sitting there like, "We think of all the different ways this could fail and we could build robustness and reliability into our system, into our technology," that again, like you say, this cache misses; I can bring it up here, whatever. But there are always going to be the things we could not predict, and resiliency is something that exists in our system that we want to be able to express the ability to have that adaptive capacity, right?

The known knowns, the known unknowns, the unknown things, all that stuff, and I think the thing we have to be cautious about is getting wound up on our own hubris of all the things – that we are smart enough to think of everything terrible that might possibly happen. We want to continue to build and refine and express – a term I like is that resilience is a verb, right? So it's a thing you do; it's not a thing that you have. And robustness and reliability allow us to have the resilience in these sociotechnical systems so that when we are brought in — again I came from PagerDuty and there was a lot of stuff that I used to get really wound up around, some of our marketing terms, and we would talk about preventing incidents. And I was like, "I hate using the term prevention."

And I know we're trying to prevent them. The reason I don't like it is I don't want senior leadership to go, "Oh, well, I thought you were going to prevent this." I didn't know. I'm sorry, I didn't know there was going to be a global pandemic, right? I'll try to be better with my crystal ball next time. But what you can have if you have a resilient system is when these things, these unknowns happen, your system, primarily made up of the people, has that adaptive capacity to adjust and minimize, and that's the thing. The point of a postmortem is not to ensure this never happens again, because you can't. You can't ensure that it'll never happen again. But what we want to do is to mitigate the disruption that if this special circumstance happens again. So it's about setting expectations around reliability, I think is really, really key. And if you dig into how you do SLOs, that's a key part of SRE, is understanding that 100 percent uptime is unrealistic, some would say impossible.

Mike Kavis:
It's impossible.

Matt Stratton:
Yeah, yeah.

Mike Kavis:
But – yeah, I use a business definition of resiliency is a system's ability to adapt to adversity, or something along those lines, how your system responds to adversity.

Matt Stratton:
And the thing I would say and, Mike, I would agree with you 100 percent with that definition as long as we understand that that system is made up of people also.

Mike Kavis:
Yep, good point.

Matt Stratton:
It is a sociotechnical system. Because otherwise we – we're engineers. We're going to sit there and we're going to say, "I need my system," and what's the thing we're going to want to run on it is a bunch of AI and machine learning. We'll be like, "Aye, well, great – AIOps." And then it will automatically be able to do all this stuff. And it's like, yeah, there's always the things, so.

Mike Kavis:
So last question, and – so we're both doing some work, helping people go to SRE. I'm doing a lot of work around op model and people are like, "Op model?. Alright, help some people out here. What's your advice when people are trying to mimic large companies, these large systems?

Matt Stratton:
So it's funny because folks end up on either end of the spectrum when it comes to that. You either end up with the, like, "I am going to do exactly what Google does to the letter," or, "I am going to call it that but I'm not going to actually do any of the important parts of it," right? And so I always like to say go back to the spirit of what that is. What are the outcomes that that thing is trying to achieve? And start with that, and then – because you think about things like the Spotify model. You think about SRE. Those are frameworks and methods to achieve an outcome. So if you are not aligned to the outcome that that thing is trying to do, then don't even bother. You've got to start with that, right? So if you want to think about adopting SRE, you need to have reliability as a core business concept across leadership as that is an important thing, because if it's not, all you're going to do go through motions.

And then likewise to your other point, I always like to joke and say there's two kinds of SRE. There's SRE and there's GSRE, and GSRE is how Google does it. And you should absolutely do GSRE if you are Google, right? If you have the exact same business and exact same infrastructure and everything's the same, then yes, do exactly that. What you can't do – and this is where the nuance comes in – is the picking and choosing. And by picking and choosing I mean the outcome stuff, and here's a great example. So we can say we want to have everything about SRE except for actually empowering our site reliability engineers to stop and deploy. You know what? If you're like, okay – so you kind of – be conscious in the things that you're picking and choosing, but they

should be that you are agreeing with the outcome, but for your particular world that implementation detail is not the best way to achieve that outcome. So the thing that you can shift when you're adopting these frameworks is the implementation detail, but if the outcomes and the dare I say philosophies – that's a little too highbrow way of thinking about it - those are what matter, because if you don't – if those are not important to you, then you're never going to have any success with those implementations, right? Those are the how, but you need to be aligned to the what.

That being said, when you think about SRE I think it all starts with service levels because all of the amazing things that you can get out of SRE, you cannot do those things until you can reason about your service levels properly. Everything – toil reduction error, budgets, and learning from incidents and all those things, if you don't have strong service-oriented SLOs that you are adapting – that's the other thing to remember. SLOs, they're not legal documents and you should be constantly reviewing them as you're learning more. If you aren't doing – that's where you start, with that stuff. And it was interesting. Google recently published a research project on kind of where people were in SRE adoption, and basically what most people have adopted is, "We sure do have some SLOs," and then you say, "Well, do you revisit them?" "Oh, no, we don't do that but that's a good part – if where you're at is you just have some SLOs now, cool. That's a great place to start, but it's a continuum, right? And it's not a revolution and it's not – by the way, it's also never done. Sorry. [Laughter]

Mike Kavis:
Yep, I use that term a lot, but as a consultant, clients don't like to hear never done. [Laughter]

Matt Stratton:
Right, right, right.

Mike Kavis:
Matt, enjoyed the conversation. I wish we could go another hour, but I think people would cut out by then. But really enjoyed it. Where can we find your podcast?

Matt Stratton:
Yeah, absolutely. So you can find us all in the iTunes store, on Spotify, search for Arrested DevOps. Or if you don't want to do that, just go to ArrestedDevOps.com. You can listen to the show there. You can click on all the subscribe buttons. We ship about two shows a month. We've been doing this for six or seven years. We bring on a bunch of just interesting folks across the industry to talk about DevOps-y things and things that are DevOps-y adjacent. And if you want to argue with me about things, you can find me on Twitter: @MattStratton. I love to get into things there in a very friendly and tongue-in-cheek way.

Mike Kavis:
Alright, enjoyed it. So that's our episode today of Architecting the Cloud with Matt Stratton. To learn more about Deloitte or to read today's show notes, head over to www.DeloitteCloudPodcast.com. You can find more podcasts by me and my colleague Dave Linthicum just by searching for Deloitte On Cloud Podcast on iTunes or wherever you get your podcasts. I'm your host Mike Kavis. You can always reach me at MKavis@Deloitte.com or follow me on Twitter: @MadGreek65. Thanks for listening and we'll see you next time on Architecting the Cloud.

Operator:
Thank you for listening to Architecting the Cloud, part of the On Cloud Podcast with Mike Kavis. Connect with Mike on Twitter, LinkedIn and visit the Deloitte On Cloud blog at www.deloitte.com/us/deloitte-on-cloud-blog. Be sure to rate and review the show on your favorite podcast app.