Architecting the Cloud, part of the On Cloud Podcast

Mike Kavis, Managing Director, Deloitte Consulting LLP

Title:	**SRE: it's not about numbers—it's about customer satisfaction**
Description:	Most organizations have implemented some version of site reliability engineering (SRE) to help them understand system performance and develop more reliable systems. However, SRE isn't just about managing to a number or a percentage. Instead, it's about defining "reliability" in terms of customer satisfaction and letting data drive decisions. In this episode of the podcast, Mike Kavis and guest, Google's Nathen Harvey, discuss how organizations can do SRE better. Nathan's perspective is that SRE is organic to each organization and that, in addition to error budgets, SLIs, and SLOs, SRE is really about leadership. Leaders set the tone and help ensure that SRE is implemented in an inclusive manner that stresses teamwork, fosters communication, and delivers reliable systems customers love.
Duration:	**00:26:46**

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 all the hot topics of cloud computing with people in the field who do the work. That's the most important part. And I'm Mike Kavis, your host and chief cloud architect over at Deloitte. Today happy to be joined by Nathen Harvey, developer advocate at Google. So, Nathen, welcome to the show and tell us a lot about yourself.

Nathen Harvey:
Yeah, sure. Hey, thanks so much for having me today, Mike. I'm really excited to be here. So, I love the sort of intro that you just gave. We're talking about I mean hot topics. But also, like, in the intro you mentioned that you talk to people that are doing the work, and I have to start with a confession. I don't do the work anymore, although I used to do the work. You know, a developer advocate – I think to be a good developer advocate really requires that you've

walked the mile in the shoes of the folks that are doing the work that you're advocating for. So, to that end I've basically kind of come up as sort of on the operations side of the house if you will. I was a systems administrator. I did a lot of infrastructure as code, a lot of automation, and I was on call. Like, I know that pain. I know the pain and the joy of being woken up in the middle of the night. Unfortunately, sometimes you get woken up about things that don't matter and sometimes you sleep through the things that do matter because you weren't woken up about those things. But, I'm sure we'll get into some of that here as go through.

But when I left that job of being on call and sort of moved – a shift in my career, I moved to Chef Software, which was all about infrastructure as code. How do you do all of this automation? And at Chef I helped build the community around the open source solutions, I managed a team of open source engineers, and I really went out and advocated for the use of Chef.

And about two years ago I joined Google Cloud, where I've been a developer advocate, really focused on sort of the ideas and practices around DevOps and the ideas and practices around site reliability engineering, or SRE. And I have the great pleasure of going into organizations and talking with them about, how do they build high-performing teams? How do they adopt these principles, practices, and technologies to actually deliver what they need to their customers in a reliable fashion? And so that's kind of what I do today.

Mike Kavis:
Cool. Yeah, I was actually on the developer side but, somehow, I had the pager strapped to me that went off every night, seemed like 3:00, 4:00 in the morning. And over time that migrated me to the operations side. *[Laughter]*

Nathen Harvey:
Yep.

Mike Kavis:
But I call myself more developer than ops, but I've lived on both sides, so I feel pain for both sides, so.

Nathen Harvey:
Yeah, for sure. Well, I think that empathy is really important, really understanding what does each side of that equation – like, what is their experience? That's so important.

Mike Kavis:
Yep. And I've hung out with a lot of the sales side, so you know, when people sell stuff that doesn't exist, so, we always got mad. But being in the room when someone says, "I'll give you $2 million if you can do this," it's hard to say no, right? So, I understand that side, too, which kind of led me to a world where how can we build systems to anticipate – someone's going to sell something, right? So, how can we build extendable systems. So, empathy is a big part. You should always be in someone else's shoes to build systems.

Nathen Harvey:
Absolutely, absolutely. And recognizing that, yes, I will take your $2 million and I will give you a system, and the system, the software isn't the thing that's going to make everything magical. At least not alone it won't. People in the system – are at least as important, probably more so.

Mike Kavis:
Absolutely. So, the reason why I asked you to be on here is there was the All the Talks conference – ah, it must be two, three months ago now. There were so many good talks. Actually, many of my guests the last few months have been talks I hand picked out of there. Yours was one. We finally got you on here, so thanks for that. But you were talking about SRE and error budgets, and I wanted to focus in on error budgets because I work with a lot of companies trying to implement SRE and error budgets isn't even in the vocabulary when they talk about this. They're more focused on creating a group called SRE.

Nathen Harvey:
Right.

Mike Kavis:
So, first, tell us what error budgets are, and then tell us why they're probably the most critical part of SRE.

Nathen Harvey:
Yeah, from my perspective an error budget really is a communication framework. It's a communication framework about, "How do we prioritize the engineering capacity that we have?" If you go into any organization, I don't care if it's a startup that has two people, or it's a huge multinational corporation that has hundreds of thousands of people, tens of thousands of which are developers. Regardless of which size organization you're in, there's one thing that's true. You are constrained when it comes to your engineering capacity. You have to prioritize the work that you're going to do. And from my perspective, an error budget really is a communication framework that gives you the tools that you need and the language – you mentioned that error budget isn't even in the vocabulary. SRE is a lot about the language, right? So, an error budget gives you that language to have those discussions around what should we prioritize right now.

So, when you think about an error budget, if you step all the way back, an error budget basically is a thing that tells you should you prioritize velocity, or should you prioritize reliability. And the answer in most cases is yes. Yes, you should prioritize velocity and you should prioritize reliability. But how much of each do you really put into play, or how much of each do you focus on right now?

When Google published the SRE book, a lot of people read that book and saw error budgets and kind of read this as, "If we are out of error budget, if we don't have capacity in our error budget, that means we freeze all feature releases and focus all energy on things that make the system more reliable. And then on the other hand, when we do have error budget to spend, we can spend that on things that make the system move faster, so we increase that velocity." And, so, a budget – it truly is this. It's a budget. It's a thing that you have to allocate. It's a thing that you can spend. It's a thing that you can overspend, you can underspend. But at the end of the day it's a communication framework that gives us signals about how our system is doing.

Mike Kavis:
So, I have also seen engineering teams adopt SRE and adopt error budgets, but the product manager wasn't along for the ride, right? The project manager's still incented to get stuff out the door. So, how do you solve that problem?

Nathen Harvey:
Yeah, yeah, this is a really interesting thing. I think that I see this a lot with teams that want to bring on SRE practices. So, the first thing we do, who want to bring on SRE practices, we create an SRE team. And then we tell that team, "Go and do the SRE." And so that team will go and create service level indicators, service level objectives, and an error budget around that, but if they do that in a vacuum, then it really doesn't matter. What matters is that we have to use this as a way to talk with everyone across the organization, specifically all the way back to the people that are prioritizing the engineering work. Usually in an organization, the people that sit down and prioritize what things are we going to work on are product managers or product owners, certainly in sort of a more modern IT organization. And, so, if we don't have those product owners in the room when we discuss what are the consequences of missing our error budget, what does our error budget look like, it's not really going to matter.

Because we've all been in that scenario. Like, if you just go back to when your team first adopted test-driven development, and maybe you talked about this as a team. You said, "You know what? As a team we're doing test-driven development, we're doing continuous integration, and so we have this new principle on our team. If the build goes red, we focus all of our energy on getting the build back to green." Well, in that scenario I can almost assure you that you've had a project manager who came tapping on your shoulder, and you said, "Hey, we can't ship; the build is broken." And that product manager said, "Yeah, but we have this press release going out that says this feature is available, so ship it. It works on my machine. When I tested it, it looked fine. Ship it." This is the crux of the issue. We aren't communicating very well across those teams. And, so, when we start talking about error budgets, we have to bring product owners into that room.

The other reason that's really important about this is that your service level objectives on the surface – or fundamentally are about your customers. What reliability needs do your customers have? And unfortunately, as operators, as sys-admins, as SREs, we're pretty far-removed from our customers. We don't necessarily understand what exactly do our customers need out of this particular product or out of this particular service. A product owner probably has a much better understanding of that. And one of the challenges with product owners or product managers is they understand reliability. They know it's important for their systems to be reliable. But they also don't view that as something that's within their control. I have a good friend who's a product manager or a former product manager who says, "You know, reliability – I knew it was always important, but I always just figured I'd get that for free. Like, I'm deploying into these production systems; they're reliable. So, I don't need to worry about reliability. I need to worry about these features."

Mike Kavis:
Yeah, that's one thing I've been preaching for a number of years, is everybody owns reliability, everyone owns security, everyone owns quality –

Nathen Harvey:
Absolutely.

Mike Kavis:
All the way up to the product owner. Like if you own designing the next car, you're not going to skim on consumer safety, right? But we seem to do that in software, and I never thought about it that way, but I think it's what you just hit on the head, is they just assume IT's going to do that because that's what they do.

Nathen Harvey:
Yes, exactly. And I also think it's interesting to think about the questions that you go and ask product owners, right? So, when you have this conversation with a product owner there's sort of an inclination to ask, "Well, how reliable does this system need to be?" If you ask someone, "How reliable does this system need to be?" what's their answer going to be, Mike?

Mike Kavis:
Very reliable.

Nathen Harvey:
Yeah! Well, and then you say, "Well, I'm an engineer. Turn that into a percentage. Like, what percentage reliable does it need to be?" What's a product owner going to say?

Mike Kavis:
A hundred percent.

Nathen Harvey:
A hundred percent. I wish that you were right, Mike. Most product owners I talk to say, "Nathen, I believe in you. 110 percent reliable, that's what I want." *[Laughter]* Right? That's kind of the starting point, but I think that fundamentally it's the wrong question. I think we should change that question to, "When do we need to make this system more reliable?" And the reason that I prefer that is that when you ask how reliable, you want all of the reliable. But when do we need to make it more reliable? This gets to the heart of the matter, which is when a system becomes unreliable, so unreliable that our customers stop using that system, or otherwise change their behavior, we don't want to get to that point. So, if it's approaching that point, we need to make the system more reliable so that our customers stay happy, that they continue to use that system. And it really is this question of when do we need to make it more reliable. "Hey, you're the product owner, Mike. When should I call you at 3:00 a.m.? What calls do you want to take from the customer?" And starting to talk and think about that really changes the conversation and gets them invested I think in the conversation, and make really smart decisions about what is the desired level of reliability for our customers.

Mike Kavis:

So, the next question I want to go into is on SLOs and SLIs, and some people listening may not know what those are so define those first. But another issue I've seen is people kind of going into paralysis analysis on that and getting to the nth degree of granularity on that to the point where they want to measure so much, they measure nothing. So, talk about that.

Nathen Harvey:
Yes. Yeah, for sure. So, first we'll start with some definitions because, so, again, vocabulary – so important. We'll start with the SLIs. An SLI is a service level indicator. So, this is some indicator that tells you the level of service that a customer is getting. And it's important there that I said customer, right? We have to take a customer- or user-centric point of view of our systems. So, at their very heart every SLI is a metric, but not every metric makes for a good SLI. So, a good metric that you're probably measuring today: What is your CPU utilization? And if you went to any of your customers and said, "Hey, how utilized do you want our CPUs to be?" no customer would care, right? It's not something that they care about. That's not something that they feel, see, interact with.

So, if you go and understand how does a customer know if this service is working, or maybe you ask a different question, how does a customer know if this service is broken? And when we start digging into that, that starts to point to, what are the service level indicators that matter. I like to say that you don't get to define your service level indicators; your customers have already defined them. It's your job to go and discover what are the service level indicators. How does your customer experience this service?

Now once you have a service level indicator, you have some measurement that you can go and start to make. We now have to figure out, well, how good does that measurement need to be? How reliable does it need to be? And that's where your service level objective comes in or your SLO. So, an SLO is a service level objective. An SLI is a metric. An SLO is how good do we want this metric to be? And then if we're exceeding, if we're better than that SLO, that means we have error budget that we can spend. If we're below that SLO, that means we've consumed all of our error budget and should maybe focus some of our engineering on reliability.

So, there's another S that often comes up in this conversation, and that's SLA. And an SLA importantly – I like to say SREs you should stay as far away from as possible – from your SLAs as you can, because an SLA is typically a business to business agreement that's written by attorneys. There's a bunch of legalese involved in that. And the truth is – I don't know, Mike, if you've ever experience this; I certainly know that I have – I've been so unhappy with a service provider, unhappy enough that I pull the SLA out of the drawer and look at it to figure out have they violated the SLA. Oftentimes the answer was no. I'm super unhappy, yet it's still within the boundaries of the service level agreement.

This is not something that we want to really focus on. If you do have SLAs, you want to be providing better service than that obviously, and that's kind of where your service level objectives come in. But really the two, I kind of think of them orthogonally. There's the SLA – that's the realm of lawyers and attorneys. And then there's the service level objective, and that's how do we make sure that we're keeping our customers happy.

Mike Kavis:
So, how do you get the right balance? Like I said, I've seen people go way in the weeds and they've got so many metrics that it's almost a full-time job just making sure the metrics are coming up.

Nathen Harvey:
Yes, yes. So, that's a really good question, and here's the thing. I think that when you set out to set your SLIs and SLOs and your error budget, I think the first thing you should do is recognize that they already exist. You might not have the language for them today, but when are your customers – like, how do your customers know if your service is working or broken? That's your service level indicator. When do they change their behavior? When do they stop using your service? You need to set your objective slightly better than that. And then let's say in your organization for this particular service you just had a big outage. How did your team change their behavior immediately following that outage? Those are the consequences of your error budget.

So, now if we bring some formalization to it, what we can do is we can recognize, one, that when we start with SLIs and SLOs, we need to keep them as simple as possible. We need to understand what is the business looking for, what are our customers looking for, and what indicators do we have today that will guide us in that path? And just get some service level objectives, some SLIs defined. Start measuring them and recognize that they are wrong. It doesn't matter what SLOs and SLIs you set, when you initially set them, they're wrong. They're wrong not because you did a bad job, but because you just maybe don't have enough practice, don't have enough exposure to your customers and really understanding of the customer. Over time you're going to iterate on these, so it is much better to put something in place than it is to sit in a room with a whiteboard, sort of, again, disconnected from your customers, designing what are all the SLIs and SLOs. Don't try to boil the ocean, as they say, right? Do something, measure it, see how it goes, and then iterate on that. You're going to get better over time.

Mike Kavis:
So, question for you. It's good to have someone from Google because I always make assumptions of what Google did, but now I have a Googler.

Mike Kavis:
But my understanding is when SRE was created it was more looking at the application, right, at the reliability of the application. What I'm seeing is a lot of SRE at the infrastructure level, disconnected from developers, disconnected from customers. But we always tell people, "You're not Google. You're not Spotify. You don't have to copy them but look at what problems they try to solve." So, I don't necessarily think having, like, platform SREs or infrastructure SREs are all that bad, but usually they're just a silo renaming themselves. But you go into customers a lot. What are their kind of op models you see them applying SRE, and which ones work, and which ones don't?

Nathen Harvey:
Yeah, for sure. I think – like, I'll cut right to the end of that question. Which ones work and which ones don't? The ones that work are the ones that teams invest in, and I know, like, maybe that's a little bit of a cop-out answer, but the truth is this takes time. It takes time and practice to grow this, like to build up these principles within your organization. And the other thing that's so important is it's within the organization. It's not within this team. It's not the SREs are over here. And from a, like, how do you implement it perspective, I think you see teams doing all different sorts of things, right? They will spin up a separate SRE team that is separate from dev and it's separate from ops; it kind of sits in the middle. There's this SRE team. Sometimes SRE practices will just

be spread out across dev, ops, product, and there isn't necessarily anyone who has the title SRE. And then also when you have these SRE teams, sometimes they're a separate team. Sometimes the SREs embed within either the development team or within the operations teams.

The truth of the matter is, like, I don't know which one is right for your organization. You know what's right for your organization or what's the right place to start experimenting. And again, this comes back to let's use data to help us drive decisions. So, let's run an experiment and see how does this work when we have an SRE embedded? How does this work when we have a separate SRE team?

Within Google we have separate SRE teams, and SREs within Google, like – we have sort of this abundance of everything, right? We have an abundance of people. We have these huge teams. SREs can go and work with a service that needs SRE support, but if that service is too unreliable and the SREs are doing too much work to make that system reliable, what the SREs will do is disengage from that team. They'll say, "Hey, developers, you own the reliability of this service. We're here to augment and to help you with that, but if you're shipping something that is always breaking, that's paging us too frequently, you know what? You have to take ownership of that. We're going to hand the pager back to you. And when you have a more reliable system, then come and talk to us and we'll reengage."

Mike Kavis:
That's pretty interesting, and that kind of – like when we talk about concepts like chaos engineering and stuff like that, I kind of tell clients, "Your systems can't be chaotic and then take on chaos engineering." So, it's kind of the same approach with chaos. Just come to me when you don't have chaos. But I've never heard what you just said, which is interesting, which is, "We'll be your SRE partner once you get your stuff under control." That's a pretty interesting concept.

Nathen Harvey:
Yeah. And it is. Like, you used that word partner. It's about a partnership. Like, we're all working towards the same goal, but you can't just say, "Well, my job" – you now, it's kind of the classic, like, QA thing, right? So, a developer's job is to write the bugs and the QA's job is to find the bugs, right? The developer can't be like, "Well, my job is to move as fast as possible, and operations or SRE, they'll take care of the reliability. Again, I get that for free." That's not the case. We have to work together to meet those customer needs.

Mike Kavis:
So, when you engage with clients who are having success, are they working with HR or someone to get shared goals and incentives? – because what I've seen is everyone gets the concept, but they're still incented to do the server, or they'll still incented to be the owner of quality, they're still incented to be the owner of security instead of everyone owning that.

Nathen Harvey:
Yeah, I think you've really hit the nail on the head there with incentives, and I think it is so important. This – you know, and whether they're working with HR or not, the thing that really successful teams do is they have the proper executive buy-in, right? So, they have executive-level sponsorship and support, and that executive is willing to go to bat for them, to go to HR, to go to their incentive plans and really help align those. And when those things are aligned properly, that's where you really start to see success instead of finger-pointing and, so, these misaligned incentives leading to really, really bad behavior within the organization. Not bad from an individual – like, they're doing the right thing. They're doing exactly the thing that's going to get them that bonus or get them promoted, but there isn't a systems view that looks across the entire system so we have these misaligned incentives. I think it is really, really important that we can step back, take that system-level view, and frankly that's the role of leaders. That's the role of leaders within our organization.

Mike Kavis:
Cool. Last question, because I get asked this all the time. A lot of companies start SRE with an SRE center of excellence, a group that goes out and figures out SRE and kind of takes this SRE as a service type mindset. I think it can work. Most of the times I've seen it, it doesn't. But what's your take on that?

Nathen Harvey:
Yeah. So, first, like, my hackles go up anytime someone says "center of excellence" because I personally have a huge disdain for a center of excellence, and here's how I kind of pitch it, Mike. Imagine that we do have a center of excellence and I'm in the center of excellence but you're not. Like, where do you work every day? You work in the pond of mediocrity, right? You're not in the center of excellence! *[Laughter]* You're over there being mediocre. Like, that's terrible. So, I think the term I prefer is a community of practice, and the practices that I prefer are a community of practice where we have a group of likeminded individuals or at least a group of people who are interested in similar principles, similar topics like SRE, and those folks get together and figure out, like, what does SRE mean for us? And the truth is, within an organization, it is best to start with one team or one service and figure out what does SRE look for here within my organization? And let's start experimenting with that. When we see that working, now let's start replicating that success across other teams within the organization.

When you have a center of excellence – a "center of excellence" – I'll use air quotes there. You probably can't see them if you're listening to the audio podcast. *[Laughter]* But this center of excellence, like, you have teams that just kind of wait for the center of excellence to lay down an edict, and then they see the edict, they see the new policy, and they realize that it's so far disconnected from the work we do every day that I can't implement this and I'm going to push back on it, or I'm just going to simply ignore it. So, it really comes down to how do we put these things into practice, and so a community of practice to me is much more inclusive than it is exclusive, like a center of excellence. And it really is about how do we put these ideas into practice within our organization.

Mike Kavis:
The ponds of mediocrity. That sounds like a great blog post title. I love it. *[Laughter]* I'm going to steal that. I'm definitely going to steal that.

Nathen Harvey:
Please do.

Mike Kavis:
Well, Nathen, thanks for you great advice today on SRE. I learned a couple things along the way, as always. That's why I like doing podcasts.

Nathen Harvey:
Absolutely.

Mike Kavis:
And where can we find you out there in the world of Twitter or if there are any presentations you've got out there we can go check out and all that good stuff?

Nathen Harvey:
Yeah, for sure. Definitely you can find me on Twitter. I'm @NathenHarvey, but just a word to the wise or a thanks to my father. My father misspelled my name, so it's N-A-T-H-E-N, H-A-R-V-E-Y. That's my Twitter. You can find me on LinkedIn, GitHub, basically all under the same name. And I do try to publish presentations. Let's see. I have a workshop coming up at Datadog Dash. I'll be at GitHub Commit giving a presentation there as well. Those are the two that I have coming up very soon here.

Mike Kavis:
Great. We look forward to seeing those. So, that's it for today's episode of Architecting the Cloud. To learn more about Deloitte or to read today's show notes, just simply head over to www.DeloitteCloudPodcast.com. You can find more podcasts by me and my colleague Dave Linthicum by searching for Deloitte On Cloud Podcast on iTunes or wherever you get your podcasts. I'm your host Mike Kavis. If you'd like to contact me, I'm on Twitter, @MadGreek65, or you can just e-mail me at MKavis@Deloitte.com. Thanks for listening and we'll see you next time at Architecting the Cloud. Thank you, sir, enjoyed it.

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.