Deloitte.

On Cloud Podcast
February 2022

The Deloitte On Cloud Podcast

Mike Kavis, Managing Director, Deloitte Consulting LLP

Title:	**It's all here: inside the 2021 "Accelerate State of DevOps" report**
Description:	The 2021 "Accelerate State of DevOps" report from Google is out! In this episode, Mike Kavis talks with Google's Nathen Harvey and Deloitte's Manoj Mishra about the report's most compelling findings. Among them: DevOps and SRE are complementary; successful SRE equals knowing your customer; documentation is critical, but it should be organic; and there's a newly-announced "Reliability" metric. The group also discusses customer satisfaction and the new "Release Management" role in DevOps/SRE.
Duration:	**00:27:53**

Mike Kavis:
Hey, everyone. Welcome back to the Architecting the Cloud Podcast. I'm your host Mike Kavis, and today on this show I am joined by repeat offender Nathen Harvey, making his second appearance, developer advocate at Google, and a colleague of mine, Manoj Mishra, managing director of Deloitte's Agile and DevOps practice. So, today what we are going to do is we are going to talk about the recent—Google's Cloud State of DevOps report, and we're going to talk about it with these two. But I just want to talk a little bit about the report before we get into questions.

This was the seventh, I believe, annual one. I have covered every single one of them in some form or fashion. I used to write a recap post every year. In my older days I'm getting lazy, and I just do podcasts on them now. It's a lot less work for me. But about 32,000 people worldwide participate in the survey that makes up the findings that come out of this, and that's what we're going to talk about today. Before I ask my guests to introduce themselves, Nathen, where can people find this report? Because we're not going to cover every finding; we're going to cover the ones that I like the most. But where can we find this?

Nathen Harvey:
Yes, of course. Thanks, Mike, and it's good to be back. The best place to find the "State of DevOps" report is to go to Cloud.Google.com/DevOps. From there you'll be able to download the report.

Mike Kavis:
Cool, thanks. So, while I've got you talking, just introduce yourself for the audience here.

Nathen Harvey:
Yes, sure. So, I'm Nathen Harvey. As Mike mentioned, I'm a developer advocate here with Google Cloud. And I've really spent—I don't know—the past decade-plus in the DevOps community helping teams and running teams that are building, deploying, and operating software. A lot of that software happens to live in the cloud, but certainly not all of it.

Mike Kavis:
Cool, cool. And, Manoj, welcome to the show—first time. Tell us a little about yourself.

Manoj Mishra:
Thanks, Mike, and it's really a pleasure to be with you and Nathen. So, my name is Manoj Mishra, and I lead the business agility practice for Deloitte. I focus on all things Agile and DevOps and how to get them to work together to deliver better software, really.

Mike Kavis:
Cool, great. Great to have you, and we've worked together over the years here delivering a lot of value to our customers. So, the first finding is one that says—one of the key findings of the report is that SRE and DevOps are complementary philosophies. And I was very happy to see that, because I can't count the number of articles that say, "DevOps is dead; here's SRE," or, "SRE's dead and here's DevOps," or, "DevOps or SRE." This is an "and." So, I'll start with Nathen. Tell us a little—I just hit the highlight but tell us a little detail about that finding and your thoughts on this finding.

Nathen Harvey:
Yes, sure. So, of course site reliability engineering, or SRE, was really a practice that was born at Google, and then Google brought out into the world. And like you said, ever since Google mentioned SRE outside of the building, if you will, there's been this fighting and this big, like, consternation. Should I do DevOps or should I do SRE? And really as part of this year's research we wanted to bring the data, bring the science to this question of SRE and how are DevOps and SRE truly related. And like you said, they are complementary.

We find that teams—in fact, we found that 52 percent of the teams that we're working with are adopting some sort of SRE practices, so 52 percent of the survey respondents. Those folks are seeing improved software delivery performance in the same way that you see improved software delivery performance when you bring in these DevOps capabilities. But I think the most important thing that really says that they're complementary is that software delivery performance drives organizational performance. Guess what else drives organizational performance: reliability performance. So, these two together really help push your organization forward.

Mike Kavis:
Great. That's a great explanation. Manoj, what's your take on this? What are you seeing? You probably see some customers where it's and/or, and you probably see some customers where it's an and. So, what's your take on this?

Manoj Mishra:
I always found it funny that people were always thinking of DevOps or SRE. That was an interesting conversation, because, frankly enough, I always felt that DevOps is not complete because it focuses on one area, and then when SRE came in—SRE does not really compete because it focuses on another area. When you merge them together, and you work with it together, you really can improve the quality of your software and how to do better development. So, really the way I see it, and I'm seeing more and more customers of mine recognize that, right, and they are seeing that you need to have both in order to be successful.

Mike Kavis:
Yes. One thing I've seen, unfortunately, is a lot of SRE implementations is the renaming of existing operations teams. What are some of the characteristics of a successful SRE implementation? I'll start with you, Manoj.

Manoj Mishra:
Yes, so you are right. I think we do see a lot of Ops teams basically becoming—renaming their monitoring and incident management team as SRE in many cases, right? What I have seen is those companies who are trying to integrate better with their Dev teams and their business, right, in order to have a really, truly product manager driving it, driving what comes in and working with them together, that's the key technique of trying to make sure that you have the complete value stream that you are looking at, and not just one component of it and making it successful. So, it's a difficult concept because many—sometimes the Operations team does not have the view of the business, right, and does not have the view of what's coming in. But I think drawing the box so that it takes you all through to the business teams is something very critical in order to make it successful.

Mike Kavis:
Yes, there's something you said there resonated with me when you mentioned the product, and I think to me that's—a key is no matter if your SRE is infrastructure-focused or application-focused, it should be product-focused. And too often a, like, central operations team who serves everyone, but no

product in specific is trying to be SRE, and it's like, “Well, what are you doing the reliability of? Everything?” That doesn't make sense. So, I think when there's alignment with product, that seems to work better. What do you think, Nathen?

Nathen Harvey:
Well, first I want to take a little bit of issue with your opening statement, people just renaming folks. Look, I think the right way to success as a former system administrator is to change your title from SysAdmin to DevOps, and then to go from DevOps to SRE. And in each one of those you take a step higher in your pay, so if you get 20 percent more with each of those title changes, that's perfect. That's all we need really.

No, no, no. All right, so seriously *[Laughter]* I think, look, one of the challenges I see with SRE and spinning up SRE teams is we now as an organization say, "Okay, now I have an SRE team. They're responsible for reliability. It's no longer my concern." But the truth is, like Manoj said, it's about bringing everyone together on this reliability journey and really thinking about who are the customers of this service, because one of the fundamental practices of SRE, or principles of SRE, is that we want to keep our services reliable, so reliable that a typical customer will be happy. We don't necessarily need to go beyond that; we just want to keep our customers happy. But if we don't know who our customers are—and I'll tell you what, your customer is not the network switch. It's not the container that you're about to deploy into that cluster. Those aren't your customers. You have to understand who's using the software.

Mike Kavis:
Absolutely. So, the next finding I find awesome, and I think this is going to spawn a really good discussion, and that is good documentation is foundational for successfully implementing DevOps capabilities. And I think there's going to be a camp of listeners who are going to go, "See? I told you. You need you that 80-page document before you go do design." And then there's going to be people on the other side going, "I'm not doing any documentation." So, we'll start with you, Nathen. Tell us a little bit about that finding and the data behind it and your thoughts on it.

Nathen Harvey:
So, here are some of the big stats that we found, Mike. Teams with quality documentation, they're 3.8 times more likely to implement security best practices. They're also 2.4 times more likely to reach their reliability targets. So, look, documentation matters, but it's not just about having the documentation. We really need documentation that helps the readers of that documentation accomplish their goals. And one of the goals that people have—the software is down; we need to bring it back up. Will the documentation help me achieve that? Is it an accurate description of the system as it is today? And that's really important. Is the documentation accurate and up to date? And is it findable? Can I find it within my organization? Is it in the places where I expect it to be? All of this really, really matters.

Mike Kavis:
So, Manoj, I'm going to pass it to you and give you kind of the hard one. Based on what he said, what do we change so that this stuff is findable, it is updated? Because have you ever seen a CMDB that was current and accurate? *[Laughter]*

Manoj Mishra:
Interesting question, right? Correct? Look my point of view is we need to somehow redefine documentation so that it does not mean an 80-page, separate document that I'm preparing in order to tell people what my software does and what my—I think that's what we need to change. With documentation, I think Nathen kind of mentioned it, that it needs to be in place to be find at the right time, right? You need to find it at the right time, and the best way to do it is to document the code where it exists, right, or document the operation where it happens, right? I mean, if certain things are supposed to run at certain point of time of the day, document it over there as to why does it run that way, right?

So, the more we can make documentation just part of the ecosystem and not make it into a large document or everything that we are creating, the better it is. I'm working with one of the health-sciences customers where we are looking at the SDLC, and they have this onerous SDLC software that they have built which makes every—whether you are the project manager or the scrum master, or the—to go to that portal and do stuff over there, and then come back and get back to your software development, right, or get back to your user stories that you are writing. We are changing that. We are rewriting that whole thing so that it just gets embedded into everything that you are doing, right, and you are not having to have a separate process for that.

Nathen Harvey:
I think all too often documentation is the obituary of what our systems used to look like, right? Let's write about how they used to be. Because we don't go back and update them, to your point, and we don't go back and update them, one, because maybe it's a difficult process and it takes us out of our flow. But I think there's also something really important, and this is especially for leaders of organizations. We have to recognize that documentation work is important, and we have to recognize that during performance reviews, during promotion discussions. We have to prioritize and incentivize the people that understand the systems writing that documentation and keeping that up to date through clear processes and clear rewards for having done so.

Mike Kavis:
So, to either one of you who wants to answer this one for me, when I think of architects establishing patterns, isn't that a form of documentation? So, like, say I need to build this highly-reliable system that leverages MongoDB and something else. If I have pre-published patterns, is that a form of documentation? Or not? *[Laughter]* Anyone want to take a shot at that one?

Manoj Mishra:
Nathen, you want to go?

Nathen Harvey:
Yes, sure, I can take a stab at that one. Yes, I think that those patterns aren't necessarily, in and of themselves, documentation, but what they do is that they help establish a clear mental model of how systems are working. So, really, I guess kind of the answer, Mike, is, it depends. It depends on how well that pattern is documented, how well that pattern is understood, but then probably most importantly, is that pattern actually followed with this system?

Because if you end up in a place where I'm trying to troubleshoot a system that's down, or I'm trying to work with a new system, and I understand what the patterns documentation says but the system is implemented in a way that's different to that, that's where I'm going to have a big disconnect. So, I think to the extent that those patterns are actually followed—and then I think the following of those patterns has to feed back into the definition of those patterns, right? It's all about closing these feedback loops. That I think is the important bit.

Mike Kavis:
So, "it depends" is a consulting answer and you're talking to consultants here, so we are hiring. No, I'm just kidding. *[Laughter]*

Manoj Mishra:
Yes, Mike, just my thought, right? I think patterns are documentation, but they are implemented as something right? And what you really need to do is to make sure that the implementation—the feedback loop into the documentation of the pattern—is automated. I think that's where we mostly lose the connection and then it becomes outdated and it's no longer useful as documentation, right?

I would rather have someone write a script to generate the pattern, right, what is being used in that case, rather than having—whether it's a PowerPoint or any other form of documentation that was created in the beginning. I would actually make it as a part, so that I generate the documentation from the system and then correlate the real document.

Mike Kavis:
Hmm, absolutely. So, the next finding—and this was another one that was welcome to me. So, we've been talking about the four key DevOps metrics for years. That's kind of been established. And there's two on throughput, deployment frequency and lead time to change, and there's two about stability, which is time to restore a service and the change failure rate. This year we added one on reliability, which I think was the missing link, right? I mean, we could hit all those metrics there, but if our system isn't reliable, who really cares, right? And, so, what I want to do is go to Nathen because he knows this report much better than us here at Deloitte. But tell us exactly what this reliability-based metric is and what brought that to the forefront this year?

Nathen Harvey:
Yes, for sure. So, those four key metrics that you talk about are really—are about measuring software delivery performance. And as you mentioned, it's not just software delivery performance that matters. We want to take a look into that operational side And, so, we look at reliability as the extent to which teams are doing those sorts of modern operational practices Are you driving your reliability metrics based on users of the system? Are you using those reliability scores or the input from reliability to help you prioritize the engineering work that you're doing? And, so, those types of things really help us identify how is operations going for this particular organization, or for this particular team.

Mike Kavis:
So, within that, one metric I always think that's missing is customer satisfaction. So, yes, we can deliver it fast, I can make it really reliable, but the customer doesn't—that's not what they want. So, why don't we ever talk about that one?

Nathen Harvey:
Yes, I think this is a really important thing, obviously. If what we're doing is measuring can our customers use the system reliably, can they depend on it, that is one measure of customer happiness, but it's not a measure that tells us are we shipping them the right features. Is this the feature set that they want? And frankly, that is an important question to ask. I might argue that it falls outside of the realm of the research and certainly falls outside of the practice of software delivery.

And it's kind of based on this hypothesis that if we get good at delivering software fast and reliably, the folks that are really focused on product fit and really deeply understanding what the customer need is, as they generate ideas and new features, we can get them into market and learn from them much, much faster. And, so, we kind of separate those two concerns a little bit, but the idea is that by improving that software delivery and operations performance, we are making it easier to run more experiments, to learn more about exactly what our customers want, and get closer and closer to delivering that value to them.

How do we take all these great performance metrics on software delivery and merge that with are we getting the customer the right stuff?

Manoj Mishra:
Right, absolutely, and that's where—you are right. I think that's where Agile and DevOps combine together to deliver the product. I am almost thinking we need to—instead of calling it DevOps, we need to call it ProDevOps or something, Product DevOps basically, right? I mean, that's the way we need to start thinking about it, because it extends and connects you back toward the customer needs, how you prioritize those requirements, how you approve them for your portfolio, and then how you're delivering that with the right quality of software, correct, right? So, I think that's why it's very, very important.

Another point, Mike—you asked about availability versus reliability, right? I always thought availability was an incomplete metric, because—and it became very prominent in the early 2000s, when infrastructure was a problem, and people were so much obsessed about five nines and six nines and eight nines in terms of availability, right?

Nowadays with cloud, that's not really a problem. Your systems are always available. But are they providing the right response time? What's the latency, right, or what are the latency kinds of issues, right? Are they working the way it's supposed to work, right? Those are all the other factors which come into the picture when you think of reliability, right? So, that's why I love these new metrics much more, because availability is just a component of it.

Mike Kavis:
And I'm going to tie this into SRE a little bit. So, a lot of what you just talked about, to me, are things that SREs should be focusing on, the "50 percent of time" they're not operating. And too often our clients are thinking about SRE in operations-only mode and not in how can I improve the overall architecture, processes, flow, et cetera, of this service, this product, so proactively it could be more reliable? So, thoughts on that? I'll go right back to you, Manoj, on that.

Manoj Mishra:

There is a very nice slide I use in many of my discussions where it talks about the SRE responsibility, right, correct, right? And it says that 50 percent of their time is in operations and 50 percent of their time is in enhancements and process improvements and on the left side that you talked about, right? And the SRE's job is to take that out and move that towards the 80/20 ratio. They should really be automating everything and getting to a place where only 20 percent of their effort is really on day-to-day operations, break-fix, and all the stuff that they need to do, but 80 percent of their time, they are working with the product teams and everyone to improve what comes into production, right? Correct if you think about it, right? So, I think—I mean, that's the way I think about it, Mike.

Mike Kavis:
Nathen, your thoughts?

Nathen Harvey:
Yes, I think you're spot on there. And I think within Google's SRE practice, we have a saying that's always used. "We're here to make tomorrow better than today," right? And, so, we're always trying to invest in those improvements. We're looking for toil in the system that we can eliminate. Toil is that work that doesn't provide any lasting value, right? How do we either get rid of that or automate that away? And it's funny, I think, because when we talk about product, when we talk about DevOps, when we talk about SRE, like, what we're talking about is making smaller and smaller changes, getting them through the system faster, and learning from them. So, it's really about taking almost the scientific, experimental approach to learning and continuous improvement. So, I think it really ties everything together, and maybe we need to rename it. Maybe you're right, Manoj. Maybe it's rugged enterprise DevSecProdNetQAGovCustOps. What do you think? *[Laughter]*

Manoj Mishra:
Yes! All of these I can see the new terms emerging there! DesignOps is a big one, right? Architectural design–

Mike Kavis:
Oh, yes, GitOps, ThisOps, ThatOps. Yes. It gets silly after a while. So, last question, and this is one because I'm seeing a lot of this, is the concept of release management. In the new world, one could argue that release management is something that can be fully automated in CI/CD pipeline. A lot of clients are standing up a release management organization, which may or may not be good. To me it's not good if it's a gate that I have to go through. It's good if it's a set of policies and practices that you must implement in CI/CD and in your cloud platforms. But I'll turn to you, Nathen. What are your thoughts on what's the role of a release management organization, or group now, in this kind of new world?

Nathen Harvey:
Yes. I think at a very high level their role is the same as everyone else's, and that is to improve those four or five key metrics, right? Improve the speed with which we can get a change through our pipeline. Improve the quality of that change when it lands in production. And, so, I think when we look at large organizations that have essentially one path to production, from the developer's workstation all the way through to production, you don't get there by waving a magic wand. You do get there by putting a lot of engineering and a lot of thought into that. And, so, I think having engineers that are dedicated to that problem, as long as they're moving towards that end goal, right—what are the outcomes that we're measuring, and we have to take that systems-wide view. Let's make some investments in the release process, absolutely, absolutely.

Mike Kavis:
Manoj, what are you seeing?

Manoj Mishra:
Oh, man, Nathen is absolutely right, and you are also right that I am starting to see this trend of people standing up release management organizations, right? If I have the say in what they should be doing, basically they should be setting up the policies and processes and all to make the releases faster, repeatable, more secure.

Those are the things they should focus on. But they should stay away from the manual intervention. Everything, every process that they do, needs to be automated and built into the process, so that the developer can really take and expedite their code through to release, right? There is perhaps a place for them because there is an outside view of what checks and balances you should have. But don't make the checks and balances manual and stage gate processes and things like that. Make it automated into the pipeline so that you can release it into production.

Mike Kavis:
Totally agree. Well, gentlemen, great, great talk today, a topic that we're all passionate about. Manoj, where can we find you? Do you have a Twitter account or LinkedIn or e-mail, whatever? Where's the best place to reach out to you and do you have any content out there that we can grab?

Manoj Mishra:
Oh, absolutely. LinkedIn is perhaps the best way, so LinkedIn, and then Meet Manoj is my short term, so I think people can use that, right? Twitter also, same thing—I think @MeetManoj is there, too. They could use that. And, otherwise, I'm also on Deloitte.com. They can search my name and they can come across the profile and send me a note through there.

Mike Kavis:
Great. Nathen?

Nathen Harvey:
Yes, Twitter and LinkedIn are both great for me. Just be aware that my father misspelled my name, so it's N-A-T-H-E-N, N-A-T-H-E-N H-A-R-V-E-Y. So, if you search there, you'll find me. If you search with a more traditional spelling, you'll find someone else, but I'm sure that they're very, very kind and would answer your question just as well.

Mike Kavis:
So, what's your Twitter handle? Is it your full name?

Nathen Harvey:
It is. It's my full name. It is @NathenHarvey, yep, yep.

Mike Kavis:
Okay, great.

Manoj Mishra:
It's perfect because your name is unique now, right? So, –

Nathen Harvey:
That's right. It's because my father was a brilliant futurist and said, "Everyone needs a unique name!" And, so, me and all my siblings, we've won that lottery, I guess, yes. *[Laughter]*

Mike Kavis:
Well, great talking with you both, great having you on again, Nathen. That's it for today's episode of Architecting the Cloud. To learn more about Deloitte or read today's show notes, head over to www.DeloitteCloudPodcast.com. You can find more podcasts by myself and my colleague David 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 via e-mail, MKavis@Deloitte.com, or you can fine me on Twitter, @MadGreek65. Thanks for listening and see you next time on Architecting the Cloud.

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.

Visit the On Cloud library
www.deloitte.com/us/cloud-podcast