Architecting the Cloud, part of the On Cloud Podcast

Mike Kavis, Managing Director, Deloitte Consulting LLP

Title: **Want better apps? Focus on value and optimize workflow**

Description: Cloud app delivery runs on tight timelines. Unfortunately, other organizational processes, like procurement or legal, often create workflow bottlenecks that hobble the software delivery process. In this podcast, Mike Kavis and guests, Chris Young and Mattia Battiston, discuss teams, workflow, and value delivery. Their advice: IT and business must work collaboratively to manage change, identify and clear bottlenecks, and understand where value lies to optimize workflow.

Duration: **00:33:10**

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, and welcome back to the Architecting the Cloud Podcast, where we get real about cloud technology. We discuss all the hot topics around cloud computing, but most importantly with the people in the field who are doing the work, and in many cases writing the book. And speaking of writing the book, yeah, I'm your host, Mike Kavis, chief cloud architect over at Deloitte, and I am joined by a couple authors today, Mattia Battiston and Chris Young,

authors of *Team Guide to Metrics for Business Decisions*. So, gentlemen, welcome to the show. Chris, we'll go to you first for alphabetical order, but just—on the first name, and just tell us about your background and what drove you to write this book.

Chris Young:
Well, first of all, thanks so much for having us on the show, really appreciate it. I was approached by Matthew Skelton, who at the time was not the kind of stellar visionary of Team Topologies that he is now. But he had the idea of pairing me with Mattia to write a book on metrics, because he'd actually done some work with me at a startup called Honeycomb.tv, where we were putting in cloud infrastructure. And before we put that cloud infrastructure in place, we wanted to have some kind of idea of, like, had we actually made any difference by putting that in place, had we moved the needle at all. So, we both shared this interest in how you measure stuff, how you use metrics to create a shared understanding of the state of things, and then when you've done a piece of work you can check in on them and say, "Okay, did anything change?" Yeah, so Matthew reached out to me, and about four years later, after a lot of—writing's really hard work, I must say—after a lot of work, here's the book.

Mike Kavis:
Cool, cool. What about you?

Mattia Battiston:
Yeah, so my day job is as an engineering manager, so I focus a lot on helping teams reach their level of high performance, working well as a team. And in this job, I see so many teams struggle answering questions like how long will this take, what should we do next, how are we improving, where should we improve next? So, I started using data metrics for helping my teams improve in this area and then did some talks and ended up in touch with Matthew and Chris, and we thought, "Why not? Let's pair up, share some of the lessons, the things that have worked well for us, and see if people find a use for it." And hopefully they do.

Mike Kavis:
Yeah, so this topic is near and dear to my heart. I've been in this cloud journey for a while, first as CTO of a startup and then eventually as a consultant these last eight or nine years, helping companies move to the cloud. And I kind of have this timeline in my mind of what I've seen. Starting back in 2008, most of the people in cloud were trying to figure out how to build, right? And then we kind of got past that to some extent, and then there was a lot of how do we do CI/CD pipelines and all of that, which I think a lot of companies that have been at it for a while kind of got that down. And then it became how do we run, because we didn't think about that.

And then some are getting better in that, but now it's about are we working on the right things, and that's where stuff like flow metrics and these things are coming in. And I was just working with a client who is pretty advanced in—if we were to do like the DORA assessment of DevOps, they would score really high. But they were struggling on working on the right things and juggling tech depth versus new features and who owns reliability, is it the ops guys, is it everybody—that's where—the companies that have been at it for a while, that's where they're struggling. Are we spending the time right? Are we working on the right things? Are we doing too much local optimization?

So, I was kind of looking for there's a lot of traditional metrics, but there wasn't stuff about—I couldn't find a lot of good stuff about are we working on the right things and is the flow right. And I stumbled across your book, and, so, that's why I was excited to invite you guys here. So, with all that said to set the context, there's a couple points when I read the book I wanted to bring out there. And you mentioned that time is the scarcest resource, and it is often wasted if we don't know what to work on next. So, what metrics help guide teams to figuring out where's the best time to spend?

Chris Young:
I think I wrote that line so I feel I should be on point to start, but I think—but I'll say a bit and then I'll hand to Mattia to say as well. It's a really difficult question. There's no shortage of good ideas for what to spend time working on. But when you have a—most teams are finite in size. Most teams can only work on so many things at once. So, deciding what to work on and when to work on it is absolutely critical, because otherwise you could be working on something that has no value at all to the business. You could be working on something that may be valuable, but may be valuable further on down the line. So, if you don't—my experience has been if you don't try and come up with some kind of model of what effective change looks like, it just becomes who shouts loudest or the hippo, which I think is the highest-income person in the room. So, it's the kind of—it becomes kind of like sort of pulling rank and saying "Well, I'm CTO so we're doing this," or, "I'm head of sales so we're doing this."

So, to get out of that kind of opinion space and to come up with some kind of—something that everyone can say, "Yeah, we agree that, even if this is not right, models are wrong but useful. If we can come up with something that somehow reflects what we think effective change looks like, what we think a functioning business looks like, then we can use that to say, “Okay, well, let's see if when we change something, the measures we see from that model are affected and things are better." So, I love the Dave McClure pirate metrics, the AARRR (the acquisition, activation, referral, retention, and revenue) because they're all about the kind of dynamics of people around the platform or the product.

But I think you have to go beyond that and you've got to work with your—if you're a tech person, work with your business colleagues to really understand what value means. And that's—what I like about that is it really encourages you to get out of your tech comfort zone—I'm a dyed-in-the-wool techie and I'd rather just be at the command line all the time, but it prompts you to go out and say, "Well, okay, so how do we know this is doing better?" So—well, we might say, "Okay, well, we can see we're having more orders for these high-value products,” or “We're spending less money on fielding support cause because the platform's working better," or that sort of thing.

So, I think you can start with a kind of generic model, and then you can use that generic model to start asking specific questions that people from different parts of the business with different perspectives then can work together to come up with a sort of more nuanced and finessed model that can then be used to say, "Okay, if we see these indicators move, we know we're doing something right.” So, if we're working on something and we know there's an area where we're weak, so maybe we're saying, “We're not seeing any sales in this geography. Can we do something to support sales in that geography?” Or, “It's taking a very long time to respond to these particular types of customer requests." You now, if it's a particular—if you're offering a service—I've done a lot of work around video coding. You might say, "Okay, we're having real problems with our dashing coding. It's taking us typically a day to get the configurations of the dashing code right for a customer." You might say, "What if we could get that down to half a day? Maybe we'd have more people using the dash service." And, so, it's coming up with something that is going to be domain-specific and going to be context-specific, but by having those

conversations, by working across tech and business, I think you build that shared understanding, you create that empathy, and that helps you kind of work better. That's my take on it. I'll hand to you, Mattia, to hear what you think.

Mattia Battiston:
Yeah, yeah, so what you explained is basically the metrics for value that we talk about in the book. And it's absolutely essentially that we know why we are working on something and that we select what to work on next because of the value that it brings to our teams, our companies. I don't think I can add much to that. You explained it so well. But part of the original question is also when should we work on something, because –

Chris Young:
A hundred percent.

Mattia Battiston:
Very often there are lots of options for what to work on next. And we can take a guess at the value of each option, but quite often it's just a wild guess. So, how do we pick what to work on next and when to work on something? And I've got an example, that is actually in the book, that I think is a good story of taking—using some of the more operational metrics for taking such a decision.

This was—I was working in a team in the broadband domain, so we were creating tools for customer support basically, tools used by call-center agents to support customers when they were having problems. And this was in a bit of a legacy call base, so for technical reasons we only released this software once a month. It was back in the days when continuous delivery maybe wasn't quite a big thing yet. And, basically, the managers of the call centers came to my team saying, "We've got this new feature that we would like to release. It's going to help us diagnose better this particular type of problem that some customers have got." But we knew that the next scheduled release was in five days' time, so the question for my team was, “Well, can you work on this and make it part of the next release? Can you finish this feature in five days? Because I need to know. I need to train my staff in the call center if it's going to be ready in time.”

And what I did with my team back then is, we looked at some of our historical data. We used a metric called lead time, which is the time it takes for a piece of work from being started to being finished, to being in the hands of our customer delivering value. And using a particular way to look at lead time, which is a metric called lead-time distribution, we could see that only 50 percent of the time we were able to finish a whole feature in five days or less. So, we had this sort of discussion with these call center managers. "We have only a 50-percent chance of having this feature ready in time for the next release, so do you want to take the risk, and should we work on this now with the risk that it might not be ready in time, and you might train your staff, and then the feature is not actually ready?" And they decided that it was too big of a risk for them. Training the whole staff on this with the risk of it not being ready in time was not worth it for them. So, we worked on something else instead, and the following month, instead, we made sure that that feature was part of it.

Mike Kavis:
Yeah, a decision based on data is better than the hippo method then, right? Someone slams their first, says, "Do this," and then you've got a half-baked solution that creates more and more technical debt and all that. Cool. So, one of the challenges I've seen—value, right, you want to deliver value to the customer, but there's different customers in the stack of this, right? Because a lot of companies will have a platform team. Underneath that, they may have an infrastructure team. Underneath that, sometimes they have an ops team. I mean, hopefully, we're moving towards ops embedded in these other areas, but I see that a lot. And every layer above you is your customer, right? And, so, they're all delivering different value, but sometimes what I see at certain clients if that, if you're not in the product layer, it's hard to get your value prioritized. How do you solve—I don't know if solve is the right word, but how do you mitigate against that?

Chris Young:
The best method I've found for this is Gojko Adzic's impact mapping, which is a—I don't know if you've come across this before, but it's a way of working from a measurable goal through to options for doing work. And the absolute genius of Gojko Adzic's method is that he—as you said, he brings in these different people from different parts of the stack. You start by saying, "Okay, what's the measurable outcome we want to get?" Then he says, "Who can either help or hinder us achieving that outcome?" And that stops you in your tracks and gets you out of this kind of pure product layer, stops you thinking about the customer—or all too often my experience is you don't go anywhere near the customer. You just go to the product manager, and the product manager's a proxy. And if you have run an impact mapping workshop, and you bring in people from product, you bring in people from operations, you bring in people from sales, you bring in people maybe from legal, or from marketing, you get everyone together in a room and you work—you say, "What's the thing we're working on now?"

It might be, okay, we want to get a particular new piece of functionality on the platform live in a month's time or something. And you look at how can these people help or hinder us, and then you say, "Okay, if they're going to help or hinder us, what's going to change in their behavior? What are they going to be able to do more or less of?" So, for the operations example you might say, "Well, somebody in operations might be able to diagnose a fault in half the time." Okay, great—that'd be brilliant, okay, if they can diagnose a fault quicker, or they can get the platform up and running. And then you can get into the options. "Okay, well, what could we do to help effect this change in their behavior?" It might be better logging, or it might be a run book that says here's how to deal with this particular problem. Or it might be a way of actually stopping the problem appearing in the first time.

So, it's I think something that encourages you to think about everyone who's impacted by your software platform or the work you're doing and getting out of the kind of abstract thinking of just the user or the customer, getting explicit and saying, “Well, we have our,”— I'll use the video example again. You might have somebody sitting at home watching video on their couch, or you might have somebody who's trafficking ads to a video platform, or you might have somebody who's selling media on a video platform. You make those things explicit, and then you say, "Okay, now bring in all the people who are the enablers of this. So, bring in operations. Bring in sales and, so, forth." And you kind of create a kind of—I think—is it called a forcing function? You bring people together and you create a kind of—almost that you sort of raise the temperature and say, "Okay, we've only got so much time." Again, time, the scarcest commodity we have. You want to get this done by this date. All these people are—they're part of this thing succeeding or failing. “Where do we want to place our chips on the table? Where do we want to do this?”

So, it's kind of—it's making it about the people and getting everyone who has—everyone who comes with their idea of, “Oh, yeah, my user or my stakeholder is the most important.” Bring that cognitive bias into the room and have it countered by somebody else. So, try and not have, like, one to one

conversations with say products or sales or ops. Get everyone together and say, "Right, you're all going to be impacted by the changes we make to the platform. Not everyone is going to get everything they want, well, certainly not straight away, so let's have a grownup conversation about this and let's leave our kind of ego and bias at the door, or recognize our bias is there." And be opened to being challenged and kind of cognitively told, "Hang on a minute. There might be another way of doing this."

Mike Kavis:
Yeah, and that requires a lot of culture change, right? I mean –

Chris Young:
It does, yeah, yeah. I'm glad you called me up on that, because it's non-trivial. *[Laughter]*

Mattia Battiston:
There are maybe a couple of metrics or pieces of data that we could use to look at this. Again, one way I would look at this is under the lens of flow, especially if we look at the flow for a company level, if we're trying to deliver a product from someone's idea to being a fully-fledged product in the hands of customers. It's not enough to get this thing to do a piece of work. We also need the platform to work well together. We need legal to do their part. We need everyone to do their part. So, if we map the whole value stream at that level, then it becomes a lot easier to, what, manage the flow, manage this whole end-to-end flow and notice when this work should get prioritized.

Chris Young:
And I was going to say, I'm a big fan of value stream mapping. I mean, I'm a visual thinker. I have to have things on whiteboards, and, sometimes I don't recognize that everyone doesn't think like that. But I'm really glad that Mattie brings up the point of the non-technical workstreams being just as important. I worked on an IOS app, and the best thing that happened on that IOS app project was getting marketing involved on day one, because they immediately looked at what are the criteria for submitting an app to the App Store? And it was like, okay, you've got to have screenshots. You've got to have copy. That copy's got to be approved by legal. And then we drew out the value stream map and said, "Okay, here's the technical stream. Here's the legal stream. Here's the marketing stream." And then suddenly people were talking to each other. Everyone knew that there would be no point in development flying along and being—the flow could be going really fast there—I don't know why I'm saying it would be development would be going really fast, but let's just say it is. But if something's stuck in legal or waiting for approval in marketing, there's no point.

So, you have to—you think of the thing as a whole and look at it—so Mattia talked about throughput before. I think throughput is the absolutely—sorry, you talked about lead time, didn't you? Again, so lead time and throughput are my two actually, sort of anchor metrics. Without them, you can't really make any kind of decisions or actions on flow. And if, say, the lead time for getting something signed off by legal is two weeks, maybe, because you've only got so many people in the legal department, there's other things they need to do, then that you need to get your request in early, or that there's no point putting it in the day before the submission to the App Store or whatever.

So, I really—I like what Mattia's saying there and I'm really glad he brings up value stream mapping, because that's a—that, and there's another sort of type of thing called a—I only discovered this recently—called a service blueprint, which I love it. To your point before about the different people in the stack, a service blueprint is kind of like—it's like kind of an architecture diagram, but then with the people at the top, and you see all their calls kind of going down and going through, like, “Okay, here's the web tier. Here's the backend tier. Oh, here's the ops tier. Here's somebody looking at some logs.” Or whatever, or all that sort of stuff. So, those kind of—sort of system-level descriptions and, sort of pictures of the work really—they really help me.

Mike Kavis:
Yeah, I agree. I think that visualization of work and responsibility is pretty powerful. One of the first cloud transformations I worked on as a consultant—this was back, like, 2013, 2014—we had a 90-day initiative to stand up what we called the landing zone, which is the base foundational infrastructure with their guardrails. And during the process, they chose a particular SaaS log-in solution, and then we found out that the procurement process was six months, right? So, 90 days, six months. So, from that point forward whenever I start these things, we—like as you mentioned, we bring in legal, we bring in procurement, you bring in the outside—you discover those things early.

The other one I want to mention was a platform team within their sprint delivered a feature, but because they weren't in line with the product team, it couldn't be released yet. So, it sat there for a couple releases, and then the feedback we got when we were interviewing them is, we had so many things we didn't work on to get that done. And then by the time they caught up to us on that, there was changes, right? And I see that a lot. That goes back to are we working on the right thing.

Chris Young:
Well, so that—I probably sort of annoy more people than help by saying this, but I always say to people, "What are you choosing not to do?" And it's the priority—it's very easy to say something's a priority, but if you say everything's a priority, nothing's a priority. So, you've got to be focused, saying, "Okay, we're going to do this, and before we commit to it, what are we not doing?" And I love the point you make about things—the world turns and you find the feature that was ready and was passing all its tests and good to go two weeks ago. Now it's no longer needed because regulations have changed, or some other customer segment is more important. So, it's—and that's, too, again to Mattia's point about lead time. If you can get small lead times on things and you've got predictable lead times, and one of the ways you can do that is by working on very—on the smallest possible batch of work. So, you work on what's the smallest thing that can effect change. And you can't always do little things.

Sometimes you do have to do a number of things in concert and do a kind of ta-da release. But if you can get to a point where you can sort of drip-feed things into the platform, and feature flagging's a great way of doing that, kind of getting your changes in early and often, then you don't end up in the situation where you have code—kind of to use the sort of goal-to-goal analogy, you don't have code kind of sitting around on the shelf gathering dust or kind of going stale. You have code that's earning its living, and if it's not earning its living it's struck from the record. It's like one of my favorite refactorings is delete? It's –

Mike Kavis:

[Laughter] Remove-dash-RF, yeah.

Chris Young:
It's great. I love it. I love it. It's like, "I don't need this code anymore."

Mattia Battiston:
That's why it's so important to have that good flow, that short lead time, because you need to be able to get something out there as quickly as possible, really. A short feedback loop enables you to test your ideas as quickly as possible, gather feedback, and immediately realize, "Actually, what? I'm building something wasteful. Let's not spend our energy on this. Let's do something else instead."

Chris Young:
That's it. I think it's treating everything as an experiment. And I was working with somebody just recently. It was actually—the actual output was a document. It was a document that was a response to request for proposal. But in order to—from a big European trade body, but in order to put in that request for proposal, we had to write quite a lot of code to validate what we were saying in the proposal, and also to provide some screenshots and, some, like, shots from load testing and stuff to say, "Yeah—no, this is not just smoke and mirrors. This is a real thing." And throughout that engagement—it was about a month's worth of work—I kept saying to my sort of co-conspirator, "Remember, all this software here is a liability. It may look like we've got things stood up and working, but until that document gets submitted to the trade body, this software is a liability."

And lo and behold, right at the last minute, something went wrong with the load testing software and we had to leap around and find a different platform, and it was really—it was like—I was getting messages saying, "Hey, I want to send the document out tonight. Have you done the last load test?" It's like, "Give me an hour." And it's just—it was that real feeling of, like, “My God, we've invested so much in this, and we've worked iteratively, and we've pushed stuff to production. And it's like, yeah, but until the whole thing is finished and the fat lady sings, it's like—.”

Mike Kavis:
Right. *[Laughter]* So, in the last couple minutes here I just want—because I'm not sure if all my listeners are familiar with flow as a metric. So, what are the typical flow metrics? And where are people typically using them? So, I'll go to you, Mattia.

Mattia Battiston:
Yeah, yeah. I can go first this time, give Chris a break.

Chris Young:
Mattia is very much the brains of the operation on this bit.

Mattia Battiston:
Oh, no. When we say flow, just to clarify here, usually the way I would define flow is the movement of work from the start to end of a process, the movement of value through your process, whatever that process is, from starting something to finishing something. There are a few metrics that—in the community nowadays—we tend to refer to as flow metrics. Some of them are about—are a direct measure or indication of how good the flow is in your process; some of them are more indirect. So, the most famous ones that a lot of people will have heard about nowadays are maybe a more indirect indication of flow, but they would be really affected by it.

So, we're talking about lead time, which we've mentioned already in our conversation, lead time being the time it takes for something from start to finish. And, throughput is the other really common one, which some people call delivery rate or story count because it's literally just counting how many things you've done in a particular period of time. These two things are maybe proxy metrics for flow, but they give you a really good indication of how well work is moving for your process, because, if you have a bad flow, if you've got lots of bottlenecks, lots of wait times, then you're going to have a long lead time, poor quality, low throughput.

There are also some more direct measures of flow with diagrams like a community flow diagram, which can be a bit tricky to read sometimes, but buy our book. We'll teach you how to read it. *[Laughter]* There are a few tricks you can learn to spot patterns in a community flow diagram. Or, there are some easier things to look at, like a net flow diagram, which shows you whether you're starting more things than you're finishing, which is always a bad sign, or flow efficiency is such a simple metric to calculate but tells you—tells you a lot about how things are going in your team.

Mike Kavis:
So, a lot of the people talking about flow are talking in the context of building a product. Where I'm coming in is a lot on operating model design and saying, "How can we design an operating model that doesn't deteriorate flow? How can we maximize flow with an operating model?" So, for example, for one client we worked with there wasn't a clear accountability of ownership when something went down, so when something went down, you had the famous war room with 40 people on a call. And we tried to design an operating model to clearly outline who was responsible for what, and we had a customer-engagement level so everyone went there, and they worked the whole organization and gave them kind of the cloud service-provider model, right? You don't get to call the product owner of a service. You talk to your account manager, right—that type of thing. So, do you see people looking at flow metrics for organizational design? This is probably why Matthew reached out to you about this book, is he'd wrote a book about organizational design and then you got the flow part.

Chris Young:
Yes. So, it's kind of why I'm now a DevOps specialist. I mean, when I was 12 I started writing 6502 Assembly. I went up, like you, to end up being a CTO for a startup. Now I work entirely in DevOps because it's the area that I think gets neglected the most, and it's the area that if you don't get this right, no amount of building the world's best product is going to count for anything. And I'm really glad that you bring this up, talking about sort of operations and service, because you can absolutely apply these metrics to it. You can look at when we get a defect raise or when we see a 500—you could say, "We see a 500 error in the logs."

Okay, how long did it take us from seeing a 500 error in the logs to understanding why that was there and stopping it from appearing again? Or if we can correlate that 500 error to a customer complaint or something going wrong. So, I think it's really important when you raise errors to the customers you give them a unique ID so they can say, "Hey, the system doesn't work. I'm going to take this ID to the helpdesk." You can then look at the cycle time at the helpdesk and you can say, "Okay, how long did it take me from ticket being raised to ticket being resolved?" or you can look at mean time to recovery.

When the platform goes down—and I say when, because despite your best efforts, if you're in the cloud, you're dependent on a whole load of other stuff. You're dependent on, say, your cloud provider's DNS. You're dependent on the electricity supply to the datacenter, all this stuff. So, when something goes wrong, you need to be ready. As you say—you talk about the war room thing. One of the things that I talk about in the book is identifying how many people does work need to pass through before you get it resolved and can you do anything to optimize that? If you've got a problem where it's got to pass through six or seven people to resolve it, you've probably got what Chris Matts calls a staff liquidity problem. You've got towers of knowledge, and you can draw up a skills matrix and you can say, "Okay, in order to deliver this service, this part of our platform, what skills do we need and who has them? And if you haven't got them, would you like to learn about them? Or would you rather never touch this at all?"

So, it's kind of again, to Mattia's point, it's kind of bringing it back to being about the organization and the people. But it is exactly to your point. It's thinking about the flow of service and operation. It's kind of like there's flow in building the thing, and then there's flow in operating and running it. And what I love about LEAN and the LEAN metrics is you can look at them in manufacturing with Toyota. You can look at them in service design with John Seddon. You can look at them in product development with Don Reinertsen. And you can look at them in software with Dave Anderson. They all fundamentally have these same qualities of, “We're trying to affect change, we're trying to make something happen, or make something, and we don't want to spend—we want to know how long it takes us to do it. If we work on fewer things at the same time, we tend to get more stuff done and do it to a higher quality.” And we want to tell when are we working on stuff that's of value and when are we working on stuff that's detracting from value.

And it's really—you can take away all of the Japanese words, you can take away all of the jargon, and it comes down to very simple focus on what to do next, don't do too many things at once, finish things before you start other things. And it's—I don't want to use the word common sense, because a lot of common sense is just kind of habit or kind of—it's kind of biased to wherever you're coming from. But I think it is simple and it's fundamental, and it's something I use in many aspects of my life far beyond programming. Sorry, Mattia. I'm talking too much again, so let me hand to you.

Mattia Battiston:
No, no. I mean, you touch on my favorite subject, limiting work in progress. I have yet to see a single team or a single company that wouldn't benefit from limiting work in progress. It seems like common sense, and yet it's still such a controversial thing. So, few teams nowadays get the concept of, you do fewer things at one time and you'll get them done faster, better, or with faster feedback. It's still, “Well, I need to coach more teams when I start with a new thing.”

Mike Kavis:
And I think that—I don't know if you saw the latest *State of DevOps* report, but that's where things have evolved. And I think this is about their tenth year doing it, and I actually interviewed one of the people who did the report a couple of years ago and walked him through the topics. And this is where we are now. For companies that have been at this for a while, have reached a level of maturity, now it's all about people and process, right? It's about how do we—the way we do work. I think we've kind of solved the technology problems; now it's organizational, people problems.

Yeah, so great, great conversation. I could go on forever on this. Unfortunately—yeah, but most importantly, where can we get your book? And I'm sure the book has a website. Where can we find more content and information about the stuff you guys wrote about?

Chris Young:
It's definitely available on Amazon, and I think the website is BizMetrics.com—is it BizMetrics.com, or –

Mattia Battiston:
BizMetrics.com—sorry, BizMetricsBook.com.

Chris Young:
There you are, BizMetricsBook.com.

Mike Kavis:
And do you guys got Twitter handles we can handle?

Chris Young:
I'm @WorldOfChris.

Mattia Battiston:
And I'm @BattistonMattia.

Mike Kavis:
All right, we will definitely follow you, and I already got the book but I highly recommend people read that book, especially if you're in that part of your journey, where you're trying to solve those cultural, process, ways of working problems. So, again, thanks for your time, guys. To learn more about Deloitte or read today's show notes, head over to www.DeloitteCloudPodcast.com. You will 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. Again, I'm your host Mike Kavis. You can always find me on Twitter @MadGreek65 or reach out to me directly, MKavis@Deloitte.com. 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.

Visit the On Cloud library

www.deloitte.com/us/cloud-podcast