Deloitte.

Architecting the Cloud, part of the On Cloud Podcast

Mike Kavis, Managing Director, Deloitte Consulting LLP

Title:	**The cloud CTO challenge: making decisions that add value**
Description:	What's it like to be a CTO? Is it days spent in the trenches, or is it purely a lead-from-the-front gig where the job is more cerebral than hands-on? In this podcast, Mike Kavis and CrossBorder Solutions' Jim Ford take a look at the job of the modern CTO. In their wide-ranging discussion, Mike and Jim take a deep dive into the process CTOs use to make mission-critical decisions every day. They debate issues such as how deeply the CTO should be involved in the actual work, considerations of future tech vis-à-vis current decisions, and what criteria should be used in determining the right mix of products and services to obtain from cloud providers. Jim's advice? A CTO's decisions should be made on the basis of how much business value their choices create for the organization.
Duration:	**00:23:22**

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 around cloud computing, but most importantly with the people in the field who have done the work. I'm your host Mike Kavis, Chief Cloud Architect over at Deloitte, and today I'm joined by an old friend, Jim Ford. Jim is Chief Security Architect, Head of Information Security at CrossBorder Solutions. Jim, welcome to the show. Tell us a little bit about your background, and then we're going to get into some cool topics about cloud, Kubernetes, all that good stuff.

Jim Ford:
Well, thanks, Mike. I appreciate you letting me come and pontificate with you. My career's been kind of a long meander. I started out doing things for IBM and the Sunlight Education world where the Kodak, then Kodar, video would've been the size of a whole rack. And we'd have a full rack at each end with full teams running just to get this video going, not the modern internet. Went from there to learn and support AIX and commercial Unix and C, and went to ADP, built stuff for the internet, brought five of the first six internet products to market for them over a 24-year career. And, as you know, we met when we were embracing Docker for containers and moving forward, and the 2016 DockerCon where my friend Keith Fulton made the comment that we had chickens and needed chicken McNuggets on the application, and –

Mike Kavis:
I remember that.

Jim Ford:
Then everybody in the place, look, going, "Really?" *[Laughter]*

Mike Kavis:
I remember I was like, "Man, I'm hungry now."

Jim Ford:
I was glad he switched to the ice cream metaphor quickly.

Mike Kavis:
Yeah. *[Laughter]* Well, it's great to have you on here. You gave me a lot of flashbacks when you went through your career. It seems like everybody our age worked at IBM at some time for some reason. I don't know. Maybe that's all there was back in the '80s.

Jim Ford:
Yeah, but you know, I've got to tell you. It's really novel because now I've moved on to a recent startup, or a tax modernization company, that is innovating with serverless and cloud native on Amazon. I was always, "Let's not get too close to any one cloud provider," Here it's like, "How deep can we go? How much can we consume higher up the stack? How can we avoid anything that looks like infrastructure?" And since I'm doing security and compliance now, I love being able to point at the Amazon SOC 2 and the Amazon ISO and say, "Here! Here's the basis of the bottom layer of my program!"

Mike Kavis:
Yeah. Well, when you don't have all the legacy, when you're building greenfield you can go up that stack and –

Jim Ford:
It's a beautiful thing. I feel bad for any company with legacy.

Mike Kavis:
Well it's an interesting topic. So, my introduction to cloud was through a startup and the last maybe eight years has been all through consulting, so I'm dealing with really large enterprises. And they keep talking about the lock-in and you've got to be agnostic, and I keep going back to one example where—again, where a startup—people say, "Well, that startup doesn't count," but forget about the company. Talk about the topic.

We only had about ten folks, and half of them were offshore just doing the UI. The rest of us were doing the middle tier and the transaction processing, which was all the work. And at the time we were on Amazon. RDS didn't come out yet, or it was out, but it was beta and it couldn't meet our requirement. So, we spent the—the three guys who really built most of this stuff spent all their time managing Postgres across zones. It's a lot of work and then RDS became real, and we moved there and all that work went away. And then we could actually do stuff that had value, right? And yeah, I know, lock in, lock –

Jim Ford:
And that's a great point, because the question is Amazon and public cloud have the value prop of remove the undifferentiated heavy lifting. As we move up the stack we start to say, "Well, what is undifferentiated heavy lifting? What is my value add?" My value add is not running infrastructure in the public cloud using IaaS. My value add is not managing a Kubernetes cluster, unless I have some strict regulatory requirement to run my own. You know, I can find use cases where it's valid.

But realistically anyone who's doing something that new is insane to take on that work. There's a fixed number of people who understand how to run Kub. Most of them are going to the cloud providers and the large engineering firms. If you're not one of them, what are you running Kubernetes for? You're basically pretending you know what you're doing, and that's dangerous. You're better off building functionality your users and business are paying for, and that differentiates your value to the end user, than building infra.

Mike Kavis:
Well, it provides value in two ways. You can call yourself a DevOps engineer and you can put Kubernetes on LinkedIn.

Jim Ford:
Well, resume-oriented architecture—yes, I understand the term well. My resume needs to have Kubernetes. I'm going to put Kubernetes out there.

Mike Kavis:
ROA—that's my new acronym. *[Laughter]* All right, so the first question is going to be about an interesting article you wrote about—it's called "To Code or Not to Code," and it was really talking about the CTO level, and when you wrote that you weren't working for startups so it's going to be interesting if your answer matches the article now. But it was really talking –

Jim Ford:
It still does.

Mike Kavis:
Yeah, I was going to—talk about the depth of, should the CTO be hands-on keyboard coding? Should they be more governing, setting, policies? Is it somewhere in between? So, I'm not going to steal the thunder. Tell us about that article and your thoughts.

Jim Ford:
Well, sure. I was smarting a little bit, because I had recently gone through some interviews, and I was having a debate with some folks over why it was important that I should be current in a programming language and be able to sit down and write a coding test in some language as part of a modern interview where it's ten people and five hours or whatever. And my basic pretense is, if you need somebody who's going to understand the crosscutting concerns, the business touch-points, the drivers, you need them to have coded. You need them to understand technology in a fundamental way. But we actually haven't invented much in the way of new technology since you and I started. We've re-platformed it, we've re-implemented it, we've reimagined it, but what is an OCI-compliant container? It's a modern version of IBM's LPAR, skinnied down to where I can put it on a thumb drive. These things aren't new; they're just reimagined.

Mike Kavis:
And we'd write that to vSAN tapes.

Jim Ford:
Exactly! *[Laughter]* So, my main point is, if you're going to lead a team of engineers to build a product, and it's more than a startup where you're called the CTO because you're the third guy in, you really have to–

Mike Kavis:
Hey, now, I can relate to that, so…

Jim Ford:
Well so I've kind of refined my answer to an, "It depends." But if you're to the point where your engineering team is more than about 15 to 20 people, your CTO should not be doing prototyping or production coding. They should be out doing some high-level vision, high-level direction, high-level explanation, but they should not be the expert coder.

Mike Kavis:
Right, but they should be able to look at an architecture and call BS.

Jim Ford:
I agree. I'm not saying you can go ahead and get somebody who's purely academic. I'm not saying it's only a Visio architect. They have to have the chops. They have to have done the thing. But should they be current in it? Should I still be writing—should I be writing JavaScript? Should I be writing .net? Should I be learning Rust? Maybe, but most of my concern is in the pattern, the interaction, the metaphor, the pattern of how things are going to work together, how they queue, how they dequeue, how they scale, and that's not really down in the weeds of the actual code in my opinion.

Mike Kavis:
Agree. The next topic is we're going back in the day. It's always build-versus-buy, right? Do we build this or do we buy a package? But now we're in the cloud and it's all services. It's really, do we build or do we consume? Yeah, so we have this dilemma today. You know, the cloud providers have a ton of services, and in the last two years they're really moving up the stack beyond PaaS to business processes as a service, right?

Jim Ford:
Sure.

Mike Kavis:
What are those decision points on should I go up that stack and leverage these things versus it just doesn't have everything I need; let me go build this? And there's consequences of both, so let's just talk about that for a bit.

Jim Ford:
Well, let's start with the idea of, is software ever done? This was the problem I always saw working in large enterprises, that there's this idea that you're doing a project. When the project's over, you redeploy those coders to go work on new stuff and they can forget about the old stuff. Nobody ever takes into account the maintenance and the aggregate debt that comes, the fact that there's re-platforming and sunsetting of technology, and there's a lot of stuff that you end up getting screwed on in enterprises as you acquire more and more bespoke tooling in the quest to get perfect. There's companies out there that are still running homegrown CRMs. So, there's a value to it, but there's a danger to it. There's also a danger even if you're just consuming opensource because not all opensource is created equal, and how do you recognize the difference between a rising sun of a product and a waning sunset of one?

I think we have a real problem because we're good at adopting stuff. We're really bad about letting stuff go. So, when we look at the long tail of legacy in the datacenter from a company that's been around a long time, the modern company in the cloud isn't necessarily that much better if they're using something with a strange dependency change. We all live way up here in the extraction ladder. We're not living down near the metal anymore. We're not in the days of Command.com and NetBIOS. We're way up here in the, "I'm running an interpretive language on VA in the browser and it doesn't know anything about where it's running." Well, that means we don't—since we don't care we don't know, and we're always amazed when we get compromised or intercepted or man-in-the-middled. But we don't understand that layer three down there somewhere doesn't have the controls that everybody else had, or whatever it is.

So, I think consume is dangerous when you don't understand what you're consuming, but I think focusing on business value is the most important thing people can do. And the least amount of code they can write to do that, the better off they are in the long run.

Mike Kavis:
Yeah, I agree with that. That goes against the core of a lot of developers who love to write code, right? But to me, the architect is the one that figures out when not to write code.

Jim Ford:
I had a guy, a very wonderful developer, and he used to argue with me all the time how the best code was no code. And I would fight him tooth and nail, and now I look back and I say he was a genius. *[Laughter]*

Mike Kavis:
So, talk about no code. Now there's no code, low code, right? And the first thing when this came out I started thinking—man, we're old. I started thinking about Ashton-Tate code generator for COBOL, how that was going to solve all our problems, right? It generated all the code –

Jim Ford:
Well, all the 4GL, SEER—code to write code was the answer; metadata was the answer; abstraction's the answer.

Mike Kavis:
Yup. So—and then back to your comment where we're good at adopting things but we never sunset them, and I was thinking maybe we should start our own reality show called The IT Hoarder. Instead of the hoarding thing, IT hoarding—we could have a lot of shows there.

Jim Ford:
I kind of wonder if we should start publishing a health index for open source projects and do some kind of a heuristic against contributors, rate of change, supportability, criticality, and come up with a score of red, yellow, green. Yeah, use these. Get off those because the community's moved on. It was only one guy to begin with and he's moved. Okay, what do we do? *[Laughter]*

Mike Kavis:
He retired, yeah. So, the next question—this goes back—like you said, we met at DockerCon after you guys' presentation, and back then it was early days of I'll say container 2.0, because container's been around forever, but Docker kind of made it a thing that people cared about beyond Linux Kernel people. But—so anyway, so you guys were early to the game adopting containers, and back then the container orchestration war wasn't settled. So, there was Swarm and Mesos and Kubernetes, and like I said, you guys were an early adopter. You made some choices.

So, my question is—so that was 2016, which probably means you made those choices in '14 or '15, so we're talking about six years. Now we fast-forward six years. Say you're still in 2015, but the technology fast-forwards to '21. Now you have the choices you have today. You have—all the cloud providers have containers as a service. Kubernetes kind of won. Where's Docker today? So, with all that, how do you think—would any of that impact the decisions you made? And there's a point to this once you answer that.

Jim Ford:
Okay. Well, first let's kind of divide the universe into two parts. What Docker did for us in 2014, 2015 was it got the developers exciting about the right ones run anywhere, and, really, it was the developer tools and the whole initial concept that got the developer mindshare that really enabled a lot of the shift to begin with. So, I think that, plus the fact that the container image became an OCI format that was a standard that would run across a Kubernetes or a Mesos or a Cloud Foundry, or whoever at that point.

So, making the decision to go to containers would probably have stuck. Making the decision to use Docker tools for the Devs very well may have stuck. I think Docker's big mistake was when they tried to do the build-run part and moved too far to the datacenter and try to make the Ops folks happy too quickly, because, ultimately, Kub has kind of won that battle and that's great. But I think that, with Kubernetes, we run the same risk we ran when we built the datacenters 30 years ago, that we're going to miss the wrong boundary for our security boundary. And I know Open Policy Agent's supposed to solve this, but eventually we're going to be running Kubernetes in the background. You're not going to know it's there. And it's not going to be a thing. Think about how Kub control captures everything on a command line. How long before we teach the AI what you do when you need to do it, and take away the command line, and make it so that the thing that comes beyond Kub just runs your stuff?

Mike Kavis:
Yeah.

Jim Ford:
So, I think we would've done a lot of the same stuff. Would we have jumped straight to Kubernetes instead of Swarm? Possibly. Would it have saved some time? It might've made it faster. But I think, ultimately, the main decisions that were made were to break things down into containers and try to get the portability, and to start isolating away from the VM. And I think that was valuable no matter what, and it allowed us to get to a unit of delivery and a unit of deployment that was much more self-contained and predictable. So, I'd like to think we would've done most of the same thing. We just would've done them better, because the tooling would've been better.

Mike Kavis:
Yeah. So, the reason I ask, and a lot of this has to do with the speed at which the cloud providers—even though we're talking about containers here—improve their APIs, actually add more features to it. So, a few years back we were working with a client, helping them move off of on-prem, kind of

packaged big data solutions through like AWS EMR, right? And one of the requirements was you need parity, and there's a—if you look at the whole Hadoop world there's Pig, Hive—there's all these things that get cobbled together, right? EMR didn't have all those things. There were certain things missing. So, we were cobbling together all these things to give parity, and as we would get one solution Jeff Barr would put a post that says, "Here's this part," and it would just keep happening. So, a year later I think Amazon looked at the solution and said, "What the heck are they doing? We've got this, this, this, and this." Well, we didn't have that then.

The point then is—and–

Jim Ford:
Rate of change.

Mike Kavis:
Yeah, the rate of change, and sometimes you have to acquire third party solutions to fill a gap, and a year and a half later you don't need that. And it goes back to IT hoarding, right? That lives forever. So, do architects today, as part of being fiscally responsible in the cloud, which includes shutting stuff off and making smart decisions, do they also have to be thinking about—reevaluating decisions every year, 18—whatever that interval is, going back and looking, saying, "Is there value in taking out legacy?" And legacy may only be two years now instead of 20 years. So, I don't know. It's just --

Jim Ford:
It could be less than two years, and no, it's super valid. And the other challenge is that sometimes the cloud provider's solution is far superior. When I look at Fargate or Lambda on Amazon today, they're all underpinned by Firecracker, which is the new micro VM that only supports three system devices, so it's immune to most of the X86 attacks that are leaky containers that most people still have are running. So, you get benefits by moving to people who are focused on doing it at scale that you'll never get trying to get it yourself.

And then you also have to probably get to the point where you're close enough to your cloud provider that you're getting the roadmap briefings and you're getting the NDA stuff so you can understand what is the stuff that you're really going to have to build because it's way out, and how much of it can you be cranky customer on and get Amazon to move. And it surprises me, because you think of Amazon being so huge that they don't really listen, but they're actually pretty good at listening to the clients and what they're doing and seeing.

And I think that going ahead and taking in a solution that might be possibly a little more expensive or a little more different than what you wanted by going with the PaaS layer, or going with the managed service probably helps you in the long run because by getting into the ecosystem, by getting articulated into the tooling, when you want to turn on Redshift, or a data lake, or a new SageMaker model, or some new add-on thing that you never had before, you don't have to go back and think about, oh, well, how do I index this, how do I access it, how do I address it, because the cloud provider has stitched a lot of this together already.

And if you're storing this stuff in your S3 already because you're using their XYZ service, well, then you can point Athena at it, or you can point something at it, consume that data again, and you start to get some of the benefits of say a LinkedIn does with their digital exhaust, without necessarily having to plan ahead or doing a giant Kafka kind of situation that will become a Frankenstein of its own. You kind of get that for free if you kind of buy into pieces of the ecosystem and then think about how things are being sequenced, consumed, and strung together. It used to be if you wanted to turn on a data analytics, data warehouse kind of project, you went to the board and asked for two million bucks.

Mike Kavis:
Yep.

Jim Ford:
Then it became, “Okay, now I can go to the board and tell them I'm going to spend a hundred grand while we reformat the data through the data lake.” If you're "cloud native" and you're using an RDS, or you're using an Aurora database, now you say, "Fine, tell SageMaker where it is and run." Now you're spending 20 bucks to find out do you tell the board at all. *[Laughter]*

Mike Kavis:
Yea. Yeah, that's a good point. Back in the day, late '90s we built a data warehouse, probably 18 months before business could get their hands on it. And in the last few years we can walk into a client and do a proof of concept and a data lake in three, four weeks.

Jim Ford:
The real key here is that we have to remember that we have to push back and understand the business driver and understand the business pain, because if we just become order takers we build those bespoke monsters and we should say, “Well, if you can live with this until next Tuesday I can get you that which will give you 80 percent more of what you wanted,” rather than trying to meet it all on some arbitrary deadline that somebody pulled out of the ether.

Mike Kavis:
January 1. All my deadlines used to be January 1. I'm like, "You know, I really wanted to take time off around Christmas."

Jim Ford:
I prefer April 1, because then we can debate whether or not I was serious if I deliver or miss it. *[Laughter]*

Mike Kavis:
There you go. All right, man, well, great catching up as always. Where can we find you? I know you've got a lot of good content out there. Where can we find your blog and what's your Twitter handle and all that good stuff?

Jim Ford:

Sure. You can find my blog when I do occasionally post. It is on my website, www.Pareidolia.rocks. By the way, pareidolia is the habit of the human brain to see things that aren't there. So, after a year and a half at home pareidolia does rock. *[Laughter]*. My Twitter handle's @FordJamesC, and I'm on LinkedIn at James C.—Caldwell—Ford. You can find me out there.

Mike Kavis:
All right, thanks for your time. That's it for our episode of Architecting the Cloud with Jim Ford. To learn more about Deloitte or read today's show notes, head over to www.DeloitteCloudPodcast.com. You can find more podcasts by me and my colleague Dave Linthicum just by searching for Deloitte On Cloud Podcast on iTunes or wherever you get your podcasts. I'm your host Mike Kavis. You can always find me on Twitter, @MadGreek65, or you can reach out to me, e-mail at MKavis@Deloitte.com. Thanks for listening. 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