Deloitte.

Architecting the Cloud, part of the On Cloud Podcast

Mike Kavis, Managing Director, Deloitte Consulting LLP

Title: **Workflow management, serverless, and cloud: how to do it better**

Description: Workflow management is often a thorny problem, and it doesn't get easier with a move to the cloud. In fact, different cloud-provider requirements and languages may exacerbate workflow management issues. In this episode of the podcast, Mike Kavis and guest Cohesion.Dev's Soam Vasani discuss workflow management in a serverless environment. They take a look at how companies are using workflow management to enhance AI/ML and how to execute the process better, no matter where it occurs. Mike and Soam dissect the advantages and issues with different tools and techniques, as well as cloud-specific workflow management topics, and Soam gives his take on how companies can make the process easier, faster, and more effective.
Disclaimer: As referenced in this podcast, "Google" refers to GCP (Google Cloud Platform).

Duration: **00:20:00**

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/Deloitte. 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 Architecting the Cloud Podcast, where we get real about cloud technology. We discuss all the hot trends in cloud, but more importantly we do it with the people in the field doing the work. I'm Mike Kavis, your host and chief cloud architect over at Deloitte, and today I am joined

with Soam Vasani. And Soam leads the effort at Cohesion.Dev, a new company which we'll talk about briefly here, and has a long background with serverless, Kubernetes, functions as a service, all that good stuff, worked on a Fission workflow system back in the day, and he's working on all kinds of cool stuff that is relevant to what we're going to talk about today. So welcome to the show. Tell us a little bit about your background, and then tell us what's new – what you're doing over there at Cohesion and what led you to build this solution.

Soam Vasani:
Thanks, Mike. It's great to be here, and hey, everyone. I'm Soam Vasani, so thanks for the intro. So I'm working on Cohesion, which is a set of developer tools for serverless workflows, so let me unpack that a little bit. Workflows are a really common thing in various areas – you know, data engineering, machine learning, CI/CD pipelines. But a lot of the workflow tools that exist are fairly hard to operate. And having worked in the serverless space for a bit, I saw the ease of use, the time to market advantages, all of the good stuff from serverless, and I saw the things that it abstracts away, and have always been wondering about how a workflow system might benefit from serverless in general. So Cohesion is filling that gap. It targets AWS and lets programmers work in Python, just ordinary code that's familiar to them with very little extra configuration. They don't need much knowledge of the functions in Lambda and so on. And Cohesion can then turn that into a workflow on AWS, which you can then run with all of the serverless advantages.

Mike Kavis:
That sounds cool. You started to talk about machine learning and analytics, and a lot of people talk about functions. At least the travels I've been on, they're not really talking in the areas of machine learning. So through your experience, how are people leveraging functions as a service in the area of machine learning, AI, and those places?

Soam Vasani:
Yeah. I mean, so I think a lot of the early usage – so Lambda is now pretty old, and a lot of competitors have come up, right, including like on-prem stuff like Fission, which I worked on. And I think a ton of usage of functions as a service is around infrastructure automation kind of stuff, like Rhombus Webhook or a like, transform certain data to glue two services together, that sort of thing small stateless services. There is a certain amount of simple Cloud APIs, but I think the ML and AIs, like the model training – well, not so much training but model inference and stuff, that is a relatively new usage with machine learning, I think. With Fission, also, we saw a certain number of people doing machine learning on functions on Kubernetes. With Lambda, now and then I hear about it.

And there's some limitations of Lambda that make it hard to do, but there are some advantages of Lambda that make it very attractive to use also. So one, the core limitation is that Lambdas are small in both size and in time, right? So a Lambda is a software artifact, I forget the exact limit, but it's not very large, and you add enough TensorFlow and all of the libraries on top of it, you might have blown your limit already. And there's the other problem of the 15-minute time limit, so that is usually not enough time to train a model. It might be enough time for inference, but that brings us to the other problem, which is most machine learning models have a large memory footprint. And it's expensive to keep loading that from storage into memory, and that's what functions are optimized for, for going to sleep and waking up, right? This cold-start problem in serverless.

So those are the challenges for machine learning on functions. But in spite of those challenges, some people do it because fundamentally the promise of serverless functions is that you have a very granular billing structure. In case of Lambda you pay for every tenth of a second, right? If you use your models for half a second, that's – you know, what you pay for. And the other advantage is that you have enormous amounts of compute available to you when you want to use it without having made any up-front investment for it, right? So I was just reading this post by someone from Bustle– which is a company that uses a lot of serverless stuff—where they do a ton of model training on Lambda, and they like it because they can parallelize it a lot. So they can invoke a thousand Lambdas concurrently and it obviously has no problem running your workload on a thousand different CPUs. And all you pay for is the few seconds that you've used it, or even a few minutes, right? So now imagine having the old way, having deployed a whole bunch of VMs. If you need a thousand CPUs, you have to deploy that many VMs to have that, and that's expensive and it's also kind of hard to manage the scaling up and down of those clusters.

Mike Kavis:
I was going to say, another challenge has always been warming up right?

Soam Vasani:
Yes.

Mike Kavis:
And I think sometimes you call a function, but it takes a long time to warm up the systems underneath it. And I think there's been some advancements recently, features recently where you have pre-warming, but when it comes to machine learning they may not need that millisecond response time like some other systems might. I don't know. What's your thoughts on that?

Soam Vasani:
Exactly. So I mean, it depends. If a company is building an API where the core API requests parts it needs to run a model, you know, now they're counting the milliseconds that that thing takes to respond, so they probably want to keep that model in memory all the time, forever. But if it's any kind of background thing, like we want to asynchronously check if so-and-so thing that happened is an anomaly, right? It's a really common thing to do. Or we want to build up some sort of recommendations to show on – I don't know, like the Netflix home page or something. All of those things are in the background, so there it doesn't really matter if it takes an extra few seconds.

Mike Kavis:
So I want to move to the world of operations, which I spend a lot of time on, which is ironic because I grew up on the app side. Now I'm paying for what I did to everyone when I was a developer, so. [Laughter] But when we talk about serverless, there's functions, and then there's fully managed services like BigQuery, right? And a lot of greenfield systems, including one I have a team working on today on the Google platform that has no VMs because it's using all the fully managed Google APIs and using functions. First it freaks out a lot of the security people and a lot of the old school operations people because they're used to a certain set of dashboards and logs, and now a lot of responsibility is being pushed the vendor. So with these newer systems where there's few VMs or even no VMs, how does this change how we operate and run and monitor serverless systems?

Soam Vasani:
It's a really good question. You know, the change is real. At some level I don't know how things are going to change. And you know, a lot of details are changing. The exact tools are changing. But what's really interesting is that the pattern of operations has not changed, right? So if you think about how – you know, there's this idea of the interloop of development where a single developer on a single machine coding environment is making a change and testing it, right? And then there's the outer loop where the change leaves that one single developer's system, goes into some kind of version control, goes to some kind of CI system, goes through reviews, maybe even staging, production.

The production has to be monitored. You might have some sort of continuous deployment. All of those things as a pattern exist in the world of VMs and outside it, okay? So simple example: I mean, let's talk in the context of function. Let's say you make a change to your function. Let's say you have various good practices. You have lots of unit tests. You run those. You still need to think about is my change going to break production, right? Fundamentally that did not change. So maybe you have candidate deployments. Maybe you have some way to do a quick rollback. And evidence in all these details that the tools and the costs are different.

So for example, with Lambda it's really easy to do candidate deployments. With Fission we had candidate deployments built in using Prometheus and a custom router to direct traffic, things like that. And many systems on Kubernetes support this kind of intermittent deployment. So that's functions, and sort of my claim is that more – mostly the same things if you do enough, but the details of the tools have change.

With managed services, it's a bit different because you simply don't have that much control. But I've heard about people using managed Kubernetes, and then because certain things are abstracted away from you, including for example details of how the cluster is set up and started, that people keep two clusters, and then when a version has to be updated actually migrate to a new cluster slowly and split traffic between them and so on. So maybe that's a pattern for managed service as well, where you could actually candidate from one to the other. But I mean, it varies from service to service I think, the operations for those things. And a lot of them simply don't give you enough information to make good decisions. So again, the kind of monitoring you are doing plays a big role in whether you're able to actually understand your system.

There's a favorite quote from the Lightstep people, that it's a tracing and observability company. They say, "With microservices, every single operation's problem is like solving a murder mystery. You're kind of investigating hundreds and hundreds of clues, and it's really hard." And I think, yeah, the problem is worse in microservices, but with managed services also you have some of those problems. There's a ton of information in different places and it can get really hard to debug.

Mike Kavis:
Yeah, I've seen organizations trying to take their datacenter-centric monitoring tools to the cloud. And maybe they can get away with it when they're lifting and shifting a three-tier app, but when you're building these new apps using managed services, and everything mutable and everything's kind of self-healing. It's different. So the big change for me is, one, it's a skill set change, right? The people who are running, whether it's the same team building it or not, need to really understand cloud and how these things are working. So it's a big challenge for people, and I see companies figuring out the build part really quick, but not figuring out the run part. And when stuff breaks even though their stuff may have not been great in the datacenter, at least they knew how to fix it, right? And now they've got all these fancy stuff in the cloud and it breaks, and all of a sudden their SLAs go down the tubes because they don't really know how to respond because it's kind of a greenfield operation. That's kind of what I've seen, and people forget that.

Soam Vasani:
Yes, for sure.

Mike Kavis:
Right? And it's like, hey, this stuff's got to run, too.

Soam Vasani:
Exactly, yeah. And the quality of these services varies even on the big three cloud services app much better than the others, right?

Mike Kavis:
Yeah.

Soam Vasani:
So, you kind of just learn from experience to avoid certain services.

Mike Kavis:
Yeah, yeah. And they're at different levels of maturity. I think one of the things the cloud vendors do, and I'm jealous of, is they don't tell you what's coming out next. They just release something because they have good feedback loops, but it usually has a minimal set of features. And then as people consume them, they learn, and they add the next feature and the next feature. So depending on where that API is in its lifecycle, some of them may not be ready for prime time yet.

Soam Vasani:
Yes, that's true.

Mike Kavis:
So let's talk workflow now. So I am not an expert on workflows and functions as a service, but I look at it like stringing together a bunch of commands to form a unit if work. Is that a right way of thinking about it?

Soam Vasani:
Exactly, yeah, yeah. Take a bunch of simple things and make a likely more complex thing out of them, a composition basically.

Mike Kavis:
So if we go back to operations discussion, it can be threatening and scary that there's all these microservices; there's all these functions. Is the plan really to string these together and orchestrate them as a unit of work and manage the unit of work at that level and then drill down? Or is it hey, there's a hundred things running; how do we follow it?

Soam Vasani:
So broadly if you look at this problem of stringing things together, the idea of orchestration and composition in general, I think different patterns are applicable to different use cases. So even just if you forget about the general idea of services and just talk about function, there are so many ways of having one function call another, right? You can simply call that function over HTTP. That call could be synchronous or asynchronous. You could put a message on a queue and that function could be attached to a listener for that queue. That queue may be persistent or not persistent and have various different delivery guarantees.

And then there's the question of you could have a workflow system where the actual composition is owned by neither of the functions. Both functions are just called by something else, and that something else is the workflow system that's part of the one stateful thing in the system that's deciding what should happen next at each step. And workflow systems are probably one of the more complex ways of solving the orchestration problem, the composition problem, but what they give you is that it's a sort of clean abstraction. The implementation of the workflow system is fairly complex, but as the user of the workflow system you only see calls into your functions, and you have to specify that workflow itself. So basically it's a clean way to abstract away the composition of function.

And then a lot of these workflow systems support things other than functions. So you have step functions function support. I think you can run a container in ECS which is a container cluster service. It can also run in things like database queries statically, so you don't have to write a database query inside a function and stuff like that. So the vocabulary of the system keeps growing and you can have a big set of tasks that you can call from them. So that also means you write a little less code on each task.

Mike Kavis:
At the beginning of the talk you talked about how some of the workflow in the big three cloud providers may be limited in functionality, and that's the point you're addressing. What are some of those limits? It feels like it's still hard to string a bunch of stuff together, so talk about what some of those gaps are that you're trying to fill.

Soam Vasani:
Right, right. So fundamentally, each of the clouds has their own slightly different workflow system. Microsoft in fact has two of them. And now the ones that are serverless, the ones that don't make you deploy a cluster, they eliminate the whole cluster management stuff, which is great, but all of them have really strange programming models. So if you think about a workflow and the idea of saying, "Do this, do that, if this happens do something else," that's programming, right? The idea of expressing that series of tests, that is programming. And any complex enough workflow starts looking like the flowchart of a program. So the language that these systems use to express the workflow becomes a limit on how well you can work with them. So that function, for example, uses JSON. I speak to tons of people who say, "I like the idea of serverless workflows, but I do not want to write programs in JSON." And I believe Microsoft has something that's programmed in YAML and similarly.

So what I'm trying to fill first, what Cohesion is trying to fill first is the programming experience. And what we're saying is that if workflows are programmed, they should be programmed in a really familiar programming language, and so we chose Python. And the programmer can write a Python program and we will actually transform it into the JSON multi-step function and into a few AWS Lambda functions, and we'll wire all those up together. And we can even let you click a test button and you will see what will run. So you get all the properties of serverless workflows without actually having to deal with that kind of crappy direct (Inaudible) that those workflows have.

Mike Kavis:
Yeah, I can't count the Twitter feed responses to groaning about YAML and JSON that I've seen over the years.

Soam Vasani:
Yeah, for sure. I might have been somewhere in there as well.

Mike Kavis:
Yeah, you may have.

Soam Vasani:
It really is no fun to write a thousand lines of YAML and – it's extra bad to come back there after six months and try to find the one little bug, you forgot a quote or a comma or something. Yes, not fun.

Mike Kavis:
Well, cool. Well, that's all the time we got today. I appreciate your time and look forward to seeing what you guys crank out there at Cohesion. We'll stay in touch and keep tabs on that. And for you Python freaks out there, this sounds like something right up your alley. So that's it for Architecting the Cloud today with Soam Vasani. You can follow Soam on Twitter @SoamV. 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. If you'd like to connect with me directly just e-mail me at MKavis@Deloitte.com or you can always hit me on Twitter @MadGreek65. That's it for today. We'll see you next time on Architecting the Cloud. All right.

Soam Vasani:
Thank you so much.

Mike Kavis:
All right.

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