EPISODE 743

[INTRODUCTION]

[0:00:00.3] JM: Functions as a service allowed developers to run their code in a serverless environment. A developer can provide a function to a cloud provider and the code for that function will be scheduled onto a container and executed whenever an event triggers that function.

An event can mean many different things. An event is a signal that something has changed within your application. When you save a file to an Amazon S3 bucket, that creates an event. When a user signs up for your app, that can create an event. Functions as a service are allowing people to build applications completely out of managed cloud infrastructure. Apps can be fully serverless with managed databases, managed queuing systems and APIs tied together by these serverless event-triggered functions.

Today, there is not a consistent format for these events that are across different applications and different cloud providers. The lack of consistent formatting makes it more difficult to stitch together these events across different environments. Ideally, events would be lightweight, they'd be easy to deserialize, they'd be easy to operate with, to interoperate with. The cloud events specification is a project within the cloud native computing foundation. It has the goal of creating a standard format for events.

Doug Davis is the CTO for developer advocacy of containers at Microsoft. Doug joins the show to discuss how events, an event-based programming works. He also talks about the need for the common format across cloud events. This show is a great complement to the other shows we've done about serverless computing, event-driven computing and serverless infrastructure in general.

We are looking for sponsors for Q1. If you're interested in sponsoring Software Engineering Daily, we reach about 50,000 developers on a regular basis. If you are looking to advertise to developers, you can send us an e-mail, jeff@softwareengineeringdaily.com, or go to softwareengineeringdaily.com/sponsor, or you can tell your marketing director about it.

We are also conducting a listener survey. We'd love to know what you are thinking when you listen to the show. What do you like? What do you dislike? What could we do better? If you take that survey, there is a place where you can enter your e-mail address and you can potentially win a piece of Software Engineering Daily swag; a hoodie, or a t-shirt, or a mug from the Software Engineering Daily Store.

With that, let's get on with today's episode.

[SPONSOR MESSAGE]

[0:03:15.2] JM: Clubhouse is a project management platform built for software development. Clubhouse has a beautiful intuitive interface that's made for simple workflows, or complex projects. Whether you are a software engineer, or a project manager, or the CEO, Clubhouse lets you collaborate with the rest of the product team.

If you're an engineering-heavy organization, you will love the integrations with Github and Slack and Sentry and the other tools that you're feeding into your issue tracking and project management. To try out Clubhouse free for two months, go to clubhouse.io/sedaily. It's easy for people on any team to focus in on their work on a specific task or project in Clubhouse, while also being able to zoom out and see how that work is contributing towards the bigger picture.

This encourages cross-functional collaboration. That's why companies like Elastic and Splice and New Bank all use Clubhouse. Those companies have all been on Software Engineering Daily, and we know they are competent engineering organizations. Stay focused on your project and stay in touch with your team. Try out Clubhouse by going to clubhouse.io/sedaily and get two months free.

Thank you Clubhouse for being a new sponsor of Software Engineering Daily.

[INTERVIEW]

[0:04:45.4] JM: Doug Davis, welcome to Software Engineering Daily.

[0:04:47.5] DD: Thank you very much. Glad to be here.

[0:04:49.7] JM: We're talking today about functions as a service and events. Let's start with the topic of functions as a service. We've done a lot of shows on this. Why are functions as a service useful?

[0:05:01.8] DD: When I first was introduced to functions as a service and serverless in general, it was one of those things where I looked at and said, "Okay, you're just applying code. Why is this so special?" right? Someone actually gave me a really, what I would call a killer use case that have really solidified it for me. They said, let's say you have a – say an ACP server sitting there and it's your normal application, you send get, put, post, request, do it all over HTTP, all that good stuff. What if you realized that you only needed to really support say 10 puts at a time per second, but you need to support a 1,000 gets at a time? Because your people are – your users are doing – opens more queries than they are actually updating things, right?

In a normal world, if all that's one application, or one bundle, you got to scale that thing up to support a 100, or a 1,000 per second, regardless of whether you're doing gets or puts. Well with functions, you can now split that apart and say, "You know what? I'm going to scale just the get side of things, not the put side of things as much."

That's when it started sitting, or solidifying in my head and said, "That's cool," right? Because it takes that splitting up your monolith into microservices one step further and say, "You know what? I got this tiny little bit over here that needs to scale, but I don't want to scale all the infrastructure behind it necessarily," so that's where a function in my head really plays a role is the easeability of scaling bits of your larger business logic, if that makes sense.

[0:06:21.0] JM: On these different cloud providers, they've been implementing functions as a service, whether it's AWS Lambda, or Google cloud functions, or OpenWhisk on IBM. There's a set of problems that each of these providers has to solve. As the open source function as a service platforms have come up, they're also solving these problems. There's some problems around scheduling and rate limiting and the cold start problem and all of these problems around if you're loading a function onto a container, on demand, and then making it available to a developer to hit with the request on demand, there's a number of problems. Tell me about those problems.

[0:07:04.0] DD: I think you actually touched on I think probably the biggest one, which is okay, in a serverless world, things are supposed to be to scale down to zero, so that you give the economy a cost, right? You only have to pay for when you're actually using it. You scale down to zero, no one's hitting your function, life is good, you don't have to pay for it.

The minute something comes in, if it's going to take a long period of time for that first instance to scale up, then yeah sure, you may have been saving money, but is your business going to be hurt because it takes, let's take an extreme case, 10 minutes, for that first instance to scale up. No one's going to want that, right? I think that's the biggest issue people are going to run into is how fast can these things scale up, in particular zero to one, but of course it does apply for the rest of it in case you get a huge burst beyond one. Zero to one is use the big ones that people are really, really concerned about.

[0:07:51.6] JM: Each cloud provider is employing their own function as a service platform today. How consistent is the experience across them for the developer?

[0:08:02.1] DD: I probably can't list comments about specific ones. I think it probably is just fair to say that you're right, a lot of them do have their own proprietary APIs. You are seeing some people pick up the open-source version of things. You have OpenFaaS, OpenWhisk and I'm sure there are other open source ones out there. Obviously, any platform that picks up those, you're going to get portability, interoperability and stuff. The options are out there, right? People may choose to roll their own. I don't know. I just know some people are. I can't comment about much beyond that, to be honest.

[0:08:32.2] JM: If I have a function that I want to run on one cloud provider and then I want to go and take that function and run it on another cloud provider, am I going to have different experiences in terms of latency, or other kinds of issues I might encounter?

[0:08:46.1] DD: Obviously, you probably will. That's true of everything, right? Every cloud provider is going to have different sets of performance characteristics and stuffs on it. I wouldn't imagine functions are any different there. You got to look at each provider to see what each one offers, I would imagine. Yeah.

[0:08:58.5] JM: Sure. Right. Now function as a service are interesting, because initially they came out and they were on the cloud providers. More and more, you see smaller companies, not small companies, but smaller companies than the major cloud providers bolting on a function as a service platform. Companies like Salesforce, or MongoDB, or Auth0 are adding serverless functionality, that maybe they want to call in response to somebody authenticating, for example. Why is that useful? Why is it useful for all of these other service providers to also implement function as a service platform?

[0:09:37.1] DD: I would imagine that's just speculation on my part. I would imagine, it's for the same reason we talked about earlier, which is you get the benefit of the economic cost of people that you quickly scale up bits of the application without having to do the entire monolith all at once, kind of a thing.

I assume it comes down to basically that aspect of it, right? Because that's what people tell that has the benefit of serverless or functions, if you live in a serverless equals functions world, the entire reason to break things down to from monolith, to the microservice down to the function is because you want to do usually scaling, right? You don’t want to have scale your entire monoliths, you break them on at microservice.

Same thing with function. I would imagine that's why those guys are doing it as well, to see a better economic value in that way, or performance as well, because it's easier to scale something if it's smaller, right? You can spin it up faster.

[0:10:27.1] JM: Many of the patterns around using serverless functions involve evented models. Can you explain what an event is?

[0:10:37.2] DD: It's interesting. I think you might get a different answer depending who you talk to, right? I think in the most puristic form based upon what I’ve heard people talk about, an event is basically a one-way fire and forget type message, right? I think to other people that have a little bit more liberal attitude towards it, where to them in a function space yeah, they think of it as oh, an event comes in, but they're allowing a response to come back sometimes.

In a puristic world, maybe that's not an eventing type of an operation where we request response, but I think – in the end, in my mind, it doesn't really matter. It's just some message comes in to your function and you're going to process it. If there's a response, how the response goes back, to me is a secondary concern. I don't personally get hung up on the puristic model that says, "Oh, if there is a response per se, it has to go back over another asynchronous channel." I personally don't care if it flows back with a HTTP response flow.

[0:11:32.0] JM: There is this relationship between functions as a service and events. Explain what the relationship is there.

[0:11:39.4] DD: Typically, a function is going to get kicked off by an event, right? Because you function – if you think about what people talk about with functions and especially with serverless when it scales down to zero, it's like, okay, you go down to zero, or when you bring up that first instance, something has to happen to kick it off. That's usually "an event," right? It could be an HTTP request coming in, or something over transport coming in. It could be a cron job or signal thing. Anything that basically comes in that acts as a trigger to force the function to get invoked or instantiated, that to me is an event.

[0:12:10.2] JM: What are some examples of events being triggered and propagating to triggering other services? Give me a typical, a prototypical model of some event creation and event triggering of serverless functions.

[0:12:26.7] DD: I think, one of the classic ones I've heard is a one system that has the notion of a new user being created, right? A new employee gets added to the system or something like that. An event gets fired that says, "Hey, employee Joe Smith has now been added." Other systems now get notified, because they subscribe to that type of event and then they can take the appropriate back-end processing to propagate whatever is necessary to get that employee into all its various systems and to the HR or whatever. They subscribe to those events and then take the appropriate processing basically.

[0:12:56.1] JM: If I work at a cloud provider, I'm responsible for managing tons and tons of events that are being created. If somebody is – if I'm running this authentication service for people logging in, and every time there's a login, there's an event that gets created. I need to put those events somewhere. I need to manage how those events notify the other services in the system. How are cloud providers handling all of these events that are getting created across their system?

[0:13:24.8] DD: I'm not an expert in this particular space, so this is figurative speculation on my part basically. Basically, I would assume it's basically some of the lines of a message bus thing, right? You put the events into a queue, a channel, whatever you want to call it, and then it's probably going to get persisted. Then something is going to either send it off to someplace else, or someone's going to come in and pull it off the queue or something like that. I would imagine, it's a message bus type of environment, a classic middleware type stuff.

[0:13:49.1] JM: You might have just a publish/subscribe system where the events are published to and then other systems are subscribed to.

[0:13:56.6] DD: I mean, that's definitely one of more popular models. Yeah.

[0:13:58.5] JM: Okay. What if I want to use an event in one cloud provider to trigger a function in another cloud provider?

[0:14:08.3] DD: The short answer to that one is I don't see the problem, right? Because typically, when you start talking about events, talking about something flowing over the wire, as long as the receiver or consumer knows what to expect, right, the format of the events; what metadata is supposed to be there, so we could do proper processing of it, routing up and stuff like that? Whether you're crossing cloud providers, or it's all in one cloud provider, technically should not matter.

[0:14:31.1] JM: In reality, different cloud services do produce different types of events, different formats of events. Does that create frictions in having cross-cloud eventing models?

[0:14:42.7] DD: I wouldn't look at as a problem of being a cross-cloud problem, because either way, whether your consumer is in the same cloud, or a different cloud, the consumer still has the problem of knowing what to expect, right? Whether, take the simple case if you just go and look at the documentation for the event producer, right? You still have to know what to expect, so whether the consumer is living in one cloud versus another, that's not going to change the code he has to write in terms of the processing, right? I don't look at it as a cross-cloud problem, as much as just how does a consumer know what to expect from an event producer.

[0:15:15.7] JM: I think this brings us to the cloud events spec. Can you explain what the cloud event spec is?

[0:15:21.3] DD: The cloud event specification is a spec that's being developed under the CNCF. It's a sandbox project. It's basically designed to help in the routing of an event from one place to another basically. The analogy I like to use is if you look at HTTP, it's a very simplistic protocol, even though there's lots you can do with it. At its very core, what you have is a bunch of HTTP headers and in particular, a very small set of required or standard ones, right? You got content type, content length and then the header itself that says HTTP, POST, whatever.

You have very small set of headers there and then you have the body. For the most part, HTTP doesn't say, or actually it doesn’t say at all, what goes in the body, that's application data, right? It just says, “Here's a couple headers to help you with the routing.” That's what cloud events is trying to do. It's trying to define a set of standard bits of metadata that allows you to receive an event and know how to process it, or route it properly to its proper destination. It's not going to get into defining what the body of the message is supposed to look like, or your cloud data, application data is supposed to look like.

It just says, “Here are four or five properties that are help you to get the event from point A to point B.” It has things like, a URI that defines what is the event type. In case you have routing through different processors based upon whether it's a Github type of event, versus some other object store type of events, you can properly route it based upon that. There are a couple other ones, which for some unknown reason escaped me this vague moment, but something simplistic on those lines that are there just for the routing purposes. Once it gets to the application, then the application can do is normal processing on the application specific data.

That's all we're doing is defining the core little bit of properties and how those are serialized, both in formats, like JSON, or how they appear on the wire and things like HTTP, or MQTT and stuff like that. That's really all we're doing. Very minimalistic to try to ease a little bit of the interoperability between producer and consumer in the eventing world basically, if that makes sense.

[0:17:19.9] JM: Well, tell me more about what you mean by routing.

[0:17:22.9] DD: Like I said, when an event hits an endpoint, that endpoint needs to be able to route it to the proper location, right? In the same sense in HTTP, when you look at the first line of the HTTP request, it says it's get versus put, and then there's the tail part of the URL, right? Well, the HTTP server is going to use that tail part of the URL to route it to their proper – use java term, servlet, right? It's going to route to the right spot.

Well, you may need to do that same logic within your function infrastructure, right? The event type comes in and it may say –I'm going to probably bastardize what Github actually does. Let's say it says github.issue.create, right? Maybe you have a function that specifically set up the handle, new Github issues, right? Not necessarily updating Github issues, or deleting Github issues. Or maybe they're separate functions.

By using that one little piece of metadata, now the function infrastructure can route it to the proper function and let it do it the processing at a smaller scale, right? I don't have to have one, forgive me, monolithic Github processor that handles all Github type of events. I could have one that does at Github create scale differently, than a Github delete type of operation, or event I should say. That makes sense?

[0:18:30.4] JM: Well, so there's no specification that is widely used today, I don't believe. What are the penalties that we're paying in routing infrastructure, because of the lack of a specification?

[0:18:44.1] DD: Yeah. Basically, I think what it comes down to is now every bit of function infrastructure has to pretty much do this on its own in a sense. Or it has to have advanced knowledge of the types of events it can do, or it's going to be asked to process if it wants to that processing within the function infrastructure, right? Because if the function infrastructure doesn't have this knowledge, or doesn't have advanced knowledge of the types of events coming in, all it can pretty much do is pass it on to the other system, or you can invent some proprietary logic that says, "Oh, based upon reg expressions, does the event coming in look like this with these types of headers?"

I can as an application writer, maybe add some specialized logic and through an extensibility point, into the function infrastructure to do the routing for me. It may all be possible, but my gosh, what a pain in the butt that is to have to do it every single time, and every provider has to recreate the wheel on that stuff, right? Why not say, "You know what? If you want to do routing based upon type of events, that's where it's going to sit every single time when it's an HTTP request." Just won the URI in this HTTP header, and now you can do routing based upon that. It makes life easier for the function consumer, I'm sorry, the event consumer basically.

[SPONSOR MESSAGE]

[0:19:58.9] JM: HPE OneView is a foundation for building a software-defined data center. HPE OneView integrates compute, storage and networking resources across your data center and leverages a unified API to enable IT to manage infrastructure as code. Deploy infrastructure faster. Simplify life cycle maintenance for your servers. Give IT the ability to deliver infrastructure to developers as a service, like the public cloud.

Go to softwareengineeringdaily.com/hpe to learn about how HPE OneView can improve your infrastructure operations. HPE OneView has easy integrations with Terraform, Kubernetes, Docker and more than 30 other infrastructure management tools. HPE OneView was recently named as CRN's enterprise software product of the year.

To learn more about how HPE OneView can help you simplify your hybrid operations, go to softwareengineeringdaily.com/hpe to learn more and support Software Engineering Daily.

Thanks to HPE for being a sponsor of Software Engineering Daily. We appreciate the support.

[INTERVIEW CONTINUED]

[0:21:22.2] JM: I think it's worth pausing here to talk about the protocol format. You have HTTP, AMQP, MQTT. This might sound really simple and I'm sure some people really know this already, but maybe you could just explain what HTTP is in contrast to these other message formats, or these are the protocol format, sorry.

[0:21:44.3] DD: Okay. I'm not an expert in the other protocols, and so I'm going to really dumb this down and –

[0:21:48.7] JM: That's fine.

[0:21:50.4] DD: - just say, there are just different ways to serialize a message across the wire, right? HTTP has the format of, there's headers, space and then basically, binary data, right? AMQP or MQTT, whatever other ones we have, they all have their own format for how things are going to appear on the wire and that's really all it is. It's just a very clear definition of when you start reading by itself the wire, what to expect, where and how to interpret it.

[0:22:17.4] JM: The headers are unserialized data?

[0:22:20.1] DD: Unserialized data.

[0:22:21.8] JM: Or yeah, are the headers like, you can basically learn context about the internal information by –

[0:22:27.8] DD: Oh, I'm sorry. Yeah. In the HV case at least, yes. The headers are there to help you understand and process from a infrastructure perspective the incoming message, right? The HTTP headers for example, will tell you where to route the message in terms of what's servlet to send it to, or function. They will also include a little bit of information for example, of how to interpret the body of HV message, right? What is the format? Is JSON? Is it XML? Is it some binary format?

That you don't necessarily have to complete – the infrastructure doesn't necessarily have to understand all different types of messages and formats that come in. All it has to know is how to interpret these little bit of metadata, so it can send it off to the next chain or next thing in the chain for processing, like the servlet as an example, or some queuing system to stick it into a queue thing, right? It just needs to know how to add a process at the infrastructure level to get it to the next hop in its processing.

[0:23:22.1] JM: The routing infrastructure wouldn't need to deserialize the entire message. It just needs to deserialize the header and then understand where to direct this message to?

[0:23:32.4] DD: Exactly. Thank you for explaining much better than I did. Yes. I totally agree. Yeah, that's a great way to think of it. Yes. Is the infrastructure just needs to understand and know how to deserialize those headers? Everything else after that is application data, it doesn't have to touch and understand. Exactly. Yes.

[0:23:49.0] JM: Is the cloud event spec similar to this? Were you just trying to make a header specification?

[0:23:55.1] DD: That would actually be a fair way to think about it, just to get your mind around it. Yes. We are defining a set of properties that an ACP case can appear as HTTP headers, that are really just there to help the infrastructure get a basic level of understanding of what this thing is that came in. We could pass it along to the next step in the processing model.

In the cloud event space, or the cloud event spec, we do allow those properties to appear in different ways. For example, an HTTP we have the binary format, which allowed them to appear as HTTP headers, if that's the way you want to receive the message. We also allow it to appear inside of the HTTP body. If for example, you want the entire event serialized in say JSON, then we allow for those properties to appear in the JSON payload itself, and that way you don't have to necessarily deal with HTTP headers if you don't want to.

We basically give you the choice. For some people, HTTP headers is an easier way to go, because then, you don't have to pay the cost of JSON de-serialization. You can just pick them off as basically strings as HTTP headers. We give you the flexibility to do what's best for your particular use case.

[0:24:56.1] JM: Well, maybe you could just describe what is in the cloud event spec. If I'm making my event and I want it to conform to the cloud event spec as it stands today, what does that look like?

[0:25:09.9] DD: Right. As I said, the cloud event spec defines, I think basically four properties and then there's a fifth one that says what level of the cloud even spec these things adheres to. Those four properties are basically very simple things, like the time the event was sent. Hey, I'm starting to remember. The event type, the event time, a unique identifier for the event, so you can do deduping if you want do something like that. You have those fields I just described.

Really, in order for an HTTP message coming into a system, to be a "cloud event message," really all you have to do is add those four properties as HTTP headers with the names that we provide in the spec and suddenly, it's a conformant cloud event HTTP event. That's really all it is. Add those properties and you're good to go.

[0:25:52.9] JM: If I'm a cloud provider, let's say I have a bucket storage system and people are uploading images to their buckets and they want to see events that are created from those buckets, because maybe they want to trigger a function that compresses their image that they just put into the bucket. They want to have this evented workflow. You want to have the function that does the compression. If I'm that cloud provider, why do I want to make the events that are created by that bucket storage system? Why do I care about the cloud event specification? Why do I want to make my events adhere to that?

[0:26:30.1] DD: From my point of view, it comes down to reducing the friction for people using your system, right? If you have an event consumer that wants to hook up to particular object stores in this particular case, they may choose one that allows them to use their existing code base, right? If their existing code base supports cloud events, then they may be more likely to pick another event producer, in this case the object store system that is already producing events in that particular format. It just eases the pain of doing the integration.

I'm not going to kid myself, right? It's not going to necessarily completely sell somebody on that particular cloud provider. I mean, it is just a tiny little thing in the big scheme of things in terms of what you're going to look at in terms of choosing a cloud provider, but every little bit helps. This is just the first step in the process for the serverless working group in the CNCF, right?

Some different things we could look at in terms of reducing the pain point, increase interoperability and the event format was one thing we thought, "Okay, maybe we could start, here's the low-hanging fruit." When this is done, we're going to look at what is the next hopefully, piece of low-hanging fruit that we could pull off and start looking to harmonize. At that point is again, just slowly reduce the pain of interoperability and integration between systems.

[0:27:41.7] JM: Looking at this another way, if I'm a service owner today and I want to build my system to respond to events, what are the frictions that I'm encountering because I'm trying to build these evented systems and I'm not getting a consistent experience?

[0:27:59.1] DD: Yeah. I think it just comes down to how much work you're going to require people hooking up your system to do, is what it comes down to. If every single cloud provider, which actually we're seeing probably today basically rolls their own events and everything has their own formats and stuff, yes, people survive, but it's not optimal and it forces people to think way too much about a layer that they should not have to think about, right?

Every step we take to allowing people to focus more on their business logic and less on infrastructure, I consider a step in the right direction to making the developers life easier, so they can be more productive and get their products out there faster to market, basically.

[0:28:34.6] JM: If I'm that developer today that's building something that responds to events from a cloud provider, I'm still having trouble understanding what the lack of a specification, like if there's no specification widely adopted today, why is this impacting me? I'm not writing routing infrastructure, right? I'm just saying like, "Hey, event. Come hit my system." I don't see any of the middleware in between.

[0:29:00.6] DD: You're right. Yeah. I think you're probably right that if all you're concerned about is the tail-end of the processing of the event, unless I'm missing it, you are probably right. You probably don't care too much about that. If you have to do any routing at all and maybe this is more in the middleware layers, then I think this is where it plays a key role, right? Because then, because that is your purpose in life and as middlewares to do the proper routing to send to the next hop.

If you are the last hop and all you care about at that point is the business logic, then you're right, it may not play as a critical role there. This may actually be more for the leading parts of the chain of processing.

[0:29:36.9] JM: I see. If that part is handled by the cloud providers, right? The routing infrastructure for the events.

[0:29:44.6] DD: Yeah, the function infrastructure basically is hopefully there to help route the event hitting the system to the proper function. Yes.

[0:29:52.3] JM: Right. Okay. If I'm that cloud provider and I need to route the cloud events intelligently to the right services that are subscribed to them, what does that routing infrastructure have to do? Does it have to deserialize certain parts of the cloud event, because it's consistent metadata, and so we know where all these different fields are? What is a cloud provider building to be able to respond intelligently to cloud events?

[0:30:24.8] DD: Yeah. I think you basically nailed it on the head, right? Every time an event comes into a cloud provider so the cloud provider can route it properly, he then has to understand what that data coming in looks like, right? To say it's just HTTP may not be sufficient, right? He may have to – because he may provide to his users the ability to say at this particular URL, if you see this type of event, send it to this function. If you see this type of event, route to that function over there.

If the cloud provider has to understand every different type of message coming in, even though they're always GP, but if the metadata that he's looking for is in different places based upon the event producer, that's a pain in the butt to have to support all those different event producers, right? If we can get them all to say, “No, we're going to stick the event type for routing purposes in one particular place in the HTTP message,” that makes their life easier and they don't have to add a different innocence routing logic for every different type of event consumer out there. Or I'm sorry, bringing that producer.

[0:31:24.5] JM: The end result is probably going to be for developers, that these events process faster, or how does my experience as a developer change?

[0:31:35.3] DD: That's an interesting question, because actually to be honest, this is something I haven't really thought much about, because I think about more from the – I guess, the more the middleware perspective, because based upon the previous question you're asking, right? For the tail-end of the processing, that thing that's going to receive the event, it's going to look at the business data and process it from a business logic perspective.

At that point, it may not necessarily care about the cloud event properties at that point. From the function developer’s point of view, if he's not doing any routing and doesn't necessarily need to necessarily understand those common properties, it may not help him much. The question is whether there are certain applications that would still benefit from that information, right? Maybe they're not necessarily doing routing, maybe they're doing something else with it for some other reason that I just can't think of right now. Having that common metadata that's available for every single event coming in still may be useful.

For example, there is an ID in there, right? Maybe they first want to see – have I seen this event before, right? Maybe the function as a service infrastructure isn't going to do deduping, but your business logic may need to, right? That event ID being in a consistent spot allows you to perhaps, put that into a common function that all your business logic can now use as a utility thing. Just something else top my head, as a possibility. It may impact them from that perspective.

[0:32:56.2] JM: Sure. Whereas today, different cloud providers have bespoke eventing identification systems, so maybe even if you have an ID for all of your different functions, or all of your different events, if they're coming from different cloud providers then maybe the idea is in different places and you have to deserialize the entire event, in order to find that ID.

[0:33:22.2] DD: Exactly. Actually, that that's an excellent point. We mentioned this earlier, you're right. It not only does it provide a consistent spot to look for the information, but because we extracted it from the business logic itself, you can very quickly look at that metadata and do some quick processing on it, for example, delete the message because you've already seen it based upon the ID, without having to one, understand the business logic data and to deserialize it and understand it to find out where that ID actually is in the message. Yeah, you're nailing it right on the head. Yes.

[0:33:53.4] JM: The reason we're talking is because, I think you're – you're in the cloud events working group. Is it a working group?

[0:34:00.0] DD: It's not a working group. It's a project. It's a sandbox project in the CNCF. Yes.

[0:34:04.3] JM: Okay. How did that come together and who is in it?

[0:34:07.6] DD: I'm trying to remember how far back it was. I want to say about a year and a half or so ago. The CNCF technical oversight committee, like everybody else started hearing about serverless and they wanted to know more about it. A working group was formed, called the serverless working group, whose purpose in life was to basically explain what is serverless, how is it different than platform as a service, container as a service, what's the difference between function as a service versus serverless, when would you use each one of them. Basically, just lay out the landscape of what it's all about, okay?

Also provide information about what people are doing out there today, right? What are the open source serverless providers out there, or product, or open source projects out there? What are the proprietary offerings out there? What are the serverless, or function as a service services that people use as back-end services types things? Basically, lay out the full landscape for the people.

Then the key thing that we also did was lay out some recommendations for the CNCF for what they should do if anything about serverless. Maybe we could have come back and said, "Eh, interesting technology, but it's not worth our time," right? We didn't say that obviously. One of the recommendations however we did have was we should look for opportunities to find some level of harmonization, or standardization, right? Can we reduce some of the pain points that we've been talking about so far?

One of the low-hanging fruits that I mentioned obviously was cloud events and said, “Okay, most of serverless involves function. It's not just functions.” To some people, serverless can be other things besides functions, but a lot of people do look at it as functions, and a lot of functions are kicked off by events. Can we provide some little inoperability for the eventing infrastructure we've been talking about the entire time here?

We started looking at that. We came up with a rough draft specification for what the cloud events might look like, or the cloud event spec might look like, and then we took it back to the technical oversight committee and said, “We want to form a project around this as an incubator project.” They said, “Yeah, looks good. Go off and do it.” We created a sandbox project under the CNCF and there we are. That's how we got started. It was an offshoot from the serverless working group.

It's basically the same group of people within the serverless working group. In terms of players there, since you asked, you pretty much have all the major players there in terms of cloud providers. I'm not going to rattle off names, mainly because I'm afraid of leaving one off and upsetting them. We have all the major cloud providers and we also have a fair number of end-users too. Big companies who are not copywriters, but actually use this stuff and they're contributing on a regular basis to make sure that their use cases are understood and take into account as we go forward in this space.

[SPONSOR MESSAGE]

[0:36:49.3] JM: Software engineers can build their applications faster when they use higher level APIs and infrastructure tools. APIs for image recognition and natural language processing let you build AI-powered applications. Serverless functions let you quickly spin up cost-efficient, short-lived infrastructure.

Container platforms let you scale your long-lived infrastructure for low cost. How do you get started with all these amazing tools? What are the design patterns for building these new types of application back-ends?

IBM Developer is a hub for open source code, design patterns, articles and tutorials about how to build modern applications. IBM Developer is a community of developers learning how to build entire applications with AI, containers, blockchains, serverless functions and anything else you might want to learn about.

Go to softwareengineeringdaily.com/ibm and join the IBM Developer community. Get ideas for how to solve a problem at your job, get help with side projects, or think about how new technology can be used to build a business. At softwareengineeringdaily.com/ibm, you will find the resources and community who can help you level up and build whatever you imagine.

Thanks to IBM Developer for being a sponsor of Software Engineering Daily and for putting together a useful set of resources about building new technology. I'm always a fan of people who build new technology, who build new applications and this is a great resource for finding some design patterns and some new ways to build and leverage these technologies, like serverless and containers and artificial intelligence APIs.

Thanks again to IBM Developer.

[INTERVIEW CONTINUED]

[0:38:48.8] JM: When did the cloud event spec start focusing specifically on the set of metadata, as opposed to deciding other things, like I don't know, the size of the event, or the message of the event, or –

[0:39:04.5] DD: Basically from the beginning.

[0:39:05.7] JM: From the beginning.

[0:39:06.2] DD: Yeah. The entire notion here was as I said, I think one of the first questions, our goal here was not to boil the ocean, or to create yet another common event format that has been tried many times in the past. We wanted to basically look at what is the bare minimum to help get the message from point A to point B. The focus on in order to do that, let's look at the metadata needed to do that routing. That's basically how it got started. What is the bare minimum data a receiver needs to basically process the message to pass it along to the next hop? That's four or five bits of – piece of properties, metadata things.

[0:39:40.8] JM: Is the size a constraint, or it doesn't matter how big these events are?

[0:39:44.9] DD: Well, obviously to the infrastructure it may matter, but to the cloud event specification, it does not matter. Actually for example, we don’t have any property called content length, like HTTP does. We don't care about that. If the transport, or the function infrastructure, or the producer versus consumer have those constraints, they're welcome to solve that problem themselves. We're not there to help solve that particular problem. We're there just to get the message from point A to point B. If it doesn't involve routing per se, we're not going to touch it and we're not going to try to tell people how to get those other jobs done. Maybe that's something we look on the future for other things, but that's not the purpose of the cloud event spec. There are other – other people are going to look at those problems.

[0:40:24.5] JM: Tell me more about the deliberations that go on in the project.

[0:40:30.9] DD: Can you elaborate what do you mean by that?

[0:40:32.4] JM: Sure.

[0:40:32.8] DD: How to make decisions kind of thing?

[0:40:33.7] JM: Yeah, how do you make decisions and what are the – I guess, what are the debates right now and how willing are people to adopt the cloud event spec?

[0:40:42.1] DD: Those are two very different questions. Okay, let start with the deliberations first.

[0:40:44.1] JM: Yes.

[0:40:44.9] DD: It's pretty much what you might expect in terms of people come forward to the group with an idea. We're doing this all through Github issues and per request. Someone says, “I have an idea.” If it's still in its infancy, they may open up a Github issue and then we have some discussions about the issue. Ultimately, if someone is really serious about it and wants to be a champion of that particular change they want to see, eventually they're going to open up a pull request, which is pretty much a pull request to change the text in the specification and then we have discussions about it, typically through the PR itself, through comments. We have weekly calls and we discuss some of those pull requests on there.

Because we only meet once a week for an hour, we actually try to limit deep technical discussions on the phone calls, because otherwise you could rattle really, really easily as you can imagine, right? When this seems like there's strong disagreements in terms of how to proceed on a particular problem, more often that what we'll do is we'll say, “Can you guys who really care passionately about this issue take it offline? Go discuss it. Take as long as you need, but then come back with basically a joint proposal for the rest of the group to consider.”

That way, to be blunt, all the contentious work is done by those guys who really, really care about it. Then the working group – I'm sorry, I shouldn’t call it working group. The project can then basically look at the end result and say yes or no to it and they don't have to necessarily get into a deep dive just arguments about it. They may have a deep discussion about it to make sure they understand it, but typically by that point, it comes down to a yes/no vote thing.

It's relatively straightforward process. I think we've been very successful with it. We haven't actually had that many disagreements, which is really bizarre, because as you might imagine with most anything, people tend to have very strong opinions about certain things, but we've been really lucky. Most people have had similar goals here. They want to keep it down to the minimal thing. When you only have four or five properties, there's not a whole lot to bicker about, right? A lot of it comes down to things like, “Oh, are we going to restrict the character set of this or some way? How are we going to make life easier?”

For example, we had a long discussion about the property names. It may sound silly, but that was one of the longer discussions was do we allow upper and lowercase letters? Do we allow underscores? All the various things that go into it. It wasn't that we all necessarily disagreed, as much as we had a long discussion to try to make sure that we pick the right character set to make it as easy as possible for people to support it. Because for example, not all transports for example, are case-sensitive. Some are, some aren't.

That's going to be pain in the butt, when you have to take the cloud event and moving from one transport to another. You may lose some information as you move from case-sensitive to not case-sensitive, right? Those things came up. It wasn't really that we were even necessarily bickering. It was just we had to take the time to make sure we got the right answer and yeah, make sure that actually solved the right problem going forward.

That's how we work right now and it's been a really, really enjoyable experience. Because one of the things that's also a risk when you start getting collaboration from so many companies like we have, which by the way on the average, we have 30 people on a regular basis on average every week joining our phone calls, which in my opinion is actually really, really high for other working groups that I've seen. That shows the commitment that we have in the group.

I've seen very little politicking, put it that way, because a lot of times in these groups, people may want to push their own agenda, or their own products, needs and stuff like that. I've seen very, very little of that at all. It's been a very, very fun group to work with in that perspective.

[0:44:11.4] JM: Cool. Okay. The latter question, how –

[0:44:14.8] DD: That one I was trying to avoid.

[0:44:16.8] JM: Well I mean, it's worth even explaining it, why this would be annoying to implement, if you are a cloud provider? Or why you would not want to implement it? You've already got events being created across your infrastructure. Why wouldn't you want to adhere to the cloud event spec?

[0:44:32.5] DD: Honestly, I have no idea. To me, it's a no-brainer. It is an incredibly low-cost thing to support. Now, I'm not going to sit there and say every single event, that every single event producer should produce – event that they produce should be in a cloud event format. That's a choice they're going to have to make on their own. I think it does make sense to perhaps allow for people when they subscribe to specify that they want it in a cloud event format, right? That way as a consumer, they can make life easier. I mean, ideally yes. I would love it if everybody looked at this and said, “You know what? I can add these additional HTTP headers to my message. It's not going to impact anything. It's only four little headers. If it makes my thing more interoperable and makes people – even the inkling more inclined to subscribe to me as opposed to somebody else, or to use me as their service provider, why not do it?” It’s like I said, it's four little properties. If it makes my life easier for people, why not do it?

[0:45:21.6] JM: Well, let's say you're the function as a service provider who has completely dominant market share, like hypothetically.

[0:45:29.6] DD: Hypothetically. Those exist. I didn't know that.

[0:45:31.7] JM: Those exist. If I'm the function as a service provider with 95% market share and I've got a bunch of other cloud providers that are suggesting to me in this group like, “Hey, what if you just add these other headers? It would really help out in interoperability and so on.” What do I have to gain for doing that?

[0:45:52.0] DD: Let me answer it this way, because it gets into area that I – to be honest, I’m not even sure I feel comfortable answering it. Let me answer it this way; I think in general, the consensus from the community and our users in particular, is that interoperability and portability, other applications between providers, or be able to interrupt with your different back-end system – not back-end system, but your various cloud service providers is a goal that everybody wants, because people do not want vendor lock-in.

I think to everybody's benefit to reduce those friction and pain points, roadblocks, whatever you want to call them, for people so that they can focus on their business logic, I think in the end cloud providers that allow people to focus on their business logic and less on infrastructure are probably going to be the ones that people might gravitate to in a long run. That's the way I just tend to look at it.

[0:46:46.8] JM: Tell me more about what you think multi-cloud looks like. I get the sense that in another six months, 12 months, 18 months, 24 months we're going to be talking a lot more about multi-cloud.

[0:46:59.3] DD: Okay. Obviously, we're leaving the boundaries of cloud events and serverless –

[0:47:02.5] JM: Well, to some degree. I mean, there's –

[0:47:04.8] DD: I mean, multi-cloud to me is basically not being forced to use one cloud provider for everything, right? You're spreading your workload across different cloud providers. Again, that's where I think interoperability, portability is all going to come into play. You can choose the right cloud provider for that part of your application, if you choose to spread it out.

[0:47:21.6] JM: How will those decisions be made? If I'm already on one cloud provider, what are going to be my reasons for choosing services on other cloud providers?

[0:47:30.9] DD: I would assume it's pretty much the obvious things, right? Functionality, cost, performance, whatever characteristics you want to lump into those. I think those are the things people are going to obviously look at first.

[0:47:44.1] JM: Yeah. What do you think is going to be the longer-term impact of people moving to functions as a service for development? How is it going to change architectures?

[0:47:52.9] DD: Honestly and this is just my personal opinion here. I don't know, because I'm not sure we as a community completely know when functions versus microservices versus just flat out containers versus VMs, I'm not sure we actually have a clear story yet on when each one is supposed to be used definitively yet, right?

I mean, well actually I guess, that's not a 100% accurate, because I think a lot of times people do know, okay, VM maybe overkill for some things. Container might be better. I think it's a little bit fuzzier when you start breaking it down all the way down to a function level, because obviously if you go too far and you try to take your monolith and split it up into nothing but functions, you're going to have a whole different set of problems at that point, right? You now have one thing to manage and deploy, you now have a thousand things to manage to deploy. You don't have to worry about the networking management between all those things.

It's not something that you necessarily want to jump into lightly, right? When you choose to do a function versus a container as a service versus a application at the platform service level, I'm not sure there's a clear answer yet for those, so I'm not sure what the architecture is going to look like going forward. I think we're still in this investigative, educational period for everybody; consumers, as well as providers.

I think some projects like Knative are actually going to help – possibly help form that answer, because – this may sound almost contradictory of what I was saying, but when I look at projects like Knative, it's almost blurring the lines between some of those things. Because Knative in my mind, or maybe it's a whole wishful thinking, is basically looking more at it like, just give me your code, or your container if that's what you want, and I'll host it for you. I'll auto-scale it for you. I'll scale it down to zero if I can, if that's what you want me to do for you, right? I'll set up the routes for you, right?

Whether that thing that you're hosting is a "function," whether it's a traditional application from a past perspective, or just a container from a container as a service, it becomes less interesting from a infrastructure perspective. Now it's more back on the user to say, "How do I want to break up my application? What is best for me, right? Which pieces do I want to scale independently from the rest of it based upon my business needs, or whatever criteria I want to place on myself," right?

I can now start worrying less about my infrastructure and figure out what's best for my business. I think that's where the education comes to play and best practices to guide people to help make those decisions. I think that's going to really help shape things out. How it's going to look? I don't know for sure.

[0:50:21.9] JM: I'm doing a show about Knative tomorrow, actually. Can you tell me, what is exciting about it to you? I've done a little bit of research on at this point, but –

[0:50:29.0] DD: I think to me, the exciting thing is actually what I just said, in the sense that it's blurring the line between PaaS, CaaS and FaaS. Because I think from an end-user point of view –

[0:50:38.8] JM: By the way, PaaS being like?

[0:50:40.2] DD: I'm sorry, platform as a service.

[0:50:42.0] JM: Well, that's like Kubernetes as a service versus container instances as a service versus functions as a service.

[0:50:48.3] DD: I look at it slightly differently. I know some people look at Kubernetes as a PaaS. I tend to look at it more as a container as a service. Maybe it's because I'm too low-level and I'm a Kubernetes developer. I tend to look at PaaS more like Cloud Foundry, right? Where you basically give us your code while Cloud Foundry does support giving them a container, they started out saying, "Just give me your code, I'll compile it for you and I'll host it for you," right? That to me is a traditional PaaS.

Then you have the container as a service, the CaaS stuff. That's more along the lines of I don't care how you compile it. You as the developer going to compile it, you just give me a container, I'll host it for you, right? Now in the FaaS base, it's funny, we're actually getting more back towards the PaaS world, right? Because a lot of function as a service people, and including Knative are saying, "Give us your code again," right? You've gone full circle for that perspective, but they do support giving a container and not just code.

What excites me about Knative is that it is blurring those lines and it gets the application developer away from this thinking about their infrastructure, right? More about what is best for me and how do I want to write my application? In the same way, they can choose whatever language they want. I want them to be able to just say, "You know what? I want this part of my application to be only this big and scaled independently from the rest." I don't want to necessarily have to care about whether is it a “function?” Is it going to be doing as a container? isn't going to be done as a PaaS thing? I don't want them to have to think about that.

If Knative can get us towards that model where they just focus on take the code, give it to somebody else, or if they really want to give and take a container, fine. Give them the container and they'll host a forum and they'll figure out or manage all the other stuff for me; networking, whether it's how it’s scaled, do all that for me, that makes the developer's life easier and that's what really excites me about it.

[0:52:33.1] JM: Okay. Doug Davis, thanks for coming on the show.

[0:52:35.9] DD: Thank you for having me.

[END OF INTERVIEW]

[0:52:39.7] JM: GoCD is a continuous delivery tool created by ThoughtWorks. It’s open source and free to use and GoCD has all the features you need for continuous delivery. Model your deployment pipelines without installing any plugins. Use the value stream map to visualize your end-to-end workflow. If you use Kubernetes, GoCD is a natural fit to add continuous delivery to your project.

With GoCD running on Kubernetes, you define your build workflow and let GoCD provision and scale your infrastructure on the fly. GoCD agents use Kubernetes to scale as needed. Check out gocd.org/sedaily and learn about how you can get started. GoCD was built with the learnings of the ThoughtWorks engineering team, who have talked about building the product in previous episodes of Software Engineering Daily, and it’s great to see the continued progress on GoCD with the new Kubernetes integrations.

You can check it out for yourself at gocd.org/sedaily. Thank you so much to ThoughtWorks for being a long-time sponsor of Software Engineering Daily. We’re proud to have ThoughtWorks and GoCD as sponsors of the show.

[END]