EPISODE 617

[INTRODUCTION]

[0:00:00.3] JM: Serverless is a word used to describe functions that get deployed and run without the developer having to manage the infrastructure explicitly. With serverless, instead of creating a server and installing the dependencies and executing your code, the developer just provides the code to the serverless API and the serverless system takes care of the server creation, the installation and the execution.

Serverless was first offered with the AWS Lambda service, but it has since been offered by other cloud providers, like Google cloud functions and Azure functions. There have also been numerous open source serverless systems. On Software Engineering Daily, we've done episodes about OpenWhisk, Fission, Kubeless. All of these use Docker containers and probably the Kubernetes container management system. The Kubernetes container management system is useful for building these tools, like a serverless platform, or a distributed database. Kubernetes is quite useful for building higher-level distributed systems

Chad Arimura is the VP of serverless at Oracle, where he runs the FN project which is another open source serverless platform built on Kubernetes. Today, we're going to be exploring FN and serverless in more detail. In the past, Chad ran iron.io, a message broker platform and Chad is also joined by Matt Stephenson, who is a Senior Principal Engineer at Oracle and who has worked at Amazon and Google. At Google, he worked on the Google App Engine, which was arguably one of the first serverless platforms. It was great talking to Chad and Matt about the different serverless tools built on Kubernetes and the trade-offs that these different service tools are exploring.

[SPONSOR MESSAGE]

[0:02:09.0] JM: Test Collab is a modern test management solution which integrates with your current issue manager out of the box. For all development teams, it's necessary to keep software quality in check, but testing is complex. There are various features to tests, there’s various configurations and browsers and platforms. How do you solve that problem? That's

where Test Collab comes in. Test Collab enables collaboration between software testers and developers. It offers wonderful features like one-click bug reporting while the tester is running the tests, collaboration on test cases, test executions, test automation integration and time tracking.

It also lets you do a detailed test planning, so that you can configure platforms and browsers and configurations or any variables in your application and divide these tasks effortlessly among your team for quality assurance. All of the manual tests run by your team are recorded and saved for analysis automatically, so you'll see how many test cases passed and failed with plenty of QA metrics and intelligent reports, which will help your applications quality.

It's very flexible to use and it fits your development cycle. Check it out on testcollab.com/sedaily. That's T-E-S-T-C-O-L-L-A-B.com/sedaily. Testcollab.com/sedaily.

[INTERVIEW]

[0:03:47.7] JM: Chad Arimura is the VP of serverless at Oracle, and Matt Stephenson is a Senior Principal Software Engineer at Oracle. Guys welcome to Software Engineering Daily.

[0:03:56.6] CA: Thanks for having us.

[0:03:57.3] MS: Thank you.

[0:03:57.9] JM: I'm going to start with a little background on both of you. Chad, you built and led iron.io, which is an event processing system and a message queue. This is a managed service that was not built within a major cloud provider. How do you build a managed service from outside of the major clouds and what are the difficulties and the benefits of that process?

[0:04:23.8] CA: Yeah, that's a great question, Jeff. Yeah, we built iron.io back in 2010-2011, back before serverless or any of this stuff was even a thing. One of the big challenges of that is we're relying on all the cloud providers to provide infrastructure for us, but it's also an opportunity because a big vision that we had was creating open-source multi-cloud platform. We had the advantage of having a development layer on top of multiple clouds where our

customers can easily run workloads and move them between different clouds. It was pretty early, but it was a pretty strong vision and we're proud of what we built there.

[0:04:59.1] JM: What cluster manager did you use at iron?

[0:05:01.6] CA: This was before all of the big cluster managers that you know of today. We built everything basically proprietary, but when Docker came on in 2014, we used that to containerize everything. We made that the first class citizen package manager for the workers themselves that customers would ship code. Docker became a central point to our infrastructure, but we built everything on our own.

[0:05:23.5] JM: As the container orchestration wars unfolded, I guess you were bearing witness to that. You probably were faced with the choice of do we go with Mesos or Docker Swarm and eventually Kubernetes like everybody else. Did you have any experience like that in the container orchestration wars period?

[0:05:42.3] CA: Yeah. I mean, definitely everybody was trying to get our attention and everyone was saying that they were going to be the future of the operating system for the cloud. Honestly, they're great technologies all of them, but we took a step back. When we launched iron functions, we said we're going to be completely orchestration agnostic and we're going to build it in a way such that it handles its own scheduling, but you can utilize any of you orchestrators to provision and help with the orchestration behind the scenes and then help in provision infrastructure. Outside of that, we kept it all above the orchestration layer, so that we handle that all.

[0:06:16.4] JM: This is one thing that took me a while to understand is that there is scheduling and then there's also orchestration and there's also just the API that you're given. When I was trying to compare these different container orchestrators during the container orchestration wars, when people were really trying to make a decision for which of these Kubernetes, or Mesos, or Docker Swarm, or whatever else, there were a bunch of other ones, which one to use, there was a lot of different axes to compare them on, but I think the reason that Kubernetes took all of them by storm was probably a combination of network effects and the limited API constrain and there was a clear bounded context to what Kubernetes was trying to do. From

your perspective, what led to Kubernetes taking more mindshare from the other container orchestrators?

[0:07:14.5] CA: Yeah, honestly a big part of it is perception and the fact that, and Matt will have a great perspective on this having been at Google before, but my take on it is that a big part of it is that when Google says they use something to handle its millions of containers per day, or maybe even billions per day, the community listen. That's a pretty big stamp of approval to put behind a specific technology and if you give it the airplay and you give it the developer relations and you built a good codebase with Google stamp of approval behind it, the perception is that this thing is more battle-tested than the others. It's got a really solid foundation that could be backed from, and we're going to put our trust in it. That was a big part of it. Obviously, there's a whole, like you said, a whole bunch of axes there, but I think that was a big part of it.

[0:08:00.3] MS: Yeah, I would tend to agree as well. The thing that I think makes Kubernetes stand out against other Google projects that are similar, leading up to it is perhaps that it's gotten a lot – it has a much nicer and cleaner user interface. The primitive constructs are a lot more straightforward, like contrasted with Ganeti, which was a virtual machine management open-source project from Google as well. I think that the actual engagement with community was a lot stronger with Kubernetes and they reached out to other organizations to try and form, to leverage the skills and experience it with people outside of Google, which I think is really critical for successful open-source projects that Google has shipped over the years.

[0:08:42.1] JM: Matt, you've worked not only at Google but also at Amazon. When you were at Amazon, you built a scheduler and a build system. Tell me what you have learned about building internal clouds from your time at Amazon and Google.

[0:08:57.8] MS: Well, I guess at Amazon that was early in my career. I learned the lesson that build and deployment and scheduling and runtime, it's all the same thing. You start off that whole experience of a developer writing the software and building the actual artifacts that are going to be run later on, it all needs to be integrated together. I think that was probably one of the biggest takeaways I got from Amazon. Then at Google, I leverage that to build a lot of support through tooling of app engine. Yeah, anything specific?

[0:09:28.2] JM: Well, I guess the two of you have a bonding fact over the experience of having to build your own orchestrator, or fully integrated orchestration and scheduling system from scratch. Have you shared any war stories with each other?

[0:09:45.1] MS: Yeah.

[0:09:46.4] CA: We all have our scars.

[0:09:48.9] MS: I think recently in a chat channel someone called me the Colin McLeod of cloud because of all the stories that I've been able to rattle off.

[0:09:57.8] CA: Yeah. From my perspective, I think we're fresh out of KubeCon Europe and already starting to focus on the efforts going into KubeCon US and Seattle in December. It's clear that Kubernetes has become a big winner in this space and there's a lot of effort going into it. There's there I think a lot of raw spots too and it's still really early for you. I think we're still, to use the Gartner hype cycle, we're still at the top of the hype cycle, the peak of inflated expectations with both serverless and to some degree Kubernetes. The question is does an enterprise invest in just Kubernetes? Does an enterprise invest in a full integrated cloud solution, or is it a combination of the two?

My battle scars come from having built this stuff, like we want to avoid building this stuff as much as possible here, which is why we're putting a ton of investment into Kubernetes, aligning FN with Kubernetes, launching the Kubernetes service on top of Oracle cloud running on top with bare metal. We literally just got out of a conversation yesterday about what are all the different surface areas that we want to intersect FN Oracle functions with Kubernetes, and how do we align ourselves really tightly with that community, because there are so many things that it does really, really well that we've all built in the past so many times and why should we reinvent the wheel when Kubernetes is going to tackle a lot of these hard challenges. I think that's where the community is today, but we still have some way to go there.

[0:11:16.6] JM: Chad, when you were at iron, iron was itself a serverless service. It was a serverless message brokering system, event processing system, and you saw firsthand that people do not want to run their own message queue. They want a message queue abstraction

to interface with, but they don't want to have to think about scalability for example, and then you take that notion to the logical extreme through the things like database-as-a-service, message queuing as a service, data engineering as a service, when you think about something like Amazon Redshift, or Google BigQuery. You take it to its logical extreme and you get to the serverless function idea. Clearly, people want this serverless at every layer of the stack from everything along the gradient from these functions as a service where you have primitive elements of code that are deployed in a serverless fashion to highly integrated serverless constructs, like a serverless message queue like you were building on iron.io and serverless database. What other beliefs have you developed around the idea of serverless?

[0:12:31.9] CA: I'm not sure if you're familiar with ironworker, but was actually our flagship product at iron. In 2010, we released something called simple worker, which was the concept of zipping up some code, including your dependencies, using our API to send that to us and then we would run that for you and we actually called it serverless, which is I know that Simon Wardley talks about how he built a FaaS in 2006, 2006, but he actually call it framework as a service. We're very much aligned with being early in this market.

If you think about what that – what we had built in 2010, fast forward to 2014 when Lambda came out, I mean, that is almost exact replica of what they built. However, we were primarily asynchronous code batch processing as opposed to more synchronous API development. When you think about when customers came to us and they said, “Well, we really are interested in this,” because our developers don't want to deal with infrastructure, they don't want to deal with orchestration, they don't want to think about that stuff. What can we do with it? The answer is always, “Well, you can almost do anything. It is just code.”

We leave it up to you to figure out what you want to do with it. That is both a blessing and a curse, especially for a startup. When you tell someone they can do anything, suddenly you're talking to nobody specifically or no one solution. At a place like Oracle where our customers are doing everything under the sun and we can invest heavily in this over the course of many years as its market develops, really the answer is you can do anything with serverless and we're starting to see more and more architectures move over like you said, all the different components are moving to serverless, so you're standing up databases and not thinking about the individual VMs that run them. You're standing up queue services, you're standing up

notification services, you're standing up e-mail services and text messaging and now functions, and you're never having to think about any of the underlying infrastructure. It's a very powerful thing for developers once they figure out how to wire these things together.

[0:14:21.4] JM: You're both at Oracle now. You're working on the FN project. What is FN?
[0:14:27.9] CA: Well FN is – if you look – if you're using a Mac and you look at the bottom left corner, there's an FN key that stands for function. That's basically what the FN project is. However, the FN project actually encompasses more than just functions as a service, but at its core, it's a functions as a service platform, a 100% open source under FN project Github account or org, and anybody can take this and stand it up on any infrastructure and run their own serverless platform akin to like a Lambda, Google cloud functions, etc. They were providing a bunch of tooling to make that very easy and we're operationalizing that at Oracle as you can imagine.

There are other adjacent projects as well, such as FN flow, which is more of a workflow type product. Again, a 100% open source, but it uses primitives directly from the language printers themselves. For example, Java we use completable features from Java aid and that allows you to build more complex workflows. It's basically a project that's looking to operationalize and bring serverless to the enterprise, both in the cloud as a service and on-premise as well. That's the spirit and the goal of FN.

[0:15:36.4] JM: We've done shows at this point with Fission, with Kubeless, with OpenWhisk from IBM. These are all different open source serverless platforms. We've also done shows about the Microsoft service platform, the Amazon serverless platform. There's a lot of options obviously, but I think we should at least for the sake of the first part of the conversation focus on the class of open source serverless platforms that are built for Kubernetes, which in itself encompasses the FN project, Fission, Kubeless, several other projects. Why are there so many of these things? Why are there so many serverless platforms that are built for Kubernetes?

[0:16:28.4] CA: Anytime you strike on something really valuable and really useful and really interesting, there's going to be a lot of, I don't want to call them clones, but there's going to be a lot of options, and then you see a consolidation. This is the pattern that we've seen over time with every technology that's come to market. As soon as the hype cycle starts to move up,

everybody wants to jump in and build something, everybody's got a slightly different solution, or a slightly different take on it, or a slightly different vision for how the future goes and they release something, it becomes popular, or slightly popular.

Then once you really – once this hype cycle starts moving into the trough of disillusionment, you start to see consolidation. It's no different with functions as a service. We released iron functions at iron, I think back at about 2016. We were one of the first platforms on. We've always had the vision of bringing this to market. We decided that iron wasn't the best platform to do that, but when we met the folks at Oracle, we realized that, “Hey, here's a major cloud provider, a major infrastructure provider that's growing like crazy, and they are really interested in serverless and they're also interested in doing this with an open source angle to it.” That's when we came to the idea, let's bring iron functions to market as FN project inside the cloud group here at Oracle, so that we are one of the major, only major cloud providers to be bringing a fully 100% open-source platform to market and then operationalize that on top of Oracle. We've got a whole bunch of strategic advantages, but that's really the big one right there.

[SPONSOR MESSAGE]

[0:18:00.7] JM: Flatiron School is an outcomes focused coding bootcamp that trains people to launch a new career in technology in as little as 15 weeks. Flatiron School trains students across the full stack, from the front end to the back end and everything in between over 15 rigorous weeks. Students learn to think and build like software engineers, from developing coding skills to gaining an understanding of how products are designed and managed. Flatiron school has both online and offline courses. If you want to get started with an online course, you can go to flatironschool.com/sedaily and get $500 off your first month of Flatiron’s online web developer program to get started with a career in software engineering.

If you like that online course, you might enjoy checking out Flatiron Schools in-person courses that are the 15-week immersive courses. On their website, Flatiron School has a jobs report that details the outcomes of students who have gone through Flatiron School and they report a very high percentage of graduates getting a job offer and accepting a job offer after their curriculum at Flatiron School.

Check it out at flatironschool.com/sedaily. If you want to get started with the online web developer program, you can get $500 off your first month and get a jumpstart in software engineering. Thanks to Flatiron School for being a sponsor of Software Engineering Daily.

[INTERVIEW CONTINUED]

[0:19:45.9] JM: The idea that we're going to eventually have some convergence around the vertical of serverless platforms that are built on Kubernetes, there's some debate around this matter. Obviously we had convergence around Kubernetes and we reached a point where every major cloud provider is building a managed Kubernetes service. You could imagine a world where eventually every cloud platform offers some API into the function management and deployment side of things, and maybe it'll just be an API that's defined as within the community somehow, or maybe it'll actually be an implementation, like the FN project, or OpenWhisk, or whatever else.

It seems like there's a race to be able to disintermediate the AWS Lambda side of things, because what people are afraid of is that AWS Lambda could potentially be a mode of lock-in for people, a very low level mode of lock-in on AWS. I guess, one thing I'm having trouble understanding is how compliant are these other serverless platforms, the open source serverless platforms, how compliant are they with AWS Lambda? Would I be able to – if I've written all of my glue code on AWS Lambda and I want to port it to an open source solution, how much pain is there in ripping out the tight couplings to the Amazon platform?

[0:21:19.8] CA: I don't think that the functions themselves are really exactly where a lot of the lock-in is going to happen. I mean, we've even got an API endpoint called FN Lambda, that will import Lambda functions directly from AWS into FN, whether you're running FN on AWS itself on top of the primitives, the server primitives, or you're running it on prem. I don't think that's really the source of the stickiest lock-in.

I think that the secret there is that Lambda, what it does or serverless, what it does is it ties together all of these different services and they've got nice integrations and the experience of serverless is all about the integrations that you have with all the different services and all the different tooling pieces that you're using. When you look at AWS and or Google or Microsoft or

any of them, it's a really great experience to integrate these with all the different, like DynamoDB and SQS and S3 and all these different things tie together, but suddenly you've got 10,000 functions, all of which are tied to 20,000 different integration points with all these different services and the gravity of your data, and just a sheer gravity of all of this entanglement, that's the hard part to move off.

If you're not thinking about your architecture upfront, I think that's the biggest opportunity that Kubernetes and the CNCF and the cloud and the serverless working group and cloud events, really the big challenges that we're tackling there inside the working group is how can we create an architecture using open source components that you can take this thing and deploy it to any of the primitives on any of the clouds, but not get basically tangled into that mess? Also have the opportunity to take advantage a lot of the services that the different providers bring to market. I think that's what we're all trying to figure out.

[0:22:58.6] MS: Yeah. I would add that a lot of Google, Amazon, they both provide APIs that are very convenient to use for various nitty-gritty operations, like searching, or GIS type operations and things like that. The nice thing about something like FN is that our primitives, our containers, and so there's always open source projects available that can provide a lot of these sources, the same functionality. It's pretty easy using like what we've built in the open to stay mostly in the open and not necessarily have to tie yourself to these various APIs. Back it app engine, it was a pretty big thing for one of the biggest use cases we added App Engine was generating PDF files, like people would create apps that just you call it and generates a PDF file. We provided a service for it for a while and didn't really have that much uptick. People just wanted to still run their own EDF generation.

[0:23:52.7] CA: This is a lot of the work we're doing with cloud events, where a lot of these things represent events. If we can standardize on what that message looks like and what that payload looks like and what representation of what an event looks like, and standardize that across the industry. There are literally every major vendor has been contributing to this effort. If you're not familiar with cloud events, it's an incubation, our sandbox project inside the CNCF now that we've been helping steward alongside some of the major vendors. FN announced a native integration of cloud events back at KubeCon EU. By the time this podcast airs, we'll have

announced some VS code integrations if you're using Visual Studio code to develop – to do your development.

We want to provide an open source experience, but standardize around things like cloud events to where we can essentially help customers unlock from the entanglement that they have on the different providers. As Oracle, we obviously want to provide one of the best infrastructure platforms to do that that's well integrated with all the other Oracle apps and everything that you're using, but ultimately Kubernetes is a big piece of that platform story of unlock from the providers.

[0:24:55.6] JM: The cloud events idea, this is important because the way that people stitch serverless functions together with other things in their infrastructure is oftentimes an event triggered system. You have an event that your serverless function will trigger on, and the API for that event if it's specific to one cloud provider, then you would have to build a way for your services to emit events that are compliant with each cloud provider you're using and it would be much easier if we just had some common language around what an event is. Is that an accurate representation of what the cloud events project is all about?

[0:25:46.4] CA: Yeah. I mean, that's a pretty accurate representation. I think the serverless working group really exist to talk about how as major vendors, oftentimes major competitors, but how can we harmonize around a set of primitives so that, or such that the application experience, the developer experience is unified and we can begin to unlock these entanglements? Cloud events was the first major piece of this. If you can standardize on what the, essentially the communication layer looks like for events, what the protocol looks like, what the event format looks like, then what that does is unlock a lot of functionality, or a lot of innovation opportunity for even upstream providers, like IoT devices. If they emit cloud events, then you can automatically all the different services and different providers can understand how to consume those. If they're then emitting cloud events, then other downstream services know how to how to speak cloud events, so there's just so much opportunity there to speak a common language.

We think that that will unlock a lot of innovation in a way that's not all cooped up on a single provider, but happens almost across provider. Ultimately, the benefit is for developers and the organizations consuming serverless.

[0:26:56.5] JM: I want to throw something out there. When I was at KubeCon, I was at KubeCon Europe as well, and I really enjoyed the conference. One thing I sensed at the conference was that there were a lot of enterprises that were very willing to spend money to invest in their Kubernetes installation. I think the vendors at the conference were very happy, because they were finding customers that wanted to pay for that vended software, and there were enterprises that were happy to pay. It was a very nice marketplace was the sense that I got, so I'm wondering first of all if you two got the same impression, but also more generally is the lesson here that even if you are a major cloud provider, perhaps a major cloud provider with as much heft as Amazon.

The lesson here is that we would have had this world two years ago if we wouldn't have been having the container orchestration wars. I mean, the container orchestration wars were probably important, because a lot of ideas got worked out and so on, but my sense is that customers were a little more gun-shy back then, because they sensed there's some friction and there's some churn going on here. Now you really get the sense that everybody is since they're aligned around Kubernetes, the enterprises are happy to make purchases around that. Did you guys conclude the same lessons, or maybe you have some other lessons around the idea of this business marketplace that has developed around Kubernetes?

[0:28:23.0] CA: It's definitely a balance between adopting early so that you can get a strategic advantage and deliver a lot of your business value earlier than your competitors, and being too early versus being too late. Or on the flip side, if you're too early and the market changes and you're stuck with technical debt that you then have to go and either continue to support or make the decision to rip and replace. Tis just happens through every technology cycle.

I think my perspective, again to reuse the Gartner hype cycle, KubeCon EU for me really felt like the first real effort to move out of the trough of disillusionment into more of the slope of enlightenment, where now we're starting to see real use cases, real workloads, real vendors. I mean, don't get me wrong, there's a lot of hype and it's still a bit early, but I think it's certainly

safe to say that this technology is going to be around for years to come and it's something that's going to be supported by a lot of major vendors. I mean, last time I looked, there was probably 48 different vendors that were providing some type of Kubernetes service. Those don't go away tomorrow if a new technology comes out that's slightly better than Kubernetes. I think that there's definitely an opportunity to invest here and that's the feeling that I got from KubeCon.

[0:29:34.0] MS: Yeah, I would say that I think that the container wars left us with a good understanding of what can be counted on now from different container orchestration provider to a different container orchestration provider, right? Kubernetes is nice, because it gives you some pretty clear constructs of how things are going to be different, say running on Google container engine, versus any other one. You've seen that before in other cycles of the IAS wars with OpenStack and all of that.

We look at each stage developers learn what they can count on when they move their software from one platform to another, and I think that's what I saw at KubeCon is that that's like now becoming a reality. People are more comfortable with locking in a little bit to a provider.

[0:30:23.6] JM: The serverless conversation that we should be having, let's get to that. The engineering of a serverless on Kubernetes platform, to give people an overview of this category of software, it's like if I want to run my own serverless platform, I could use Kubernetes as the engine that is spinning up containers that I deploy my service functions onto it. If people want more of an overview of the idea of serverless, or serverless open-source platforms, we've done previous episode. I'm going to assume that people have a bit more of a sophisticated understanding of this than I might otherwise, but when I talk to the other people, the Fission people and the OpenWhisk people and – well, OpenWhisk is not on Kubernetes, I realize that, but maybe you can put it on Kubernetes, but Fission and Kubeless, I think the big difference between those two and maybe, correct me if I'm wrong, but is their approach to the cold start problem.

That is the cold start problem being you write your code and you deploy it as a serverless function, but that doesn't necessarily mean it's stood up as a container that is running and serving traffic, it's code that's sitting in a database somewhere, and when you have a call to the serverless function when an event triggers that serverless function to run, you actually have to

spin up a container in order to run the code. You have to pull that into the database, put it onto a container, schedule the container, route the traffic to it, and so you have that cold-start problem where the first time you're responding to that function call, you have to spin up the actual container and then route the traffic to it.

Different platforms have different approaches to that cold-start problem. Maybe you do have one container that's constantly warm that can respond to this traffic. Maybe you could tell me what your vision is for the correct approach to the cold-start problem.

[0:32:24.9] CA: Well, it's definitely – you definitely identified one of the major areas of serverless that everybody has to deal with, whether it's open source projects all the way through to the major providers that have services today, cold-start is just a nature of the ball game, because when you don't run these things consistently and you only pay for the milliseconds that your functions run, well you got to fire them up and you have to ensure that you're not always running over capacity, I mean paying for that capacity as a provider, or as an open source project.

You hit the nail on the head of what some of the major challenge areas are, and everybody has a different take on it. The industry is getting better and better and I think the algorithms to be able to predict workloads is getting better. I think that it's really just going to come down to operationalizing these things, putting them into production, understanding what the workload dynamics look like, even to specific customers and specific workloads that they're running and understanding hey, this customer spins up this workload every day at this time. I can use machine learning to be able to better tell when that customers is going to spin up these workloads and bring them up at that particular time. Or I've seen this pattern before with this customer, I can predict with a high level of confidence that this pattern is going to apply also to this customer.

I think we're not there yet. I don't even think some of the major providers are there yet, but with the amount of data that we'll be able to get running this thing on on top of our cloud, we'll be able to start to make a lot of those decisions and get the data to be able to fill a lot of those machine learning algorithms to be able to do that. As an open source project, we'll be able to

pass a lot of those learnings downstream to the open – or I guess upstream to open source, so that our community can get advantage of that as well.

[0:34:03.1] MS: Yeah, I think that this actually hit the nail on the head for even before I was here at Oracle, at Google, on App Engine, that was the biggest problem that I ever had to deal with was how to get the Java Runtime to run faster. There's a lot that's happened. There's a lot that still is happening in the overall community, the software community for trying to minimize start time of various runtimes, especially in Java. We've released the GraalVM and that's really cutting down on startup time.

The thing that I learned dealing with this in my past and that has shaped some of the decisions we've made on FN and some of the decisions I made on open source projects outside of necessarily my work at Oracle, is it really the limiting factor when you have an environment like any cloud environment like this is that you're mostly constrained by IO in most cases. Things like very large containers are going to take a long time to start, and that's just a reality you're going to have to live with. You're never going to get away from that.

There's certain things that you can do to say minimize how much you're having to read in order to get your application started up. I think that's really an overall software architecture principle that we've taken a heart on our team to minimize the amount of time it takes for any individual piece that we're putting between your software and the outside world and make sure that that time to start is absolutely as small as possible.

[0:35:27.0] CA: Yeah, and you brought up a great point Matt that the JVM can tend to be fairly large and the applications deployed to JVM can be quite large. Oftentimes, a cold start is not spinning up a Docker container that takes, I don't know, 300 milliseconds to spin up plus a few seconds to load the code. I mean, it could be a minute, two minutes, three minutes to load some of these things up. This is a really big opportunity for us specifically here, because we're literally sitting sometimes feet away from some of the major language architects of the Java and the Java org. It's a big opportunity for us to tackle those problems, so that as the Java community moves into the slope of enlightenment, we can continue to use that institutional knowledge to reduce these times, using things like Graal and other tricks to be able to minimize the cold start time, so that serverless can really be about powerful use case for lots of enterprises to utilize.

[SPONSOR MESSAGE]

[0:36:27.7] JM: Today's podcast is sponsored by Datadog, a cloud scale monitoring platform for infrastructure and applications. In Datadog's new container orchestration report, Kubernetes holds a 41% share of Docker environments, a number that's rising fast. As more companies adopt containers and turn to Kubernetes to manage those containers, they need a comprehensive monitoring platform that's built for dynamic modern infrastructure.

Datadog integrates seamlessly with more than 200 technologies including Kubernetes and Docker, so that you can monitor your entire container infrastructure in one place. With Datadog's new live container view, you can see every containers health, resource consumption and running processes in real-time.

See for yourself by starting a free trial and get a free Datadog t-shirt at softwareengineeringdaily.com/datadog. That's softwareengineeringdaily.com/datadog. Thank you, Datadog.

[INTERVIEW CONTINUED]

[0:37:36.4] JM: When you're trying to solve that cold-start scheduling problem, is it a promising approach to keep some containers with the NodeJS environment on it for example, like you've already got a container with NodeJS, so therefore if somebody schedules a NodeJS serverless function, you've already got that container ready, and so you can cut down the cold-start problem a little bit. Is that a useful approach, or does it significantly cut down on the cold-start time?

[0:38:05.6] MS: Yeah, I would say so. If you think about how Docker works, right? Which is usually the container engine you're using under Kubernetes. If you're using say the same base image and just applying a little tiny layer on top of that to put your code in there, you're going to definitely reduce the amount of time it takes to pull that image across whatever network you were pulling your image from, right?

If you've got 80 different functions, they all are various different node functions, then optimizing your base image, making that pretty small, but as long as you're using a very similar one, then you can reuse that underlying layers across your entire fleet of machines that are running Docker, then I think you're going to really speed up cold start.

[0:38:47.3] CA: We take it a step further too, rather than just keeping an environment around the FN load balancer, we call it the FNLB, actually has memory and knowledge of where all the different customer containers are running on which VMs. It essentially keeps a map around of knowing where to send the traffic, so such that the containers are still running for customers. We can actually continue to send traffic to a single pod, or a single container that's running for a customer and that will just basically respond in real-time as opposed to having to load any container up at all, because even Docker can take 200, 300 milliseconds just to spin up. If you have a synchronous function that's too much, because the user is left waiting for something. There's a lot of interesting and hard challenges that are opportunities for us here to solve in that area.

[0:39:31.5] JM: Right. The FN load balancer you mentioned, this makes traffic get routed to hot instances. Explain in more detail how this FN load balancer works.

[0:39:42.9] CA: Well basically, it keeps essentially a map of where all the customer functions are running, and it can essentially know where to place the workloads that are coming in. First request comes in, it does a quick scan, it knows that a customer – it's customer A, function 1. I mean, it can send it to a specific nodes that are running function A1, but the nodes will actually produce back-pressure if they're too busy. It will let the router know that it needs to start to spin up more functions on other nodes and it can then start routing traffic there. It handles a lot of the intelligence, so that's something that we've been working pretty hard on and that's another piece of the project, which is called the FNLB.

[0:40:23.2] MS: Something that about one of your earlier points is that what used to happen to speed up cold-start for JVM-based passes, I think definitely App Engine did this, but some of the open-source stuff did it too. Was any of these things like nail gun to spin up the actual JVM itself and then apply a war file later on. I think that was how – we did that with Apollo at Amazon a

little bit, we tried getting that to work, so to restart the JVM between app redeployments. That's some neat stuff that you can do for optimization.

[0:40:52.8] JM: Indeed. When we're talking about the basic deployment process of you get your function written, you deploy it to your serverless platform, whether it's running on your Kubernetes cluster or some managed Kubernetes cluster at a major cloud provider, what happens to that code? Does it just get put into a database, or does it get stored in a container running somewhere? What do you typically want to do upon the deployment of a serverless function?

[0:41:22.3] CA: Yeah, there's a little bit of magic to FN deploy, but none of the magic is behind a curtain that you can't see. It's pretty straightforward. What you'll notice with FN is it has a set of commands, FN tests, FN run, FN deploy, and one commonality across all those commands is that's doing a Docker build. It's either doing a Docker build with a Docker file that you provide if you want to bring your own Docker file, or Docker image. Or it will use a pre-built gold container, a gold image if you do FN build or FN run, it will use let's say Go, or Java, or node, or whatever language that you want, it will build a container and then it will send it to a container registry, could be running local, it could be running on Docker, it could be running on some private registry, and then it will do behind the scenes in FN routes update, or FN update routes, I think we changed the order of those verb-noun.

Then we'll update the route to tell it, "Hey FN API, you have a new image and it's been tagged with the new version. Next time you execute, pull over that new version and operate – execute that code." The commonality is that everything is built around a Docker process, so this opens up the possibilities, the Docker ecosystem, you can do Docker container scanning, you can use CICD framework solutions at all encompass Docker and make that a first-class citizen. It's a pretty powerful model.

[0:42:39.2] JM: You guys are working on this at Oracle. Why is this project valuable to Oracle?

[0:42:45.0] CA: I think for a lot of reasons, but Oracle has made a really big bet in cloud, so if you've seen a lot of the announcements coming out, we have something called Oracle cloud infrastructure and then it specializes in bare metal nodes, and GPUs, and it's one of the fastest

cloud, it's one of the least expensive cloud, it has enterprise SLAs. A lot of it is made of compute storage and networking.

As you can imagine, we're building a whole lot of other services that developers are asking for and serverless is obviously a really major component to the future architecture of software. I think they recognize this really. We started talking to them, and the reason that we did it open source is because it's a differentiator for customers to be a major cloud provider that's taken an open source approach to serverless and Oracle is really interested in this vision, because I think that a lot of the customers were going after both enterprises and developers that really like to see an open source alternative and Oracle believes in that vision, and that's where we found ourselves.

[0:43:43.5] MS: Yeah, to speak to both of your prior two questions, I think that the fact that there is no special magic for where we put your software. It's a neat thing about how we built this. There's nothing magic there. You can go look at the code in the open source project and you'll know your codes going in a very specific location. It's just going into a Docker registry, so it's not like other platforms that I've worked on in the past, where we have these massive internal projects that you don't really know about as a customer. No one really knew about spanner back in the day at Google, so that's a big differentiator, I think.

[0:44:17.6] CA: Yeah, and it's interesting being here at Oracle, because we're releasing a lot of the services and some of the other providers have, but we have the luxury of be able to see what was done right and what was done wrong. I think there are some really key areas that we're going to innovate and out lead the market. Then there are other areas that we don't need to be innovators, but we can have the advantage of knowing what worked and what didn't for some of the other providers in those areas. All of that value then gets passed to both our existing customer base and our future customer base as well. Expect a lot of cool things happening.

[0:44:48.8] JM: Are there any examples of services you can think of where there are learnings that have come out of previous cloud provider experiments, where you feel like maybe the benefit of not being the first mover can manifest?

[0:45:04.1] CA: One of the key areas is how the integration experience looks like with various services. When you think about some of the other providers, they had a bunch of services and then their function service came out and then they're like, “Uh-oh, how do we get this thing integrated?” Then suddenly, you look like a service that bolted on and superglued a bunch of pieces together, and the user experience, there's some areas that are left lacking there. It looks like you're using 40 services to build an application. Whereas, I think one approach we're taking at Oracle is it's going to be a much more beautiful experience for application developers and it's going to be built more on open source components around Kubernetes and the CNCF projects where it's more of an integrated approach and it feels like a platform as opposed to a bunch of rubber band superglued services. That's one thing that we're focusing really hard on.

[0:45:52.8] MS: Yeah, I would say I definitely pull a lot from working on App Engine when it comes to decisions around tooling and such. I mean, when I got on that team at Google, one of the things that I noticed was we still were recommending people build their war files using ant when maven had been definitely the leader and build tools for Java for years at that point. Yeah, we're trying to build everything so that existing tools that people are using in the industry work. We're not going to build something where like disconnected from our community.

[0:46:25.2] JM: App Engine is a pretty interesting historical example, because it was – well, I mean, it depends what you classify as serverless, or how liberal you wanted to use that term, but I think many people would say it was the first serverless deployment platform. Yet, I mean, it did get widely used, I think Snapchat even uses App Engine, but I think Heroku maybe took a little bit of its thunder. Heroku seemed to have done something a little bit different, maybe it was a focus on the dashboard, or something to do with the APIs. I don't know. Do you have any other added perspective from that time of work on App Engine which was arguably the first serverless platform?

[0:47:05.9] MS: Yeah, I think that there were some really big customers. Of course, yeah, Snapchat was, like I'm out of my NDA timeframe. Yeah, Snapchat was definitely the biggest one that we had as far as I know. One of the neat things about working on that at Google was that we had a lot of internal projects and pet projects of side companies that would run on us, because it was a lot cheaper to try and provision infrastructure through us than it was to try and figure out how to get it all the way through it, to get some priority and Bourque.

One of the neat things that also people brought to App Engine a lot was various gaming platforms. A lot of the times you would play like, there are a whole lot of video games out there that the underlying actual runtime of the game was running inside of App Engine. You never noticed it, because you were inside of an app or something like that on your phone, but tons of that was actually running on top of App Engine. One of the things that I guess, I learned there was that people really wanted the heavy workloads and they wanted to build this stuff and run it without having to do any infrastructure management, because there were companies that didn't traditionally have a big infrastructure group or were founded in places, parts of the world where they're not in Silicon Valley. You can't go just find a DevOps engineer off the street, pull them in.

I think that that has led me to a lot of decisions here on FN that are mostly around, like how do we make it so that we can make an incredibly flexible compute platform that's really easy for people to use, they don't have to work too hard to use it and they can bring big heavy workloads to us and we'll do it.

[0:48:41.2] CA: Yet, it's interesting you use that example that was actually App Engine was a big inspiration for iron.io. It's one of the reasons that we started the company, because App Engine was pretty early and they had this integrated approach to these different services. For example, the queues were integrated with the database, were integrated with that. There were certain constraints that you had to follow, but if you followed those constraints you had this infinitely scalable application and we looked at that Travis, and my co-founder and I looked at that and said this is amazing. This is where the world is going. I can trust Google to run these things and I never have to worry about infrastructure again.

Then we realized that there were some crank constraints there that kept us from doing some of the things we wanted to do. I think a lot of other customers ran into those similar constraints, so that's when we developed simple worker, which was more of a zip up your code, send it to us and we'll run it, and then we'll figure out a way to integrate it with some of the other services. Then obviously, you mentioned Heroku came storming on where they took both the rails community and the git community and created git push Heroku master and that suddenly changed the ballgame, because it took these different waves and amplified them and everybody fell in love with it.

It is a marathon. It's not a sprint and none of these technologies can be successful with just sprinting. You really have to stay the course, and that's I think where we're just now starting to see some of these serverless architectures now reach the enterprise, and it's happening in the CNCF, it's happening with Kubernetes, it's happening with the cloud vendors and that's we're really excited to be a big part of that.

[0:50:07.5] JM: All right guys, well I think that's a good place to wrap-up. What should we expect from the FN project over the next year or so?

[0:50:15.9] CA: A lot of innovation, a lot of reliability improvements. This thing is operationalizing here. We've got customers using it. We're pushing towards GA this calendar year and there's a lot of momentum behind the project. We're currently engaged with partners and customers and obviously we're building a service internally. There's really exciting things happening with FN, so there's a – I would say we're really looking for community and contributors. We've got developer relations team, we've got a community team, we've got core engineers working on both open-source and service. You can expect a lot coming from the FN project. If you want to get involved, we've got a Slack room, just Slack.FMproject.io. Come talk to us. We're there basically 24/7, so tune in.

[0:50:59.3] JM: Okay, guys. Well, it's been a real pleasure talking to both of you and I wish you the best of luck with the FN project.

[0:51:04.9] CA: Thank you, Jeff.

[0:51:05.8] MS: Thank you very much, Jeff.

[END OF INTERVIEW]

[0:51:09.7] JM: Azure Container Service simplifies the deployment, management and operations of Kubernetes. Eliminate the complicated planning and deployment of fully orchestrated containerized applications with Kubernetes.

You can quickly provision clusters to be up and running in no time, while simplifying your monitoring and cluster management through auto upgrades and a built-in operations console. Avoid being locked-in to any one vendor or resource. You can continue to work with the tools that you already know, so just helm and move applications to any Kubernetes deployment.

Integrate with your choice of container registry, including Azure container registry. Also, quickly and efficiently scale to maximize your resource utilization without having to take your applications offline. Isolate your application from infrastructure failures and transparently scale the underlying infrastructure to meet growing demands, all while increasing the security, reliability and availability of critical business workloads with Azure.

To learn more about Azure Container Service and other Azure services, as well as receive a free e-book by Brendan Burns, go to aka.ms/sedaily. Brendan Burns is the creator of Kubernetes and his e-book is about some of the distributed systems design lessons that he has learned building Kubernetes.

That e-book is available at aka.ms/sedaily.

[END]