EPISODE 651

[INTRODUCTION]

[0:00:00.3] JM: At Google, the job of a site reliability engineer involves building tools to automate infrastructure operations. If a server crashes, there is automation in place to create a new server. If a service starts to receive a high load of traffic, there is automation in place to scale up the instances of that service. In order to create an automated response to one of these infrastructure problems like a server crashing, or perhaps a voice assistant responding in an incorrect way, you need to have insights into that infrastructure. Every service needs tools around monitoring and alerting and debugging and distributed tracing, and one benefit of working at a large company like Google for example, is that an engineer building a new product gets this tooling by default.

If you start a Maps product at Google for example, you are going to get monitoring and alerting and debugging and distributed tracing right out of the box, just by virtue of setting up that service within Google. In contrast, if I'm hacking at home by myself on some random project, I have to set up all kinds of tools to help me diagnose and resolve problems. I have to set up my own little debugging system and my monitoring system. Setting up all this tooling can take time and it can require expertise.

Stackdriver is a set of tools and instrumentation that allows developers to monitor and debug and inspect infrastructure. Stackdriver is a tool for engineers that's based on the internal observability tools that were built for Google. Mark Carter is a Group Product Manager at Google and he joins the show to discuss site reliability engineering and the creation of the Stackdriver product.

Before we get started, I want to mention I'm hiring for a new company that I'm starting, and I can't talk about the product quite yet, but I'm very excited about it. I'm looking for an engineer in the Bay Area to join me. I'm looking for somebody with significant experience in React JS and either AWS, or Google Cloud. You can e-mail at jeff@softwareengineeringdaily.com, or you can check out the job posting at softwareengineeringdaily.com/jobs if you're interested.

[SPONSOR MESSAGE]

[0:02:31.9] JM: You listen to this podcast to raise your skills. You're getting exposure to new technologies and becoming a better engineer because of it. Your job should reward you for being a constant learner and Hired helps you find your dream job; hired makes finding a new job easy. On Hired, companies request interviews from software engineers with upfront offers of salary and equity, so that you don't waste your time with a company that is not going to value your time.

Hired makes finding a job efficient and they work with more than 6,000 companies from startups to large public companies. Go to hired.com/sedaily and get $600 free if you find a job through Hired. Normally, you get $300 for finding a job through Hired, but if you use our link hired.com/sedaily, you get $600, plus you're supporting SE Daily. To get that $600 signing bonus upon finding a job, go to hired.com/sedaily.

Hired saves you time and it helps you find the job of your dreams. It's completely free. Also, if you're not looking for a job but you know someone who is, you can refer them to Hired and get a $1,337 bonus. You can go to hired.com/sedaily and click ‘refer a friend.’

Thanks to Hired for sponsoring Software Engineering Daily.

[INTERVIEW]

[0:04:14.4] JM: Mark Carter, you are a Group Product Manager at Google, welcome to Software Engineering Daily.

[0:04:19.2] MC: Great to be here. Thank you, Jeff.

[0:04:20.8] JM: You work on the Stackdriver product, which is a suite of tools that's developed in parallel to some of the tools that have been developed in Google's SRE team, the site reliability engineering team, and we've done some shows about site reliability engineering. Can you describe some of the types of internal tools that were built under the site reliability engineering brand of software engineering at Google?

[0:04:47.4] MC: We tend to think about SRE as what happens when you treat operations, the same way you deal with software products. SRE is about optimizing the performance, the reliability, the availability and providing end-to-end observability and reducing toll. At Google, we've been on a journey for the last 15 to 20 years and we have managed to build multiple billion-user services that are trusted by our customers. Many customers tell me that when they want to know if the internet is up, they go to google.com, and that's the level of availability we inspire for.

To take the toll out of the work that our engineering do and provide integrated observability, we build a set of tool in Google that encompasses the range of what a software engineer will need to operate the service. It starts with when a software is deployed at Google, the monitoring, logging, tracing, all the golden signals that are required to evaluate the availability are automatically instrumented as part of the software deployment. The dashboards needed are automatically generated. There are tools to manage incidents if and when something goes wrong. There are tools to automatically manage capacity and automatically scale, and a set of tools the deals with how do you take a service from one instance to billions of instances running across the globe.

[0:06:14.8] JM: If I stand up a service at Google, maybe it's an internal service that some piece of middleware, or it's a service like Gmail, or more likely some micro service that falls under the purview of the Gmail product; when I stand up that service, it automatically gets provisioned with a bunch of different pieces of tooling that I'm going to want out-of-the-box, regardless of what my service does. Is that correct?

[0:06:45.0] MC: Exactly, Jeff. We believe that separating the undifferentiated work from the software engineering work, allowing the software engineer to focus on providing value and automatically integrating things, like security, monitoring, tracing, automated scale-up, availability, independent of how the software is built is the way for us to achieve scale and availability. As such, as an engineering Google, all I know is I build my service, my service provide this function and I say how much do I need to scale and all the properties of my service, and the rest is handled automatically by the system.

[0:07:23.4] JM: It seems there are some subjective decisions that an engineer can make when they're setting up observability tools, because under some circumstances, maybe I want to do lower-level monitoring, maybe I want to have more verbose log management, or I want to have a specific expiration time for my logs, maybe I don't want to do log rolling as aggressively with this service, or maybe I want some specific machine learning related observability tooling. What are the kinds of subjective decisions and the kinds of configuration that an engineer might be needing to make when they are setting up that observability tooling?

[0:08:05.5] MC: When a service is first initiated at Google, [inaudible 0:08:08.1] the development of the service, it's part of our SRE practices. The SREs that will operate the service sit down with the owners of the service and ask the key questions, such as what data will your service need? What's the level of availability that you are aiming for and why? What are the critical user journeys? What signals do you need to collect and for how long do you need to keep them? What kind of logging would you need, and who needs access to those logs?

In Google, we put privacy and security first, so all data that get omitted by a service is strictly controlled. One needs to think about, should that data be generated to begin with? Once all of those are defined, we create a template for observability for that service that defines, that specify all the things that the service need to behave. That defines things like retention, how long the information is kept, the sensitivity and the information and provide abilities to dynamically change the verbosity level of the information that is generated.

[0:09:10.9] JM: You talked about some of these different sets of tools, the things like distributed tracing and logging. Can you outline in a little more detail what sorts of situations some of these observability tools help with, or what are the typical problems that an engineer is going to encounter and what kinds of observability do they need to solve those problems?

[0:09:36.4] MC: When you get to billions of containers running worldwide, the key challenge is there's no more books that you can sign in to and troubleshoot your service. There's no more some ways that you can do tile minus F and just watch the log go by, because which of the billion boxes would you choose?

What happens is we consolidate around what we call the SRE golden signals, which are things like availability, traffic, latency, saturation, do I have enough resources to meet the demand, so them saying, what is the error rates that we see for the service? When one of those golden signals hits the target SLOs that we define, and that SLO is defined by what do we want to provide to our customer, at the SRE team, or the engineering team get called and an incident gets created.

At that point, the question becomes, okay, who's experiencing the problem? How big the problem is? How do I pinpoint where in a highly complex systems that connect thousands of services involved is the problem? We use a set of tools that we also make available to our cloud customer to Stackdriver. Those include three tools, which are part of our APM suite. One is distributed error tracing, the second one is profiler and the third one is debugger.

Tracing provides the ability to figure out, show me my entire system, show me the flows, show me how the communication goes between them in real-time and show me where the problem is in the entire call chain and let me go through the entire flow and figure out which component is having the problem.

The second one is profiler, which gives you the ability, once you pinpointed where in the chain the problem is, to in real-time profile the system and figure out what component within the service is having the problem. Then finally, debugger, which is one of the really magical things in Google, allows you to debug production services in real-time without impacting the user, and very quickly at logging without redeploying the service, as well as immediately get the same powerful experience that you get from an IDE in a production environment.

[0:11:50.2] JM: You touched on something there with the fact that you can have a ton of containers under representing one single service, as opposed to the idea of a single host being responsible for a service.I remember when I was doing some shows about Prometheus, when Prometheus was – the Prometheus project, that infrastructure monitoring project that was modeled after Borgmon, I remember talking to the people who were working on Prometheus and they were saying that one aspect of Prometheus is that it represented a shift, I think if I remember them correctly, a shift from host-level monitoring, to service-level monitoring.

Meaning that, instead of focusing on looking at the logs of a specific host, you might look at the aggregated logs for all of the containers for example, that are symbolic of what that service does, so each of the replicas for example. Am I recalling that correctly? Is that the idea of service-level monitoring, versus host-level monitoring?

[0:12:57.9] MC: That's a very insightful observation and you are a 100% correct. I think we've been, you and me and the industry in a transition, which is an amazing journey really. From the point where we have pets, we have the server and we know what it does, to vast level of resourcing across public routes and our data centers, where you can deploy a thing and immediately make it available worldwide. You can have true things like Kubernetes and Istio be able to provide functionality, where you don't care about one machine; you care about service.

When we transition to those kind of thinking, projects like Prometheus and Google has been very supportive, or for me just one of the key maintenance work here, as well as things like Stackdriver provide you the ability to instead of saying what's the health of this machine, go and think about what level of service am I providing to my customer? What's critical for that service and how is that service doing against what I want my customers to experience?

[0:14:00.8] JM: With that service-level monitoring specifically, as opposed to host-level monitoring, if I notice something that's going wrong with my service, doesn't it become host-level response? Wouldn't I want to kill one of those containers, if the container – if I find out – if I'm doing that service-level monitoring and I noticed there's some anomaly, is my course of action going to identify the container that's behaving anonymously and kill it and let a new container start up to replace it and hope that that solves the problem? Is that a realistic order of operations?

[0:14:37.9] MC: That's a super interesting question. I don't know that there's one-answer-fits-all, but I can tell you that in our experience in running very vast production environment, just responding by, okay this thing is having a problem. Let’s shoot it and spin up another one, can have disastrous implications, because it can create a chain where things start dying all over the place. That's not a place any of us want to be in.

We believe in RSE is – the better approach would be to ask, okay what change, what would the set of events when our golden signal started changing? In order to address that, we have a very strict way of how do we handle an incident. We have an incident commander, we have a technical lead, we have a communication lead, where it's very clear what decision are being made and what information is presented to you.

Then combining that with a set of continuous integration and delivery tools, like cloud VMs that we announced last week at Next, allows you the ability to say, at that point when things started going wrong, this version of the software for this component was in the process of being deployed, and they're using the transaction logs to say, bring back the previous version of the software, which is a much more targeted solution than just say, “Let's kill that thing.”

[0:15:54.6] JM: You've mentioned this golden signal idea a couple times. Can you talk about that in more detail? What is a golden signal?

[0:16:00.4] MC: Again, golden signals is just our experience of what we found to be most insightful about our services. Customers mileage may vary, but we think that there's a lot of common sense in them. I'll go through a bunch of them. The first one is error weight. An error weight relates to a concept we hold dear in Google called error budgets, which really says, okay, a certain service, let's take Gmail for example. Let’s say that you a Gmail user and one key user generally would be well, I want to send e-mail.

We clear that out of, let's say every million requests to the send e-mail API, only a thousand requests can fail. We say that's acceptable, because well, you know retry in the Gmail client and most of our customers will never notice. There's no such thing as a perfect availability and you should stay away from there, but you need to define a certain acceptable error rate and define an error budget that the team is allowed to experience.

One thing that we track across all services is what's the acceptable rate, what's the common error rate and start to engage our SREs when we're seeing that the error rate is trending towards violating the service level objective.

Another example is saturation. We think a lot about the cloud as this unlimited vast resource. But even in the cloud, there's limit to resources, there's things like noisy neighbors that might impact you, so situation really says, "Do I have enough resources allocated to that service, to be able to service the SLOs that I'm looking for?" If I'm saying that the stress on the resources allocated is exceeding a certain level that I define as reasonable, then I should stop and reevaluate and reallocate resource.

Maybe lastly, really quickly, I will touch on latency, which I personally feel is one of the most important signals, because latency really says, "How long does it take to achieve the functionality that I wanted?" We all know that when you open a browser and go somewhere, you will only wait a few seconds before you say, "Oh, this is a really not interesting website. Let's go somewhere else."

Being able to treasure and being able to track the latency we provide to our customers and immediately alert when we see that the latency is trending towards an SLO violation is one of our key goals and signals.

[SPONSOR MESSAGE]

[0:18:36.8] JM: Cloud computing can get expensive. If you're spending too much money on your cloud infrastructure, check out DoIT International. DoIT International helps startups optimize the cost of their workloads across Google Cloud and AWS, so that the startups can spend more time building their new software and less time reducing their cost.

DoIT International helps clients optimize their costs, and if your cloud bill is over $10,000 per month, you can get a free cost optimization assessment by going to doit-intl.com/sedaily. That's D-O-I-T-I-N-T-L.com/sedaily. This assessment will show you how you can save money on your cloud, and DoIT International is offering it to our listeners for free. They normally charge $5,000 for this assessment, but DoIT International is offering it free to listeners of the show with more than $10,000 in monthly spend.

If you don't know whether or not you're spending $10,000 if your company is that big, there's a good chance you're spending $10,000, so maybe go ask somebody else in the finance department.

DoIT International is a company that's made up of experts in cloud engineering and optimization. They can help you run your infrastructure more efficiently by helping you use commitments, spot instances, right sizing and unique purchasing techniques. This to me sounds extremely domain-specific, so it makes sense to me from that perspective to hire a team of people who can help you figure out how to implement these techniques. DoIT International can help you write more efficient code, they can help you build more efficient infrastructure. They also have their own custom software that they've written, which is a complete cost optimization platform for Google Cloud, and that's available at reoptimize.io is a free service, if you want to check out what DoIT International is capable of building.

DoIT International are experts in cloud cost optimization. If you're spending more than $10,000, you can get a free assessment by going to doit-intl.com/sedaily and see how much money you can save on your cloud deployment.

[INTERVIEW CONTINUED]

[0:21:00.3] JM: Speaking about latency, there are some types of latency that can be really hard to diagnose. For example, tail latency, where if you take a large sample of response times and you find that at the tails, you have some very severe latency cases and it can be really hard to diagnose why this is happening, and it may be due to something like a noisy neighbor, or there's some hardware specific issue that only happens every 30 minutes.

Can you talk about some of these issues that are particularly hard to identify in a cloud environment, and also how – if you wanted to, because you're working on a product like the Stackdriver product that we'll get into momentarily, but this product is to give people observability and help them track down some of these errors, how do you build a tool that provides somewhat standardized solutions to something like tail latency that's pretty hard to ascribe general causation to?

[0:22:04.7] MC: That's a tough one. Our approach within Google for years has been standardized the way you instrument your software. You will be able to consistently identify issues. Internally in Google, we use something we call census. We have open source that in over the last few years, you've seen a lot of software is coming out of Google as open source that is the open version of what we do internally in our production.

Open census is one key example, which is a consistent set of libraries and specification that allows you to very quickly by including a library in your application when you compile it to automatically omit trace information and metric information that lets you diagnose such problem in real time.

What happens in our production environment when one of those things happen, since census omits traces as requests happen, and since all of our production environment is instrumented with a profiler and debugger that let us profile and debug in production, you can do such a thing as say profile all the instances of my service. When one of them hits a latency that is above a certain thing, automatically I activate the profiler and give me a profiler analysis of which functions are causing this problem.

If that information is not enough, we can create a breakpoint in our debugger and say, when something hits that function, please create a memory snapshot of what was going on and let me debug that in my free time without impacting the user, and figure out what's going wrong there when that wait state is happening.

We have been very fortunate in Google Cloud, because of our global distributed resource management. The issue of noisy neighbors almost does not exist in GCP. In Borg which is our production environment, we have not invested as much as we invested in our cloud customers and we have come across many scenarios where noisy neighbors have resulted in cases where we had to identify who is the noisy neighbor, and then enforce most strict resource allocations.

[0:24:13.0] JM: You talked a little bit there about the externalization of services and the open sourcing – open sourcing and externalization of services. With Borg, the infrastructure management tool that has been active in Google for many years, the system, the Borg system

was tightly coupled to how Google does certain things. With Borg, Kubernetes provided an open source version of Borg, but it was basically a rewrite and another rewrite of Borg.

Was this also the case with the – because Stackdriver is another one of these externalizations of Google infrastructure. Was this another case, where the internal tools were a little bit too tightly coupled with how Google does things, the internal assumptions of Google and there was a lot of rewriting that was necessary? Or were you able to reuse a lot of the internal systems in the internal code from Google's monitoring?

[0:25:11.1] MC: I would say that it was an interesting conversion of bringing stuff from the outside. For example, Stackdriver, the monitoring experience started off as an AWS-based software as a service solution that Google acquired four years ago. We have consistently made sure to keep Stackdriver hybrid.

Stackdriver today monitors a hybrid environment, multi-cloud environment and continuously expanding. In that fashion, the customer facing UI experience have come from the outside in. On the other hand, the backend which is our time series database that stores all the metrics, as well as a logging platform is the exact same backend system that is used for Google production services.

We run at Google what I think is probably the largest time series database in the world, with tens and hundreds of petabytes of time series that are being consumed at any given time. Our logging platform supports – we have customers that sends in petabytes per day of data, and is able to deal with rapid scaling of thousands of percentage. On the other hand, when it comes to integrating with our customers, we made sure to continuously use open source projects, so Google adopted FluentD, which is our logging agent. It's an open source Apache project that Google helps maintain.

We adopted collected as a monitoring agent, again an open source Apache project. We are partnering with many most of the leading vendor in the monitoring space around open census, which is the externalization of our internal instrumentation as an open source project.

[0:26:54.8] JM: Okay, so Stackdriver is a combination of different observability tools into a single tool. The idea is if people are spinning up infrastructure, they're going to want some observability tools out of the box, much like you get at Google. I used to work at Amazon and there was something similar within Amazon, where there's just a given that you're going to want certain things like distributed tracing, or logging at some level when you spin up a new service.

I think what's cool about the idea of Stackdriver is if I'm just a startup, then I don't – historically, I don't have the backlog of tooling and infrastructure to be able to roll my own collection of distributed tracing and monitoring and logging tools, but it's nice if I can just get that from the cloud provider out of the box. Can you talk a little bit about the design goals of the Stackdriver project? What were you trying to achieve when this project was getting off the ground?

[0:28:02.8] MC: I think we will focus on two things; one is how do we provide integrated observability throughout the software development lifecycle? Operational visibility will not be an afterthought, that once you deploy to production where I need to monitor that, but rather it be something that you can bake into your software, and the signals that are needed for you to maintain your software will be automatically omitted and managed.

The other piece was how do we do integrated root cause analysis? Just tracing is not enough, because you don't have enough visibility. Logging is way too much data. How do I figure out what's important, then what's not? Monitoring provides you the ability to do trending, but okay, now that I figured something is wrong, how do I go in?

Really bringing all those signals together around the construct of services and application and being able to say, the collection of resources, this BigQuery table, or this Big table, or this spanner together with those set of Kubernetes containers on GKE and this GCS bucket, altogether make up my application, and be able to visualize that and attach a service level agreement to that. Then when something go wrong, right? They can say, “Show me what happened when the thing is beginning to go wrong,” and have an integrated view across tracing, logging and monitoring is what our developers at Google and to expect, and what we're trying to provide to our customers.

[0:29:28.7] JM: Let's talk about some specific engineering problems you have to solve when you're building an observability tool like this. Logging for example – logs – we've done some shows about logging, and logs can get to be a big data problem, because logs get generated on a host and you've got to figure out how often are you going to shuttle those logs from the host to some centralized logging solution. You've got to figure out are you going to index these logs, or how you're going to index them to make them searchable, how available do you want to make them, how quickly do you want to make them available, how long do you want to keep them around before you garbage collect yesterday's logs, or the last week of logs? Describe the architecture for the logging side of Stackdriver.

[0:30:18.9] MC: I would start by saying that I have the privilege of talking to many of our large customers enterprises throughout the world every day, and what amazes me is that no two customers are the same. The needs, the way they arrive, where they arrive, the way they think about their business is completely different. The nice thing about Stackdriver is whether you are cutting-edge SRE organization, like Snapchat, or HSBC, or whether you are more traditional business, you still can get to the same level of log disability with Stackdriver.

From an architectural standpoint, things start with our local action agent, which is FluentD. It's an open source project with a long set of extension that lets you manipulate the logs as they come into the agent and being able to deduct logs that are irrelevant, be able to do compression, be able to do things like annotation, and we work with the open source community to enable you to do new and interesting thing in the agent every day.

Then the FluentD agent speaks to our back-end API, which are distributed throughout the world to make sure that one, we can automatically scale as our customer businesses scale and second, we can handle any availability scenarios that might happen where an availability to zone fails, or network connectivity issues happen. Then as the logs get into the backend system, at the backend of Google, we have hundreds of thousands of hosts that deal with the incoming logs and continuously write them to both a backup system, as well as in-memory to allow immediate querying of the logs as they come in.

One of the key features that our customers love in Stackdriver is the ability in our log, we have to say, just stream and show me the flow of the logs as they come in. When you think about is

that we monitor hundreds of millions of hosts, that's a really powerful thing to do. It's really the equivalent of tail minus F, but on millions of hosts.

Then as the logs get older and you want to do long-term analysis, or long-term storage for compliance reasons, Stackdriver provide the ability to export those logs and keep them for very, very cheap price on things like GCS buckets will be able to send them to BigQuery and combine them with other a business data, in order to do analytics. Or many of our key partners like Splunk, or Sumo Logic integrates with Stackdriver to prop sub and in real time send the logs that are collected to their specific solution to provide unique experiences through their services.

[0:33:00.5] JM: FluentD story is interesting. FluentD is this agent that runs on the customer's infrastructure, that runs on a host and gets configured to do the – some of the log management, or some of the filtering, or some of the flagging. We had one show about FluentD a while ago, that was pretty interesting and the guest talked about some of the chains of monitoring, or the chains of logging filters and layers of infrastructure that people sometimes set up, where they will collect some logs and send them to a Kafka queue, and then have more processing take place on that Kafka queue, and then shuttle them somewhere else.

I guess, depending on the situation, some people want pretty complicated set ups where there are different parts of logs getting teed off and into different places, but the general use case is probably figure out some general use case for how people want their logs sent around. I guess, you have some defaults that make it at least easy for people to get started with shuttling their logs around.

[0:34:11.0] MC: Yeah, that's a great observation. I'll throw some interesting things that we hear from our customer. One is yes, you mentioned the amazing flexibility of FluentD. The second thing is in Stackdriver, as the logs come in, we provide something we call exclusion rules. Let's you say, out of the data that is coming in, I can apply filters using [inaudible 0:34:31.1] and say, this data I want to see in Stackdriver. This data, I want to throw away, because I don't care about it, even though it was sent in. It provides both cost control and the ability to reduce the mental load on what you need to analyze. Finally, this data I just want to keep for historical purposes.

The second things that we hear a lot from customers is two features that they love about Stackdriver is one is log-based metrics, which is the ability that as the logs come in, provide change what you see in the log and create a metric out of that. For example, tell me what the latencies that I'm seeing for disk reads, from all my VMs across the entire cloud and shot that into a graph. Really turn unstructured data into structured data.

The second one is what we call error reporting. Think about, let's say that I'm a big e-commerce vendor and I provide the ability for customers to do shopping on my side. I want to be able to track as I deploy new versions of software, is the number of errors, like 300, 400, 500 that my customer is seeing, are they changing over time and are they changing as I deploy new software?

Error reporting automatically extracts error information from logs and give you the ability to do trending and analysis and alerts on the errors within the logs, without actually looking through each log entry.

[0:35:57.0] JM: This product can require a lot of integration. You have to keep up with integrations from other cloud providers, you have to integrate with new open source projects, you have to be able to integrate with different host types, whether the user is on some VM, or they're on a container within a VM, or a container within bare-metal infrastructure. How do you structure the team, the Stackdriver team to be able to write the required integrations and to be able to keep up with all of the integrations and all of the different levels of this kind of product?

[0:36:38.0] MC: I tend to think of our integration team as a group of superheroes. There's X-Men, Superman and really they do the possible every day. It's a really, really crazy software world out there. On a serious note, there's two parts for that; one is think of Google Cloud itself. We have over a hundred services today and new services getting launched every day. On day one, customers want those services to be observable; meaning, to get logs and metrics for them.

It's both keeping up with all the amazing innovation within Google Cloud, as well as keeping up with commercial applications, like databases, other crowd windows, on-prem. I can't say that we have a magic formula for that. We're still working on time machines, so we can do 60-hour

development a day, instead of 24. Until somebody at Google comes up with that, really a core power is the open source ecosystem. Example is our integration with Prometheus, that if you want Kubernetes and you use Prometheus, everything you instrumented with Prometheus automatically go into Stackdriver. It gives us a force multiplier that everything that was ever instrumented with Prometheus, suddenly amazingly lights up in Stackdriver and the community empowers us.

Another great example there is you mentioned, FluentD collect these and they have a lot of plug-in. Open census, open census today is instrumented in many of the production-ready databases. For example MongoDB, for example MySQL automatically include open census instrumentation. Really, the only answer I can give here is as Google alone, we can do so much, with the community, we can do so much more.

[SPONSOR MESSAGE]

[0:38:31.2] 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.

[INTERVIEW CONTINUED]

[0:40:07.0] JM: Well certainly, the fact that the agent uses FluentD allows you to leverage some of – standing on the shoulders of a giant that's already been adopted by the community, and you already have some of the large cardinality of integrations taken care of probably due to the FluentD leverage there.

Getting Stackdriver to general availability, required you to get it to a place where you felt quite comfortable with all of these different sets of customers and the different types of infrastructure that Stackdriver could potentially be used with, which includes not just Google Cloud, but also other cloud providers and also, I believe on-prem infrastructure as well, if you want to monitor on-prem infrastructure. What was the process of getting it to a place where you felt comfortable with putting it into general availability?

[0:41:05.7] MC: It's been a journey. We're still on that journey. I'll first admit that as a Googler, and everyday customers come to me and say, "Mark, you still haven't done that, and we expect you to do that." The great thing about observability is it's a never-ending journey. For us, the bar to get to production was for us to say, we as Google can trust Stackdriver to do monitoring for ourselves. Can it scale to Google scalability?

When we acquired Stackdriver, it was a very innovative ahead of the market solution, but it was a startup. There's a lot of really, really cool startups there. Every day, I see a new company come with new innovation and it's amazing. The challenges is when you're one of the world's largest, if not the largest cloud provider in the world, there's a certain level of scale, of availability, of security, of privacy, of establishing trust with our customers that was important for us.

We kept Stackdriver in a better state for a very, very long time. We can say when we [inaudible 0:42:14.6] that we are able to meet the same availability, security, privacy and scalability requirement that any of our billion user services meets every day. Again, it's a journey. There's still much for us to do, but I'm very – we get feedback from customers every day, that for them to run something like Stackdriver them self would have required them to hire a 100 people, and that we remove the overhead from them of dealing with stuff that is not code to the business, and that makes me excited every day.

[0:42:50.7] JM: One of the reasons that people listen to this show is to find out about different management structures. You are a Group Product Manager at Google. What exactly does that role encompass and what's your interaction with the rest of the Stackdriver team?

[0:43:09.1] MC: I think Google is a very unique company. I've been in this industry for almost 30 years. I worked for some of the world largest companies and had three companies that I founded myself. I think Google is the most unique company I've ever worked for. In a way, it's a set of really, really smart people that want to solve really cool problems that can have a positive impact on the world.

Google does not start from, let's build something because we can make a lot of money out of it. It starts with, what can we do that will delight our user? What can we do that move technology forward? We know that if we will make it right, people will be happy with it. I think within my team, within the Stackdriver team, we have I think something like 15, or 16 PMs. Each of them runs their own business. They're responsible for their own customers and the partner who is their engineering team to say, “How can I make a difference tomorrow?”

For me as their manager, it's really a learning experience every day. Every person that joined the team, teach me something. Many of the PM on the Stackdriver team went on and did really cool other things in Google, in databases and Google assistance. That's the unique thing about Google is that if you’re passionate about making the world better, you can make a difference anywhere.

[0:44:32.5] JM: You said you've started three companies before working at Google?

[0:44:37.1] MC: Yes.

[0:44:38.0] JM: Could you contrast? I mean, this is off topic, but can you contrast your experience starting companies, versus working at a big company? I mean, the thing is that the big companies that you can work for today, it's a unique time, because some really cool stuff going on in the big companies, even relative to the coolest stuff going on at startups. I mean, I guess this has always been the case to some degree, but there's a really cool stuff going on right now; I mean, in both places. Yeah. I mean, maybe you could just contrast your experience doing three different startups with working at Google. Like your level of satisfaction and what you get and what you don't get.

[0:45:19.5] MC: I would say, I worked in a few big companies before. I won't name names, but corporate has been a very different experience for me than a startup. Because in starting my second startup, my business card said janitor. Because honestly, as a startup founder, you do everything, including washing the dishes.

In a big company, it's different. People expect you to do one thing and do that well and stay out of everybody else's way and partner with them. Because it's about moving the air force carrier, it's not about the speedboat. I think what's unique with Google, it's a vast navy of speedboats; each of them racing towards doing something super exciting. Since all the people in Google are really passionate about what they do, and what amazes me every day is that I sit in the room and I meet people that are so overqualified for what they do, and have such an amazing life story; people like Melody Meckfessel, who's our head of DevOps, or people like Brian Stevens, our CTO, or Eyal Mano who created the cloud services platform and Goldberg from Kubernetes.

Those are amazing people with such a depth of knowledge. Even in places where we disagree, we come and sit in a room and say, "Okay, tell me where you coming from." I have never been in a meeting in Google where the discussion was about ownership, or territory, or politics. The discussion is always about, how can we get to a shared understandings that make both of us successful? I've never worked in a 80,000 plus company, where people are so invested in each other.

[0:46:57.8] JM: Cool. One thing that I think came out recently from the Stackdriver project was this transparent SLIs thing, which is I think SLI is that's – is that service level indicator?

[0:47:12.2] MC: Correct.

[0:47:13.1] JM: Yeah. Service level indicator, this was one of the things that was in the Google SRE book, the different service level components. The transparent service level indicator, I think this feature is interesting, because it's basically if you instrument something in Stackdriver, so Stackdriver is this cross-cloud, cross-infrastructure platform monitoring system. If you instrument some piece of your infrastructure with Stackdriver, you can get these transparent SLIs, which can actually help you identify, for example if a cloud product is meeting its – meeting its guarantees, if it actually is providing you the number of nines that it claims to. Could you describe the transparent SLI feature? Explain what a transparent SLI is.

[0:48:07.8] MC: I'm really excited to talk about that, because for years, ever since the first SRE book came out and now that we have the second one that talks about practices, customers have been coming to me and say, "We love the book. How can we do what you do, and how can you expose to us what you have internally? Like, "Okay, you sold us on the dream. Now how do we realize it?"

Service level indicator, this transparent SLI is really one of our most important milestones towards that. In every cloud vendors that you go to, at least the big one, they provide you things like, well if you use the database as a service, they can tell how many queries you're running, and maybe be able to say, "Well, how many errors did you see?" Each of the services provide some monitoring metric that is unique for that service.

What you do not have is the ability to say, since this is a multi-tenant public cloud vendor, how is the service behaving? Is the service provider really providing the service levels that I'm expecting? When they have a problem, how do I know if it's Google's fault, or is it my fault as a customer? Is it my applications in this book? With the service level indicators, what we do is we provide a standard set of metrics along the line of the golden necessary signals we talked about earlier, things like error rate, things like latency. It's the exact same indicator across all of Google Cloud services.

Even beyond that, we don't just say, “Well, here's a service dashboard and our service right now meets its SLO.” We say, “Here's the error rate in the latency, for example, spanner that we provide specifically for you customer and specifically within this specific project in Google Cloud.” It's not an average of what all our customers is saying, it's the service levels that I'm providing to you right now.

I was really excited last week to be on stage with Ben Treynor, the guys that invented SRE ahead of Google 24/7 and [inaudible 0:50:09.4] from Snapchat, where they demonstrated how they use transplant SLI to very quickly add those issues and figure out is it Google, or is it Snap and make sure that their snap steps remain snappy during the World Cup, which was by far, one of the most popular events in the world.

We think that as SREs, we want to automate everything, because when you get humans into observability, things get slower, right? You need to pick up the phone and open a ticket. When you have an indicator that is programmatic, when you can look at the dashboard and say, well things are going wrong. The indicators from Google are still green. Well, things are going wrong and Google indicators are going bad, you immediately know is it Google? You can immediately, when you pick up the phone, or e-mail Google, you can say, “Here's the dashboard and your indicators are saying that something is wrong on your side.” It removes the entire step of troubleshooting and immediately pinpoint the problem, which improves availability to all of our customers.

[0:51:10.8] JM: For an example of this transparent SLI thing, so if I was – let's say I'm Snapchat and I'm built on Google Cloud and maybe Google Cloud spanner, the spanner database is on the critical path for some aspect of my product, maybe I have on my dashboard an SLI for the – how long it's taking for to get an act from the database when I when I write to Cloud spanner, how long does it take to get an act back from the database that my write preceded, and my write was accepted to just spanner. Then I could at that chart and maybe I'm going to see, “Oh, I've got some high latency right now.” I know just by looking at that SLI that if my service is failing, it's due to cloud spanner or not due to cloud spanner, just by looking at the dashboard.

[0:52:06.5] MC: Exactly. We think that that's super powerful, because it allows a customer to combine their server level indicator with ours and immediately be able to get to the root cause of the issue.

[0:52:19.3] JM: I was at Google Cloud Next last week myself. It was my first time there, so I was amazed by two things. One thing, whenever I go to one of these software conferences, I'm always just amazed by how big the software industry is, because you just wind up in conversations with people where you're like, “Oh, you're using cloud? Well I guess, why not? Of course you are.” Somebody from a candy company, right? Just long-tail customers.

Also I talked to some of the people in the vendor booths and you just – you see vendor booths that are building really big businesses that sound like they're in very niche areas, but because there are so many customers in the cloud, even what sounds like a niche can actually be a pretty good business. Maybe pretty optimistic about the cloud infrastructure business as a whole that's just among some reflections I had from the conference. Did you have any reflections from being at Google Cloud Next over the last three days last week?

[0:53:21.5] MC: I think my main takeaway was wow, to summarize it in three characters or less.

[0:53:27.2] JM: It’s really big.

[0:53:28.5] MC: Yes. Last year, we've seen thousands of people. This year, we had tens of thousands of people. The stories as you mentioned were amazing, like ranging from people like HSBC and PayPal came on stage, and the fact that I had more customers on stage than our executive, the facts that we had huge partners like Accenture and Rackspace coming on stage and talking about how do they help customer do transformation.

Google is known worldwide. Everybody knows who Google is, but to hear the world's largest enterprises and the world's largest partners saying we bet our business on Google, to be honest, it was the proudest time for me as a Googler, because it means that all the hard work and sweat that we're doing makes a difference.

The other thing is I looked at all those people with the amazing stories. As much as the world has been speeding by over the last few years and things have been crazily advancing, I think we're just in the beginning of what we can achieve in the next decade or so.

[0:54:33.9] JM: Mark Carter thanks for coming on Software Engineering Daily. It's been really great talking to you.

[0:54:37.1] MC: It's been a pleasure, Jeff, always.

[END OF INTERVIEW]

[0:54:42.3] 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]