EPISODE 557

[INTRODUCTION]

[0:00:00.3] JM: Software engineers have interacted with operations teams for decades. In the 1990s, most operations teams worked with physical infrastructure. They made sure that servers were provisioned correctly and installed with the proper software. When software engineers shipped bad code that took down an entire company, the operations teams had to help recover the software systems and this often meant dealing with the physical servers. During the 1990s and early 2000's, these operations engineers were often called sysadmins, database admins if they worked on databases, or infrastructure engineers. The nomenclature has changed over time.

Over the last decade, virtualization has led to many more logical servers across a company's infrastructure. Cloud computing has made that infrastructure remote and programmable. The progression of infrastructure led to a change in how operations engineers do their work. Since infrastructure can be interacted with through code, operations engineers are now writing a lot more code.

The dev ops movement can be seen through this lens. Operations teams were now writing software, and this meant that software engineers could now work on operations. So both software engineers and operators could create deployment pipelines, monitor application health and improve the system scalability all through written code. That's the infrastructure as code movement having its effect.

Sight reliability engineering, or SRE, is a newer point along the evolutionary timeline of operations. Web applications can be unstable sometimes and SRE is focused on making a site work more reliably. This is especially important for a company that makes business applications which other companies rely on.

Mike Hiraga Arocha is the head of sight reliability engineering at Atlassian. Atlassian makes several products that many businesses rely on such as JIRA, Confluence, HipChat and Bitbucket. Since the infrastructure is at a massive scale, Mike has a broad set of experiences

from his work managing SRE at Atlassian and doing systems administration and infrastructure engineering at many different roles prior to Atlassian.

One particularly interesting topic is Atlassian's migration to the cloud. Atlassian was started in 2002 before the cloud was widely used, in fact, before AWS had even been invented. Atlassian has made a more recent push to move their applications into the cloud, which has all kinds of ramifications for the sight reliability engineering team.

Full disclosure; Atlassian is a sponsor of Software Engineering Daily, and they are hiring. So if you're looking for a job, check out Atlassian jobs or send me an email directly. I'd be happy to introduce you to the team in Atlassian.

[SPONSOR MESSAGE]

[00:03:20] JM: If you are on call and you get paged at 2 a.m., are you sure you have all the data you need at your fingertips? Are you worried that you're going to be surprised by things that you missed, errors or even security vulnerabilities because you don't have the right visibility into your application? You shouldn't be worried. You have worked hard to build an amazing modern application for your customers. You've been worrying over the details and dotting every I and crossing every T. You deserve an analytics tool that was built to those same standards, an analytics tool that will be there for you when you needed the most.

Sumo Logic is a cloud native machine data analytics service that helps you run and secure your modern application. If you are feeling the pain of managing your own log, event and performance metrics data, check out sumologic.com/sedaily.

Even if you have your tools already, it's worth checking out Sumo Logic and seeing if you can leverage your data even more effectively with real-time dashboards and monitoring and improved observability. To improve the uptime of your application and keep your day-to-day run time more secure, check out sumologic.com/sedaily for a free 30-day trial of Sumo Logic.

Find out how Sumo Logic can improve your productivity and your application observability whenever you run your applications. That's sumologic.com/sedaily.

Thank you to Sumo Logic for being a sponsor of software Engineering Daily.

[INTERVIEW]

[0:05:05.3] JM: Mike Hiraga is the head of SRE at Atlassian. Mike, welcome to Software Engineering Daily.

[0:05:10.3] MH: Thanks, Jeff. Great to be here.

[0:05:11.4] JM: SRE means sight reliability engineering. That's a term we've done a few shows on in the past and it's something that evolved out of systems administration in dev ops that lineage the operation side of things. You've been working as an infrastructure engineer since that role was called sysadmin, which was several years ago. How would you describe the evolution of the industry role from sysadmin to the more modern term sight reliability engineering?

[0:05:43.5] MH: Right. Well, this goes back quite a ways. My first experience dealing with a sysadmin was back in my university days when we're just doing stuff in the computer labs, we had sysadmins in there. I recall when I first met these folks thinking how awesome they were, because just the amount of breadth knowledge that they had dealing with hardware, dealing with networking, they could write code, they could do all kinds of things. Just looking at some of the things that they dealt with on a day-to-day basis was quite intimidating to me at that time.

Walking into one of their labs and seeing one of these giant slide machines which was the size of a refrigerator, made lots of noise, consume lots of power, lots of air conditioning. It was just a very overwhelming experience for me. I always had a great picture of sysadmins in my mind.

As I went off and became an infrastructure engineer, which was basically kind of the next — An interpretation of sysadmins, I started appreciating the job quite a lot more. At that time, the job was very operational in the sense that we dealt a lot with hardware operating systems and how we'd make those operating systems accessible to people that needed to put things on there and

run businesses on top of them. Sometimes we would deal with services, but that wasn't necessarily the focus.

Now, as things evolved in the industry, internet services started to proliferate, mobile devices — Everything started just expanding at a phenomenal rate, which started started putting a lot more pressures on the system's ability to scale. As a result of that, SRE type role started to evolve, and most notably with the Google SRE, which obviously SRE's birthplace was over at Google. To this day, SRE's are still loosely defined as engineers that employ development and software engineering to solve operational problems. A lot of this — Yeah, I see that as being born directly out of having this next generation problems of scale and folks on services.

[0:07:31.5] JM: Yeah. In terms of services, I think you mean the tooling. So today we have cloud services, and that's a dramatic shift from the technology stack in an average tech company 15 years ago when you started working in technology. What are the changes in the operations practices and the cultural changes that have contributed to this more recent evolution in SRE?

[0:08:00.2] MH: As you mentioned, cloud services and cloud platforms is definitely one of the big drivers. A lot of these was also going back to my comments about scale. The industry began quickly moving towards having very large systems running in on-premises which might not be your own or things that you don't necessarily need to have a direct control over.

Because of this, the need was there for features to iterate very, very quickly in the customer facing services space because of your competition and other driving factors like that. The pace of development needed to basically accelerate. Using kind of the old operations, sysadmin way of doing things, a lot of that didn't exist. A lot of these practices of scale and quickly being able to facilitate development velocity was important.

Because of that, one of the cultural shifts was how do we basically take all these infrastructure that we might run our premises or might be in the cloud and basically shorten the amount of time it takes in order to get those resources available for features and development purposes. A lot of that was basically how do we keep up with the pace of development in order to facilitate? Like provide your business with very quick turnaround time for market.

[0:09:08.9] JM: Yeah. Now, in SRE, there's more an emphasis on writing code compared to the sysadmin roles and perhaps even the other operations roles, the infrastructure engineering roles in the past. Is that emphasis on writing code, is that more because of this shift, shift toward the cloud and the infrastructure as code, because you can codify your infrastructure?

[0:09:35.7] MH: Yes, that is correct. Basically, when you're dealing with many, many, many different systems, which could distributed across many locations around the world, having automation and tooling and writing software to manage all that is the only way a team can basically survive. SREs don't typically scale linearly with an organization. SREs rely ultimately as being — As what's been mentioned before, force multipliers, by relying on tooling and software in order to expand their capabilities and make themselves be able to do more and be responsible for more.

[0:10:06.1] JM: What's the interaction between software engineers and SREs?

[0:10:10.8] MH: SREs and software engineers should be very, very tight especially in the environments where I worked at, software engineers and SREs effectively function as team members. Just, basically, the SREs obviously have a different focus on stack, but for all intents and purposes, they regard themselves as being peers and team members. So the interaction is very tight.

[0:10:33.5] JM: You're the head of SRE at Atlassian. What are your responsibilities as the head of SRE?

[0:10:40.2] MH: Right. I see oversee the overall direction and priority or the SRE organization. I take what the company priorities are, do a little bit of team [inaudible 0:10:49.9], do a lot of meetings and discussions and I facilitate the contribution of the SRE organization towards those goals. I know that's probably kind of very vague, but basically I look after what the company is doing, what the needs are. I look at what our capabilities are in SRE and I work towards providing our support towards meeting those goals and a lot of that is basically in delivery and service reliability.

[0:11:14.4] JM: Are the SRE practices — Atlassian has a bunch of different products. Are the SRE practices standardized across those different teams and across those different products?

[0:11:24.1] MH: Yes, they are. What we have so far, we have things like operational readiness practices, incident management and things of that nature. We are also keen on expanding that portfolio into other areas that maximize, being able to scale. In addition to tooling and automation, our practices also enables Atlassian engineering teams themselves to be a little bit of SREs in their own right. In that manner, it makes ourselves as a team much more scalable and increases our reach by that much more.

[0:11:54.3] JM: You have standardized practices, do you also have standardized tooling, like standardized logging and monitoring and the pipelines for those logging and monitoring systems?

[0:12:04.0] MH: Yes we do. We have a team that's a part of SRE, which basically develops and maintains all of our monitoring and login systems.

[0:12:11.4] JM: is there like a platform engineering team?

[0:12:13.3] MH: This is not a platform engineering team. This is actually a part of SRE, although all their tooling does integrate with the existing Atlassian platform team.

[0:12:21.3] JM: So if I'm somebody in a team at Atlassian and I want to spin up logging and monitoring for my service, what kind of interface do you want to give that team or that engineer?

[0:12:34.7] MH: Typically, what we want them to do is use the Atlassian platform. We do have a platform as a service organization, which SRE partners closely with. When you use their tooling and when you provision resources tools, a lot of the instrumentation happens through that process. So that is the fastest and easiest path, least resistant path to it. If you're outside the platform, and some services still are, then yes, there is more elaborate set of instructions and guidelines in order to use these resources, but our best recommendation is use our platform and a lot of your pain goes away.

[0:13:09.6] JM: What do you think are the pros and cons to having that central team that provides the tooling that gets standardized across the different team, that versus allowing the teams themselves to go rogue and spin up their own logging and monitoring?

[0:13:27.7] MH: Right. Going rogue, and when you have lots of different teams basically duplicating effort, that adds up to quite a bit. For example, if we take the thousand or so developers here — Let's just say we have 50 teams, and if each one of those teams spends one person's worth of time working on that, then immediately you have 50 people, which are now working towards exactly the same thing.

Obviously, the duplication adds up very, very quickly. Also, a lot of our products, it's important that we do a lot of analytics and studies across our different products. If we all have monitoring and logging and all these different telemetry systems independently managed, it becomes that much more difficult to merge the data and do a lot of useful things with it. Consolidating into a central team solves all those problems, but also the problem with that is that it is a central team, which means that — Concerns with the resourcing constraints of a central team, and a central team might not be able to address every last one of a team's requirements. We have to focus on what delivers the most value to the organization.

[0:14:36.5] JM: Probably most of the people listening work at a company where there is not a centralized team that's well-organized and giving them a logging and monitoring platform. Can you talk at all about, I guess, suggestions or — It's 2018. What are the contemporary strategies that you recommend around logging and monitoring?

[0:14:57.8] MH: I would very generically say, if you are a lot of splinter team is doing the same thing, then I would say talk to people and try to write a business case around justifying a centralized team forming around it. I think that if you pay a little bit of the cost upfront, it's very, very easy to show the long term value and ROI on such an endeavor.

[0:15:18.2] JM: SRE teams, they set service level agreements for their teams to communicate the reliability of those services. We've talked about that on previous episodes. What do you do if a team fails to meet its SLA? They've got failures all the time. Do you punish them or do you adjust the SLA number? What's the response to that situation?

[0:15:40.6] MH: The SLA should always be aligned with what business objectives are. So I don't think changing them is necessarily the best thing to do unless they were miscalculated and set too aggressively to begin with. For example, if you — Without any — Don't put too much slot on this and say, "Hey, I want to set my service level objectives for this endpoint which only gets 10 users per day to be 5-9's." The effort in order to do that is going to be astronomical and your ability to hit that might actually be very implausible.

There would be cases where, yes, you do look at your SLAs and understand, "Well, was this correctly set?" Assuming it was, and you should never regress on your SLAs. In fact, you should actually look at what's causing your business to mis-assess SLAs and work on basically doing a 5-Ys and taking apart the problem and improving it, right? If this means like you take corrective action — And I don't believe in necessarily saying initially through punishment, but more education and understanding and how the situation came up and just working towards attacking the natural root cause issue and not the proximal causes.

[SPONSOR MESSAGE]

[00:16:53] JM: Speaking of reliability, do you find yourself worrying about system downtime or missing an alert while you're on call? If so, VictorOps is the incident management tool that you need. VictorOps integrates with a large number of the monitoring, alerting and messaging tools that you already have in place to help your dev ops teams communicate better, diagnose incidents and resolve any problems that come up all in one place. On both your smartphone and your computer, you can view a highly contextual detailed alerts that will help your on-call engineers to understand and respond to incidents more quickly and effectively.

Head to cictorops.com/sedaily. That's victorops.com/sedaily, victorops.com/sedaily, and see how VictorOps can help you. Be victorious with VictorOps. That's victorops.com/sedaily. Thanks, VictorOps.

[INTERVIEW CONTINUED]

[0:18:05.9] JM: SRE is all about automation and there's always a manual process that an engineer could spend a day automating. You can come into work and you've got a big list of things that you could potentially automate. How do you choose? How do you prioritize which processes to automate?

[0:18:22.9] MH: The engineers and teams have to understand at all times exactly what they're working on, how it contributes to the goals of the organization itself. If you a very good mind of what your metrics are and what needles you are moving, then the productization becomes a little bit easier after that.

For example, if we set a goal for hitting some number of availability, then you need to look at everything that you're doing, all the tooling and automation and think to yourself, "Okay. Well, what gets me? What will move that needle the farthest?" and I will spend more of time on that and those things at a very basic level.

[0:18:52.7] JM: Yeah. Sure. Quite hard to generalize there. Incident response is obviously a core component of SRE. Can you tell me about an interesting incident that you've been a part of at Atlassian?

[0:19:04.1] MH: Interesting incident at Atlassian. I think that the most interesting ones are unfortunately the ones that are somewhat out of our hands. When we look at some failures of some of our providers, and these things happen, it's unfortunate, and watching how all the different teams in Atlassian coordinate and communicate with each other. Those have been extremely the most fascinating for me.

As a company, which builds software to help empower and unleash the potential on all teams, being a part of a major incident, which I see a lot of different teams involved in the incident, trying to fix things, trying to bring things up, trying to understand the signals, what's going on? Communicating and collaborating on a common goal towards restoring services, mitigating impact, communications. Then on top of that, using our own tools that we built at Atlassian in order to facilitate all that communication. Those are extremely fascinating to me, because there's a lot of improvements that we could claim from within our own process internally, how we functions as teams and how do we make ourselves more efficient.

[0:20:06.4] JM: When an incident occurs, how does it bubble through the organization? Atlassian is a big company. As the head of SRE, do you hear about most of the incidents or do you just hear about a minority?

[0:20:19.7] MH: I hear about all of them. We are very transparent when incidents happen, and we have major incident managers who come in and take over communications and basically in managing and distribution of information. I think that easy to say that not just me as head of SRE. I think everybody at the company hears about these incidents and knows all the details about what happens.

[0:20:41.0] JM: Now, the way that an organization implements SRE has a measure of subjectivity to it and it depends on the tenants of the company. When I was at Amazon, I worked at Amazon briefly, and the core tenants of the culture affected everything, including operations and on-call and all the other things that we would associate with SRE. How do the values of Atlassian affect SRE?

[0:21:08.8] MH: The values of Atlassian almost line up perfectly with what I believe the charter of an SRE or an organization should be. We have some very interesting values, very, I think, unique values. The ones I feel that are the most relevant to what we do is don't F the customer, and that is basically —

[0:21:27.9] JM: That's one of your value?

[0:21:29.2] MH: I'm not allowed to say that word, but don't F the customer. Really that's what SRE is really about here at Atlassian, is we want to make sure — Our team is here to help maintain trust in the reliability of our services that we provide for our customers. What we do, everything that we do in some way, shape or form lines up to maintain that trust so that we don't F our customers.

Also, if we think about other value of play as a team, well that's, again, what SRE does. We work very closely with our platform teams, with our product development teams, with finance, with all kinds of different teams here and we are not going to be successful as an organization

and as a team and as a company unless we play as a team. Even as SREs, we are an engineering team, but we are also a team that works though consulting, advice and basically influence, and a lot of that is built on trust and we can't maintain that trust unless we play as a team.

Also, another one is open company. Open company, no BS. I think this goes back to our transparency and how we deal with incidents and how we manage just our infrastructure and how we manage engineering here, especially within cyber reliability, and engineering transparency is basically a must. We need to be upfront about our mistakes. We need to learn from our mistakes. We can't blame each other about anything that happens, and we just need to have a very open mind to how we solve these problems and make sure we come up with the best and most innovative ways of doing things.

[0:22:50.7] JM: Tactically, you've done infrastructure and operations for pretty long time. Is there anything, any unique practices that have come up at Atlassian that differ from other SRE or operations teams you've had in the past?

[0:23:05.0] MH: I would say that the way with which we manage incidents I think is something that we're definitely proud of. We do actually go to a lot of conferences and we talk about our incident values and how we deal with problems. We have our own Atlassian incident value set of detect, respond, recover, learn and improve.

As simple as these might sound, they're actually very prolific in how we deal with things and how we learn from our incidents and how we manage communications. I think that if anybody has a chance to look into this more, yeah, they're very fascinating and that's something that I think that we're very proud of. I can't say it's necessarily unique to what we do to how we do things, but they definitely line up with our company values as well. I think going back to the company value's aspect of it, those company values actually just drive quite a lot of what we do as a team culturally, how we'd prioritize things and what are our pains on many areas.

[0:23:56.8] JM: It's probably useful for any company to codify their incident response values and plans. I think that piece of advice in and of itself would be useful. I want to get into talking about the cloud, because you're in the process of migrating to the cloud, and I think SRE in the

context of moving an organization into the cloud is a worthwhile discussion. So the products in Atlassian were started at different times in Atlassian's life cycle. The company is — What? Like a decade old or almost a decade old, something like that.

[0:24:32.8] MH: The company — It's definitely more than a decade old.

[0:24:34.0] JM: Okay. Yeah. Okay, let's say 11 or 12 or 15 years old. I'm not sure. But the products in the company were started at different times in Atlassian's lifecycle, and so the products, I assume, are backed by different kinds of infrastructure. To set the stage for the discussion of the cloud migration, how does the infrastructure vary across the different teams across the organization?

[0:24:58.3] MH: Right. A little bit of history. Obviously, Atlassian's core products were JIRA and Confluence, and then later we picked up other products such as [inaudible 0:25:07.5], Trello, and [inaudible]. Yeah, a lot of these did develop in different times by different engineering teams and often in different parts of the world. They're tech stacks and how they managed things and how they developed things varied quite a bit, but we have been working towards migrating everything into our cloud platform for a while now.

I would say right now at this point, most of our infrastructure is now out of like — Atlassian operated data centers. I think that we're, for the most part, in the cloud at this point. There is a lot of variance in the technologies that we use, but our platform, our Atlassian PaaS and the usage of it is growing quite a bit. The value of it had so far been very, very clear in terms of making the [inaudible 0:25:50.8] experience easier and actually even improving reliability on all the services that have adopted it.

[0:25:57.0] JM: This is purely an internal PaaS that you designed for developers at Atlassian?

[0:26:02.7] MH: Yes. The internal PaaS is basically — Yes, internally designed by the PaaS platform teams. They're basically aimed at building share components to take the pain away from common development tasks. Their goal is to make sure that the product development teams have more time to focus on their unique product challenges rather than have to think

about how do they release code, how do they provision resources, how do they look into logging and monitoring and things like that.

[0:26:28.6] JM: Interesting. Are you moving on to a cloud provider or you're moving on — You're just in the process of moving to your own internal PaaS?

[0:26:35.8] MH: We have our own internal PaaS, which is basically an abstraction into — Which is basically a layer between ourselves and AWS. It just basically makes it — Yeah, you could put a lot of things like processes and releases and things like that in between. Again, the goal of basically taking the pain away from these common tasks.

[0:26:53.7] JM: I see. Is the goal to eventually have access to the cloud for like seamless burst capacity or access to the AWS services, if you wanted to use Kinesis or something that. What's the motivation for having access to the cloud?

[0:27:13.8] MH: The services, they are provided. Keeps us from having to build our own things, right? A lot of those problems are solved. They provide a very rich set of services that we could use and enables us to improve our capabilities and improve what we do with our products.

In my experience, going back as far back as almost two decades ago, running physical infrastructure is difficult. It's difficult, and as you grow and scale, sometimes you get to a point where sometimes you need the capacity, sometimes you don't, and having that ability to have it available when you need it and scale back when you don't need it and being able to do so very, very quickly and not have to worry about any of the other common infrastructure operations tasks that go along with it is extremely beneficial. It really helps you focus on what your business needs to focus on.

[0:28:01.8] JM: Is there still some significant migrations to go internally?

[0:28:05.7] MH: Yes there is. I would say that we're not over the hump by any means yet, but although our progress has been extremely good so far.

[0:28:11.4] JM: Cool! Can you talk about more specifically what phase of that migration you're working on?

[0:28:16.3] MH: What phase of the migration? We have some our major products already migrated to our cloud platform. I think right now what we're doing is working through how we get the rest on there and how we do so — As an SRE organization, we're evangelists of this, because, again, we see the benefits to reliability, and doing so we see the benefits just to scaling. We see the benefits to how quick things can happen in terms of getting features out to the market. Again, that allows SREs also focus a lot more on the more long-term improvements that we could be making, allow the tooling automation we can do, because it allows [inaudible 0:28:53.7], which is now taken care of in tooling [inaudible 0:28:55.3] platform.

[0:28:57.1] JM: From migrating different teams, is there like a planning process that you go through to decide what is going to be a roadmap for a specific team getting into the cloud?

[0:29:08.8] MH: Yes. There is a very detailed planning process that goes through it. A lot of things are considered. Everything from just what is that product doing in terms of what does every roadmap look like? How is the landscape of technology that that product relies on changing? What are its needs are from a capacity and customer perspective and how do we make all these line up with what the platform and the cloud service providers like AWS can provide for us? Yes, there is a lot of detail planning that goes into this.

[0:29:39.7] JM: Can you describe the plan in more detail? Because I'm sure there's people out there that are listening that are — Maybe they work at a bank or an insurance company and they are looking at a large scale migration. They're trying to scope out what their strategy should be. What advice would you give around the planning process?

[0:30:00.2] MH: The advice that I would give — And this advice doesn't actually come from my experience here at Atlassian, but other places as well, is when you're looking at this, make sure you understand exactly why you wanted to do this. I think there is a temptation for companies to want to embark on this without fully understanding the reasons why. Really think about what your pain points are and how does moving to a cloud provider solve these pain points, and are you willing to kind of float the upfront effort and costs and in your term in order to achieve what

the long term benefits are going to be. You have enough — Can you maintain the focus on this enough to actually get yourself to that end state? That is the most basic thing that I would advise any company do, is understand what you're trying to solve first. Understand it very, very well and make sure that you have a proper understanding about what your return is going to be on that effort, otherwise you don't want to figure this out when you're halfway through it and you realize, "Oh my God! This is not going to work where we expect it to." I would suggest, first start off with the basics. Understand where your problem is, understand what you're trying to solve.

[0:31:03.8] JM: Now, is that to say that there are organizations that you would not advice moving to the cloud?

[0:31:10.5] MH: Yeah, there definitely could be cases like that. For example, there could be companies which are at such an extraordinary large scale that they run out of — The cost and benefit is no longer evenly balanced. There are other companies out there which have a business motivation in order to keep things internal to their own infrastructure. For example, you have logged major players out there in the market, like Google, Amazon and so on, and they work with each other, but at the same time they also compete with each other in a lot of different products bases, right?

I think there might be — And Microsoft, for example. All three of these company compete with each other, and it might not make sense to them from a business perspective for them to basically run on each other's infrastructure and pay for each other's R&D against each other, right? There could be situations like that out there.

[SPONSOR MESSAGE]

[0:32:09.3] JM: Failure is unpredictable. You don't know when your system will break, but you know it will happen. Gremlin prepares for these outages. Gremlin provides resilience as a service using Chaos engineering techniques pioneered at Netflix and Amazon. Prepare your team for disaster by proactively testing failure scenarios. Max out CPU, black hole or slow down network traffic to a dependency, terminate processes and hosts. Each of these shows how your system reacts allowing you to harden things before a production incident.

Check out Gremlin and get a free demo by going to gremlin.com/sedaily. That's gremlin.com/sedaily to get your demo of how gremlin can help you prepare with resilience as a service.

[INTERVIEW CONTINUED]

[0:33:08.5] JM: How does the SRE team play a role in a given migration? So if there are some products, so like maybe JIRA, for example. If JIRA was moving on to the cloud, what would be the role of the SRE team in supporting that migration?

[0:33:25.0] MH: In supporting that migration, the SREs would take on tasks such as building some of the tooling and automation around like the migration of tenants and resources from one platform to the other. Making sure that we properly test the foundations of the new platform to make sure that like requirements are met, things like war gaming, where we do failure testing and so on is done. Helping the development teams make sure that they get to you, operational readiness standards, have proper incident management practices, and we have a proper engagement, roles and responsibility engagement strategy between different organizations. Just overall, just keeping an eye out for anything that could break or could be improved.

[0:34:04.8] JM: That failure testing is a growing trend. How do you do failure testing and why is failure testing important?

[0:34:12.1] MH: We have a — First of all, it's important because you don't know what's going to break until you start breaking things. Even though you think you have an idea about what your moving parts are, a lot of times the best way to fully understand how they actually depend on each other is to try to simulate as many different ways of breaking things as possible. Not only do you discover new unique things about how your services run and how they depend on other things, but you also get an understanding about how you as a team and as an incident response organization can function and how efficient you are.

If you break things in a war game type of situation, you could measure how long does it take for me to triage something? How long does it take for me to even open up communications with other teams, escalate things? How long does it take me to even identify what the problem is, and then how do I identify — How long it actually takes me to fix things? Also through this, you

identify what's on the more common breaks could be, and you could work on fixing them proactively or you could work on building some tooling and automation that works around it or auto fixes and things like that, and you could refine your detection to these mechanism.

We have a war gaming practice here where we have different core teams, development teams, SREs, get together in a room and just go through basically an exercise where we take something and we break it and we see what we learn from it. A lot of these happens before we launch a service. Some of these things happen on a more routing basis, but it is a very common well-appreciated practice here.

[0:35:41.7] JM: There is always a tradeoff between migrating or updating infrastructure versus developing new features. Has that been an issue when you're trying to balance the migration with the new feature creation across the organization?

[0:35:59.7] MH: Yes, that's always going to be the question, but I think — The balance in terms of migration is always going to be on making sure that we could execute such things without F'in the customer. The whole point of us moving to — One of the points of us moving to our platform is the boost in reliability as well. For not keeping an eye on that during the migration process, then we're F'ing the customer, right?

[0:36:23.6] JM: What about capacity planning? Does an individual team have to make planning for how much capacity their product is going to grow to in a given quarter, for example?

[0:36:35.8] MH: Yeah. Currently, that's generally the state of things. Different teams kind of have their own capacity and make decisions on how much resources they allocate based on that. I think longer term, that could change to be something to have more of a consistent model across different teams. But for now, yes, different teams who understand their customer base, who understand their products base, maintain their responsibility.

[0:36:58.9] JM: Are there any particularly difficult parts of that migration, like when you're moving a team to the cloud, like refactoring their networking stack or migrating their database? What is particularly hard about the migration?

[0:37:14.7] MH: A lot of it is in the services and how they deploy and how they build the services. That's one of the areas. I think there's a whole lot of stuff that's probably worth another entire discussion on. Yeah, I think a lot of the things that you have to — Yeah, there's a lot of refactoring that's involved. On the top of my head the biggest one is basically how do you actually deploy, how do you build and deploy your services, how do you containerize, things like that are the first things that come to mind when we deal with [inaudible 0:37:42.8] platform.

[0:37:44.4] JM: What are some best practices there? Like if a team is trying to determine how to containerize their large product that covers a lot of surface area, how do they determine the scope of different containers?

[0:37:58.1] MH: I would say that like if you're undergoing a process like this, it's probably worth thinking about just how your service is architected in the first place. If you're already on a very clean — If you're already on microservices and things like that, then the situation is much different if you have a larger monolithic service, then you might not have so many options. There's always going to be the question of whether it's possible to do tackle any low hanging fruit by decoupling some of your services from each other.

But that, again, really — That answers depend vastly on what you're running and how you build things to begin with. That actually can swing in either direction very, very easily.

[0:38:35.7] JM: Are there any efforts to move to something like Kubernetes at Atlassian right now?

[0:38:40.8] MH: Yes. That is something that's very interesting to us. But I comment exactly since that team is not in SRE and how that's being addressed. Yes, Kubernetes is something that's very interesting.

[0:38:51.8] JM: Yeah, it'd be interesting to see how that factors into — Because you have this homegrown PaaS. Maybe it'll be working Kubernetes into that PaaS. Well, again, I guess you're not the person to ask about it. Tell me about your day-to-day, like you walk into the office on a given day. What does the head of SRE do?

[0:39:12.7] MH: I go get some coffee first to wake myself up. No, really, the first thing I do is I take a look at what's some of the incidents have been over the last day or whatever. I look at some of the low priority ones. If there's anything that was higher priority, I always have to take a look at those and I start following in. Another part of my day is usually a lot of planning meetings, thinking about what do we do given the data that we're collecting about incidents, about all the different metrics we're getting from our services. How should we be planning our next quarter, our next year and what are the needs of the organization evolving towards, and also how does improvements, reliability incident reduction, TTR reduction, SLO attainment and things like that factored into the overall organization plan. That's generally how I spend my day. Obviously, other meetings, more tactical things sprinkle in here and other administrative things. Yeah, generally that's how it plays out.

[0:40:11.3] JM: Do you have any advice for how to conduct a planning meeting for an SRE organization?

[0:40:17.0] MH: Advise would be start with making sure you understand what your metrics are, what are your goals for what you're trying to accomplish? These should be obviously informed from what your company goals are going to be, but start with that. Understand exactly what you're trying to change, what you're trying to influence and what impact you're trying to delivery through your organization for the quarter, for the next year. That's your basic starting point.

If you have that and you have your strategies outlined for that and you understand what your target numbers are, the rest of the planning, in my opinion, is a lot easier after that, because you'd understand what your targets are and then you could go off to all the different SREs and talk about, "Okay. What are the great things that we could do or that we need to do or that are being driven from the product side, driven from the infrastructure side and driven from the SRE side that could help accomplish these goals?

[0:41:09.5] JM: What advice do you have around managing SREs?

[0:41:12.9] MH: Managing SREs. I think SREs are very interesting group, because they spend a lot of their time dealing with operations. They spend a lot of their time doing engineering and software development, and maintaining a proper balance between that two is very, very

important. Now, I think the problem with a lot of traditional operations groups is that the amount of toil that builds up within those teams and manual tasks and toil, just toil, builds up very, very quickly. Unless you actively work towards prioritizing things which reduces toil and eliminates toil, the SREs are going to have a very difficult time. One of the things I would say is — The most important thing to say is make sure you have a proper balance in your team.

That said, toil is sometimes unavoidable and you have to do it and sometimes it's even fun, but longer term it's something that you want to always actively work towards, reducing, controlling and automating away.

[0:42:05.5] JM: What about new SREs? What's your process of onboarding new SREs and setting them up for success?

[0:42:12.4] MH: We do have like a lot of kind of boot camp type of training sessions. We have them paired and partnered with other SREs, and what we've started to do is having more rotations of where we have SREs that work — A SRE team works closer to the Confluence team or another team. We'll have those SREs actually embed with that development team so that they could do some diligent work, learn the code base and get more insights into that service from that development team. Then they'll bring those learnings back to the SRE team and they'll start doing normal SRE work. Yeah. Again, I encourage rotations between different teams, because that's a great way or learning, distributing the information.

Also, what we also do is we have SREs that rotate between SRE teams periodically as well, and I think that's important because it brings the learnings and insights from one group into another. SREs [inaudible 0:43:04.9] kind of like an information, like communication bus across the company.

[0:43:09.0] JM: What about interviewing SREs? If you're recruiting SREs, what are the kind of things that you want to ask them and vet them for?

[0:43:16.4] MH: Again, it also depends on what your needs of the job role are going to be. Some companies need SREs who are more operational. Some need SREs for a more development focus. Again, it goes back to understanding your needs. In general, for our team,

we look at making sure that they have some operational experience and have good coding and development practices and experience.

[0:43:41.1] JM: To circle back to that cloud migration a little bit as we wrap up, what kinds of value do you see deriving from the migration? Once it's entirely completed, what will you be able to leverage and where will you be able to go once you're entirely migrated to the cloud?

[0:44:00.0] MH: I think a lot of our — Scaling becomes a lot easier once we're in the cloud. Also, because we are effectively — Back to my point about taking away some of the time spent on things are not commonly provided, that freeze up a lot more of the product team's time, any product team's time for that matter and just focusing on the problems with their space and making their features great. The return on that I think is — The potential return on that is pretty large.

Also, in doing this, I think if you have a lot of different things solving the same problems independently, they're not going to have the same expertise in any one of those problems. As a larger team, that is just thinking about those problems, if that makes sense. In that way, it's going to make those common services and features more robust, more reliable and I think — And that team will be able to kind of look ahead of the curve and see what different product teams might need, start getting ahead of technology trends and so on. That, again, is just going to have a cascading acceleration effect across all the different teams that use the platform.

[0:45:09.7] JM: Right. We would get economies of scale if you have all the people on the shared infrastructure and you just have this infrastructure team that's building out the cloud services platform, that everybody can take advantage of that.

[0:45:22.7] MH: Yes. Again, you'd get a team of experts who are building a platform rather than everybody fending for themselves and trying to figure out the same problems on their own. Yeah.

[0:45:33.2] JM: Mike, I want to thank you for coming on Software Engineering Daily. It's been really great talking to you.

[0:45:36.3] MH: Thank you. This is a lot of fun. Thank you very much for having me.

[SPONSOR MESSAGE]

[00:45:41] 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 plug-ins. Use the value stream map to visualize your end-to-end workflow, and 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 longtime sponsor of Software Engineering Daily. We are proud to have ThoughtWorks and GoCD as sponsors of the show.

[END]