EPISODE 951

[INTRODUCTION]

[00:00:00] JM: Software bugs cost unexpected problems at every company. Some problems are small. A website goes down in the middle of the night and the outage triggers a phone call to an engineer who has to wake up and fix the problem. Other problems can be significantly larger.

When a major problem occurs, it can cost millions of dollars in losses and require hours of work to fix. When software unexpectedly breaks, it is called an incident. To triage these incidents, an engineer uses a combination of tools, including Slack, GitHub, cloud providers and continuous deployment systems. These different tools emit updates that can be received by an incident response platform and the incident response platform can allow the on-call engineer to have the information that they need centralized to more easily work through the incident.

On-call rotation means that different people will be responsible for dealing with different incidents that occur. When an incident happens, the current engineer who is on-call may not be aware that a similar incident happened last week. It might be easier for the new engineer to triage an issue if they have insights about how the incident was managed during the first-time.

Chris Riley is a DevOps advocate with Splunk. He joins the show to discuss the application of machine learning to incident response. We discussed the different data points that are created during an incident and how that data can be used to build models for different types of incidents, which can generate information to help the engineer respond appropriately to an incident.

Full disclosure; Splunk is a sponsor of Software Engineering Daily.

[SPONSOR MESSAGE]

[00:01:46] JM: If you are selling enterprise software, you want to be able to deliver that software to every kind of customer. Some enterprises are hosted on-prem. Some enterprises are on AWS. Some enterprises are on cloud providers that you've never heard of and every cloud provider works differently.

Gravity is a product for delivering software to any kind of potential environment or data center that your customers want to run applications in. Think of Gravity as something you can use to copy-paste entire production environments across clouds and data centers. Gravity is made by Gravitational, and Gravity works with on-premise data centers and on different cloud providers.

Gravity can get software to your biggest customers without the pain of developing individualized deployment systems for every single customer. Gravity puts a bubble of consistency around your application so that you can write it once and deploy it anywhere, and Gravity is open source so you can look into the code and understand how it works.

You can also listen to the episode I recorded with Gravitational CEO, Ev Kontsevoy. Gravity is built to solve the problem of software delivery. Gravity ensures compliance and lowers the cost of development. You don't have to write your code to support every platform. It is as easy as copying and pasting your deployment each time.

Gravity is from Gravitational and it's trusted by leading companies including MuleSoft and Anaconda. Go to gravitational.com/sedaily to try Gravity Enterprise free for 60 days. Gravity uses Kubernetes under the hood and the Gravitational team knows Kubernetes well. If you go to gravitational.com/sedaily, you can sign up for a free consulting session about cross-cloud Kubernetes security. This is in addition to the 60-day free Gravity enterprise trial.

If you feel like you need to get a better understanding of Kubernetes security, check out gravitational.com/sedaily for this offer of a 30-minute free Kubernetes consultation along with a 60-day free Gravity Enterprise trial.

Gravity is a system of securely delivering your applications into any environment, and you can try it free by going to gravitational.com/sedaily. Gravity Community Edition is also available on GitHub and it's free to play with. If you are curious about how Kubernetes will change software deployments, I recommend checking out the Gravity repository, and thanks to Gravitational for being a sponsor of Software Engineering Daily.

[INTERVIEW]

[00:04:48] JM: Chris Riley, welcome to Software Engineering Daily.

[00:04:50] CR: Yeah! Thanks, Jeff.

[00:04:52] JM: We're about 7 years into the "DevOps" movement, and this has had a large effect on software. The goal was to have developers and operations work together more harmoniously, but one downside that I've seen for some companies is that now everyone is an operations person and everyone is drowning in notifications and emails. What have been the downsides of the DevOps movement?

[00:05:23] CR: Yeah, it's a great question, and I could tell right away where you're going with that. Obviously, modern applications themselves have gotten more complex. Even though we spend a lot of our time talking about how to make developers lives easier, but I would say that the rate of complexity of applications has increased either faster or at the same rate with all of the new automation and so forth, so microservices, two pizza teams. Now we're introducing this whole concept of shift left where, like you said, developers are now getting more operational responsibility.

The reality of how that's played out differs for every organization, but one of the big challenges today is clearly just noise, the noise ratio that everybody is facing. You have noise from your application. You have your noise from alerts, the systems and the processes that you're dealing with. With microservices, you have this whole other abstraction where you're thinking about your services as a holistic thing with its own lifecycle, its own set of data, its own set of management tools and release cycles, and then the broader application where everybody has to play as well, because if you don't work together and play together nicely, then the overall application is going to have issues.

We have added a lot of new challenges. Now, fortunately, I think even today a lot of organizations have found ways to deal with those, and then looking into the future, obviously, there's a lot of great things coming out that are going to help support an make sure that we're not making developers lives worst ultimately.

[00:07:11] JM: People sometimes get a little dogmatic about the definition of that term DevOps. I've met people who believe that there is no such thing as a "DevOps" engineer, because that suggests that DevOps is being siloed into the responsibilities of that person, whereas if you're dogmatic about it, well, I mean, shouldn't DevOps be like the unification of these different responsibilities rather than these siloization of them into a single role. Are we structuring teams correctly today or do you think we're like in some kind of transition mode where we're figuring out the different roles and the structures of engineering environments?

[00:08:03] CR: I look at DevOps, I feel like there are two definitions of DevOps, and whoever says that DevOps engineering isn't or shouldn't be a role, please tweet at me now, because I'd love to have that debate. I have a lot of great DevOps engineer friends. But I think there are two, so there's DevOps the principles and DevOps the practice. I think a lot of times we're guilty of having conversations that focuses on one versus the other.

When we talk about DevOps practices, the word Jenkins and release automation and all of that comes up. When we're talking about the principles, the dreaded word culture comes up and getting people to play nice with each other. In both matter, and both are very real. I would love to have the argument again of somebody telling me that either of those aren't very real things.

Where I like to use the word stewardship as the blend between these two, because I think that part of the problem is if you get too far into the weeds of the DevOps tactics, the DevOps engineer, where it is, the siloed rule, their job is not just spinning up Jenkins. Their job is also stewardship. Their job is also to make sure that the entire organization understands what's possible, what's available and how to use it.

One of the models that I really like, and I don't think we're structuring. I recently did an interview of a distinguished engineer at a large financial institution, and this is true at USA net publishing as well and several other organizations I've seen where they actually build a system of stewardship, and there are several ways to do that.

One is the classic shared services model, which you'll find in United and a whole bunch of other companies, but then you have the model of the DevOps engineering team where it's not just a rule, it's a whole team, is vetting and building tooling that they are champing across the

organization. If you don't use it, that's fine, but you're not going to get supported if you don't. There's not only the stewardship, there's the, "Well, if I don't leverage these best practices and these tools, then I'm going to be left behind from the entire organization."

This is happening at scale in very large organizations. I wouldn't say it's right out of phase where we're restructuring where we're going from the unicorn DevOps engineer to the more mature thing, but there is this reality that's happening today. Like you said, seven years later, where DevOps is kind of becoming boring and we're really focusing – And I like that. We're really focusing on solving real problems today, and the conversations I have now are about the nuances of that, which are fun to have.

[00:10:54] JM: I think we can all agree that the cloud has made operations easier in many ways, but it's also created new operational problems. What are the new operational problems that the cloud created?

[00:11:13] CR: It's hard for me to answer that question without kind of putting it on the lens of the organization. I mean, the organizations that are moving from like co-lo data center, monolithic applications that just have a tremendous amount of considerations around security, re-architecting their applications. A lot of times the answer is it's going to us seven years to re-architect our application, which just doesn't make sense to the unicorns or the more cloud native applications, which also are facing challenges.

One of the challenges that I think that is happening today is largely around security, because I think it's everybody's biggest fear, but also to jump on the bandwagon of the multi-cloud and the hybrid cloud, because a lot of companies have kind of pick their de facto cloud. As a result of picking their de facto cloud, they will limit themselves on the types of services that they will use whether they're available in the cloud or not. There several issues with this. they might choose a service that isn't the right thing for their application or they may not be leveraging thing that could be tremendously beneficial to their application just because it's not with their public cloud service provider.

I think that this is going to be an ongoing challenge that organizations are really going to have to consider, and multi-cloud is going to matter not only from a high-availability, even legal

perspective, let's say your public cloud provider and you become enemies for some reason, but also from an, "Is this the best thing from my application in all regions in all cloud providers type scenario?"

[00:13:06] JM: What about Slack? Has Slack made operations easier or harder?

[00:13:10] CR: Man! Yeah, Slack is one of the unicorns. I mean, there's no doubt. Slack is one of the organizations I can say that the shift left, they are living it, and it's cool. You build it, you support it, two pizza teams, everybody's on call for the code. The culture is focused on making sure that the services that the organization puts out there are satisfying the user and the developers are not only just thinking about get through a backlog and delivering their applications, pushing code multiple times a day. I can't speak from being within the organization, but I can speak from what they –

[00:13:49] JM: Just to be clear. I'm talking about the product. Has the product itself –

[00:13:53] CR: Oh! Okay. Got it.

[00:13:56] JM: Because you hear these people who say like I am drowning in Slack notification. I don't know how to configure them correctly.

[00:14:06] CR: Yeah. The noise ratio, yeah. Yeah. What is Slack? Slack is a better interface to a very old thing called IRC. Now people are going back to discord suddenly. All of those tools have problems. The problem is information architecture and noise. If you don't think about how you structure your communication inside of Slack, then it could just become the same thing that email was that we all complained about, but also it becomes very often even though you can isolate channels and so forth. It becomes a one-to-many type chat ops tool, and that is very problematic, because it relies on the users to decide if something is important to them or not or if their involvement matters. That's when we get blindness.

We get blindness for everything in the tech field these days if we see too much of it. We get blindness for ads. We get blindness for chat notifications, everything. Yeah, absolutely. I think

chat ops in general, not just Slack, has created a big problem for teams because it's so easy to adapt, which means it's so easy to populate with noise.

[SPONSOR MESSAGE]

[00:15:24] JM: Monday.com is a team management platform that brings all of your work, external tools and communications into one place making cross-team collaboration easy. You can try Monday.com and get a 14-day trial by going to Monday.com/sedaily. If you decide to become a customer, you will get 10% off by coupon code SEDAILY.

What I love most about Monday.com is how fast it is. Many project management tools are hard to use because they take so long to respond, and when you're engaging with project management and communication software, you need it to be fast. You need it to be responsive and you need the UI to be intuitive.

Monday.com has a modern interface that's beautiful to look at. There are lots of ways to use Monday, but it doesn't feel overly opinionated. It's flexible. It can adapt to whatever application you need, dashboards, communication, Kanban boards, issue tracking.

If you're ready to change the way that you work online, give Monday.com a try by going to Monday.com/sedaily an get a free 14-day trial, and you will also get 10% off if you use the discount code SEDAILY.

Monday.com received a Webby award for productivity app of the year, and that's because many teams have used Monday.com to become productive. Companies like WeWork, and Philips and Wix.com. Try out Monday.com today by going to Monday.com/sedaily.

Thank you to Monday.com for being a sponsor of Software Engineering Daily.

[INTERVIEW CONTINUED]

[00:17:14] JM: I think of the cloud, and Slack, and many other SaaS products as kind of the changing face of software development, because I think 10 years ago we might've thought of

the process of building software was the composition of open source software. Today, the composition of open source software and SaaS tools and everybody's got a million open source projects that they're working with and a million SaaS tools. What are the other primary tools? What are the other tools that have changed the process of operations and software development?

[00:17:57] CR: Yeah. Isn't it cool that today, again, we don't have to argue for the value of DevOps? We don't have to argue for the value of open source as much as we used to. I think we're getting a better understanding of the variations of open source licensing models as well. But those are all really a part of like what is your application comprised of, which developers love to talk about.

What I like to think of is that your software delivery chain is a product as well, and whether it's the DevOps engineers building it, whether it's the development team collectively building it. In your software delivery chain you have different components. You have your source repository. You have your release automation. From A to Z, and at Z you have your incident response, incident management. You have monitoring in there. There're all these things that you need to put together, and there's open source and there's commercial.

Then the big benefit of commercial is really around support. But you're really building – When you build out these pipelines and slot these various tools in, you're building an application in itself that you're offering to your delivery team as a service and you need to be able to not only slot those tools in seamlessly. You need to be able to remove them seamlessly.

Integrations in how tools connect is really important across that entire delivery chain. As people get more advanced with their delivery chains, we're seeing even a new suite of tools coming in the area of data ops and AI ops and to use every acronym out there. There's a lot that comprises your delivery chain. A lot more than there ever used to be.

[00:19:44] JM: One part of the modern software experience is incident response, and this is the process of dealing with an outage or a large bug. When an incident happens, different companies respond in different ways. What are some best practices for responding to an incident?

[00:20:06] CR: Yeah. So this plays into the chat ops question, and a lot of times when we think of alerts we think about thing happened, X-thing happened, and then there was Y-response. There's actually a lot that happens in between there and it really is analogous to an ambulance going to some sort of event or tragedy. All the things that those EMTs do are the same things that we can kind of think about when we are actually responding. The very first thing who acknowledges it first? Is that the right person? Then what do they do after they acknowledge it? Do they bring other people in to the firefight? Do they get more support? Do they start to look at run books to address the issues themselves? Do they look at other issues that have come across? Because when we're talking about noise, a lot of times, critical system events or application events will cascade. So it's not just one event. It starts a snowballing and builds on to others.

That's why it's so important to get to meantime to acknowledgment and understanding of the issue as quickly as possible. The first thing is just making sure that you get the right people engaged. At every level of response, the triage, the getting the right people, the remediation, etc., that meantime to acknowledgment of the right people is going to determine how successful you are as you go down the chain up into the point of where you do postmortems.

I think that the best practice there is people oftentimes just think about the firefight. They don't think about what they do after the firefight and how to build on top of it. It's not that they don't do postmortems. They don't take them seriously and they don't put enough context as a part of their postmortem to be able to build on it in future. It's kind of like a police officer pulling somebody over and they don't want to file all the notes related to the ticket or whatever. They just kind of haphazardly put it all together and they're doing themselves a disservice when they do that. That is just the big thing.

Then the other big thing is, and it relates that everything, this blameless mindset. When I say blameless, I don't just mean like from a top-down is somebody going to get in trouble for this? I mean, actually, when an incident happens, the team understanding that we're not here to go and find the person who broke the thing. We're here to work together and find the immediate solution and long-term solution.

[00:22:54] JM: How did tools help with the incident response?

[00:22:57] CR: Yeah. I'd say, still today, the vast majority of organizations have a knock and email spreadsheet type system where they're literally going down and finding the right person to email, or they have a shared mailbox, or they're a little bit more advanced than they're using chat ops like Slack and they have a channel and they all watch it. That's the one to many approach.

What proper incident, management incident response tools do is they change that into a one-to-write approach. First thing, getting to acknowledgment. Finding the right person. Finding the right responder based on know where the incident came from. The type of incident. Also, when you get alerts, it's not just an alert. It's not just tapping somebody on their shoulder saying, "Hey! Something's broken over here. Come fix it." It's, "Hey! Something's broken. This is where it came from. These are all the details. This is the payload." You have a lot of contexts.

Also, using modern communication techniques – I mean, we're all very used to being mobile and we have our own ways that work for us to get our attention. Unlike chat ops and email where everything is kind of dumped in one spot, responders should be reached out to in the way that they're going to best pay attention to. Is that a phone call? Is it push notification? Is it a notification on their watch? Is it a text message?

Then also be able to say, "What happens if I don't react to the first notification? How do I want to be reached out next?' You have the whole sequence of events that need to come together, and that's really where the incident, management incident response tools come in. Then after the issue has been resolved, they have all of that data. They have all the sequence of events and everything related to an issue that they can then roll up into a really strong powerful postmortem.

[00:25:11] JM: Tell me more about how people collaborate during an incident. What is the tooling around collaboration and the workflows?

[00:25:19] CR: Yeah, there's a lot. There's a lot. There's chat services built in to incident response tools directly, which is always super powerful. There's the chat ops themselves. There

is a concept of having kind of like war rooms where you aggregate everything together, all the people, all of the information together to respond to incidents.

What's important, and this goes back to best practices. Wow! I really said that in a very enunciated way, is that the hierarchy, like how you determine what type of alerts get routed to what people, the type of information, the structure of the payload of the alerts that you want. How you reference run books and so forth. What's important is making sure that that's set up correctly, and people a lot of times expect the tools to do that for them and they don't. You have to be deliberate and actually think about this stuff. When you do that, then you're aligning it to your organization's best communication style.

[00:26:28] JM: You work with VictorOps, which is an incident management tooling company. Explain what VictorOps does.

[00:26:35] CR: VictorOps is all around offering site reliability engineers, even developers access to building that framework for as quickly as possible, getting the right people into the firefight when it comes to incident response. We refer to the whole thing as incident response, because the most important thing when something breaks is who responds and how you respond. We have a lot of very specific tooling and intelligence around how to do that in the most efficient way.

[00:27:17] JM: Overtime, you've gotten into machine learning, because these tools are generating lots of data from logging, to monitoring, to chat messages, to the kind of information around incident response that you get out of a tool like VictorOps. Ideally, there would be systems that are aggregating this data and learning from it. How is machine learning being used in ops tools today?

[00:27:44] CR: Yeah, it's a great question, and I understand that AI and machine learning for the typical practitioners, one of those phrases that you kind of roll your eyes at, and I happen to be one of those people as well. But the point of talking about machine learning is to really talk about its value. The first key value that machine learning provides in the world of incident response is being able to intelligently identify responders based on a history of responses and issues that happened.

So that's one thing that we already do today, which is really cool and something that we're building on top of, is we can tell you, we can recommend to you responders based on your routing keys, and routing keys relate to generally like your stack, like the type of technology that alerts are coming from. Not only that. What we can do is make sure that it aligns with your on-call rotations and your on-call schedules. So we don't just tell you the right people that could be brought into the incident to help you. We make sure that there are our people that are actually available so that in the thick of the moment you're not just paging everybody in the organization who's a specialist in Nagios. That the first place that machine learning is really helping incident response.

Then when you start to talk about into the future what can happen, you get into really interesting conversations, some hypothetical, some very real, where this technology ties together with monitoring tools and analytics. You have systems that start to become self-healing and where the first responders might actually be bots and not actual individuals.

[00:29:48] JM: Do you have an example workflow of how that could actually happen?

[00:29:51] CR: Again, I want to preface and say this is technology. We're speculating here about the future of where things could go. Incident response in that scenario has to be tightly coupled with monitoring, and more specifically, infrastructure monitoring and APN, application monitoring.

Again, back to what I was saying before, context matters a lot. Having full observability, full distributed tracing in combination with this tool that can identify the experts in terms of who to bring in and respond to issues. In that scenario, you can very conceivably see a tool that has kind of the sequence of events that it can choose from where the first response in real-time is to go through these events. It might be scripts. It could be something as basic as did you tried turning it off and on so it resets a server or something of that sort? Two is advance is running full scripts on your infrastructure, because it knows that previous times there was an incident of this sort. When this script is run, everything was fixed.

Then the next step could be even further from that where you have kind of the inverse of Chaos Monkey, really, where you have the bots that have a path that they can follow to solve the problems themselves. It could be something specifically related to infrastructure. It could be something specifically related to the codebase, especially now that people are using feature flagging, which is a very modern design pattern. It could be as simple as being able to identify that this feature flag was when the problem happened. Flip the feature flag, redeploy, etc.

A lot of intelligence can happen, and if that fails, then go to the, “All right, let's get a person involved.” But at that point, even once you get a person involved, you have this whole kind of thought process build up that you can share with the responder to help them understand what's been tried and what hasn't been tried and then what they should do next.

Then once we get to the postmortem and the remediation stuff, in the postmortem, the team can go, “Oh! You know what? We can train the system. We can help the system with these scripts or these set of instructions to resolve this issue in the future.”

The goal here is that I don't think you're going to replace the human element for a while. The goal here is that any common issues that are repeatable or you can anticipate happing based on historical data that you address those without human intervention and via automation.

[00:32:58] JM: Let's talk about today. What is possible today? What can we actually do today with machine learning?

[00:33:05] CR: Yeah. Today, I think what you already see in the market is you seeing monitoring tools deploying machine learning to have anomaly detection, so things that are out of order. They understand the steady state and they understand when things breach that steady state. You have predictive analytics to say, “Last time we saw this trend of activity, something bad happened.” You already have that. It’s being used. It’s very real. A person always has to act on that. So they see it, they act on it.

Then on the incident response side, as I said, at VictorOps what we have is the ability to very intentionally identify the right responders based on historical data. So that, again, it's historical data. We know when these types of incidents happen, we brought in this firefighter. This

firefighter was able to resolve the issue five times faster than whatever. You have all these data and you can build up to make a very intelligent decision on who to bring into the fight. What I'm suggesting is that you bring these together in addition to actions, because it's all about action. This data is great, but you have to do something with it.

In addition to actions, and those actions are automated in the form of you can call them bots, you can call it scripts. You can call them whatever you want, but it's automated. Very quickly as you build these together, you get this series of events that happen in real-time or maybe in advance based on those predictive analytics and hopefully get to a resolution before a human even needs to be brought in. That's where we're getting.

I mean, people throw around the term self-healing. I've yet to see a fully self-healing environment today. But there are very basic things that we can heal on and do very basic things like restart services and toggle feature flags and etc., etc. But it's going to get much more advanced, and we're going to start to look at a scenario where we're going to be talking about at what number of repetitions should this be like fully autonomous?

Hopefully, it becomes very low where the types of issues we are addressing are very high-impact. The human is getting involved in very high-impact stuff that is going to support the platform long-term in not just these common types of things that pop up on a regular basis.

[SPONSOR MESSAGE]

[00:35:58] JM: Every software engineer writes integration. Whether we're integrating Stripe, or Slack, or Google, or Facebook, we write code to leverage the APIs and tools of the software world. As an application gets bigger, more and more of these services exist in your app. You have Twilio, and HubSpot, and Zendesk, and Salesforce. You begin to want integrations between these different services and the amount of integration code you have to write grows and grows.

Zapier can simplify the integration process between your apps and services. Zapier is an online automation tool for connecting two or more apps. For example, I can use Zapier to integrate stripe with Google Sheets, and every time a user signs up and pays for a subscription with the

Software Engineering Daily mobile apps, their Stripe email address can be put into a spreadsheet, and Zapier can make that Google Sheet easily import those email addresses into our MailChimp newsletter, Software Weekly. Then Zapier can make sure that every reply to the MailChimp newsletter sends a message to our Slack. If that newsletter subscriber is also in our Slack channel, we could send them a message and start a more real-time conversation with them.

If you're looking for a single service that centralizes all these integrations into simple workflows called Zaps, Zapier is the easiest way to automate your work. Find out how Zapier can help your software integrations by going to zapier.com/sedaily to try Zapier free for 14 days. That's Z-A-P-I-E-R.com/sedaily.

There's probably a way that Zapier could make your software run more smoothly, and if you are just a technical person, you probably have enough spreadsheets, and Gmail accounts, and social media management that Zapier could save you some time personally even if you don't have a business.

Check out zapier.com/sedaily right now through November and learn how your API integrations could be managed more easily. Try Zapier for free.

Thank you to Zapier for being a sponsor of Software Engineering Daily.

[INTERVIEW CONTINUED]

[00:38:27] JM: Walk me through an example of how that works in a modern context. If I want to create an automated response to an outage, maybe that automated response is to loop in the right people. Maybe it’s to do something to my CICD pipeline. Maybe it's to start monitoring or sending alerting emails more aggressively. Walk me through like an example.

[00:38:57] CR: Okay. Yeah. I think that we’ll focus more on the something breaks scenario, because there're also scenarios like deploying code that has vulnerabilities in it and so forth. Something breaks. You have an issue with your backend. It's become nonresponsive, and that

will trigger an alert and it's probably going to go to your backend team. That's where the alert is immediately going to go in a normal scenario.

Now, in the scenario first of when we start to bring machine intelligence in this, instead of going to that team, you have a sequence of events that checks the various systems that have to be in place for your backend to be running. You have the service itself. You have the infrastructure, etc.

Let's just say very basic, it's the infrastructure. Infrastructure is down. Let's restart it, or it's a matter of rerouting traffic and doing some smart re-juggling of where the requests are going from and to. That's a very basic scenario. I'm trying to think of something more complex. I'm big on feature flags these days, and feature flags are used a lot for Canary releases, and Canary releases are great because it moves upstream and you can do deployments upon every commit, which is like the radical place to be. A developer commits their code, it goes to productions.

Now, if the organization has used feature flags properly, what you can do is not only AB testing where your deployments are going to different subsets of users, but you can have – If something breaks when you push a feature out into production and you're doing this at scales where there're lots of features going out of production in various times. Instead of just shutting it down and reverting back to a previous deployment, you can have intelligence go in, and the first alert is for a bot to go in and turn off X, Y, Z feature that they see as potentially being the cause of the issue. Then there can be an additional sequence of events where it then maybe does revert back to a previous deployment or who knows? Maybe it starts to evolve where it knows how to kind of actually fix the application so that the issue isn't there, because maybe the feature, the issue with the feature is something related on the infrastructure side.

There was a demand on the infrastructure that wasn't address previously with this new feature and it automatically spins up a new instance or creates a new Redis database or whatever needs to do to address it. You can take this in a lot of different directions. I think that the big point is that the technology, the core technology is there to make intelligent automation on-the-fly with production environments, but everything needs to start to come together across the

delivery chain to do that, and incident response is one big part of that where the response element of it could be bots and artificial intelligence versus immediately humans.

[00:42:32] JM: I'm actually very glad that you mentioned feature flagging, because I feel like this is something we didn't talk about at the beginning in the changes to “DevOps” that are occurring. I have had some conversations with people recently about just how influential feature flagging can be, because it can decouple the deployment process from the risk of enabling the features from the deployment process. Can you just talk a little bit, reflect on how feature flagging has changed the software development process.

[00:43:12] CR: Yeah, and I'm big on the future flag bandwagon right now for sure. One thing that's challenging with feature flags is it does assume some level of DevOps maturity. When I talk about DevOps maturity, I don’t mean in terms of the principles. I mean in terms of execution, like you have automation. You have to have automation.

Initially, feature flags were pitched as this I test in production technology and the Canary releases, which is really cool, and AB testing, all of that is cool. I'm not going to question that, but the problem with AB testing and Canary releases is that it assumes that your application has this large enough user base, enough transaction volume to make that testing valuable, because you have enough data to actually produce meaningful results. That's all great, but that's not true for all applications. I think feature flags are a design pattern that are useful for all applications. I don't think that this is just something that you use for some applications and not others.

In the context that we’ve been talking about it, I’ve been talking about it from a support perspective, really, which is how do you release functionality and be confident that you can put it out there, and if something breaks very quickly, bring it back. That's a critical aspect of it.

But not only that in your delivery chain when you talk about continuous integration, continues delivery, and you talk about application quality and quality engineering, feature flags also give more autonomy to your quality engineering team where they don't have to have this regular communication with developers about what's coming. They can actually see the feature flags and use those in their testing process and they can decide through testing whether or not a feature flag is enabled or not versus going back and being blockers of functionality.

Then, finally, we still – Even more in microservices world, features are not necessarily features in a service. A feature could be a feature application wide as defined at the product management level. With feature flags, what you can do is you can actually allow microservices teams, each service team, to release functionality a part of a broader feature, but not turn it on until everything is there for the application. So you don't slow them down by making them have to collaborate with every team that is involved in this big feature, whatever it is. You can let them build it as they do. Then once everything is there, you flip on the future flag for that whole feature and now it's available broadly.

[00:46:08] JM: We have all these integrations. We have feature flag integration, Slack integration, GitHub integration, logging immigration, all these different things are integrating and they're creating different forms of data. How can we use that data from the different integrations to have a smarter system for dealing with our operational processes?

[00:46:31] CR: Yeah. Sometimes in this world it feels like we're just chasing tools and chasing integrations, especially as it relates to incident response, because it's such critical to have your tools tied together so that your delivery chain is comprehensive.

What happens is that all of the data these systems pull out, whether it's exception monitoring, real user monitoring, application monitoring, release automation, infrastructure monitoring, cost monitoring, vulnerability scanning. All of these stuff. All of these tools are great at producing data. The problem is it's so much data that no human in themselves can go in and make the mappings. Even if they did – So when I say mappings, like map what happens X exception leads to Y critical event, and then the response associated with that and then how it's responded to.

It's really cool that we have all these data. But like I said, it's not feasible to expect humans to create a configuration in any tool that can map all those things together if not for the reason much less the amount of time it takes compared to also the value it gives, but also it's changing too much. It's changing too frequently. Developers should be able to choose the tool that they need for their job.

The only way to do this is to have some very clear intelligence that can do the mapping on its own. We're really big on context. We're really big on making sure that what you get in an incident helps you resolve the incident. When we talk about self-healing and what's possible with machine learning and in the future where incidents may be responded by bots, they need to be able to aggregate all this data from all the systems. Intelligently relate them together based on historical information and then take some action towards it.

One might say that the answer is less data and more structured data, and we already saw how that played out in the database back in the world where people moved from relational databases to NoSQL databases. This is even more unstructured data and it can be changing. Really, what you need to do is have really strong analytics, predictive analytics models that can relate all of the pipeline data from your pipeline and all the tools in your pipeline. Correlate those into historical incident data and the solution to those alerts, which is more structured data and start to design an, "Okay. When all these chaos came through the system, we were able to correlate these events and we are associating it with this type of resolution." The resolution may be human-based, but it could also be machine-based.

More data is not always a good thing. But if you have intelligence, which we do – I mean, in the big data field and all the models out there and all the things that are possible, we do have that intelligence. If we tie those two together, keep the same amount of data with the intelligence and the ability to respond, then we have some really cool things where our delivery chains become a lot more autonomous than they ever were.

[00:50:13] JM: Have will the process of on-call change within the next five years?

[00:50:19] CR: I would love to tell you that it's going to change dramatically. It's going to change dramatically for the organizations who take it seriously. I think that for a big chunk of the companies out there, they're just realizing how important it is to have an incident response solution, and the adaption of that is still going to happen.

But I think what's going to happen is that on-call is going to become a lot more stressful. It's going to become less frequent, but more stressful, because I think what's going to happen is that when a human gets paged, it's really bad. It's going to be like catastrophic. A whole region

is down. Your CDN has shut down. It's going to be big, big things, really bad, super stressful and you have to have a human involved. It will improve the lives of people on-call, because it's going to be much less frequent and it's going to be much more targeted.

When they get paged, when they get an alert, they are going to have a lot more at their disposable, a much bigger bag of tricks to solve the problem much faster and go to sleep. I do think – Because it is a very stressful life, and I do think that their lives will get better in the respect that they will be engaged less often. But I think when they are engaged, it's going to be much bigger, much more catastrophic things that need their involvement, and a lot of SREs live for that fire fight. They live for solving those really hard problems. So maybe they'll be happier.

I also think the way it's going to change the world is that it's going to roll into application design and architecture even more so that the true feedback loop of production back in the product will exist, because there's going to be a lot more data in context to know how we improve the infrastructure we run on. Do we go into multi-cloud?

All the things we want to do to improve our infrastructure and applications, we can now relate to actual real-world events that happened in production, which today you don't see a lot of. You don't see that feedback loop. You still see that wall where it kind of stops. Issue was resolved and it didn't go back to product or it didn't go back to the ops team to figure out a different infrastructure setup to support the application.

[00:52:54] JM: Chris Riley, thank you for coming on Software Engineering Daily. It's been great talking.

[00:52:56] CR: Yeah, it was great. Thank you.

[END OF INTERVIEW]

[00:53:07] JM: If you want to extract value from your data, it can be difficult especially for nontechnical, non-analyst users. As software builders, you have this unique opportunity to unlock the value of your data to users through your product or your service.

Jaspersoft offers embeddable reports, dashboards and data visualizations that developers love. Give your users intuitive access to data in the ideal place for them to take action within your application. To check out a sample application with embedded analytics, go to softwareengineeringdaily.com/jaspersoft. You can find out how easy it is to embed reporting and analytics into your application. Jaspersoft is great for admin dashboards or for helping your customers make data-driven decisions within your product, because it's not just your company that wants analytics. It's also your customers.

In an upcoming episode of Software Engineering Daily, we will talk to TIBCO about visualizing data inside apps based on modern frontend libraries like React, Angular, and VueJS. In the meantime, check out Jaspersoft for yourself at softwareengineering.com/jaspersoft.

Thanks to TIBCO for being a sponsor of Software Engineering Daily.

[END]