Incident Management with Adrienne Walcer

Adrienne Walcer discusses how to approach and organize incident management efforts throughout the production lifecycle.

Viv: Hi everyone. Welcome back to the SRE Podcast. This is a limited series where we're discussing SRE concepts with experts around Google. I'm Viv, and I'm hosting with MP today.

MP: Hello.

Viv: Hi, MP. We have a special guest today: Adrienne. So hello, and... introduce yourself?

Adrienne: Sure. Hey, y'all, I'm Adrienne Walcer. I'm a nine-year Googler and a Technical Program Manager in Site Reliability Engineering. I'm also the program lead for Incident Management at Google. You may have seen me before, through— I wrote a book thing? I'm still self-conscious about it because I'm pretty sure my mom has just downloaded it several thousand times. She's lying to me about it. It's called *Anatomy of an Incident*; that is with O'Reilly Publishing. And I also had the pleasure of hanging out at the last USENIX LISA conference, for all of you large-scale distributed systems nerds. Wow— what a weird and specific thing to be nerdy about. Yeah! How are y'all today?

Viv: We're great. We're glad to have you here. And I am glad you know so much about incident management because that is what we are here to talk about today.

Adrienne: Nice.

Viv: So in our previous episode, we talked a little bit about just generally being on call, and now we're kind of at the point in the process where, hey, it's whatever your favorite day of the week is, and you're at home, chilling— or if you're me, taking a nap— and your pager goes off. So I guess the big question is: then what? Or I don't know if you want to back up a little bit, but where do we start when we talk about incident management?

Adrienne: I've noticed that there are two primary schools of thought around incident management. The first school of thought is that you manage an incident and then it's over; you move on. Incidents are a single point-in-time event that occurs that you need to get through in order to go back to doing whatever you were previously doing. The second school of thought is that incident management is a practice that you do every single day with every single piece of engineering that you touch. Incident management is a continuous cycle that will exist throughout the lifecycle of your system.

I am a firm believer in number two. So even though your pager is going off at one moment, I believe pretty firmly that you are managing the incident from the second that you wrote the root cause into your technical stack.

Viv: I like it. Definitely good to back up then. If you are always in the cycle of incident management, then how do you know where to start when you want to get into best practices?

Adrienne: Great question, Viv. So there's the difference between your perception of a system and how the system is actually functioning. Usually, we first get involved in the incident management process when a signal has been raised to indicate that our system is no longer in the expected state of homeostasis; something has changed. An alert has gone off, a user sent you a nasty email saying "what the heck?? I lost all of my Angry Birds" or something. You get an indicator that something isn't normal. Ideally, these are automated, but that's a problem for later us. And this indication that your system is in some kind of hazard state, some kind of outage state, some kind of incident state, that's when usually we start engaging with the incident management process, because something has been tipped off that, oh, we need to act and we kind of need to act now.

But realistically, we build our incidents and our outages into systems as we engineer them. Every time we create a dependency, we create a possibility that that dependency won't be available. Every time we create a config, we create the possibility that that config is gonna be out of sync with what your system really needs. So on a continuous basis, it's the cycle of triggers and a matrix of root causes. And these root causes are gonna be the underlying challenges or issues that your system might have. And these triggers are environmental or system context shifts that enable those underlying system hazards to become problems, to become incidents, to set off those alerts and set off those pages. And what those really are, are indicators that something isn't the way it usually is.

This doesn't always mean that an incident is necessary and that it's time to raise the alarm and wake everyone up. No— it means that it's time to apply a little bit of thinking and diagnosis and triage to that system and identify whether or not this marker that has no longer been moving in a normal direction, whether that's indicative of something really problematic or whether it's just an environmental shift that your system knows how to handle. And then the incident management process from there is working through, responding to that incident if that incident exists, helping to mitigate whatever challenges are thrown your way— helping your system recover, become more stable, grow further. And then eventually another incident will happen again.

So it's a bit of a cycle and we move through these phases of planning and preparation of incident occurrence, of response, of mitigation, and of recovery. And usually these recovery actions— these things that we take to build a more stable system— are the same things that we should be doing in order to better prepare and train and be ready for incidents, 'cause those hazards, they already exist. The state of building something means there is uncertainty as to that thing's future.

MP: It sounds like you're using hazard as a bit of a term of art. Could you define that for us?

Adrienne: Yes. So when I'm thinking root cause and trigger or hazard... so a hazard can exist in a system for an indefinite period of time. The system environment needs to shift somehow to turn that hazard into an outage. And essentially "root cause" kind of equals the "hazard." In reality, a root cause is usually a very complex matrix of factors, of timing, of user behaviors. But saying "root causal matrix" is kind of a mouthful. So when you say "hazard state," that means that your system is in a state in which something is vulnerable to error. And that's kind of cool, but that system environment needs to shift to transition that hazard, which can exist in a stagnant kind of non-moving state for a while. Something needs to happen for that to become a problem.

I mean, think of it a little more broadly. A gas leak in a house can go on for hours, but it's not until an electrical switch flips or a burner lights up before that gas leak turns into an outage or a problem, or an incident scenario. There needs to be some kind of environmental shift and that environmental shift, usually we refer to that as the trigger. So it's related, but it's a separate thing. And these two work in concert in order to create hazardous or dangerous scenarios. And when you think about prevention efforts, sometimes it makes sense to work on and make more safe one of these underlying hazards within a system. But sometimes it makes sense to build stronger prevention around these trigger conditions— essentially building a safer environment for the system to operate in. So you could get more of what you want, which is safety, which is security, which is reliability.

I know it's super nerdy, *super nerdy*, but these are some of the language bits that we use in the field of systems engineering to talk about broad systems. They aren't inherently software engineering terms, but, you know, SRE is a combination of systems engineering and software engineering. So it makes sense to borrow from both fields of study. And there's a lot of really interesting literature and nerdiness, general nerdiness around the creation of hazard states and fault analysis within systems. And all of it creates really juicy learning tidbits that I try to get everyone to consume. And sometimes it's like getting kids to eat their broccoli— unless you have one of those weird kids that loves broccoli, in which case, jealous. But sometimes folks see the value of some of this deep nerd dumb. And I'm hoping that a lot of folks will take this as what it is. It's a language that can be used in order to describe incidents and incident management and start some of these more critical, important conversations either on your team and your company, maybe around your house. I don't know what you do in your free time.

MP: Going back to the stages of an incident that we were talking about, there's mitigation and recovery and kind of that improvement side of the story.

Adrienne: Mm-hmm.

MP: I'm now hearing about this framing of hazard [that] is brand new to me. And now I'm thinking about on the recovery side that you can either try to eliminate a hazard, but you can also try to manage a hazard. You don't necessarily need to— at least that's my initial intuitive response— is you don't necessarily need to eliminate hazards; you can manage them. Is that intuition true?

Adrienne: Yeah. I mean, hazards exist everywhere and a lot of them are totally out of our control. Like things we can't inherently control are user behaviors, whether or not your app or video goes viral because of the shout-out on a local news show. We can't control the speed of light, the size of the earth— which is woefully inconvenient sometimes. But all of these things create, essentially in some weird way, hazard conditions that we need to work around in order to optimize our system for what we think, or maybe what we know, or maybe what we hope our users need. And here's a really bad analogy: being outdoors, kinda hazardous. There's bad weather, maybe there's animals, maybe there's bears, you get bugs. Just weird stuff happens outdoors. I'm an engineer, Viv. I'm indoorsy. Don't laugh at me.

Viv: No, we're all indoorsy here. I think it's funny because the way you said it made it sound like the outdoors is just this perilous, like, nightmare. Don't step outside, you know, the mosquitoes will get you.

Adrienne: I do believe the Google Australia sites are hiring if you're interested in a thousand things that can murder you. But, outdoors: many hazard conditions— so many hazard conditions. But how do we work on these set of hazard conditions in order to keep our human bodies safe? We live in houses or in buildings or in tents or yurts. I don't know what people do. But that's essentially a mitigation from the series of hazard conditions that exist for living in the world.

I know— so deep, right? I am a puddle. But it brings up some really interesting things because incidents happen. They're weird; they're confusing. Maybe we like them, maybe we don't, but we need to figure out what is the best work that your team can do after an incident. And by thinking about things in terms of this language of hazards and triggers, you could figure out where your team or where your org or whatever can exert the most control in order to make the most efficacious changes on that system and on that system environment. You know, your incident has been mitigated, your system is stable again; does it matter that you understand the underlying problems? Yes. Because you want to prepare for the next problems and you don't want the same problems that you just dealt with to come back again. Something that our SRE vice president Ben Treynor has said on numerous occasions is that he only wants new incidents. You know, if we've seen something before, we don't wanna see it again. We're not into the replay or repeat button on the record player. Wow, record players. I am old. Doing the work to understand what is shifting within your system and how it works, that'll help you from having those same repeat issues over and over again.

Viv: That makes sense. So you mentioning that you don't want repeat incidents reminded me of— I think we might have jumped a little bit over this, but I'm curious if you have thoughts particularly. So you said, okay, yeah, your incident is mitigated. Sometimes that's easier said than done, right? Especially in the moment. I guess I'm curious before we even get to the "how do we wanna move forward?"... how do you make sure this doesn't happen again, you know, et cetera, whatever steps come afterwards when you're there in that moment. I you're looking at something that is totally new— new trigger, a hazard you didn't realize was around— what can you do? I don't know. What is your advice in this case? Or, you know, another nice outdoor analogy, but I'm a little lost on the analogy right now.

Adrienne: No, no, I feel ya. Like, I think the most terrifying part of the incident management lifecycle is that moment when an alert or your pager goes off and there's those few seconds of, "Oh gosh, is this a thing I can deal with?" And then there's always that like, "Is this a thing *I* can deal with?" 'Cause those are the moments where I always have that instinctive feeling of like, well, time to go and cry to my manager and I'm gonna hope that they have sufficient magical wizardry, that they can handle the big evil problems.

So an incident happens; you're staring at something you don't know how to deal with. I have two things that I love for people to think about at this point in time. Number one is figure out immediately, or as quickly as possible, if you're having any user impact. Because realistically, if your system is sufficiently robust, a little bit of outage might not be super noticeable. But if your system is quite sensitive to its current balances and configurations, if it can't take a lot of really sudden fluctuations, what you wanna make sure of is that your end user is having an acceptable time and interacting with your system. So for there, we wanna do things called mitigations, and Jennifer "Macey" gave a great overview on mitigations and mitigative activities. I believe she appeared recently on the SRE Prodcast.

Viv: She did indeed. She's our first guest.

Adrienne: Awesome. So if you see immediate user impact, like I love to prioritize having some kind of Band-Aid to bring about a little bit of stability, such that you have time in order to figure out what's really happening within your system. And depending upon the size of the problem and how many folks you're gonna need in order to resolve that problem, that's where some of the art of incident management comes in.

So here at Google, we use a cool variant on the FEMA incident management system. We call it IMAG: Incident Management at Google. And we believe that incidents are an issue that have been escalated and require kind of immediate, continuous, organized response to address it. This means you're gonna need to organize your team in order to make progress on this issue. And this means putting together some kind of organizational structure such that you can make progress in parallel with your teammates, but we can get more into that later. I said that there are two things I like to think about when a page immediately goes off.

So the second thing that I like to think about is knowing who is accountable when an incident is happening. When pagers go off, it's really simple. Whoever's pager is going off is somehow responsible or accountable for that incident. But when you think about broader things, it can get really confusing who owns what, or what piece of organizational muscle should be activated in order to work on and make progress on a given scenario.

So figuring out in advance who is accountable, who owns the thing, who works on things when something happens, those are the things that can buy you a lot of really valuable time during an incident. Because that deer-in-headlights response of, "Oh gosh, who does the thing?" When you multiply that by the number of people on your team, the number of people having that same response, what initially starts out as seconds in figuring out who's accountable can turn into minutes, can turn into hours, can sometimes turn into days of figuring out what subteam or what person is accountable for resolving an incident or an issue. And that's where you can really lose organizational velocity in addressing your users' needs. And it's incidents like those where everyone's trying to figure out who's responsible where the person who ends up looking like a hero— and I normally don't condone any kind of heroism— is the person who's just willing to raise their hand and move first. 'Cause when really messy things happen, any kind of leadership is helpful. You just need someone who's willing to take the first step.

And particularly at Google, we've got a massive organization, like [a] hundred thousand plus employees. Figuring out who's in charge of things and who makes a move when bad stuff happens, that's a continuous and ongoing challenge. But that's also been some of the places in which we've been most successful during major incidents. During a couple really recent big blowups— you think of the Log4Shell vulnerability in JavaScript where, you know, you just get a little bit of remote code execution, it's cool, we all do it. A major security vulnerability, like threatening the whole internet. One of the cool things that Google was able to do was we were able to mobilize really quickly. It took a matter of hours before we had some teams on pretty much every product area working to address these issues and bring things to closure. And I almost have no idea how it happened, even though I was working on it? But the ability of a company to mobilize quickly, figure out who's accountable, and make some concise steps forward— that's a strength. That's a huge strength towards business continuity.

And if you and your team ever have that deer-in-headlights kind of moment: number one, totally okay. Everyone does it. Everyone does it. It's cool. But take a little bit of time and diagnose those challenges. What are you feeling when you first see an incident? Is there anything that you wish you were better prepared on? Is there anything from previous incidents that now scares you to work on them? Because incident management is about building good habits all across that lifecycle. And if you can take some of these best practices and turn them into habits, you won't be relying upon looking up a playbook to figure out how to resolve something. You'll have that internal, intuitive understanding of what those next steps should be and how we can communicate and work together to resolve an incident. And that means you will have more uptime and everyone will be happy and will hold hands under a rainbow. It'll be beautiful.

Viv: Even though the rainbow is outside, just to clarify.

Adrienne: [Laughs] The rainbow is outside. It is dangerous there. We have also built windows in order to see the rainbow while being inside, because innovation is key for incident management. Oh my gosh, I'm gonna make myself puke.

MP: Something you said there that stuck with me a little bit: that getting slowed down by determining ownership of an issue.

Viv: Ooh, yeah.

MP: I think I'm usually, in my mind, I'm more familiar with that in the recovery stages, where it's like, "It's been mitigated". And then it kind of, it gets lost in the morass of everything else going on. But I wanted to think more in the moment where I have heard it phrased a few different ways. My manager likes to use the phrase, "You have the keys to the car." I've heard other folks phrase it as temporary director-level authority of, like, that SRE that has the pager: "You're the one with the pager. You own it. This is your problem until it's over."

Adrienne: Yep. But then you get messy distributed stacks where problems can, like, slip through the cracks or something can happen across multiple systems and you need to determine who's actually in charge.

Viv: Yeah. Yeah. [Laughs]

Adrienne: Oh, that's a messy one, but that's where categorizing different types of incident responders can be kind of helpful if you have a big enough and messy enough technical stack. Here at Google, oh yeah, it's the messiest. I love it. But we have essentially two types of incident responders at Google. We have component responders, and these are incident responders on-call for one component or system within Google's overall technical infrastructure. And then we'll also have systems of systems responders. And these are folks that are on-call to support incidents that might span multiple component systems—incidents that fall between system boundaries. Or sometimes they're just the folks around when anything gets messy.

I mean, the reason that we hire teammates is because often the scope of a technical stack grows beyond one person's capacity to understand and maintain state. So we split up that technical stack, such that multiple component responders can provide coverage on a single component of the whole stack. For example: you look at our Ads team. We have folks on call specifically for video ads, specifically for YouTube ads, specifically for mobile ads, specifically for—what do we got? Google AdSense, Google AdX. We got all the ads, and there's somebody different who's usually holding the pager for each. And these are gonna be examples of component responders. We maintain kind of a limited scope primary responder to resolve some issues. But, there, we get a risk of remaining ignorant to production issues that span maybe multiple components or between system boundaries, or we also get a chance of not providing individual component responders with sufficient support if an issue proves to be beyond their expertise. And it's important when somebody is going on-call that they don't feel alone. Oh man, what an experience that must be. So adding a second line of defense that's a little more holistically focused can provide some real advantages.

So I talked about a couple of different potential component responders for our Ads team, but we also have an overarching Ads incident response team. And these are a group of folks that just have a lot of tenure working on Ads products, and if something arises that goes across multiple components that seeps between multiple ad systems, they're there to provide overall coverage and structure for the whole Ads product. And we do this in a couple of levels at Google, but by differentiating between individual component, as well as systems of systems folks, you're able to scale your incident response and build an organization really well designed for the technical enterprise that you've built. I can spill more details if you need, but... I don't know. [Laughs]

MP: I'm just thinking about mapping that to my own experience where I almost feel like my team's a little in between, because we have a lot of large-number— like it's all one system, but it has a lot of, like, subsystems all maintained by different developer teams. So it's kind of like each binary is an entirely different development team that don't necessarily really talk to each other a lot, but it's all one giant thing. And then we have the one to two dozen binaries that we're responsible for, of this whole chunk of systems. And then we talk to external teams!

Adrienne: So that sounds messy.

MP: That a lot of our job being on-call is, what is actually broken? 'Cause usually the thing that annoys us isn't necessarily the thing that actually is having the misbehavior. And then okay, who can actually fix it? Can I fix it? Do I need a developer to fix it? Do I need Spanner SRE to fix it?

Adrienne: Yeah. And I mean the idea of "component" is gonna be flexible based upon the size of your stack, the size of what you're handling. So it sounds like you're sitting on top of a really big component, and the cool thing is that you build a lot of experience in troubleshooting different types of systems. So you've become a more skilled generalist, you learn a little bit more holistically-focused engineering. So how do all of the things work together? You also get to learn some cool stuff about triage and how to organize others in messy scenarios. You learn how to command complex situations and diagnose systemic behaviors. So your team is in a position to give you some real career advantages. But at the same time, I would imagine it would feel a little bit invalidating if you're told you should be expert on all the things, because it's not possible.

MP: It's not remotely possible for my team at all.

Adrienne: I mean, but it's possible for *you*, because you are brilliant. I say that with 100% genuineness; no one gives themselves enough credit. But it's tough. And in those types of scenarios, being able to regularly review your team's operations, comparing notes on what's causing problems, of what's going outside the quality control thresholds, looking over stuff like postmortems and practice as a bulk, is gonna be more important to identify patterns from across your whole team of all the problems that you're seeing, because you're not just looking for behaviors within one more isolated system. You're looking for patterns of behavior across multiple systems— across multiple systems that sit together and talk together with different dev teams involved— and being able to coalesce all of those nuances that you've now witnessed into really concrete ideas of what's the best thing to work on now.

That's a position of power, but it's also a lot of work, which is why it's important really not to burn your teams out. Incident response is *exhausting*. Holding a pager is *exhausting*, and because it's such a human-expensive activity, it makes a lot of sense to do it as sparingly as possible and really work on prevention, work on preparedness, such that incident response isn't as human-expensive as it can feel. Because sitting at the helm of 12 systems, like— MP, I gotta ask, is a single on-call shift tiring?

MP: Depends on the day. We're kinda spiky. We tend to like, we'll have a week where— or like a week or a week and a half where the pager will just be really silent and nothing will happen. And then some hazard will start finally manifesting, and then it will just wreak havoc for a little while, and then things will quiet down again.

Adrienne: [Laughs] I feel that. Launches never go the way we want. But it's tiring work and it's, you know, it makes sense to do it as carefully as possible.

Viv: Yeah, absolutely. [Sighs] Yeah. Lots of complications. Another thing I was thinking about, not to jump back too far, but when MP was talking about all the complexity and we were talking about different types of incident things, you also— I'm just saying this from experience, 'cause it's happened to me recently— you also get these scenarios where maybe multiple teams get paged at the same time and it becomes a bigger thing. How do you coordinate that?

Or, you know, in my case recently, somebody paged me for something they got paged for and then I paged somebody else. It can also get complicated too, not just in the general expertise before and after, but while you're there, as you mentioned: having organization and having control of who's gonna do what in the moment is important and it can be really tricky.

Adrienne: It can be.

Viv: Yeah. I don't know what the leading advice on that is, but—

Adrienne: This is where having a really solid incident management protocol that all of your team kind of knows about equally can be really helpful. So for Google's incident management protocol, we think about the three C's of managing incidents, which is:

- **Command**— so making decisions and keeping the team or subteam focused on the same goals
- **Control**— know what is going on, coordinate people, be continuously aware, and
- **Communications**— so taking notes, being clear and ensuring that everybody has the same context.

Being able to fall back onto the same set of things can be really, really helpful. When you're pulling people in, it means that you don't need to explain who is doing what and how. You just see the role title and you understand, and you know how to move through things. So the version of the FEMA incident command system, it has defined roles like incident commander, scribe, communications. And by using a shared and clearly defined process, we build really positive emergency response habits, including maintaining active state, a clear chain of command, and just overall reduction of stress. Everyone understands who to go to in an incident and how to hand off.

So essentially, a chess player can't drop a bishop on a Mahjong table and make sure that everyone knows what to do with it. You know, in urgent situations, it's important that all players are playing the same game. So by using a common incident management protocol, you can page in a bunch more people. And you're all gonna understand where to go, what to look at, how to talk to each other, who's generally in charge of what, and it's in part the role of the incident manager to make sure that this information maintains in a rigid, concrete kind of state as folks move in and out of the incident. It's maintaining that protocol— maintaining that incident management structure will help to drive clarity through the escalated state.

And when things get really big and really messy, and they totally do, there isn't often enough mental space for one incident responder to work out the appropriate mitigations and coordinate, implementing those mitigations and communicating to everybody that's dependent upon the system and manage expectations. Bringing in more people is helpful. No one can do all that on their own. So having like a structure that you can fall into, maybe like some standardized tools or plans that you leverage in order to track how this protocol is going, these are some of the tools that are gonna allow you to move smoothly as an organized unit through really, really messy times.

So when you get that, like, deer-in-headlights response, sometimes even a sloppy first step is to be like, all right, I'm the incident commander. I know how to do "incident commander." What are the things that I remember to do? Okay, let's start by broadly checking things out for context. That's an incident commander role.

And hopefully, you'll be less awkward than I am when explaining it, but it gives you a little bit of muscle and a little bit of habit to fall back on. You know, it's playing a game that you're very used to, that you know the rules of. And in that messy scenario, taking the first step forward is gonna be the most important step.

Viv: Yeah. Thanks for diving into that a little bit.

Adrienne: Sure!

MP: So if there's one lesson or concept that you could instill in the hearts of all of our incident responders— all of our on-callers, our pager holders, our SREs— what would that be? What's your number one takeaway for incident management that everyone should try to internalize?

Adrienne: Everyone should try to internalize: incident response is a really human-expensive activity. Multiple people need to be involved in driving an incident from initial alerting to resolution. And the act of incident response is to put a mitigation on either that root cause, that hazard, that problem scenario while it's happening in order to buy some time to make judgements about priorities. And it's disruptive, it's tiring, it wrecks all of your normal plans, but by thinking about incident management as a broad lifecycle, by working on preparedness, by working on recovery, these are some of the things that over time can increase the amount of time between incidents, maybe reduce the frequency of incidents and build a more robust system.

So I guess my number one takeaway is: do as little incident response as possible. Focus on great engineering— building really sound products that can handle a wide variety of user behaviors. Build in reliability from the ground up, use [incident response] sparingly, avoid burning out your team and start doing that engineering work needed to fix longer-running issues or risks. You have a lot of tools at your disposal, so make it happen.

Viv: Thank you. I will definitely keep it in mind and I hope everyone else will too. So it was really great to have you on the Prodcast today. Thank you again so much for being here and sharing all of your helpful advice and tips. And as you mentioned, you do have a book out, *Anatomy of an Incident*, and if I'm correct, you wrote that with Ayelet, right?

Adrienne: Yes, and she's the coolest, and it is available on the Google SRE website under Resources. And if you consider downloading it, my mom will be really proud of me. Please help.

Viv: We'll all download it, but pretend to be your mom, or I don't know. But yes, everybody, please take a look, and I bring it up because Ayelet is actually gonna be our next and final guest on the Prodcast to wrap us up with postmortems. So thank you again. And we will be back next time with postmortems and to close out this first season of the Prodcast.

MP: And hopefully I don't need to use anything I learned today while I'm on-call this weekend.

Viv: Ooh, yeah. But stay inside. The outdoors is scary.

MP: Thank you so much.

Adrienne: Yes. May the pagers be silent and the bears be very, very far away.