Alerting

Amelia Harrison advises on when and how to alert, ideal coverage, and tuning.

MP: Hello and welcome to the Google SRE podcast, or as we affectionately like to call it, the Prodcast. I am your host for today, MP, and here with me is Viv.

Viv: Hi!

MP: Last time we talked about monitoring, so today we have someone here to talk to us about monitoring's close sibling: alerting. Amelia, why don't you go ahead and introduce yourself.

Amelia: Hi, I am Amelia. I work on AutoAlert, a service at Google for alerting services across Google if there's something going wrong.

MP: Wonderful. So to start us off, how would you distinguish alerting from monitoring?

Amelia: So I like to think of the main distinction between monitoring and alerting as being one of timing. So monitoring is fundamentally asynchronous: at any point as a service owner, you should be able to go look at some graphs, some dashboards, and answer questions that might arise about your service. On the other hand, alerting is synchronous, so timing really matters. You wanna get alerts when something is going wrong and is actionable, and not get them when things are working as intended.

MP: In the last episode, we talked a lot about all of the different ways to get data, and we talked about—I think some of the terms that popped up were like client-side monitoring, workflow monitoring, server monitoring. There's just so much data that we have from our monitoring systems. How do I get from that to meaningful alerts?

Amelia: So I think there are sort of two sides to this coin. Like on the one side, there's sort of the service-specific aspects of alerting where you really need—as a service owner, you really need to sit down and think about the user journeys, the user interactions with your service and what matters, maybe set SLOs for your service and hopefully they're user-centric. And you want to make sure that you have alerting coverage for when those journeys aren't working as intended. On the other hand, I think there's a sort of infrastructure level of alerting that ideally is somewhat service-independent. So like, if you are using common infrastructure resources, like database storage or load balancers, or if you're using a common probing solution, then by virtue of using those resources, ideally you'll have sort of 'alerting as a service' set up for those resources to let you know when things go wrong—when your probes start failing, when you're at risk of running out of storage quota, when your load balancer starts seeing an abnormally large percentage of errors, for example.

Viv: There are a lot of things to consider

Amelia: [laughs] Yes.

Viv: Or there are a lot of things that could go wrong, I guess. It's a big world out there.

Amelia: Yeah. There are a lot of things that could go wrong. I think that—like, in an ideal world, I think the onus should be on service providers to think about not how things could go wrong, but how things going wrong could manifest as user impact, and make sure that if it is manifesting as user impact, that there's alerting coverage for that. But then, on the other hand, I think if you're building a service using common building blocks— like a common storage resource— then ideally it shouldn't be on the service provider to ensure that they've thought about all of the

ways that those building blocks can fail and have alerting coverage for those failure modes.

Viv: So you mentioned user impact there and, you know, how alerting kind of gets you ahead of users perceiving failures in your overall service or, you know, whatever the system is. I'm guessing you probably don't want to alert after users are seeing failures. So when do you alert, right? Like, when is too early?

Amelia: Mm.

Viv: And where is that sweet spot?

Amelia: I think that the answer really depends on the kind of failure mode. And also, I think it's kind of a controversial question because I think you'll find some people that would take the line. If you're alerting before you have a user issue, it's too early. I mean, ideally we'd like to avoid all outages, which is why, for example, alerting when you are approaching storage quota is a reasonable thing to do. Like if hitting quota is gonna cause an outage, then you should know before you get there.

But in a lot of cases, it's not reasonable to expect that you would be able to catch a production issue prior to it impacting users. I mean, you can do all sorts of things like having staged rollouts and having your binaries go through some sort of canary analysis or dev and staging environments. But sometimes— especially if the cause of an outage is not internal to your service, but it's because of a dependency— that's not going to be sufficient for you to be able to catch an issue before it starts.

So yeah, in summation, I think the answer is sort of, it depends. There are certain failure modes that you can reasonably expect to catch before they impact users, but there are others where that's a less reasonable expectation.

MP: What about subtle failure modes? I can brainstorm a lot of ways I might expect users to see an outage, but I always know there's going to be something that I've seen in my own experience: failure modes that are so hyper-specific and,

like, almost nearly impossible to capture in any succinct way other than, like, digging through logs.

Amelia: Hmm. Do you mean it's impossible to anticipate the failure mode, or do you mean that it's impossible to detect the way that users are perceiving the failure mode?

MP: We had signals that accurately reflected the user experience, but it was nothing that we would've ever, off the top of our heads, considered like, "oh, this is a way that this can break."

Amelia: Mm.

MP: We were able to notice it after the fact, after, like, some significant period of time, but it kind of was lurking in our data all along.

Amelia: It was the issue that the impact was just not widespread enough?

MP: Yeah. It was a very localized sort of impact.

Amelia: Yeah. I mean, it's an interesting question, right? It's like, where do you draw the line between bug and outage? Like, I don't know, how *do* you draw the line? Like, one way to draw the line is, what percentage of users are affected? How much of your target population is affected? Another one is severity. Like, how impactful is the problem? I guess, at the very least for high severity and widespread issues, I think that these are the ones that are easily qualified as outages, right?

MP: Mm-hmm.

Amelia: And you definitely want to make sure that you have alerting coverage for those sorts of failure modes. And this could be in the form of black-box monitoring with probes. This is, like, a fairly reasonable approach to detecting these sorts of outages. But I think for what it sounds like you're describing is a

situation where it's kind of in the gray area. Or was it clear? Was it very obviously an outage?

MP: I don't think it was an outage because of the particular way it manifested for the user. It was just like, a certain workflow would crash on the client but succeed on the server. And then when the user went back, the effect of the completed workflow was still there on the server side, so they never saw the workflow again. Like, it was a special one-time thing. And so there were these client crashes that were just happening at some rate and we just didn't really notice them.

Amelia: So in the aftermath of that issue, did you try to somehow formalize expectations around the workflow, like put an SLO on it, or...?

MP: Not me specifically, but some counterparts that I was working with added some extra monitoring and alerting around that specific workflow.

Amelia: Yeah. I mean, sometimes it's just a matter of, users use your services in unexpected ways. And so your notion of what the critical user journeys for your services are may be not really accurately reflecting the way users are using and interacting with your service. And so, I guess, kind of what this is getting at is that even if you haven't explicitly set an SLO on a certain way that a user interacts with your service— if a bunch of people are doing it and it's, like, normally working, then there is an implicit sort of expectation there. There's almost an implicit SLO, right? And I guess that these can be the ones that are issues— these implicit SLOs are the ones that can be really hard to get ahead of because you haven't explicitly defined an objective for the service with respect to that interaction.

Viv: But I do like the thought that you can. I think it's interesting that, you know, alerting probably changes as maybe your service evolves or users evolve in terms of the ways they use things, right? This is maybe only tangentially related, but I thought of this because I think we mentioned bugs versus outages. And I guess another question is, who gets the alerts? Maybe, you know— is it always SRE? Is it the developers sometimes? And I guess, are these pages, are these tickets sometimes? And is it bad to get a lot of frequent tickets to try and catch a bunch

of things, but only get, like, really important pages for outages? Or what is your opinion on all of that balance?

Amelia: Yeah. Yeah. It's a really good point. I guess we should talk a little bit about what we mean by alerts, right? I don't think of alerts as just being pages, but I think of them as being any automation generated—so any automatic monitoring-generated notifications. It could be bugs. It could be emails. It could be pages. I don't know. It could be like a notification in an IRC or something. The delivery mechanism doesn't really matter as much. And I think that ideally, if you have a paging alert, it's really important that it be urgent and that it be actionable. So reflecting an issue that needs to be addressed quickly, and that can be addressed by the person who's responding. On the other hand, the mechanisms that my team uses the most are either paging or tickets.

MP: I feel like email alerts are where alerts go to die.

Amelia: [laughs] Yes. They might as well not exist, probably.

Viv: I have gotten forwarded some email alerts before. They're... interesting.

MP: This just turns into email filter hell.

Amelia: Yeah, definitely. So one of the things that we've struggled with on my team is that there's sort of an interesting line between what we mean by urgency. If the dichotomy is: you either have paging alerts or you have ticketing alerts, then how do you decide what's urgent enough to be paging? And the criteria that we've used is, okay, if this alert fires on Friday evening at 5pm, will you have enough time to react and prevent it from becoming a major outage in the next 48 hours? Like, is it actually okay if it waits till Monday? So you need that 48-hour window. So urgent doesn't necessarily mean it needs to be handled this minute, but it means we cannot wait for the dev team on Monday. It may be that there's a few hours before it actually becomes a problem and maybe there's 12 hours, but if it's less than 48, then it probably should be paging because it can't wait the weekend, essentially. Did that answer your question?

Viv: Yeah, sort of, I think. And so I guess in that scenario it would mean that maybe you'd get a ticket, and then if the circumstances were to change with whatever it is, then your team might get paged anyway, or, you know, maybe that page would never happen and you would just come back to it in 48 hours.

Amelia: Yeah. Yeah.

Viv: I get nervous when I get tickets sometimes because I'm always like, is this the precursor to a page? You know, does this mean that I need to be acting fast to avoid like we said, large user failures, et cetera. But I think that being in the philosophy of "this is a ticket because of that" is interesting and is a good point.

MP: I feel like there's a weird class of ticket that I've encountered where it's disruptive— internally disruptive— but not an outage where it's like, you should probably deal with this as soon as you can 'cause it might make people's lives difficult, but it's not actually an outage. Users will never notice that this particular thing is broken, but you're gonna make some other people's lives difficult— I have a very specific example in my head— and failure to act increases risk later.

Amelia: But you're still talking about, like, an automated alert, not like a—?

MP: Yeah, yeah. But this is definitely, like, ticket-level in your bug queue.

Amelia: Yeah. And so the question is like, what do you do with these? Like, is it appropriate that it be ticketing versus—?

MP: Yeah, well, I know the fact is, if the justification I would use for bringing a ticket was that if I was to get one at like 7pm, that's not getting looked at until the next morning and more or less people can wait until the next morning. And it can— it could definitely wait a weekend, too, in this case, 'cause mostly it's internal workflows [that] are disrupted. It's not something that anyone's gonna really be bothered by on a Saturday or a Sunday.

Amelia: Yeah. I think that one of the things that this brings up is that really a lot of these decisions are service-dependent. There's not, like, a good generalized

answer. There are rules of thumb and sort of frameworks for thinking about what urgency is appropriate. But ultimately it really sort of depends on your service. And one of the things that can inform it to you is the, like, existing pager load. If your on-call rotation includes a whole bunch of different services and the load is very high, then it might not make sense to add these sort of preemptive, "nothing is broken right this moment, but it will be broken in 36 hours if nobody takes any action." It might not make sense to add those to the pager load. Hopefully, nobody's in that situation. And probably there's some more fundamental rethinking of how to increase the service reliability to bring down the pager load so that you can catch those things earlier.

MP: I think at this point I wanted to ask: if you could have your perfect alerting system just exist, how would you sort of structure that? What sort of a thing would that look like?

Amelia: I think that the major difference between the current state of the world and the perfect alerting system in my mind is that I would really like to see something that's more data-driven. I think that a lot of the existing approach to alerting, it often looks something like: you're a service owner, you think about what's important to your users, what their critical journeys are, you set up monitoring and alerting for those journeys, and then the alerts start to fire and you're like, "oh, you know, this didn't indicate a real problem, this was too noisy," and you tune the alerts. Or you miss an outage and you say, "oh, you know, we had monitoring, but our alerting just wasn't sufficiently sensitive to this failure mode." So you add an alert on that monitoring or you make the existing alerting more sensitive. Your team and every other service support team or every other SRE team is going through this exact same exercise of tuning alerts, making them more sensitive, less sensitive. Sometimes it's the same alert, you know. Sometimes it's like, "oh, too sensitive; not sensitive enough; haven't found the sweet spot yet."

MP: This all sounds painfully familiar.

Amelia: [laughs] Yeah. The problem is that individually exploring the threshold space, you know, the parameter space for an individual alert, we're not able to

make really data-driven decisions about what good alerting looks like. And we're not able to leverage other people going through the same exercise to sort of converge on good alerting. So I think ideally, you have a sort of system where you have alerting as a service and you're able to set various SLOs for your service, but beyond that, you don't really have to think about the details or tune the parameters for the alerting to make sure that you're able to uphold those SLOs. That can be handled separately by the alerting service, right? And that service will have visibility into alerting across a whole bunch of different services and, as a result, is able to leverage all of the data that you get from that sort of high-level visibility.

Viv: So perhaps from seeing all these examples across various services of when an alert isn't sensitive enough, it would know to maybe tune it, I guess... is that odd? I don't know which direction we're thinking, but, you know, something like that.

Amelia: That's right. And also it can learn about common failure modes, like sometimes alerts fire and it's a false positive that has to do with the way that the monitoring data is stored. It's not even related to the threshold, but it's like a blip in the monitoring in itself. Or it's, like, an issue with the monitoring repository, right??aybe there's a problem with your time series database or something. And so everybody gets a false positive alert because data got dropped. I don't know.

Viv: I know that one.

MP: Yeah, I know. I know I saw one where it was like a particular export server wasn't working and I'm getting pages and tickets for half of the services I support and I'm like, "this is wrong."

Amelia: Right. And so is everybody outside Google. When you have a centrally-provided alerting as a service, then that team has the resources to think about that common failure mode and make sure that the alerts that they're developing are robust to it. Versus every single team having to discover that for

themselves and then tune all of the alerts or make modifications to all of the alerts that they own to make them robust to this monitoring failure mode.

Viv: So when users would initially use alerting as a service, you mentioned that they would set up or agree on a service level objective. Would that mean that as, you know, alerting as a service adjusts, they would suggest perhaps a new SLO based on what they're actually seeing?

Amelia: Yeah, absolutely. I think that that's another thing that makes alerting as a service particularly attractive— is that we talked a little bit earlier about implicit SLOs. We know that customers are gonna use our service like this and that we want to provide this level of experience for them. If you have alerting as a service, then you can allow users to specify— or users, in this case, is service providers— you can allow service providers to specify their objectives or not. You know, maybe they just specify their SLIs— the indicators that they know are important indicators of service health— and then the centralized alerting service monitors those indicators and knows what the usual performance is and can let you know if there are severe regressions.

MP: That reminds me of one of the things that kind of always is sticking in the back of my mind is that worry about that alert I wrote three years ago and whether or not any of those thresholds are still meaningful. And every, like, alert you add creates more work, 'cause at some point, you have to go and make sure that alert still works.

Amelia: Right. Right. Absolutely. I mean, assumably if they're not meaningful in the sense that they're way too tight, then someone else has come along and turned them down. But if they're not meaningful in the sense that they're too loose, then maybe you're really missing big problems with your service and you just don't know.

MP: Yeah. I'm thinking specifically in the case of that implicit SLO and like, you launch your service and you're like, "okay, we're gonna start at two 9s" and you launch your service and you set up everything for two 9s and then you spend the next year working on availability improvements. And now you're running at four

9s but your alerting is still stuck at two 9s and now users have come to who expect four 9s and now your alert's not gonna fire until it's really bad.

Amelia: And if you drop down to two 9s you're gonna have to find out about it from users, which is not fun. Right?

MP: That's the stuff that gives me nightmares.

Viv: It's hard. There are a lot of things to keep track of. I mean, obviously, that's why there are other automated things that help us with this. But even still, there's a lot of challenges in making sure you're up to date with everything. How far away do you think—sorry, this is kind of jumping back, but—how far away do you think we are from your ideal alerting as a service?

Amelia: Definitely at Google, we have strong proofs of concept that this can work. We have a foothold, and I think we're moving in the right direction. We also know what doesn't work, or we have some good ideas about what doesn't work. So, for example, I think what we've seen is that generalized anomaly detection for alerting doesn't generally work. And I think part of the issue there— or maybe it could work, but what we've seen so far is that it is not very effective. And I think part of the issue there is that metrics are not created equally, right? There are important service level indicators that really tell you something about the overall service health, and then there's a whole bunch of other metrics that may be useful for monitoring that you're probably retaining somehow, but anomalies— they are likely to just lead to wild goose chases and be a distraction.

MP: Like the inherent noise of the signal is too high for you— like for anomalies to actually be a meaningful concept is, sort of, what I'm hearing.

Amelia: Right. Or even just the signal just doesn't matter. Like, it just doesn't tell you that much useful about your service. So I think that you'll never really get away from the need for service owners to have input about what they care about, right? Like, service owners identifying the important metrics that really reflect the health of the service. I think that we've seen that we're probably not going to be able to get away from that and also instrumenting the service with those metrics,

right? Because there's just, I think, in general, not a one-size-fits-all solution to that problem.

MP: So I think on that note, if no one has any more comments or anything, I think we should say thank you to Amelia for her time.

Amelia: So happy to be here. Thank you for having me.

MP: It was a pleasure.

Amelia: Likewise.

Viv: See you around.

Amelia: Bye.