Deloitte.

For Cloud Professionals
June 2021

For Cloud Professionals, part of the On Cloud Podcast

David Linthicum, Managing Director, Chief Cloud Strategy Officer, Deloitte Consulting LLP

Title: **Fail forward and inject chaos to improve cloud performance**

Description: Cloud operational complexity often breeds failure points in the system. How do you analyze failures and stop them from reoccurring? With incident management and chaos engineering, which help companies understand the human and technical causes of incidents and better prepare for them—before they become disasters. In this episode, David Linthicum talks with Jeli's Nora Jones about how companies can use these two interrelated disciplines to improve performance and achieve higher cloud ROI.

Duration: **00:26:18**

Operator:
This podcast is produced by Deloitte. The views and opinions expressed by podcast speakers and guests are solely their own and do not reflect the opinions of Deloitte. This podcast provides general information only and is not intended to constitute advice or services of any kind. For additional information about Deloitte, go to Deloitte.com/about. Welcome to On Cloud, the podcast for cloud professionals, where we break down the state of cloud computing today and how you can unleash the power of cloud for your enterprise. Now here is your host David Linthicum.

David Linthicum:
Welcome back to the On Cloud Podcast, your one place how to find out how to make cloud computing work for your enterprise. This is an objective discussion with industry thought leaders who provide their own unique perspective around the pragmatic use of cloud-based technology.

David Linthicum:
Today on the show we are joined by Nora Jones. Nora Jones is the founder and CEO of Jeli. She is a dedicated and driven technology leader and software engineer with a passion for the intersection between how people and software work in practice in distributed systems. And she created and founded the

LearningFromIncidents.io website movement to develop opensource, cross-organizational learnings and analysis from reliable incidents across various organizations and the business impact of doing so. Wow, Nora, drill down on that a bit. What's that all about?

Nora Jones:
Sure, yeah. Hi, David. Thanks for having me. So, I've spent my career in site reliability engineering. I worked at Netflix, Slack, and Jet.com. As I was working at those places, I noticed a big need for understanding and breaking down patterns that we're seeing in our incidents and how much of a world it can open up to our organizations if we understand it a little bit better. It can point to things like, wow, this team is understaffed over here, or quarterly headcount, financial budgeting—it can point to all kinds of things if we're looking at it under the right lens. And I started looking at incidents under a different lens while I was working at Netflix and working on the chaos engineering team, because I realized they could be a catalyst to understanding what's important to the organization and where things kind of need work.

So, that's why I founded the LearningFromIncidents.io website, because I honestly thought it would be beneficial if organizations throughout the software industry were sharing some of the problems that they were having. I had experienced the same console issue at Jet.com that I experienced at Netflix about six years later, and we were misusing it in similar ways. And I don't understand why the industry needs to be so tight-lipped about some of the incidents that we're having and certain reliability practices when we could all benefit from a little bit more. So, what's why I founded the LearningFromIncidents.io website. It's about 300 of us from all different companies throughout the industry sharing our incidents and what we're learning from them and how we're growing our organizations afterwards, and it's been really great.

David Linthicum:
Yeah, one of the things I was thinking about as we're getting into the world of AIOps and the ability to kind of automate operations through intelligent systems is, what if we had the ability to share the knowledge? I mean, there are certain patterns and certain things that we see that are leading up to outages and incidences in terms of things we can correct on the fly and become more proactive moving forward, but we don't share anything. And it really is a shame because, ultimately, we do share health information that's anonymized, and therefore we can determine whether someone's going to have a heart attack or not. But we're unwilling to share information which determines whether a server will go down or not. Do you think that trend is going to end?

Nora Jones:
Yeah, I'm certainly hoping so. I think it's going to benefit all of us. Obviously there—I understand where some companies are coming from. They might be reluctant to share some of their incident information because of how it might make them look, right? And I'm sure we've all been parts of organizations where we've had something really bad happen and no one's noticed, right? Why would you want to highlight that kind of thing to the world? But you're right. We can obfuscate some of these things and they can be helpful and shareable and we'll all level up as an industry.

But the software industry itself is still kind of new compared to other industries that have incidents. If you think about like aviation and healthcare and maritime, and all those industries have been studying accident investigation for years, and we actually have a lot that we can learn from them as an industry. And I see us kind of reinventing the wheel in a lot of ways in software on how we're trying to surface patterns and learn from incidents. And it's not only that. I think we end up being tight-lipped about them internally within some of our organizations, too, when really it's actually a nice way of leveling up expertise if you can surface some of those patterns. It's a really nice way of onboarding new employees and seeing where some of your weak spots are internally.

David Linthicum:
Yeah, I couldn't agree more, and I think we've got to be better at sharing information moving forward. And I think ultimately that'll lead to more reliable systems. So, what is your core work at Jeli? Why did you found the company and what do you do there?

Nora Jones:
Yeah, absolutely. So, Jeli is the first dedicated incident analysis platform, so we help you analyze your incidents. So, one of the things I was realizing as I was getting into the world of incident analysis a little bit more was that the barrier to entry was quite high. It's a whole other sector of work where we have to analyze and learn not only technically what happened, but socially what happened. And I realized the benefit that companies could have from getting some of this information, but how a lot of them were probably unwilling to invest in hiring like an actual dedicated incident analyst.

And, so, I saw a real need for it in the market, so that's what Jeli does. We help you aggregate disparate sources that occur during your incidents, like Slack, and we aggregate that all together to help give you a cohesive picture of what happened in your incident so that you're not starting from zero. And we're directing your attention towards things that need to be improved upon, need to be worked in, yeah, that kind of stuff. And it really got born from a lot of the work I was doing at Netflix in the chaos engineering space. I was working on tooling that allowed engineers to safely inject failure into production and see if it impacted the customer experience. And it was really amazing to work on that tool. We were able to successfully mitigate the blast radius where, if we had injected a failure that did have customer impact, the customer wouldn't notice because it would route them back to a normal experience within a second.

And, so, that tooling was really, really great to work on; however, we needed to prioritize some of the stuff we were finding. We were obviously finding bugs with it, we were finding issues, and we didn't want to create just this giant backlog of here's all the things we need to fix when other people need to work on features, too. And, so, we were like, "We need how to figure out how to prioritize some of these things." And, so, we started looking at incidents as a way to help prioritize and set context for the importance of some of the bugs we were finding, right? Like how often has this system bit us? Are we relying on certain people too much to fix some of these things? Are there incidents that have surprised us recently? Are there certain times of year where we have certain types of incidents? All that data was used to help prioritize not only these chaos engineering bugs, but it opened us up to a whole new world of other stuff that we could prioritize, too, amongst our features, amongst our OKRs. And, so, yeah, I wanted to open that up to people besides Netflix, so that's where Jeli came in.

David Linthicum:
So, this information, this data is coming off of humans as well as machines or just machines?

Nora Jones:
No, it's focusing on the social sides as well. So, we're surfacing things like, hey, these teams have never worked together before and they're working together for the first time on an incident. I think a lot of the time organizations are apprehensive to look into the social nuances because it can be a bit awkward, but you end up missing out on a lot of your action items and a lot of your action items after incidents end up being quite shallow because you're focused way too much on the technical, right?

Like, here is the exact mechanism that broke, rather than focusing on, like, wow, this person was on call for 48 hours, or wow, these teams have never worked together before until they were in the midst of this incident, or everyone in our incident right now has been at the organization for like five months. And, so, those kinds of things are really hard to surface, but Jeli is trying to direct your attention to how you can organizationally fix some things as well so that you can direct your attention better towards the technical, too.

David Linthicum:
So, who would technically be your customer at Jeli?

Nora Jones:
Yeah. Are you talking about which kinds or organizations or which users internally?

David Linthicum:
Both actually. What kinds or organizations and who in the organization would leverage your services?

Nora Jones:
Yeah. I mean, we're working with a lot of different kinds of organizations right now. Honestly caring that you have incidents and understanding that there's probably more that you can do with your post-incident reviews—I think a lot of folks are letting incidents go to waste. It's a very expensive thing that you already spent a lot of money on; you might as well get some ROI off of it. And, so, that's what we're doing. We're helping you get that ROI off of an investment that you already made.

But the organizations we're working with today, we have some, like, series B companies we're working with. We're working with thousands of people, organizations. It's all really organizations that are caring that they have incidents. They have customers. They might owe their customers RCAs sometimes. And, so, those are the folks that we're kind of finding a sweet spot with today.

And then internally who we're working with, we're working with the folks that own incident processes internally. We're helping them out. We're helping them get some of this work in the hands of more people so that the burden is not all just on one person and that they can manage the process a little bit better, which also helps the relationship with some of senior management as well, because they can see the ROI that they're getting out of it. Through investing this effort in individual incidents, they can see how they're doing and improving over time, too.

David Linthicum:
So, in researching you online, you've done a lot of talks on chaos engineering, which I kind of found fascinating because I don't think the audience knows a ton about that. So, what is chaos engineering? And what are some of the things that enterprises need to understand about the concept?

Nora Jones:
Yeah. Chaos engineering is effectively intentionally injecting failure into production systems. So, the story I was telling about with Netflix earlier, that's exactly what we were doing. So, I'll give you an example. At Netflix you'd come up with a hypothesis, like I want to inject failure or latency in between these two service calls. So, say we wanted to run a chaos experiment on the bookmark service and inject some failure into it. So, the bookmark service at Netflix essentially remembers where you left off on your stream, like, oh, David left off at minute four, and it will take you back to that minute afterwards. And, so, the hypothesis was if we fail this service, it's not going to render Netflix unusable. People should still be able to view Netflix. It should have a fallback that maybe redirects them to 0:00 or something like that. So, that was the hypothesis of that particular experiment. We'd inject failure, we'd break some of the customers into a particular experimental group, and then we would make sure that they still had a good experience.

But what we learned from that particular one was we sent them back to 0:00, and so all these customers are trying to figure out where in the stream they exactly let off. And, so, we never would run experiments that we felt were actually going to impact the user in a poor way. We would run experiments that we thought would be good to go and the user would still a good experience, but what we would learn sometimes was that they wouldn't and so we'd learn a little bit more about how the users are using our systems. And, so, it was very nice for SREs in the fact that they were getting their mental model refined on some of the stuff that they were working on. So, that's the gist of chaos engineering.

And then the learning from incidents stuff that I mentioned plays into that because it helps prioritize some of this work. It helps you find some of the patterns between some of the failures that you're experiencing.

David Linthicum:
So, how can we, if we're in the world of cloud and we're dealing with operational incidents, which seem to be a large focus right now as we're getting into more complexity, building multicloud systems, integrating systems with some of the on-premises systems. Getting in these very complex, very heterogenous distributed systems moving forward, how would we interject chaos engineering and also incident management into doing a better job at running those things?

Nora Jones:
Could you give me an example of like an exact scenario you're kind of thinking through?

David Linthicum:
Absolutely. Say I have a Google Cloud, Amazon cloud, Microsoft cloud and I'm running 15 applications in each. They're all leveraging different programming languages and different databases, things like that. And, so, I'm abstracting my operational systems over the clouds instead of leveraging native capabilities

within the clouds, and obviously incidents are being brought up. People are operating those systems. I have different DBAs, different human beings that are running those things. And, so, I'm in charge of everything and I have people who are working with me who are in charge of particular native instances, databases, storage systems, applications, things like that.

So, in, I guess, looking at incident management stuff, the ability to kind of make hay, or make sense out of the things that are coming out of that environment whether it's machines are telling me something and the human beings are telling me something, or basically just looking at how we're doing this stuff. How do I improve my operations over time leveraging incident management with that kind of scenario, a complex mess, lots of people are involved, and it's something that there's typically a lot of turnover? It's a brand-new technology. People are still cutting their teeth on it.

Nora Jones:
Yeah. Yeah, absolutely. I mean, are you—my first question would be are you having incidents today? Are you having surprises? Are you having any near misses with operating some of those things?

David Linthicum:
Yeah, that's a great question. I mean, say the database went down and no one knows why, and of course there's a thousand applications that are connected to it and those broke, too.

Nora Jones:
Yeah. Yeah, and it's probably going to take a really long time to figure out why. And, so, it's a great question. Like, first off I would give folks time and space to figure out why and how that happened without kind of the pressure of finding out those answers quickly, because that kind of stuff is—that process of learning for folks ends up being super, super valuable to organizations. But I get it, though. There is a time constraint. Like, we need to put an end date on it, but first off, I would make sure we're giving people the time and space to actually understand how these mechanisms worked together to make this even occur, right? And, so, that's one part of it.

The second part of it is when you do figure that out and you do learn some of those things, recording it, documenting it, making sure you're sharing that far and wide throughout the organization and you're actually disseminating those insights. And then that's when you can start to—when you actually have a good handle on understanding how something happened, that's when you can really start to implement chaos engineering. I've seen way too many organizations fail at chaos engineering because they do it before they have that good understanding of how. And there could be an argument like, well, doing chaos engineering can help us understand the how, but implementing chaos engineering safely is not going to work well if you're not taking time to understand what's already broken. And, so, it can give you some more insights, but you have to have some of those safety barriers in mind.

And, so, with that particular database example, I would make sure that you're giving folks time and space to actually figure out how things broke down, figure how people worked together, and share that knowledge and celebrate honestly sharing that knowledge internally. And then that's when you can kind of move to the proactive a little bit more and set up some regular experiments on that particular database, come up with different failure scenarios.

I also like to run pre-mortems. Like if this thing failed, how did it break, right? And ask folks that, and set them up in a room, like, hey, we're deploying this tomorrow. It breaks. What made it break? And opening up that space for everyone to share with each other, like, what made the thing break in this pretend scenario actually causes people to learn a lot about what other people's mental models are holding. And it's really important to do that in a room with a bunch of the folks working on it, too. So, all these things relate to each other and play together and are honestly symbiotic.

David Linthicum:
Yeah, and what I really like about this is we're dealing with the human aspect of this, which is more important than people think.

Nora Jones:
Exactly.

David Linthicum:
Most of the system failures and breaches that I've dealt with in my career I can trace back to human issues, not necessarily making mistakes and who to blame on the mistakes, but just a combination of things that occurred where it's really nobody's fault, but it's still in the end conclusion that there's a breach or an outage or something bad happens. Lack of communication poor leadership in many instances come down to this. And the reality is I think as we move forward and try to become excellent at operating these systems and try to do so and more optimized and efficient ways so that we are reducing the downtime, the humans become kind of the biggest part of the thing that we need to fix. I mean, like I always tell my clients. I'm good with the technology. I can do anything you need done with the technology. The tough part is culture, humans, processes, things like that that we have to rely upon moving forward, and the ability to kind of get those things in line with the machines is really the larger problem to solve. Am I off-base?

Nora Jones:
I get where you're coming from. I would flip it a little bit, because the humans are actually the ones making that technology work. They're the ones enabling the system to work. And, so, if we instead flip the script and think of them as something we need to fix, instead think of them as something that we need to enable and help strengthen and help understand what makes things hard for them, that's going to open us up to all kinds of different improvements that we haven't seen before. I think a lot of the time in post-incident conversations and stuff I'll see some normative language like, oh, lack of runbooks or lack of experience or lack of training, and that always smells to me like a starting point. Like, there's always a much deeper story there, but yet that's where I see so many of them end. And I think honestly it's because we're engineers. Like, we're not trained on, like, cognitive questioning or how to surface different answers out of people.

But the questions I like to ask—like, I always like to talk to the person who if it seems like they might get blamed or if at first glance it seems like it might've been their fault, I will usually talk to them, like, in a one-on-one setting and just kind of open up and say, "From your perspective, can you tell me about what happened?" Not in a way that they're going to get fired or anything like that, but just in a way, like, that they're holding this expertise. There's something that made this incident really hard for them and we need to surface that, because if they did something that seemed silly and we don't understand how and why it made sense for them to do that thing at that time, someone else is going to do it again in the future under a different flavor.

And, so, I'll ask them to tell me about what happened, how they got notified, what dashboards they were looking at, how they felt it went, what their comfortability was, how long they've been at the organization. I'll ask them to tell me what they think is important for me to know. I've learned so many things that way. I've been in organizations where honestly folks don't want you to talk to these people because they don't want them to feel bad almost, right? But it's important to open up the conversation because they are holding onto some knowledge that you need to unearth that's going to be really hard to unearth in a group setting.

I talked to an engineer before where it was a particular incident that they responded to at like 3:00 in the morning, and no one really—the next day everyone was like, "I don't know why they responded to that page. That was something that could've waited till business hours when everyone was in the office, and this caused our entire website to go down." And, so, everyone's just kind of hush-hush whispering, like, if it was a more experienced engineer, they would've known it could've waited until morning. And, so, I went and talked to that person and I was like, "Hey, can you tell me about the incident?" They're like, "Yeah, I was up debugging a separate issue, up until 2:00 a.m. that I had gotten paged for, and then I finally feel back asleep and I got paged for this issue."

And I was like, "Wow. You know, this guy was on call for two separate systems in the middle of the night. No wonder he answered this page. He must've been super tired." And then I asked about the system he got paged for at 3:00 in the morning and he said, "Yeah, my team had just gotten handed this system a month ago. I knew it had some tricky nuances to it, but I'm not fully ramped up on it yet." And his whole team was under the exact same scenario, and it made me able to look into how on-call transfer of expertise worked in this organization and it was kind of a hot potato. It really wasn't this engineer's fault. He was a product of the system that was being built.

And, so, as a result of just talking to him and understanding why some of these things made sense, we were able to improve on-call ownership and on-call handoffs. We were able to improve, like, knowing if someone was tired all night and just automatically handing their shift to someone else. We were able to improve a lot of things that, had I not talked to him, I bet someone would've just implemented a gate or just updated a runbook, and that wouldn't have been the answer there.

David Linthicum:
Yeah, and even some stuff comes back to leadership responsibility.

Nora Jones:
Exactly.

David Linthicum:
When I was running cloud companies, we had outages from time to time, and those were the days when we owned the infrastructure. And it came back to having core processes and the ability not to manage the operations or the systems in such a way which led to the outage. It wasn't the talent and the responsibility of people who were monitoring this stuff and doing the manual processes. It was leadership, in this case me. *[Laughter]* So —and it's tough to fall, but the thing is if you're unable to recognize that there's a problem, you can't correct it.

Nora Jones:
Right, exactly, exactly. And, software has so much to learn from other industries here. Are you familiar with Captain Sully and the movie *Sully*?

David Linthicum:
Yes.

Nora Jones:
Yeah. So, that whole situation—he didn't totally follow the rules, right? He—but no one died, right, and so he was kind of celebrated, whereas there's this other famous case, the Costa Concordia, where—it's a maritime investigation and I would recommend looking into it. I think it offers us a lot to learn from other industries here. But the captain kind of did a similar thing where there was a runbook he was supposed to follow in an emergency situation and he assessed the situation and he deemed the runbook inappropriate, right? Which is what experts do in a lot of situations, except in his case people died. And this captain is in prison, whereas Captain Sully is being played by America's beloved Tom Hanks in a movie.

And, so, it's like—it's a very different situation, and I think software kind of experiences similar things where, hey, why didn't this engineer follow this runbook, right? And, so, we focus too much on some of these situations rather than focusing more on the expertise. And the two captains I'm talking about in these situations were absolutely in the right. The maritime captain, had he not deviated from the runbook more people would have died, but less people died because of deviated from that.

And I think in software, like if we over-index too much on what the humans are doing right and what the humans are doing wrong, we're going to end up with people that are following runbooks to a T too much because they're scared, right? And we're going to end up with really shallow actions items after incidents, like just update the runbook rather than focus on building up someone's expertise so they don't feel like they to rely on the runbook to a T. Yeah, and that's really what we're doing at Jeli, is helping celebrate the human and helping understand what made things hard for them in the moment so that, like you said, David, leadership can help make these changes, so that their peers can help make these changes, so that they feel like they have the space to actually learn and grow as an engineer, which honestly gives so many benefits to the business, when the business takes this shift in this stance.

David Linthicum:
Yeah, I'll tell you what, humans and computers have to come together in this stuff or it's going to be a pretty rough ride moving forward. So, where can we find more about Jeli on the web and yourself?

Nora Jones:
Yeah, so Jeli.io is our website. We are pre-general available right now, but we are in a closed beta with our initial product, and so we have a few customers using it right now and having a lot of great experience with it. But we are taking on a few more before we do go general availability.

David Linthicum:
I wish you the best of luck. It sounds like an exciting project. So, if you enjoyed this podcast make sure to like and subscribe on iTunes or wherever you get your podcasts. Also don't forget to rate us. Also check out our past episodes including the On Cloud Podcast hosted by my good friend Mike Kavis and his show Architecting the Cloud. If you'd like to learn more about Deloitte's cloud capabilities, check out DeloitteCloudPodcast.com, all one word. And if you'd like to contact me directly you can reach me at DLinthicum@Deloitte.com, L-I-N-T-H-I-C-U-M. So, until next time, best of luck with your cloud projects. We'll talk again soon. You guys stay safe.

Operator:
Thank you for listening to On Cloud for Cloud Professionals with David Linthicum. Connect with David on Twitter and LinkedIn and visit the Deloitte On Cloud blog at www.deloitte.com/us/deloitte-on-cloud-blog. Be sure to rate and review the show on your favorite podcast app.

Visit the On Cloud library
www.deloitte.com/us/cloud-podcast