EPISODE 1197

[INTRODUCTION]

[00:00:00] JM: Distributed application, observability, is key to handling incidents and building better, more stable software. Legacy monitoring methods were built to respond to predictable failure modes and to aggregate high-level data like access speed, connectivity and downtime. Observability on the other hand is a measure of how well you can infer the internal state of a system from its outputs in order to trace the cause. At its core, building a system with observability means using instrumentation to provide insights on how and why internal components within a system are performing a certain way. Developers and SREs can build on that data to proactively debug potential failure modes, set service level objectives and speed up incident response.

New Relic has been an industry leader in the observability space for the better part of a decade. This year they announced a new evolution of their flagship system, New Relic One that streamlines and simplifies the functions available to help organizations achieve observability. And Lew Cirne is the founder and CEO of New Relic. He joins the show today to talk about how New Relic One helps developers move beyond monitoring and embrace observability and how he sees the future of software observability platforms.

I am investing in infrastructure and developer tools companies. If you're building something cool, send me an email, jeff@softwareengineeringdaily.com. I would love to hear from you. Also, if you want to support the show, you can go to softwaredaily.com and become a paid subscriber. Paid subscribers get access to all of our content without ads. So if you want to listen without ads, go to softwaredaily.com and become a paid subscriber.

[INTERVIEW]

[00:01:47] JM: Okay, Lew, welcome to the show.

[00:01:49] LC: It's great to be here. Thank you.

[00:01:51] LC: You started New Relic a while ago, and before that you started a different company, Wiley. Both of these companies were focused on what we now call observability. How are software applications today different than from when you started New Relic?

[00:02:10] LC: Well, yeah. I'll talk about what's different today, but also talk about what's the same, and I guess what's the same as so long as there is software there is going to be bugs and there are going to be problems that happen only in production and that'll be true forever I think so long as humans create software. And just like in the medical field, so long as there are people who get sick, then there's going to be a need for doctors. And so I think so long as there is software, there's going to be need for tooling and visibility and capabilities to help understand how software behaves when it's running under load in particular and take that understanding to improve the performance, availability, stability and the customer experience of that software.

So when I started Wiley 23 years ago, the idea was this brand new thing at the time in ‘98 was Java and the idea was let's see inside a JVM without asking our customers to change any of their source code and put that visibility into production at low overhead and capture as much data as possible and present it in an easiest way possible to help customers debug their JVMs. And fast forward to 2008 when I founded New Relic, the thought was, “Well, it's a multi-language world now and applications aren't running on two or three physical servers.” They’re running on 20 or 30 or so back in that time, virtualized hosts. And it was very early in the cloud, but those hosts were running increasingly in new environments like Amazon Web Services. So the idea was how do you put visibility into applications that are composed of, say, a half dozen services running in a virtualized environment where there's multi-language? And that really was the sweet spot of New Relic when it was founded sort of first several years.

And now here we are in 2021 and what an application looks like today is often hundreds of services, thousands of containers, more and more in Kubernetes, incredibly complex, a lot of asynchronous work, a lot of stuff going on in systems like Kafka. And so trying to make sense of a really complex system is more challenging than ever. And it seems like what's behind all of this complexity is the imperative to help developers be more productive to, first of all, have smaller, more independent teams who can you know deploy what they're responsible to with pretty good isolation and rely on good APIs and things like that to allow lots of those teams to collaborate on a large effort at high velocity. So that increases velocity. But it comes at the cost of increased complexity on how that whole integrated system works.

And the solution to that in our opinion is complete visibility into all the application microservices, all of the infrastructure and the end user experience all into a common platform that operates at massive scale. And really the guts of observability, if you want to understand the difference between observability and monitoring, I'd say monitoring is about telling you when something's wrong, but observability is having access to all the telemetry data you need to answer why is something wrong, which is you don't even know what question you need to ask next to get to the understanding of what's wrong. And in today's world, it just involves like collecting a massive amount of data and trying to make sense of it as rapidly as possible.

[00:05:39] JM: So there are a vast number of platforms, companies that have been built around monitoring and observability. And a lot of them have stayed niche or they've stayed small. They haven't scaled, and that's fine. I find it really interesting that you've been able to build two fairly large companies around this set of problems when a lot of other companies have stayed smaller. Is there some set of strategies or some philosophy that has allowed you to scale New Relic and previously Wiley where other companies stayed smaller?

[00:06:20] LC: I think it's always really important to understand that our customers are typically developers and operations people who are so darn busy and they've got other priorities that tend to contend for their time and are often are more important than spending a lot of time configuring your production telemetry and monitoring systems.

So obsessing on how to make it as easy as possible to get going is so important. Every time you add a configuration flag or a step in the setup process, you're making life harder for the customer who's got so many other things that's contending for their time and energy. And so in the early days of Wiley, it was very controversial to suggest that you'd have this startup that would offer an agent that would literally modify the bytecode of your software and other software that you were running. So it could be open source or it could be a commercial application server, but you were going to deploy this thing called an agent that was going to literally modify the code, which for understandable reasons, as a brand new technology, a lot of people were uncertain that they'd wanted to do that. But once they tried it, they were amazed at the benefit of, “Hey, I get immense visibility with virtually no effort.” And I think that's the key to it, is just try to make it as easy as possible for your customer to get to all the visibility they need, to answer questions they didn't anticipate they might even have. And then I think there's a moment of joy actually for technologists and when they discover something new about their software. And at that moment they often want to share it with a colleague, say, “Hey, did you know that it could be something as simple as like we're doing a table scan on this query and that's why this page is taking six seconds to load, and one quick index, and that could be lightning fast.” And like there is that moment of accomplishment when that data turns into an insight and then turns into a better customer experience. And where the flywheel builds is when that customer experience is so great and joyful that they tell their friends about it. And so I think we're at our best when we deliver software that has those characteristics.

[00:08:36] JM: Tell me more about your position as management of the company. How do you manage a company through a large growth period? A large long growth trajectory like you've done at New Relic? Do you have any particular lessons that come to mind about how to manage a large software company?

[00:08:56] LC: Well, some people think there are playbooks to it, but I think it is so dependent on the individual. Probably the common thread amongst it all is self-awareness. Being comfortable with your weaknesses as well as your strengths and knowing what they are. And so that it obviously informs who you need to have around the table that can complement for your weaknesses and also your passions so that as a group you can do amazing things that no individual can do on their own.

In my particular case, I’m a technologist, I love creating things. It brings me great joy. And so I love working with engineers on innovation and on building things that hopefully have transformative impact for our customers. I also love storytelling with our employees or our customers. But I don't have a gift for other parts that are really important in leading a company. And so I need to hire for people that are great at leading large cross-functional meetings to make sure that the various teams in the business are working in alignment with the company's strategic goals, right? Those are important things that need to be done. And there are some CEOs that are excellent at that, right? And so they need to say, “Okay, I need to help on the innovation side to make sure we're always thinking to the future and know where we're headed.” So that's why I say underneath all of it is really I think the more self-aware you can be, the more likely you are to have the right team in place to set the company up for success and make sure you're all headed in the right direction.

[00:10:23] JM: Do you have any perspective on making executive engineering hires? Do the executive engineering hires also need to be capable of doing this kind of cross-functional work?

[00:10:34] LC: Well, I think that executive engineering hires particularly in our category because our customers are our engineers, I think that our executive engineering leaders must have great product instincts. And we obviously need great product managers too, and often our product managers have been engineering leaders in a prior life. But just an intuition and a curiosity about what is the customer experience when they use our software? How do we help them troubleshoot faster? Understand the behavior of their systems better. That's super important. And then you also have to layer in the capability to deliver great software under very aggressive schedules at very high quality and service availability levels and all those things that great engineering leaders need to do.

But I feel like that the truly superior ones just have a passion for delivering software that people want to tell their friends about, and that's in your heart, right? That's something that – One of our core values at New Relic is passionate. And what I mean by that is we look for people who love the nature of their work. I remember when I got my first job at Apple I was fresh out of college. Actually I was an intern at Apple, and so I couldn't afford a car yet, and I used a rollerblade to work every day back when people used rollerblades. And I love the job so much. I remember rollerblading to work saying, “I'd pay to work here.” And so we look for engineering leaders who will pay well, don't get me wrong, but love the nature of their work and they feel lucky to be in this line of work. And I certainly feel lucky to be in the software business.

[00:12:13] JM: And how much do you personally keep tabs on product direction at New Relic? Because the scope of the product is always expanding, but you're the head of the company, so you do want to have some focus on product. Tell me about your perspective on product.

[00:12:28] LC: There's an ebb and flow to it, and obviously we're at a scale now where there's so much going on that no one person can have complete visibility to everything that's going on. But I love to stay as close to product as I can, and I typically have three to five projects that I like to stay very close to. And in ideal world, I’m not in the reviewing and rubber stamping role. I’m also at the whiteboard and ideating on how to make something amazing early enough where there's some real creative outlet there. So I love that, and it energizes me, and sometimes I can contribute to helping the products be better.

And this may be just a selfish thing I do, but I still like to allocate some time. I'll be doing it this weekend actually. We're coming up on a three-day weekend. We’re recording this on the 15th of January. But I like to write code still. And so New Relic’s platform is actually programmable and you can build applications on top of New Relic. And so I'll be building stuff over the weekend. That keeps me close to the code and gives me joy, and once in a while these prototypes actually turn into things that the product org adopts and builds for real.

[00:13:45] JM: Do you have any mistakes that you've made in product development that stand out? Strategic mistakes?

[00:13:51] LC: Oh yeah. I mean like life is certainly about mistakes and learning from them and getting better at it. I'd say one mistake that I can look back on is underestimating how much continual communications involved at a certain level of scale to align the whole organization around a strategic imperative that may require a change in direction. So New Relic One, I'm very proud of New Relic One and what it's done. We've re-architected our user interface for the next 10 years and we decided to do that when we were about eight or nine years old as a company. Recognizing that most technology architectures – I think I remember Steve Jobs saying this. Most technology architectures have about 10 years of life in them, and then at that point they'll continue to go after those 10 years but you're in kind of a low velocity maintenance mode at that point and you're on the risk of being leapfrogged by the market.

So we made the right decision to re-architect our user interface to be the underpinnings of the next 10 years of growth for the company. And my chief product officer and I were really aligned on it as well as the core team building that platform. But I underestimated the effort involved in aligning the remainder of the product organization around that vision for understandable reasons. In hindsight, I mean if you're an engineer that's working on what you have been working on and you're accustomed to the architecture that we had been running, and it was a Rails monolith, right? So it was very 2008 technology. So if I were to do it over again, I would have spent much more time re-communicating and aligning the product work around that vision rather than assuming for them to just get it on their own after I give one or two talks about it.

[00:15:41] JM: Interesting. Well let's shift the conversation to talking about actual engineering today and the product decisions you've made around engineering. So what does an engineering team need today out of a modern observability stack?

[00:16:00] LC: Well, we're opinionated about this, but I think the most important thing is a common data tier that natively understands and processes metrics, events, logs and traces all with the common query language at petabyte scale and very fast query speed without any need to think about or provide any indexing. Why do I say that? Well, when you put an index into a schema, you do that because you're anticipating what the question might be. When you're troubleshooting production systems, you have no idea what the next query might look like that you need an instant answer to.

So that's why we invested in and have this multi-tenant data tier, the New Relic telemetry data platform, that is able to literally scan billions of rows and do, what it's in effect, similar to a table scan and get very fast query times on that telemetry data irrespective of what's in the query. And that could be logs. That could be metrics. That could be events or that could be traces. And that could be telemetry coming from an event that happened inside a mobile application. Like somebody bought an item in your mobile app or it could be an error thrown from inside an application or it could be obviously metrics about the health of a host that the application is running in.

The point is what we see over and over again in the market today is a lot of companies are sending their metrics to one place, their events to another place, their logs to another place or traces to another place and they're struggling with having a unified view across all those data types and each of them have their own query language. And it's really expensive often to do that. We made a decision last summer that we're excited about and, our customers love it, and that is we want to be really aggressive on how we think about rolling out this telemetry data platform to make it very economical to adopt. So this isn't intended to be a sales pitch, but to give the details. We're offering that at 25 cents a gigabyte of telemetry data. And because we feel like in order to adopt observability, you shouldn't have to predict and worry about how many hosts you're running agents on. You should just think of like telemetry data as this inexpensive commodity. And we think 25 cents a gigabyte is really inexpensive. And we'll provide all of the – Not only storage, but the CPU and the network and the compute to query that in the ways I just described to our customers at basically a cost plus level.

And so if you've got all that data in one place and it's economical. And so if you want to send more there and it's not going to be – You're not having an ugly surprise in your bill. Then you're in a position to really have real correlation between like the application health and the infrastructure health and the end user experience and the ability to build applications on top of that telemetry data that could really solve some important business use cases that are specific to your company. So that's the approach we take with it. And I think it's really important. It's vital.

It's hard to imagine a company being successful in production software today without real-time telemetry data. And so virtually everyone understands that, and now the next step is to say, “All right. Well, what's the natural place to put all that telemetry data so that you're not jumping between different data stores to correlate across it all?”

[00:19:35] JM: Interesting. And can you tell me more about how you've had to adjust New Relic and you've had to adjust the platform as new engineering trends have emerged?

[00:19:50] LC: Yeah. I'd say the one that's most exciting to me is Kubernetes. So I think I mentioned at the beginning that I started this journey into what we now call observability in the late 90s, and that was a bet on this new thing called java that turned out to be where people run their workloads. The most popular place to run an application was the JVM for a good part of the next two decades. Well, now in 2021’s scale, I think a Kubernetes cluster in 2021 is akin to what a JVM was in 2000.

And so what we think about is how do you, with the most frictionless way possible, light up a Kubernetes cluster with the deepest visibility to instantly see everything going on in that cluster? And so we found this incredible little startup with the mind-blowing technology called Pixie Labs, and they do amazing things with Kubernetes that don't require you to deploy any agents. They use this technology called EBPF. And unlike other companies that have also looked at EBPF, they go deep into the application layer and they can show all the services and the network and dependencies between those services and the latency between the services and effectively map out everything going on in a cluster, all the discover, things like databases and other things running in that cluster all within five minutes of installation.

So we got so excited about it. We acquired Pixie late last year, and we want this to be something that every engineer in the world wants to use. And if you want something to be that ubiquitous, or certainly every engineer who uses Kubernetes, and we think that's going to be a meaningful portion of the professional developer market. So in order to reach that many engineers, we decided to open source Pixie so that anybody who wants to light up a Kubernetes cluster in five minutes and see that depth of visibility, they can do that instantly with Pixie. And we've just begun thinking about the integration now, but this technology reminds me of where Wiley was in 1998 or where New Relic was in 2008. It's that kind of fundamental and exciting.

[00:22:18] JM: So tell me a little bit more about what needs to be monitored in a Kubernetes cluster or in a post-Kubernetes world. Why does monitoring change so much?

[00:22:29] LC: Well, we've talked to a bunch of our customers. Some of them operate like the largest sites in the world where millions of users and many, many different teams each with their own microservice and other infrastructure related data stores and other packages that they deploy into the cluster that work in concert to deliver an application. Say, it's a video streaming site. And what we've heard from this, it’s like some of them love to deploy agents inside that application to see deep inside the microservice. Others don't or others have alternative ways. Some of them might use open telemetry.

And so what these customers were saying and what I agree with is it would be great if you could have a very deep level of visibility across everything running in that cluster without requiring somebody to deploy an agent inside the JVM or the Python virtual machine or whatever the process is running the code. Think of that as like an optional add-on to go deeper inside that microservice, but not a requirement to get visibility and good deep visibility. And so that is different from where the state of the APM segment has been until recently. And we think that it is great. We believe that there will always be a need especially if you want to do good tracing across these services and there's asynchronous stuff going on. There's always going to be a real value in putting agents inside some, if not most of your microservices. But to get up and running without requiring an agent and still get deep visibility, that is game changing, and that's why we got excited about Pixie.

[00:24:08] JM: I just did a show about EBPF and another company built around EBPF and Cilium, which is a higher layer than EBPF. Can you explain why it's so critical? Why is it an important technology? It's connection to the to the Linux Kernel and so on?

[00:24:27] LC: Yeah. It's at that layer in the stack where the visibility is incredibly rich and deep. For example, inside EBPF, you can see every network call that goes on within that cluster. So for example, you might have a microservice running inside a pod that's talking to another microservice written in a different language talking to another pod. And the EBPF can see the HTTP in between those two and it can measure the latency between the two. And therefore infer you know the application dependencies between those so you can build a map between all of these services. You can marry it with how Kubernetes describes that cluster and what those workloads look like to come up with a very comprehensive view and all that all by just observing at that layer that EBPF runs at inside the Linux Kernel.

And so doing that well takes some really smart engineering. You're at that kind of low level in the kernel where you got to be super thoughtful about the performance overhead. You certainly don't want to perturb the behavior of the application or the system, but you also want to collect the data in a way so that it's most useful and actionable. So it takes really bright engineering to do that well. What we saw about how Pixie approach it was they said, "Let's build a platform on top of that in which engineers could write little pixie scripts," in their language called PXL. It's basically Python, but that'll allow you to automate the troubleshooting and write some code to diagnose a problem and then that code can be reusable so that it goes beyond just simple dashboarding, because often these problems are so complicated that you often want to execute a sequence of steps like grab the top 10 slowest transactions that are running through this service. Inspect them for whether or not they are – If they're all coming from the same user and then see if that user happens to be from a list of IP addresses that we know to be problematic. I don't know. I'm just picking it up. But like that's something that is kind of scripty and can go far deeper and do more and then be more reusable the next time you might have a similar problem. So the pixie team was really brilliant in saying, "Hey, let's make this something that engineers can write scripts on top of and then a whole community can share those scripts so that the collective wisdom of how to troubleshoot these systems just gets bigger and bigger over time."

[00:26:55] JM: How does monitoring Kubernetes clusters compare to virtual machine monitoring which has been around for much longer?

[00:27:04] LC: I think Kubernetes clusters give the opportunity to – You can make some assumptions on an application running in Kubernetes that you can't make if it's just a raw virtual machine. So like there are deployments and there are end points and there are these Kubernetes first-class objects that help you describe what your application looks like and how it's deployed. And therefore, because it's deployed in this way, a visualization or logic that operates on that telemetry data can do more knowing that. So can kind of put a dependency map together that says, "Oh, your shopping cart service is dependent on your pricing service. And here's the throughput between them." And all of that stuff just gets easier to put into a semantic way when you know you're running inside the Kubernetes environment. You can make some assumptions. You can call some Kubernetes APIs to draw a map that's contextual. And it'll make more sense to the engineer who understands Kubernetes and works in that environment.

So we found that actually the pattern matching I see in that is, really, in the early days of New Relic, what customers loved about our APM project was how it instantly showed just the right information in a language that was easy to understand as an engineer. And the reason why we could do that is we said, "Hey, a lot of applications are running on popular frameworks." In the Ruby case, it was Rails. And in Java, a lot of people were doing stuff with Spring. And so, again, because these applications had these like constraints of the framework they were running in, we could build far easier user interfaces and far better visualizations that just made sense of the data.

[00:28:45] JM: So Kubernetes doesn't just represent a market shift from your perspective. There's also an opportunity to rebuild your own infrastructure. You've got high throughput, intense infrastructure that you need to engineer for. Have you re-platformed the internal New Relic company infrastructure at all?

[00:29:08] LC: We have. So I mentioned earlier, our core data tier, we built from the ground up as a multi-tenant database to natively process metrics, events, logs and traces. And the thing that's I think pretty unique about it is its multi-tenant nature running at massive scale. But two or three years ago we were at a point where those multi-tenants or all the North American tenants were all deployed into one enormous cluster that we were running in a data center. And what we have done particularly over the last year is migrate that workload into a collection of what we call cells whereas say a thousand or five thousand tenants will run in a Kubernetes cluster that's running in Amazon Web Services. And that cell obviously, if it happens to have a problem, we could migrate a customer to another cell. But only the tenants in that cell would be affected by that and we can scale it independently and all those benefits of kind of breaking up into a cellular architecture.

So Kubernetes and public cloud have been important parts of delivering on those capabilities. All along the way, our data ingest rates are just skyrocketing in growth, right? So as more and more customers have larger and larger workloads and they're adopting, going beyond our APM capabilities to add logging and infrastructure, we just see more and more data coming into our platform at very high growth rates. And so Kubernetes is an essential part of managing and scaling with that growth.

[00:30:45] JM: A recent product announcement you made was the New Relic One platform. So this is a significant re-release. Tell me about the product development for New Relic One and how you architected it.

[00:31:02] LC: It's a great question. As I mentioned, the pre-New Relic one, we started off with the Rails monolith and then over time we started adding new products. And they might have their own services, but they weren't as loosely coupled as we'd like them to be from the Rails monolith that was kind of the APM product and the browser product and a couple other products. And at that point we had so many different teams. You can imagine the challenge it would be in sustaining high velocity and all of that. And the aha moment for us was when we had built some internal tooling that would allow any engineer in the company to write a React component that could snap into a unified UI. And the UI, the shell for that UI and the rest of all the other products was hosted from our newrelic.com. But like your code, just like the component you're working on, was just served off your local desktop. So all you needed to do was have node NPM installed and just – Like could be just couple source code files. And every time you press save, your stuff would refresh in the browser.

And so I thought that was so compelling, because when you combine that with this data tier that I've discussed that's like this really fast database that you can dump anything with a time stamp in, now anybody who writes React can build really compelling application functionality on our platform. And I knew it was going to really accelerate our developer’s productivity. But what got us really excited was we said, “I don't think it's just New Relic employees that could benefit from this capability. Let's offer it to our customers too.”

So what we did was we said all of the tools and the components that we use internally to build, say, our APM product or our dashboarding product or our mobile monitoring product, we want to expose them as public APIs so that if you want to – And those include our charting components and our widgets and all of that kind of stuff, so that you could rapidly build an application.

I'll take you through an example. We have seen large restaurant companies that have moved their business to the mobile application. Build applications on New Relic One to show the health of the point of sale systems and the mobile ordering activity by location so they have this beautiful like geographic map that is showing their digital business in real-time of what are the orders flowing through across the country and are there particular locations that are having problems where people can't pick up their order. And who would have imagined that use case? I mean when we were building in the early days an APM product, right? So that was an easy application for our customer to build because it was just a new view on the same telemetry data and all they needed to do was write a bit of React code, reuse an open source library for visualizing that map and they were good to go.

[00:34:10] JM: Very interesting. Can you tell me more about the product push and just how you managed such a large product from later position in the – I’m like very curious about the resource allocation. You're spinning up a brand new product and you have tons of existing customers to support. It sounds like a big undertaking.

[00:34:30] LC: It was a really big undertaking. And so what we first did was we announced New Relic One. The first release of it was in the spring of 2019. And then in the fall – Which was kind of it was the first release New Relic One and it made no attempt to replicate all the functionality of all of our other products. It was just a new kind of high-level view. Think of it as your portal into New Relic and then you jump from the relic One into a New Relic APM or a New Relic Browser or mobile or any of the other products that our customers were comfortable using with. So it was the new front door. That was phase one.

Phase two was we added programmability so people could build those applications. That came out in the fall. And then phase three, which was really the completion of like everything in New Relic One was last summer through a pretty herculean effort, we managed to migrate all of the rest of the user interface of all of our products into the New Relic One UI. And we did that through doing some smart things by being able to take the views off of our other products, make minor styling tweaks and do some eye framing tricks to iframe in the long tail of all of the views and pages that made up our product suite and then we did native ports into React and the modern clean UI for our most popular views and we continued to do that over time. But it's a seamless experience. For many customers, they don't even notice what's kind of the "old stuff and new stuff". It's just one unified UI now. And so we're thrilled with how that came together and that all launched last summer.

[00:36:10] JM: Let's zoom out. We've talked a lot about monitoring. We've talked a lot about observability and New Relic specifically. Give me your vision for the future of observability. What's going to change? What is left to be built?

[00:36:27] LC: Oh! I mean – Well, first of all, as I say, as long as there's software, there'll be bugs and problems and things that go bump in the night. I still can't believe that is 2021 and Bluetooth almost works. But more often than not, the phone called gets dropped because I didn't get the pairing right fast enough. So software is kind of like that. So there's just continual improvement on troubleshooting, and that is almost like an arms race against ever increasing software architectures, increasingly complex software architectures. But I don't think that's the really exciting thing. In my mind the exciting thing is like every ounce and dollar of energy and money put into building software is purely sunk cost until a human uses it for some business purpose.

Let's say you're an e-commerce company and you're building some e-commerce software. I mean all that engineering effort isn't valuable until a human uses it. And often engineers – And this happens in large companies, but all too often engineers and product people, they're just obsessed on like getting their code done and committed and checked in and then they go on to the next piece of functionality. And I think the work to be done is to make sure we're continually reminding engineers that, “Hey, there's a human using the software you wrote last week. In fact, 300 people used the feature you wrote last week.” And as a builder software, that's really motivating to me like, “Hey, we all want to just not only build software. Build software that other people use and value,” right? And you certainly want to know if it's broken.

So I would love it if we can better – Our platform can help evolve software engineering cultures so that they're continually reminded that what they're working on, there are humans who use it and they have data that helps them know how good is the software that they've been building and they're responsible for continually improving. So that you're not just a feature factory, you're developing software that makes other people's lives better. And I think that our telemetry data has a role to play in that and our platform has a role to play in that and that's why I think we're early in the journey.

[00:38:43] JM: Do you have any perspective on how we're going to see software change more broadly? I think like serverless is a pretty dramatic shift in software architecture. For example, you've got shifting popularities in different programming languages like the popular systems level programming languages are now Go and Rust. And you've got really dramatic changes just all around. Are there any fundamental changes to the world of software that you're really keeping your eyes on?

[00:39:19] LC: Well, it's early days, but there seems to be increasing excitement around codeless software development. And this probably isn't the audience who listens this podcast. But there is a universe –

[00:39:31] JM: No. No. We've done a lot about that. We've done a lot about that. People love that stuff.

[00:39:32] LC: Oh, great. That’s awesome. I think that that's exciting because I think there's a universe of people who would love to build things on computers who may not be comfortable in VS code or even more so VI. And I kind of myself was not that comfortable in VI personally. So I think that that is interesting but has been something that's been talked about for so long. Maybe it'll really get nailed and people will build meaningful applications that continue to drive a lot of value in the coming two, three years. So that's something we're paying close attention to.

We think that also drives different, but increasingly important requirements. I'd say that – I’m sure you've spent a lot of time talking to other people about this on your podcast that a lot of software development today seems to be just Stack Overflow, find the right NPM package or pick your library and glue things together as fast as you can and see how it works, which is great and cool for rapid software development, but it really does mean that there's less understanding of how the whole thing works. And codeless just makes that even more so, right? So that's where I think, again, telemetry is going to increasingly come into play on saying, “Okay, how does this integrated system behave? Since you're becoming often a developer, is playing the role of integrating lots and lots of code, most of it open source or components with little understanding of what's under the covers. Because these things, no human can understand everything that's running inside these applications obviously. It's just too much.

[00:41:06] JM: Yeah. And even just the – I did a show with the CEO of Webflow a couple years ago, which is one of the more popular no code tools. And they spent so long building this thing with a really small team that hiring for that product is incredibly difficult because explaining how it works is so complicated. There's just this huge piece of software that produces no code applications. But I'd love to know, you've built two observability companies yeah. If you were not working on an observability company, what kind of company would you be building?

[00:41:45] LC: That's a great question. I feel like I don't know if this is my life's work. I am attracted to really, really hard technology problems, and that's why I feel like this is such an interesting space to be in. So many parts of this make this hard. You need to somehow make it as easy as possible for like the most demanding user in the world, the engineer, to instantly understand everything about their system. You need to collect an immense amount of data at massive scale, massive scale. You need to make sense of it. You need to operate in real-time. You need to be continually up to speed on what architectures people are building applications on. You need to correlate it with everything under the sun. I just feel like it's such an interesting problem space from a technology perspective. And I think of myself as a technologist first and an entrepreneur I guess as well as a company leader. So I don't know what else I do. I'd probably do something outside of technology perhaps, perhaps on the charitable side. But I'd say you know life is short. We spend so much time of our lives at work. More time than we spend with our families. And so that's why I think it's so important to just be pursuing something that just fires up your passion and your interests and that you're doing it alongside people who make you a better you.

And so what I try to do at New Relic is ensure that New Relic meets those two criteria for our employees. They love the nature of their work and they love who they're working alongside. And if we do that, that'll be a place where amazing people can do amazing things, and we believe we have a noble mission because we believe in software. We believe it’s changing the world in such dramatic ways, and that great software improves people's lives. And hopefully we're playing an important role to help software be better. So we've got a lot in front of us because we're not done with that mission.

[00:43:48] JM: Okay. Well, that seems a good place to close off. Lew, thanks so much for coming on the show. It's been a real pleasure talking to you.

[00:43:54] LC: Oh, thank you very much. It's been a pleasure. I enjoyed it.

[END]