EPISODE 1425

[INTERVIEW]

[00:00:00] JM: Splunk is a monitoring and logging platform that has evolved over its 18 years of existence. And its modern focus on observability, it has focused on open source and AI ops. Observability has evolved with the growth of Kubernetes and Splunk's work around OpenTelemetry has kept parity with the open source community of Kubernetes. Spiros Xanthos is the General Manager of Observability at Splunk, and he joins the show to talk about Spunk's modern product portfolio and his work on his own company prior to being acquired by Splunk.

[INTERVIEW]

[00:00:30] JM: Spiros, welcome to the show.

[00:00:32] SX: Glad to be here, Jeff. Thank you.

[00:00:34] JM: You were the CEO of Omnition before you joined Splunk. And that acquisition occurred back in 2019, three years ago, and observability has changed even in that short period of time. Let's take an even broader look, when you think about the last 5 or 10 years, what are the most notable changes that have occurred across the domain of observability?

[00:00:59] SX: Well, first of all, I guess observability, as a name and category is new, right? Probably, we started using it maybe three or four years ago and became popular probably in the last one, two years. I think what is driving observability, is the change in the underlying infrastructure and the way we build applications. So, if you were to take the last 10 years, as you described, what we have seen happening is that obviously, the cloud has become mainstream. But not let's say, the cloud as I am taking my VMs that I'm running on prem and moving into the cloud for more flexibility. I'm talking about cloud native application development, right?

So, containers, Kubernetes, microservices. This is a new way of building and running applications and infrastructure, everything has become software. It's a great way, let's say to build maybe software faster, to be more agile, to respond to the business needs. But at the same time, it has created a lot of complexity, right? Because now we're dealing with highly scalable distributed systems based on microservices on dynamic infrastructure that might change multiple times a day, very, very frequent deployments. We're not deploying software once a quarter or once in every six months, we're probably deploying it multiple times a day. So, a lot of change.

And as a result, what used to be monitoring, siloed monitoring approach, separate monitoring for my infrastructure, separate monitoring for my application, separate APM tools, separate tools to monitor what's happening to my end users, is not sufficient anymore. These systems are not powerful enough to help me diagnose problems have happened in a modern cloud environment, where most of the times I'm dealing with unknown unknowns, not problems that I might have faced and solved before.

[00:02:56] JM: Is there a larger data requirement for gathering the necessary granularity of observability as the infrastructure has changed?

[00:03:10] SX: Yes. So, I think, at the end of the day, observability and monitoring is a data problem to a great extent, given the complexity of the systems I described, there are many more permutations and failure patterns that might encounter in a modern cloud native environment. As a result, it's very, very difficult to know in advance what data to collect for any failure or problem you might face in production, let's say. So, as a result, we would be more intelligent about what data we collect, but also just need to collect a lot more data. So, when a new problem arises, we have, let's say the necessary signals to be able to troubleshoot the problem.

To make it very, very specific, traditionally, APM used to collect, let's say, used to sample heavily, collect maybe one in a thousand transactions that flows through an application. And that was good enough signal to understand how the application behaves when it comes to a monolithic application that may rely on a database. But that doesn't cut it anymore, and microservices based obligation, or what I said, the failure patterns are a lot more. So, we cannot like just sample so heavily and hope that when a problem happens, we're going to have the data we need.

In fact, our approach automation are not blank now, has been to have a no sample architecture for distributed tracing. So, we capture every transaction that flows through the system, process it and store it efficiently, but have it available if a problem occurs.

[00:04:42] JM: So, tracing, logging, those are the main sources of observability today. What's the synergy between tracing and logging? And maybe you could describe the process of diagnosing a problem using tracing and logging.

[00:05:05] SX: I thought that the other I guess, main signal that we usually have in modern environments are metrics. And those can be custom metrics that the application developer defines. Or it could be standard metrics we collect from the infrastructure, CPU, memory, et cetera. So, metrics, traces and logs are the main telemetry signals. And maybe it's worth saying also that, in our opinion, observability is this idea that all this data comes together and becomes connected. And at the same time, really, I can troubleshoot infrastructure, application, problems that my users might have, all in one place, right? Because all of these are interconnected, meaning my application might suffer from some infrastructure outage, and vice versa, let's say.

So, what we try to do with modern observability, is connect all these signals together to your question. So, connect metrics, traces and logs together as much as possible, so that we can synergistically troubleshoot problems. In our view, metrics are ideal for knowing that you have a problem. We can monitor KPIs. So, let's say, your response time, or latency, or error rates. But once you let's say, one of these indicators of service level are off, then you need to know what has happened. And then that's where I think tracing comes to play, because it can fully connect the dots all the way from the user, to whatever backend systems you might be using, so we know, let's say, we can isolate where the problem might be coming from.

In a microservices architecture, an end user might be facing problems. But that might be from an upstream service that the user doesn't talk to directly. And of course, logs give us, let's say, the most kind of free form disability, so that once we have isolated the problem down to a particular part of our system, maybe we can go look for anything that might be off or wrong during the period of time that we're interested in. So, all this work together to troubleshoot problems quite frequently.

[00:06:58] JM: Can you give me a sense of the usage of OpenTelemetry and maybe describe why an open source project is relevant to a generally closed source company?

[00:07:12] SX: Yes, I guess Splunk originally started obviously, as a closed source proprietary software. But we have been embracing open source more and more as a company, including, of course, OpenTelemetry. OpenTelemetry, started at Omnition, my previous company, where one of the co-creators of the project, and several of the founding members of OpenTelemetry are now at Splunk, either via the acquisition of Omnition, or people who joined us subsequently. The reason we believed OpenTelemetry, one project like OpenTelemetry was important is because, as I said, with observability, what we're trying to do is bring together the data across all telemetry signals. So, we can have more effective monitoring and troubleshooting of applications and infrastructure. And that is impossible if you rely on proprietary protocols, and data collection mechanisms, right?

So, I might have one way of collecting logs, I might have another way, still proprietary of collecting metrics, I might have an APM agent that is totally proprietary instruments, my application and collects the data. And all of these end up in some data silos, that are impossible to connect by design and definition in some sense. So, we felt that this was very important to democratize the data collection, and give the user the power of what we want to do with that data, first of all. And second, let's say, give the ability to tools to connect these data together. So OpenTelemetry is, of course, the merger of open sensors and open tracing, projects started with similar goals. But at the end of the day, OpenTelemetry is trying to essentially create a set of standards, and an implementation on top of it for most popular languages, so we can instrument applications, whereas the case or auto instrument applications with auto instrumentation agents, and data collection, let's say infrastructure for transferring all the data to whatever back end is a treasure of the user. And the back end can be a proprietary back end SaaS service, or it could be an open source solution, like Prometheus, let's say. And it's worth saying that OpenTelemetry today is the second most popular project in CN/CF in terms of contributions, only second to Kubernetes. It has been fully embraced by the industry and a lot of end users.

[00:09:22] JM: When you started OpenTelemetry at Omnition, what were the goals of the project?

[00:09:28] SX: The goal for us was, I guess we'll have the belief that anything that goes into the user environment more than a cloud environment has to be open source, otherwise the users will not trust it, right? And secondarily, we have the goal of, as I said, democratizing the data. In the past, especially when it came to application monitoring, all the technology and IP that we had created as an industry was all about instrumentation. So, how we can intelligently collect a very small subset of the data and based on that, troubleshoot an application.

In our view, in modern environments that doesn't work anymore. So, we wanted to move all that intelligence to how we analyze the data and to the analytics were able to provide on top of that data. So, our belief was, we had the goal of building an open source solution, that would be very powerful, and very simple, easy to use, in terms of like collecting data, as much data as possible. And then, give the users I said the option to use whatever backend they believe to solve the problem best. So, automation errors, applying subsequently, we focused a lot on building, let's say, very powerful analytics on top of this data, and we think that's where the value is, and we decided to contribute all our IP, let's say an effort in making OpenTelemetry successful. So, I would say that the data collection is not any more differentiation, because we don't think that's where the value or the users is.

[00:10:53] JM: Can you tell me more about how the spec for OpenTelemetry was developed? And exactly what it gave to people that did not exist in the open source telemetry ecosystem before? Because there were other open source projects around telemetry before?

[00:11:15] SX: Yes. So, first of all, OpenTelemetry is not a single spec, right? Because the project had fairly ambitious and first of all deals with metrics, traces and logs. It deals with transport protocols of this data. It deals with, let's say, the definition of spans, metrics, all of that as they are emitted and generated in the application. But the goal of the project from the beginning was to standardize how we essentially collect and transmit this data. It wasn't, we'd never had the goal. We had explicit – I guess, explicitly, we didn't want to be a back end for this data, right? We just wanted to standardize and democratize how this data is being collected from all the applications. We don't think there was an effort like this before. There was open tracing, of course, tries to monetize the tracing aspect. And there were open sensors that tried to do the same for tracing and metrics. But generally speaking, we didn't feel like there was another project that was trying to standardize this. Because that's why the project was created in the first place and that's why we put our effort behind it.

The standardization is very, very important. Because once you let's say, define the specs, then, first of all, anybody can be limitation. OpenTelemtery itself provides an implementation. But then the data, let's say, is fairly well described and structured on the source, so then you can build policy analytics on top or a back end, open source of proprietary, that can actually be a lot more powerful, because the data is truly structured at the source. Unlike, let's say, traditionally how logs look like, which were free form, and was very, very difficult to build something more than simple, let's text search and fill extraction on top. Here, let's say we have a lot of metadata that come from the source that help us connect all this data together.

[00:13:00] JM: Gotcha. And was there any connection between the emergence of the OpenTelemetry project and the growth of Kubernetes?

[00:13:15] SX: Of course, I think the correlation that for the most part is that, let's say cloud native application development typically means containers, it means microservices. And Kubernetes, of course, has become for the last few years now, the de facto, let's say, way of running, let's say, the infrastructure and I guess the applications on top of that infrastructure, right? As a result, the more I guess, we see Kubernetes being used in production, more mainstream it becomes, the more likely is that we're going to see complex applications, microservices, dynamic infrastructure, with a challenge I described earlier. And as a result, the more likely it is that people are going to be looking for more than observability solutions. And OpenTelemetry has become the underpinning open source project for modern observability. So yes, there is a strong correlation. I think we will continue to see this correlation to cloud, let's say growth, and to Kubernetes growth.

[00:14:13] JM: Describe the product suite that you've been working on since you joined Splunk.

[00:14:22] SX: So Splunk, we have what we call the Splunk Observability Cloud, which is a fully integrated solution that brings together metrics, traces, logs, and real user monitoring. And Automated Analytics on top AI ops, as we call it, as well as incident response, essentially, alerting and notifying the users, or otherwise, in terms of like, the areas that we're solving this problem is log analytics, infrastructure monitoring, APM, real user monitoring, et cetera. So, this is a cohesive solution. It looks like a single application, unlike let's say how we dealt with all these parts of monitoring in the past, single user interface, the data is fully connected, it all relies on OpenTelemetry. So, we believe that has improved quite a bit how effective and powerful, let's say monitoring and observability can be, because we bring everything in one place. It works at any scale. We have some of the largest customers in the world, some of the largest retail, ecommerce, name it, using it at a very, very, very large scale. And that's our focus as a company in general.

And also, the other big focus we have is I said, analytics and AI ops, right? Now, that we have all this data fully connected, not only we can, let's say, manually do more powerful troubleshooting, but it allows us to build analytics on top that start connecting the dots for the user. And that's kind of the other foundation for what to have built. So, OpenTelemetry, data fully connected in a single application all in one place, and analytics on top, enterprise scale, really.

[00:15:57] JM: And when you talk about developing a product that's comprehensive in that sense, can you explain how it differs from what was available maybe two or three years ago? And what are the kind of engineering problems that you've been addressing that were not available, or they were not addressed in kind of previous iterations of observability technology?

[00:16:26] SX: Sure. First of all, I think we're built observability cloud, as we call it, the way we build it, because we have been responding to the users and customers, right? What we see is because of the increased complexity of the systems, and how connected now they are, we don't monitor and troubleshoot infrastructure separately for applications anymore, and when let's say, an end user faces a problem, it’s often times connected back to the backend application, and that might be connected to the underlying infrastructure, which is usually on the cloud. So, that was the problem our users were facing, and that's what we're responding to.

The difference from the past is, typically, we used to have some system that would monitor my infrastructure, maybe had a different system that monitor, let's say, my virtual infrastructure on top. I usually had a different system that monitor maybe my network devices. I definitely had a different APM system that only monitor my application. Oftentimes, the APM provider might have given me a real user monitoring application as well. But that was also not really connected to the way I was troubleshooting backend applications. So, all of these were separate and not connected, right? So, whenever a problem matured, it was up to the user. Oftentimes, in the war room, multiple, let's say, admins have all these tools, getting together, trying to understand where the problem might be. And all the troubleshooting and monitoring, all the troubleshooting, all the advanced troubleshooting was happening in people's heads, right? Because I had maybe some data in one system, but how to collect the data in another system somewhere else, completely manually.

So, that was, and still is the life of most, let's say NOCs, network operating centers in the world. And observability is trying to change that. And I think the whole industry is moving towards the direction I kind of described, in my opinion.

[00:18:12] JM: So, when a user hooks into Splunk, and they start collecting logs, metrics and traces, can you give a sense of the backend infrastructure that's storing that information? I just love to get a sense of the databases and the infrastructure that you use to serve that data caching, infrastructure, et cetera.

[00:18:37] SX: So, first of all, in trying to build the system ourselves, but as I said, is enterprise scale and really handle data of any volume in real time, it's very hard engineering problem by itself. As I mentioned, all our IP, when it comes to data collection is part of the OpenTelemetry and in the open source. So, a lot of the additional value we're providing is in how we deal with this data. So, we're dealing usually with structured and unstructured data. So, metrics and traces tend to be very, very structured. And logs tend to be more unstructured. In any case, there's usually a processing layer for all this data as soon as arrives. So, we're trying with a very, very low latency to ingest the data and route it to the appropriate place. And then when it comes to monitoring, let's say alerting, for example, it has to happen in real time, right? Our goal is to be able to alert the user within let's say, 10 seconds from a moment a data point is generated, so they know immediately in real time if they have a problem.

We have built, let's say, our own streaming metrics and monitoring infrastructure. That was a lot of the technologies that signal effect had built actually before the acquisition, that allows us to say monitor all this in real time as it comes to the system. So, traditional time series databases tend to store all this data and then let it rest in query the data for let's say, alerting or dashboards, but oftentimes means that you have to wait for minutes until the data rests and is there available for your query to bring you back what do you need. And of course, the more alerts you create, the more challenging it becomes.

So, we have this streaming architecture that avoids that because we update, let's say, alerts and dashboards as soon as data streams in. Then we have an analytical database where we store all of our tracing for a very, very high cardinality, troubleshooting. Cardinality, meaning we can accommodate many, many dimensions when it comes to the data. And of course, we have logs indexing technology, for what Splunk has traditionally done in dealing with unstructured data. So, it's a very complex system with many, many, many microservices, many different databases, that is all kind of connected in the backend. But I guess, maybe the main point to remember to answer the question is there are many different types of storage, some that are optimized for real time responsiveness, some that are optimized for actually cost, so that our service has a reasonable, let's say, price. Some that are optimized for scale, some of that are optimized for dimensionality of the data, and I think a lot of the value will provide this, how does the maintenance system like this? Again, just getting just many, many terabytes of data per user, per day.

[00:21:10] JM: And for the analytical database, what are you using?

[00:21:13] SX: We use Druid for a lot of the analytical type of use cases, which is also a project that Splunk has been supporting for a while, coincidentally.

[00:21:23] JM: And what was the choice of – what was behind the choice of Druid over maybe a data warehouse? Or I guess you could have chosen like Pino. What was behind the choice of Druid?

[00:21:36] SX: Yeah. So, first of all, I guess, when it came to choosing Druid, keep in mind that the types of use cases we're trying to solve, were traditionally solved with time series databases, right? Which really don't work very well, for high dimensionality data. So there, we try to find something that will do very, very well with very high dimensional data, right, which usually find more like in business use cases that typically Druid has been used for. It just was seemed like with met for mentors we tried was the best tool for the goals we had, in terms of like the scale was able to achieve for us, for the high dimensional the data, and all of that. And it was a new idea for the most part, I think, when it came to like monitoring and troubleshooting as an infrastructure.

[00:22:20] JM: Do you know what was used before Druid?

[00:22:26] SX: Before Druid, most companies just had the time series database, either proprietary, or one of the open source time series databases are available. They will store the data there, and then they query it. But when it came to aggregating metrics, let's say, if I wanted to ask a question, what is my response time, let's say by type of user by user, and if I had like many, many of them, then it just simply wouldn't work. I could aggregate up, let's say, 10,000, 20,000 of these, right? But I couldn't go beyond that. So really, it was the limit and still a limit of time series databases that has not been solved. With what we build, let's say in Splunk, you can combine like millions or you have answers to for, let's say, properties that have millions of dimensions and still get an answer in real time and that's when Druid comes to play.

[00:23:12] JM: So, when you think about infrastructure for an observability platform, you have to do some – your databases have to be capable of connecting, for example, a trace to a collection of log data that might be related to that trace. How do you perform joins on those kinds of data systems?

[00:23:37] SX: So, first of all, what you need is you need, let's say, some sort of an ID that connects those two, to start with, right? Because if you don't have that at the beginning, of course, it's impossible to join the data later. And, again, that's where OpenTelemetry and standardization comes to play. Because now we can agree on a set of standards that, let's say allows us to collect this data. Now, let's assume we have it. For the most part, I guess, the way were are able to do this quickly is of course have a lot of indexing and caching technology, when it comes to let's say, frequently accessed ideas. The basic idea is that you normalize the data for the most part, right? Meaning, let's say if I have a trace ID, and I have a bunch of log messages that correspond to the trace. So essentially, I have a request scope, which one of my user requests has failed, and have a trace ID for that request. There might be a bunch of log messages that were generated in response to that request. So, those who will probably have the trace ID in question, right? So, the data is not normalized. I guess, if I have the tracer ID I'm looking for, and if it hasn't been sampled out, then I can just query and get all the assuming you have a system that does that effectively, efficiently. I can query and get all the log messages that contain a trace ID and bring it back to the user.

[00:24:51] JM: When you look at the design of UIs for observability platform, how do you decide what's the most ideal information to show to a user? Or do you just give the user the freedom to design the interfaces they like?

[00:25:14] SX: I think UIs and years of experiences is a big part of observability and something I'm passionate about. Generally speaking, most of the systems that we have used in the past to do monitoring, follow the same pattern, user experience pattern, let's say. We collected some data, we put the data somewhere, and we provided a query interface to the user. So, the user had to come up with assumptions or hypotheses. And let's say, maybe query that system to validate or invalidate those hypotheses. So, that's an iterative process that often takes a lot of time and takes a lot of intuition from the user. The user needs to know what they're looking for to even state a hypothesis. They have to be familiar with the system. So, they have to be an expert.

Within that modern observability should be more accessible and accessible to every developer, without requiring the developer to be an expert in the observability tool itself. So, the approach we're taking is that we try to be very visual, like our starting point usually, is maybe a map of the application and infrastructure, the service mark. On top of which we layer all the important KPIs and metrics. And if something is off, we usually try to highlight it to the user. We can also display other types of information. You can use the size of, let's say, your service to indicate how much traffic flows to that service, or you can use color to indicate if it's facing a lot of errors.

But generally, the idea is that we don't try to essentially make you an expert in the application itself, the observability application. You know your application, and roughly how it works, and will display that to you and try to essentially do as much as possible on top of that, and connect all the data together. It's very difficult, I guess, to verbally describe a user experience. But the point is, we try to be much more visual and we try to essentially, in some sense, maybe stay the hypothesis for the user of what might be off, as opposed to them having to intuitively figure out what questions should they ask.

[00:26:59] JM: How do tags play a role in designing and observability workflow?

[00:27:05] SX: Tags, you mean tags, like essentially, data tagging that might come from the data that flows into the system?

[00:27:12] JM: Yes.

[00:27:13] SX: Again, tags, we mean, like structure, right? Ideally, you have essentially key values that come along with the data that describe what the data might have come from. Say, you might have the Kubernetes pod, where these logs or traces might have come from or we have the service name, and a lot of other details. So, these are very, very important information, as I'm trying to slice and dice and troubleshoot an application. That's where the analytical kind of approach I described earlier comes into play as well. So, you have to be able, essentially, to store all that in a way that allows you later to aggregate data dynamically in any possible way.

Meaning, let's say I have an application that is facing some issues. Oftentimes, I might want to ask, is this for all the users or just, let's say, users that come from my OS? Or users come from, let's say, Canada? Or users that run on a very specific version of a client? And let's say isolate it down to a very specific subset of users, then I might want to ask, okay, does this happen for all the types of requests or the requests that hit the database, and maybe I isolated further, and maybe I can further ask iteratively. Okay, now that I know like, the types of users and the types of request, does it happen on all my infrastructure, or my infrastructure that runs on a particular pod, let's say. Show me all the pods and how they behave.

So obviously, understand, like, all these kind of queries on top of each other, create an explosion of dimensionality. And, of course, that's where data comes in. But you also need to have a way to be able to handle all of that at any scale. Makes sense?

[00:28:42] JM: It does. Yes. So, do you work at all on front end monitoring?

[00:28:49] SX: Yes.

[00:28:51] JM: Is there a way to connect front end traces to backend infrastructure?

[00:28:58] SX: Correct. So, the way we have built, I guess, what's called real user monitoring, and error reporting on top of that now, for errors that occur on the client side and JavaScript is that we use the same, let's say, principles, very visual way, full fidelity, so collect every transaction. And as a result, that allows us to fully connect the data to back end, because traditional end user monitoring, collect the samples, some very small sample of the user interactions, had another small sample of the backend interactions, the two were randomly collected, so it was almost impossible to connect the two, right? Every time I had an end user trace, we're very unlikely I would have the backend trace as well. So anyway, in our case, because we collect all the data, both on the client and the backend, the traces are fully connected. So, this allows me to say if my user is facing a problem, it allows me very, very quickly to know if a problem comes from the front end, or it's a back-end issue that has propagated to the front end. And then further iterate and solve the issue. It's worth noting that the data collection technology we have built for JavaScript, we're also now contributing to OpenTelemetry. So, it will be available for anyone else who wants to use.

[00:30:08] JM: How does Splunk fit into a proactive monitoring workflow? Or how do you build a proactive monitoring workflow around Splunk, rather than just having to be reactive to failures?

[00:30:24] SX: First of all, even when it comes to say, reactive monitoring, one of the principles we have is that we want – and converting is real time, right? So, if you have a problem, you will know as quickly as it happens. You can react to it much more quickly. Let's say, it won't take minutes from the moment problem happens until you know about it, assuming there is another place for it. Now, beyond that, we'll go back to what I was describing before, because I think we have now a lot of data, a lot more than we traditionally had. And generally, data has more structured, and has a lot more dimensionality in them, and start being more proactive, and that's kind of where I think observability and maybe this concept of AI ops come together.

We talked about AI ops for a while. But I don't think we were able to be very effective, because the data was not structured enough. The signal to noise ratio was never good enough with the data we had, until maybe more recently, with all the centralization I was describing. So, what we try to do is, because we have all the signal, we try to essentially, as quickly as an issue happens, we try to visually and proactively tell the user what the problem might be. Of course, if we have implemented this along or anything else, which is general purpose, then we can also help us be a lot more proactive. But the point generally, is that we try to understand how the system behaves normally, and identify abnormal behavior patterns that might indicate a problem, right? And kind of surface those and have the user take a look at where the problem might be right now, as opposed to like, all the data is available in the system.

[00:31:58] JM: Can you talk a little bit more about what you mean by AI ops?

[00:32:02] SX: So, I guess AI ops is an industry term, right? It's the idea that I'm using AI, or really, in reality, most frequently analytics, to be able to help automatically monitor, troubleshoot and resolve issues in an IT system. So, Splunk has always been kind of on the forefront of this, because we have always been more of an analytics solution than anything else. And for us, I guess modern AI ops is this idea that we're bringing, monitoring and observability and automation together, right? So, once I have all this data, the way I was describing, AI ops, is this idea that I'm using analytics and machine learning on top to be able to proactively tell the user what's going on or what the problem might be, before it manifests and becomes a real issue.

[00:32:51] JM: Is there a lot of work around machine learning to enable that?

[00:32:55] SX: There's a lot of work around – first of all, let me start by saying actually, how I think about this. I don't believe that black box machine learning by itself is going to be good enough for solving some of these problems. I think we need to have the right amount of data and the right structure in the data in the first place, so we have enough signal in the data, right? So, step one, in my opinion, is actually solid analytics so that the user can deterministically get to a problem with the data is available in the system. Once all that is in place, I think then yes, machine learning and AI ops comes to play. Because you can start identifying patterns automatically. Effectively, what the user would have done manually through a lot of hypothesis, stating and, you know, iterative troubleshooting can be short circuited and automated with, let's say, machine learning by identifying failure patterns, new failure patterns, behaviors that we haven't seen before, types of transactions that may be failing, and surfacing those to the user. So yes, there is a huge investment we have made in that area as a company, because that's where we think the value is, especially now that you have standardized data collection.

[00:34:01] JM: Tell me a little bit more about how an AI ops workflow would affect the work of an average organization?

[00:34:13] SX: So, I think of it as a maturity, as a way to measure how mature maybe an organization is, if you wish. So, in its most basic form, I guess most organizations today have some monitoring in place, so they might know when something is wrong and they might do the troubleshooting a bit manually. So, next stage is maybe they implemented, let's say, a more comprehensive observability solution so they can start bringing the data together. And if they're very, very mature, they probably started already thinking of how to automate a lot of the workflows and how to scale better. And I think the complexity and the need for better tooling and automation goes hand in hand, right? So, the more complex my infrastructure and application, the more desire I need to have to do, let's say use AI ops to make my users a lot more effective and efficient in troubleshooting problems. And ideally, maybe automatically, remediate some of these problems. So, I think we're still in the very early phases of like automatic remediation, I mean, beyond basic, auto scaling, et cetera. But ideally, when we get to that, obviously, we can run much larger scale, more complex applications without like scaling people linearly with the complexity.

[00:35:28] JM: How has the OpenTelemetry instrumentation changed since you joined the company?

[00:35:36] SX: So, OpenTelemetry has become more and more mature since the beginning, right? Obviously, the goal we've had, like, I think, some very good goals since the beginning. But I think what I've seen happening lately is that there is a lot more trust and adoption, especially since we have called the product, parts of the project GA. I think a lot of end users have adopted a lot more broadly. And of course, the more adoption we have, the faster the project matures, as well.

So initially, some might have started using it just for the SDKs for tracing, maybe some users adopted, let's say, the collector, which is kind of the open source, the OpenTelemetry agent. But now, we see more and more adoption of auto instrumentation slowly of metrics. Hopefully, soon, of logs, which still in kind of early phase, but definitely the product has matured a lot and has started becoming mainstream, I would say, because it now has the maturity and ease of use to get there.

[00:36:30] JM: When you look at the stack of technologies that you work on today, what are the biggest engineering challenges that you face?

[00:36:38] SX: So, I don't know application itself, you're asking that, is a very high-volume data processing system. A lot of the problems we're dealing with are, I would say, systems problems, right? How do you scale distributed systems? The data volume increases by multiple probably every year, both because the applications generate more data, but because we have more users, let's say to our own application. So, a lot of the problems and challenges we have is kind of systems infrastructure problems, of how do you rewrite systems? How do you, maybe, change the technology you use to scale better? The other is probably a lot of user experience and UI type of problems. Again, how you simplify the systems? How do you make them more accessible to your end users?

[00:37:19] JM: So, it's more of a high-level problem rather than a low-level engineering problem?

[00:37:24] SX: I mean, a lot of the system problems I mentioned, I guess, low-level engineering problems. By low level, you mean, I assume you mean low in the stack, right? So, a lot of determinism – yes, there are many, many low-level distributed systems type of problems that we have to solve for the scale we're operating in.

[00:37:39] JM: How do you triage those issues? Distributed systems problems are notoriously hard to reproduce, for example.

[00:37:48] SX: Right. So, first of all, we're using our own systems. So, we deploy, let's say, observability everywhere. And I think we're fairly advanced ourselves in terms of how much you're aware and the way we monitor and troubleshoot our systems. I would say, that's probably the best tool we have in place to help us troubleshoot these types of problems. And secondarily, at the scale we operate, obviously, you need a lot of experience and experienced engineers who have maybe have done this at this scale before. It's not easy.

[00:38:17] JM: Do you have an example of a recent distributed systems problem they had to solve?

[00:38:23] SX: I'm trying to think if there was something, let's say big enough, that might have come to my attention, because I'm not writing code anymore, myself. Actually, my role is that of the general manager. So, I'm mostly dealing with business problems these days. I cannot recall one that I can describe, I guess, well, but what I've encountered many, many – I mean, have many on call situations like happen on a daily basis, obviously.

[00:38:49] JM: How has that transition to manager gone? How has your role changed as your work changed? I guess you were a CEO before, so it's kind of a general manager role.

[00:38:59] SX: At the time I was CEO. Actually, my education is I'm a computer scientist. In fact, I started doing a PhD in computer science. I dropped out to start my first company. So, I have always been very, very close to technology. And I have been writing a lot of code myself. But I guess, as the community started becoming bigger, it's impossible to do both, right? It probably even like a disservice to everyone else, if you're like the CEO of a company that has more than maybe 10, 20 employees and trying to still be a coder, because I think you're probably ignoring a lot of the other aspects that make a company successful. So, I tried to, and over time, I became better and better at actually delegating, trusting others completely and focusing maybe on the next big problem.

[00:39:44] JM: As you begin to wind down, I'd like to get your perspective on the observability market as a whole. There's just kind of a lot of products out there these days. And I think staying competitive is is pretty difficult. There's something to product differentiation. I feel like Splunk is more on the comprehensive side, relative to other providers that might be a little bit more specialized, and maybe they do one or two things really, really well. Whereas Splunk does a wide variety of things quite well, but may lead to maybe a difficulty in just because there's so many products. So, how do you maintain quality of an overall product when there's so much product breath?

[00:40:36] SX: So, I think it's a fair assessment that we tried to be very comprehensive. But that doesn't mean that we still feel like we have the best technology when it comes to log analytics, metrics, and tracing. But our goal is need to be comprehensive, because I think that's what the customers and users require, right? As I described, the one task is to bring all of these together in one single system, and consolidate as much as possible, right? It's much, much easier to be dealing with one application than three, and probably even easier to be dealing with one vendor than three or four, right?

So, of course, there are many solutions out there, and many of them are great solutions. I'm not claiming we were the only one. But we totally tried to be the best when it comes to like enterprise scale, comprehensive, observability solution. And our focus has always been from the beginning to have all of these operators a single application with all the data being together. So, it's not an after the fact thought. Maybe we have the advantage that we designed observability, Splunk observability, from the beginning to rely OpenTelemetry and to be a single application, right? So, it's not an afterthought that we brought, let's say APM and infrastructure monitoring together. And that's what makes makes it possible and allows us to deal with the challenge we described.

[00:41:47] JM: Well, Spiros, is there anything else you'd like to add about your work or about observability as a whole?

[00:41:54] SX: I think it's a very exciting space. I believe we're still in the early phases of it. Widespread adoption is probably just starting. So, it's a very exciting space for anyone who might be interested to participate, let's say in building solutions, like what we do. Or even like, somebody who feels like, let's say, as a career in like running tools like Splunk observability. I think there is a lot of momentum and future and it's very interesting space for anyone who wants to specialize in something like this. Maybe getting familiar with OpenTelemetry as an example. I think there's a lot of opportunity out there and very, very interesting problems to solve.

[00:42:26] JM: Awesome. Well, thank you so much for coming to the show. It's been a real pleasure.

[00:42:28] SX: Thank you so much for having me, and I enjoyed our conversation. Thank you.

[END]