EPISODE 1467

[INTRODUCTION]

[00:00:00] ANNOUNCER: Highlight is a tool that helps teams reproduce end-to-end user sessions to better understand their application. With Highlight, engineering teams can replay errors with high-precision, which includes a complete session replay, outgoing network requests, dense stack traces and insight into the app's state management system.

At the same time, product teams can analyze user activity, collaborate with engineering teams and find the reason for user drop-offs to gain insight into UX.

Jay Khatri is the CEO of Highlight and joins the show to talk about browser observability and what he and his team have built.

[INTERVIEW]

[00:00:41] JM: Jay, welcome to the show.

[00:00:42] JK: Thanks for having me.

[00:00:43] JM: You work on Highlight. And in order to discuss Highlight, we should discuss debugging. Debugging typically involves a variety of tools in the tool chain. What are the disadvantages of the decoupled multi-tool debugging process?

[00:01:04] JK: Yeah. I mean, I think the decoupling of the whole process can kind of be split into two buckets. One bucket is from an audience perspective, right? Where if you're in like a large organization, maybe there's folks on support that are doing debugging. There are folks that aren't on engineering doing debugging. And there's kind of like a whole set of different teams that are in that space.

And then the other thing is, even just from an engineering perspective, there's also sort of a set of tools that companies tend to use. I can kind of go through the first thing. I think both are very

relevant in the Highlight space. I can go through the first thing. But I can also go through the second thing. What do you think is the best thing to sort of dive deeper into?

[00:01:45] JM: Let's start with the former.

[00:01:46] JK: Okay. I think a very common use case for just debugging in general is that when let's say a support ticket comes into your app, right? Someone through Intercom or Drift has messaged your company and said, "I'm having this issue." The sort of pipeline for this is that a support person takes the ticket and puts it in some sort of like issue tracking system, then passes this on to like the write team. And that write team is dependent on what part of the product is being used is dependent on lots of sort of factors, right?

Once that happens, then that specific write team then needs to figure out, "Okay, what type of tool do I need to debug this?" Right? Let's say it's most likely sort of a server server-related issue, server-side related issue. Then maybe they're in something like New Relic, or Splunk, or Datadog. Sort of looking through logs and figuring out what's going on, right? Or maybe it's sort of more of a client-side UI-related issue, and they're in tools like Hotjar or FullStory, trying to figure out what's going on.

I guess the high-level way that we sort of think about this is that a lot of these sort of leaves in the trees of what's happening in that life cycle can kind of be brought down and sort of simplified with a tool like Highlight. And that's kind of how we like to think about a workflow from a team perspective.

[00:03:12] JM: If you want to boil some of those tools down into one simpler tool, that's probably going to encompass a whole lot of functionality, right? That's like way too much functionality for one company to build, right?

[00:03:29] JK: I think so. I think so. And so, I think for us, it's important to focus on like sort of a subset of that. And what we think impact-wise and kind of my background-wise is that, from an engineering perspective, we can actually kind of boil that down as a first step.

Kind of how we think about it is if you think about the like tools like FullStory and the LogRockets of the world, and then if you think of tools like the Datadogs and the New Relics of the world, often, what happens is there are sort of different types of audiences that go into these tools to sort of figure out what's going wrong with your app. And what we think is that, given the sort of rise of like b2b companies and the fact that your customers are like sort of more high-value customers, it actually makes sense to be debugging things from more of a customer perspective.

And so, our sort of high-level mission is that, from a observability perspective, rather than going through like maybe one of Datadog's eight products or one of New Relic's large sort of offerings, we can actually just boil everything down to like a given user's session and give you a full sort of picture of what's going on, whether it's logs, whether it's metrics, whether it's what's being printed in the console, such that you can actually see what's going on and debug something faster. Does that make sense?

[00:04:49] JM: Yeah, definitely. I mean, I think there's a lot of people that are like full stack developers. And they don't know if they want to use New Relic or if they want to use FullStory. They might want just something more universal. Am I understanding that correctly?

[00:05:14] JK: Yeah, you are. You are. And I think the way that we think about it right now is that a lot of our users are not particularly frontend developers, which I feel like is historically what the use of something like session replay would be like bucketed with.

What happens in Highlight, for example, is we have like a frontend SDK that you can install that does all of the like client-side session recording. But we also are building out sort of a suite of like server-side SDKs. Where, for example, if you're like a full stack engineer, sure, you like sort of live more in the like server-side things in life. But in highlight, what happens is if you're debugging an issue, you likely know the type of customer that came in. And you can find the right sessions that sort of caused a bug.

But then from then, because you are sort of more in the full stack space, you can actually go even deeper and say that, ", a network request was sent on this session at this time. What was the contents of the network request? What did a trace look like on the backend related to what

was happening on the replay?" And so, I think for us, it's more like a different approach to how you like pictured observability when you're debugging something. Does that make sense?

[00:06:27] JM: Yeah, definitely. If you want to talk about the engineering behind this, I think the place to start is the insertion point. If I want to have observability, highlight observability, in my own application, am I installing an agent? Am I having some sidecar thing? What's the model?

[00:06:49] JK: Yeah, good question. The way we think about it is that the agent, and I guess being a small startup, and my experiences in the past, is installing an agent in your infrastructure. I think is slowly starting to become a little less and less relevant, where you can do a lot of the work that you would normally do in an agent in terms of like pre-processing metrics and stuff like that actually on the server itself.

And so, the way we think about it is that when people come on to Highlight, the first step is for them to install like our frontend SDK. It's super low-effort. They can install this like within a few lines of code and start like reporting sessions back to their app, right? And this is actually a great way to get like the rest of the team onboard. And it's honestly kind of viral within a company, because that's a great way to just show that, "Okay, we're proving our value," right?

But then the second step from a server-side perspective is that we actually just have an SDK where, depending on the language, we do this a little differently. But we run an agent on our own infrastructure that these SDKs then report back metrics and errors and things too. Does that make sense?

[00:07:59] JM: Certainly. If you want to do session replay – I remember talking to the LogRocket people about that. There's some kind of like change set logging system you can do for the frontend. Is that what you're doing to observe the frontend and be able to replay any session?

[00:08:19] JK: Yes, yes. That's exactly what we do. There's actually an open source like spec called the mutation observer that's supported in most modern browsers. And we use that to, one, get a full – What we call a full snapshot of the dom. And then we also use the same API to

get like divs after the fact. And so, we use that information to then to rebuild the dom after the fact.

[00:08:45] JM: Gotcha. What are the engineering challenges associated with recording and being able to play back high-volumes of user sessions?

[00:08:58] JK: Yeah. I think it's come to us in a few areas. I guess one is I think like compression and like where you store the data is very important. Because at the end of the day, it's a lot of data, right? We use sort of like cloud buckets and sort of technologies in the sort of more data warehouse space for that type of thing. And then from a compression perspective, we do like server-side compression. And then when we serve it back on the client when you're playing things back, we un-compress so that we're making sure that we are only storing sort of less data.

Then I think the second thing is that from a playback perspective, what happens is when you're collecting all this data, is that if you keep it all in-memory on the client, it's not sort of a fun time, right? And so, we're starting to actually build out projects where we're doing chunking. Kind of similar to what like a video codec would do. So that when you're going to a specific part in a session, rather than us having to like bring all the events in memory, like basically the snapshot and all the divs in-memory, we do the snapshots more often and then can actually split a session into different parts where the user doesn't know. But it's a much smoother experience for them from like a memory perspective when they're playing things back. Does that make sense?

[00:10:14] JM: Totally. Does the decompression of that compressed like playback set, is that compute-intensive? Does it like cause latency? Or do you just unbundle the entire user session when the developer wants to load it and replay it?

[00:10:33] JK: Yeah, it's actually not terrible. And I think the reason is we don't really touch that type of work because it's supported in the browser. We use Gzip, let's say, right? And so, when the request is received from a developer's perspective, it's just Gzip. The network request is Gzip. We're not like manually running a compression thing in JavaScript or whatever. The

browser itself schedules that. And then we get it uncompressed when we're playing it back. Does that make sense?

[00:11:04] JM: Yeah, totally so the decompression happens on the server-side.

[00:11:08] JK: The decompression happens on the client, but it's scheduled by the browser. It's scheduled by the browser before we actually receive it, if that makes sense. Before we actually are able to access that variable in-memory or whatever in JavaScript. Does that make sense?

[00:11:25] JM: Yeah. I think so. We don't need to go deep into the weeds into the session playback compression, decompression. Let's take a step back. Again, if I'm trying to have a comprehensive observability system, like I don't just want user session replay. I'm obviously going to want stack traces and ability to look at logs. There's a whole lot more functionality. Could you pick maybe the most – Whatever was the most challenging feature to build in. Because Highlight is kind of a suite of different observability tools. Whatever was the most difficult one I'd like to hear the most about.

00:12:08[] JK: Sure. I think for us, at this point, where we've had the most difficulty is actually likely in like the ingest for the replay. To be honest, I think the thing about replay is that it's like quite a lot of data. And the pipeline for which data kind of flows through our system is a little bit unique compared to, for example, like reporting errors and enhancing them with stack traces and things like that.

I think at the lowest level, we actually use Postgres behind the hood. And what happens is when data comes through – And, Jeff, is it cool? Do you want me to keep going through this?

[00:12:47] JM: Yeah, please.

[00:12:48] JK: Okay. Okay. When data comes through from like a session replay perspective, there's kind of a lot of things you have to take care of. For example, on the client, on a user's client, right? Let's say on xyz.com, right? Their client, when they're reporting data back to the Highlight backend, you have to be careful about how much data you keep in-memory on the client and how often you send data. And so, there's kind of like a tradeoff that you need to make

when you're sort of sending data across, right? Because you're grabbing a lot of data from the client. You don't want to slow down their app at all, right?

That's kind of one thing that we've spent a lot of time on especially early on, where we need to make sure that when we profile any person's browser, that there's no significant change in how their app is like actually being used.

Then what happens after data is sent over is there's a whole pipeline where we, for example, any given event, we need to give it an ID. And then make sure when things are getting thrown into Postgres, that those ids are sequential and there's no repeats. And we have a whole sort of like exponential back-off type mechanism on the client. So that if requests fail, we make sure that data gets across, right?

After that, once stuff goes into Postgres and a session, let's say, is over, then we throw it into long-term storage, something like S3. And so, that's kind of like what the pipeline for a session looks like. And it's kind of very different than one-off events or one-off metrics that you would normally report.

And so, I think that's kind of something that we've been tuning a lot so far, where it's kind of a fun problem to have, right? Where we're getting quite a lot of traffic, but also really interesting. I can go through any one of these things if you think they're worth talking more about.

[00:14:33] JM: Well, I was looking into the UI, particularly for the playback. And I think there's probably a relationship between how you store and render the playback data for a user session. And maybe it's worth even taking a step back and saying, like, when we're talking about playback, we're saying like there's a user that's using a web application. Say, they're using Facebook, and they're scrolling Facebook, and they're clicking like buttons, and they're commenting. A session playback tool allows you to see in gratuitous visual detail what the user is actually doing on the page.

And I think one thing that I see is pretty prominent in Highlight is that it's collaborative. And there's a lot of work around being able to comment and highlight different areas of a session playback.

I guess I wanted to get a sense for – You talked about the storage layer. And I guess since you wanted to be collaborative, the object model that loads into the frontend has to be pretty well designed so that you can have comments and collaboration. I'd like to know more about how you structure the object hierarchy of a session. And because there's a ton of information in a session. I'd like to get a sense for how you structure that object hierarchy of a session when it's going to be played back.

[00:16:08] JK: Yeah, that's a good question. I think actually a lot of people tend – Are curious about that. The reality is that the way that a session is represented in our system is actually quite decoupled from things like comments and other like features in the application related to collaboration and stuff.

And so, the way that that works is like a session is essentially represented on an iframe in the browser, like on app.highlight.run, right? And what happens is when someone adds a comment, let's say, we represent that comment in sort of two ways. One is we give it coordinates, right? We give it like an X, Y coordinate on the player. And two, we give it a time.

And so, when you add a comment, you're actually just writing that to the database on top of a given session, which has like a session ID. Then when we get back – Let's say you tag a friend, or you tag someone on your company, and they get tagged, now they just get a deep link to that session with that comment ID that had just been created and are like played back to that exact time and that exact X, Y position. And so, that's kind of at a high level how things work.

And I think the other thing, Jeff, is that a lot of the resources at a high-level in our application are kind of represented in the same way, right? If you think about network requests, or you think about messages that get printed in the console, or you think about even like backend errors, right? All these types of things have sort of deep links where, on a given session, you can get linked back to it. If, for example, one of your co-workers sends you a link in Slack or something. That's I think something really important when it comes to like observability and understanding debugging things. Because at the end of the day, it's not just you that's debugging stuff, right?

And it kind of goes back to my earlier thoughts about like different audiences when it comes to figuring out what's going wrong on your app, where you want to make the pass off to any other person in your organization as quick and as seamless as possible.

[00:18:11] JM: Speaking of the network requests, does collecting network request data get to be data-intensive? Or are there any other data intensive issues? If you're collecting all this data and shuttling it over the wire, is there any sensitivity there to just the volume of data being sent?

[00:18:32] JK: There is. There is. I think that's why it's important to like be cognizant of the type of storage you're using for that type of data, right? For example, when that data is coming in to our like ingest system, it's never, at least a long term, being stored in like sort of a hot database.

And so, it always will end up in something like S3 or something that's sort of more like long-term storage specific just so that we make sure that things are like in the right places or whatever. I think another thing worth saying there though is that it's often useful to be able to search for specific – Some of these like bigger pieces of data in Highlight. And so, we use, for example, Elasticsearch, or OpenSearch, the open source version, such that when stuff comes in, we can extract the data that we want to index by. Send that to those types of data storages. And the rest of it, and that's fair to say the majority of it, then goes into the right place. Does that make sense?

[00:19:30] JM: Yeah, totally. Can you talk a little bit more about the storage systems you use for these different data sets? I don't think you're dumping everything into Postgres, right? Are you using some kind of caching layer as well?

[00:19:43] JK: Yeah. We actually, right now, for the most part, use Postgres as sort of like a queue. We use Postgres. And then we subscribe to Postgres in like sort of other worker jobs to do that sort of S3 dump. For the most part, actually we use Postgres right now. And we do like a few thousand requests a second at the very least. And then on all the Elasticsearch and the S3 type stuff gets done like asynchronously after the fact.

[00:20:12] JM: Gotcha. The data comes in over the wire. I have some collection. There's a collection system running on the user's browser. That information is getting sent over the wire.

And you throw that data into – You're saying the primary database for storage of – I guess you send it over the wire and then you cue it immediately into Postgres after getting it over the wire? And then it goes into – And then it just gets queued into Postgres for being, I guess, structured and stored into S3?

[00:20:44] JK: That's correct. That's correct. We have some sort of mechanisms that we use to determine whether a session is over, right? And once we kind of like get that information coming over the wire, then we decide to like store it into S3.

Also another big reason, Jeff, why we like to keep things in Postgres is that we also support live sessions. For example, if you want to like be able to follow a customer live, we use Postgres actually. Because it's like obviously sort of more hot than something like S3. Such that you can like actually get live updates when something's happening. For us, it's like a perfect compromise between something like intermediary and something that's like long-term storage.

[00:21:27] JM: Okay. And so, you're saying it's useful, because on your side, you want to have a lot of triggering functionality and, like you said, hotness to be able to interface with that database because it's got so much information coming into it and as well as egressing from it. And so, when the developer actually wants to load a session for replay, you're saying it gets taken out of S3 and then thrown into the Elasticsearch type system?

[00:22:01] JK: No. Yes and no. The session payload, right? Those events that we talked about earlier, where it's like a big event and then the divs from then on? Those are all stored in S3, right? When a customer visits a session, a look up, a single lookup from S3 via a session ID is done, right? The Elasticsearch type stuff is actually done on the ingest side of things, where, for example, we get data from a session. For example, this session happened at this time. This was the user's ID. All that kind of stuff is thrown into Elasticsearch as data is coming into Postgres on the ingest side of things. Such that if a customer wants to look for a session, where a user's id was X, or where I want to look for all the sessions where the browser was chrome, that all hits Elasticsearch and then returns the corresponding session ID. The S3 lookup is actually only done for looking at a single session. And everything else on our UI is powered by Elasticsearch.

[00:23:02] JM: And so, can you describe like if I'm the developer and I've loaded a session, a user session into my browser, and I'm interfacing with it, what is happening on the server-side? Can you just describe the client-server interaction and what the query path looks like? Just describe that.

[00:23:22] JK: Sure. If I'm on a session and – If I'm about to click into a session, let's say, right? What happens is that S3 lookup happens. And we best basically like fetch the whole session payload into memory.

And I told you about sort of those chunking mechanisms. We can forget that for now, right? But let's say we grab everything into memory. And then actually all the logic for playing back a session is only client-side. There are obviously other things that we fetch, like sets of errors, and sets of network requests, and sets of console messages, which are done in the same sort of vein as session payload. But when you're playing back a session, actually all of it is client-side. And there's very little, like, after the fact, very little sort of server interaction. Does that make sense?

[00:24:09] JM: Yeah. You're saying, basically, if you load a session, it's not a problem to just load the whole session into memory on the developer's browser.

[00:24:17] JK: Yeah. Yes. But historically, it hasn't been as much of a problem. And I think more recently, in the last few months, we started working on some work to basically split a session into like specific chunks such that I'm able to like buffer parts of it if I'm seeking to a specific time, that kind of thing. Does that make sense?

[00:24:36] JM: Right. Yeah, yeah, totally. And so, I guess we can talk a little bit more about the frontend side of things. From what we've talked about so far, you've got a product for storing and fetching user session data. And it's collaborative, which is a differentiator. But I would like to get more of a sense of how it's competitive, because you have the full stories and the LogRockets of the world already. Tell me a little bit more about the other things that differentiate you from the competition.

[00:25:10] JK: Sure. I think one thing that we're working on in a pretty focused way is like error monitoring at a high-level, where I guess for the first like six months of having worked on Highlight, we were focused on sort of client-side errors only, which is something that LogRocket does. And definitely not something that FullStory does. But we've kind of like thought that, sure, this is great.

But in order to kind of replace like an error and like an existing error monitoring tool and be a differentiator with sort of competing with the sort of sentries and the Bugsnags of the world, it makes sense to actually build out sort of backend SDKs as well.

And so, this is actually something that we have in early access among like a few customers of ours. But that's something that we'll likely be launching in the next few months, where we're actually able to like monitor errors on your application regardless of where they are and then kind of give them to you on top of a given session. Does that make sense?

[00:26:10] JM: Yeah, totally. And so, if you have crash monitoring built in, then I can imagine that would make it a lot more appealing to somebody that's actually looking to have detailed debugging associated with the session playback stuff

[00:26:28] JK: Yep, yep. Exactly, exactly. And I think the cool thing about this is that it's also kind of very sticky at that point too, where we get – When any error comes into someone's application, they are prompted with like a Slack notification or an email and then can seek right to that exact point where it happened. That's kind of like a very common workflow actually, like beyond the sort of more support ticket type stuff we talked about earlier, where, right now, for most of our customers, they'll get client-side Slack alerts. And that's super great for like sort of early stage founders and stuff. But as you build out like the rest of your platform, the rest of your infrastructure, it becomes more and more relevant to support server-side stuff as well.

[00:27:14] JM: There are entire companies built around crash monitoring products. Can you tell me how hard is it or what is difficult about building crash monitoring?

[00:27:24] JK: Yeah. I mean, I think error monitoring, in general, it's a very interesting technical challenge. Actually, very similar to kind of what I talked about in the session replay realm, where

you have like code, SDK code, running on someone's client that you need to make sure is like respecting that client's compute, right? Because you don't want to be like taking over their application. And then you have something running on your infrastructure, right?

And so, there's always like a trade-off where you don't want to be kind of bringing in too much data or sending data too frequently. And there's like a common tradeoff there, right? That's one thing I think from just general like data collection perspective.

And then I think the other thing that's really interesting about crash monitoring is that, let's say, for example, a browser runs out of memory, right? How can you make sure as a platform that you send as much data as you can across to help someone debug it before that actually happens? Because the thing is, if the browser runs out of memory or the server runs out of memory, at that point, you can't send data either, right? Sure, your customer is obviously not doing well. You would want to help your customer in any way you can to like help them debug it, right?

There's a lot of things that you can do to sort of make sure. And which is why I think like the session replay realm is becoming more and more popular and is pretty exciting, is that you want as much context as you can around any given error such that when something goes wrong, you know what led up to it.

[00:28:59] JM: I guess it's worth mentioning, when you're talking about the frontend, crashes are often forgivable. Like a crash of an ad loading is a forgivable crash. Is there like a gradient between crashes that really need to be understood and crashes that can kind of be discarded?

[00:29:19] JK: I think there is. I think there is. I think actually something very common that happens on our customers front end is there's sort of browser extensions or other sort of third-party snippets that are maybe causing errors every once in a while and throwing things in the console, right? And they can often like kind of pollute your feed of errors on your product or on our product, right? When they're on Highlight, they'll get alerts for errors that they don't particularly care about and are not really affecting their app.

But I think that, for us, it's important to sort of enable teams to be able to either sort of ignore errors, which is something that we support, snooze errors, Be able to like ping a colleague in the product to say that this is something that I should ignore. But, yeah. I think there definitely is a gradient.

[00:30:09] JM: Maybe we could talk a little bit about the usage side of things. You can store a high-volume of user sessions. It's an open question as to which user sessions should you actually be loading and retrieving. As a developer, which user sessions should I actually be analyzing? Can you share some details about how to use a session replay tool? How to assess that high-volume of data for your product?

[00:30:40] JK: Yeah. Actually, this kind of goes back to sort of full story and those sort of more marketing tools. I think the reality is a lot of teams end up using those types of tools for like more of a support or debugging use case. And the problem with that is that it's actually very hard to be able to find the right session given something that goes wrong on your frontend, right?

And so, our goal is to actually be able to surface those types of sessions such that people are not sort of in the dark and not have like hundreds of sessions on their app and don't know what to do with them.

The way this kind of like manifests itself is sort of in two ways. One is from the error monitoring perspective, right? Which I think is a very important piece. And so, being able to instrument your back end and your front end is super valuable because now you get a deep link to a session and can actually play back what caused that error in the first place.

And then I think the second thing is from sort of more of a support perspective, right? A lot of our customers have turned on our like Intercom integration, or sort of other support integrations that we have, such that when any person sort of says something and says something is wrong, you can then go to the correct session accordingly.

I think for us, it's basically our responsibility to enable you to get you to the right sessions. Because the reality is correct. It is very hard to actually know what to look at. And that's actually

kind of very much a customer success thing from our perspective, where we want to make people successful, and that's a big part of it.

[00:32:13] JM: When it comes to thinking about the future and making a roadmap for what observability, what like kind of a full stack observability system looks like. If you're capturing stuff on the frontend and the backend, it kind of becomes a opportunity for a distributed tracing system. Do you see yourself as a distributed tracing system?

[00:32:41] JK: We've thought about it to an extent. I think, right now, for us, it's like we're working with tens of customers and they all really like our sort of error monitoring replay part of our products, right? And so, I think for us, it's likely that the next step in what we decide to build is something that we can kind of enable them to kind of even go a step further there.

I think the thing about distributed tracing is it becomes a very, very enterprise play very quickly, right? Where it's not very relevant for reasonably small teams. And so, I think it is possible that at some point we go into that space, right? As we build out the suite of tools, we end up like building out like APM, tracing, that type of stuff. But I think maybe in the short term, there are a lot of low-hanging fruits. Things like logging, things like supporting more backend SDKs, and that might be a better use of our time.

I think, right now, we are at a turning point where we need to start thinking like very seriously about like what the next step is. But I hope that gives you some color into like how we're thinking about it.

[00:33:46] JM: Do you feel like even when it comes to next generation observability tools that are higher-level, you still need multiples. You still need a collection of observability tools. Do you have like a typical suite that you see people using? Like a collection of tools that includes Highlight as an adjunct to other things?

[00:34:14] JK: Yeah. Historically, actually, like in the first sort of six, six to eight months of the company, we saw that a lot of people used something like Sentry, something like Bugsnag alongside the product, right? And I think the thing about those types of tools is at the end of the

day it's like more or less like a list of errors with some action per error, right? Which is actually very similar to how we thought about sessions.

And so, that's kind of where we thought, "Okay, if we sort of start to drill down into this and build more in that direction, it would make a lot of sense." And I think we're also starting to have some more suspicions in other regards, where folks use tools like Datadog in like some very specific subset that could be like a great addition to something like Highlight. That's, I think, how we're thinking about it, where it's like if there's a subset of tools that on top of replay are super valuable and are kind of a no-brainer, it's a great way to sort of, one, add something for our existing customers. But I think, also, from a go to markets perspective, be able to like open the market and get more customers, get more feedback and so on.

[00:35:21] JM: I'd like to go a little bit deeper on a few engineering subjects. We've talked in broad strokes about the session collection side and the storage side of things. And I think we've given a pretty good map of how things get collected. How things get stored. How things get retrieved. How they get indexed. Actually, remind me real quick in terms of the architecture, what does Elasticsearch do for you again? That's for indexing what exactly?

[00:35:50] JK: That's for indexing session metadata, essentially. If I want to search for a session that had a specific field, or search for a session that started within this specific time range, then that's what we use Elasticsearch for.

[00:36:03] JM: Got it. You have an Elasticsearch table for each user, each developer. Or an index. Sorry. Index.

[00:36:09] JK: We have an Elasticsearch index for sessions and Elasticsearch index for fields actually.

[00:36:15] JM: Okay. You have a cluster of Elasticsearch instances. And there's an instance for each customer. Correct?

[00:36:24] JK: Currently, we don't do any like customer-based sharding. But that's likely something we do at some point. Right now, it's actually like a cluster of like several Elasticsearch nodes that have like a sessions index and a field index. Does that make sense?

[00:36:39] JM: Yeah. But I assume I only have access to the sessions that are of my –

[00:36:47] JK: Of course, of course. Yeah, we don't expose that to you. That's correct.

[00:36:51] JM: Okay. We've laid out pretty much the whole architecture. What's the hardest part of the architecture that we haven't discussed to build?

[00:36:59] JK: Yeah. I think the toughest part about it is probably ingest now. I think you mentioned, you kind of like hinted at like do you have any caching layer in front of Postgres? That kind of thing. And I think that will start to like continue to bite us in the butt as the number of our customers increases and as we get larger customers.

And so, the kind of way we're thinking about that is like do we need to be like using something? Some sort of queuing mechanism? Like a Kafka, or an SQS, or something like that before we write things to Postgres? Or do we need to, in the first place, for this type of data? Because unlike the other maybe very simple like crud-like operations, where Postgres is good, this is kind of a very different query path, right?

And so, maybe we need to actually think about a different kind of data store in the first place. That's kind of like what we're I think starting to think about a lot on the sort of infrastructure side of things, where will this – Six months from now, if we're doing like five times the traffic, let's say, will this just continue to become more and more painful? Does that make sense?

[00:38:04] JM: Yeah. And I imagine, that's kind of like a flavor of a scalability problem. And I imagine, there's scalability problems all around the application. You mentioned the fact that these sessions can get very big. And you want to segment them and to make it easier to load fragments of a user session. That would only be necessary for very large sessions, which is kind of a tail scenario. But I guess, as you scale, you have more tail scenarios. Are there any other like tail risk kinds of things where unusual user situations lead to unusual problems?

[00:38:49] JK: Yeah. I mean, I think another thing about – There's kind of two things I think that are kind of maybe fun engineering issues that we've had in the past few months, which one is, historically, when we assigned a sessions ID, we would do that on the server. And so, when a session would start, we would send a request and then get the ID and then start sending data across the wire by that ID, right? That's kind of one flavor of things where we've had to change this recently to do this on the client side.

And I think the other thing that's really interesting is, because sessions end at some point, there are some kind of interesting things about how do you determine whether a session has ended, right? There are some browser APIs which will tell you that this happened, but they're not guaranteed to be sent across the wire for the exact reason that you want to close the tab, right?

And so, we have some set of heuristics that we determine whether a session has ended. And I think, for the most part, these types of things work quite well. But I think both of those two flavors of types of problems will start to become more and more relevant as we increase usage, as we increase customer account, that kind of thing.

[00:39:59] JM: Are there any issues on the side of storage and, I guess, like just backend infrastructure? It seems like from what you've described, it's there's nothing too in danger of falling over scalability-wise. Are there any weak points in the backend?

[00:40:20] JK: I think maybe not particularly other than that ingest thing we talked about, right? I think that is something that's more frequently and frequently becoming an issue, because we are doing so many writes to Postgres per second, right? And so, often, one week we'll have an issue related to like a large index. Another week we'll have an issue related to like locks, deadlocks, stuff like that, right? And they're starting to become more and more frequent just because like requests per second is increasing. I think that's kind of the big thing that we're really focused on.

Other than that, I think like we've done a pretty good job at making sure we're choosing like open source tools that at the end of the day have been around for quite a while and we have a

good understanding of how they work. But, yeah. I think that's kind of like where we're focusing our time in terms of the infrastructure perspective at least in the short term.

[00:41:08] JM: if I understand correctly, your backend is mostly written in Go lang? Is that right?

[00:41:15] JK: That is correct. Yes.

[00:41:16] JM: Okay. Are there any particular benefits you've been able to exude from Go lang? Or is it just what you're familiar with?

[00:41:25] JK: Yeah, yeah. I mean, I think the cool thing about Go is it's a good compromise between having like sort of performance and having like the ease of being able to like from thought to writing code, right? And so, that's kind of how we think about it, where we're able to get like the benefits of things like Go routines and the fact that the language is typed and things like that. But at the same time, it's actually quick like write code at a high-level. And it's not so limiting if you need to sort of turn off specific language barriers or whatever.

For us, we're pretty happy with it. And we have at this point, I think, maybe 10, 20 instances running in Go. And we haven't had any issues from like a distributed perspective either. It's a good language for us. And I think long-term, we'll likely kind of stick with it.

[00:42:14] JM: Well, as we wrap up are there any closing thoughts that you'd like to share on the monitoring space, the observability space that you fall under? Any outstanding issues in the space or unsolved problems in the space that you see? Give me a little bit more on how you see the overall space.

[00:42:38] JK: Yeah. I mean, I think we're kind of moving into an era where, because your application is not mostly on your server-side anymore, whereas like maybe five, ten years ago, monitoring was almost like bucketed exclusively in server-side stuff. That the way that people are like debugging their application is just changing quite a lot.

And so, I think I would encourage folks to – If they're not using any sort of replay tool, to sort of think about it at least at the very least. Because I think, at the end of the day, when you're trying

to like figure out what's going on, often, you're like in logs and you're searching through like all your traces or something like that. But maybe the easiest way to do this is actually to see what's going on from the user's perspective.

I think at a high-level that's kind of where our mission lies. But from like an individual engineering perspective, I think that's something that I wish that I had in the past. And I'd encourage people to at least think about at the very least.

[00:43:34] JM: Cool. Okay. Well, Jay, thank you so much for coming the show. It's been a pleasure talking to you.

[00:43:38] JK: Likewise. Thanks, Jeff.