EPISODE 729

[INTRODUCTION]

[00:00:00] JM: Releasing software has inherent risk. If your users don't like your new feature, they might stop using your product immediately. If a software bug makes it into production, it can crash your entire application. Releasing software gradually has many benefits. A slow rollout to an increasing population of users allows you to test your software in multiple real-world environments before it goes live to everyone.

A system of A-B testing different versions of your software lets you see how different flavors of that software can perform against similar audiences. Edith Harbaugh is the CEO of LaunchDarkly, a system for feature management. LaunchDarkly allows developers to deploy new software releases in a controlled fashion. Edith joins the show to discuss how to implement feature flagging, why an intelligent release process can lead to a more scientific predictable environment for software development and many other topics. Edith is also the host of a podcast called To Be Continuous, and I recommend checking it out if you're interested in learning about continuous delivery and devops and many other technical subjects.

[SPONSOR MESSAGE]

[00:01:20] JM: Software engineers can build their applications faster when they use higher-level APIs and infrastructure tools. APIs for image recognition and natural language processing let you build AI-powered applications. Serverless functions let you quickly spin-up cost-efficient, short-lived infrastructure.

Container platforms let you scale your long-lived infrastructure for low-cost. But how do you get started with all of these amazing tools? What are the design patterns for building these new types of application backends? IBM Developer is a hug for open source code, design patterns, articles and tutorials about how to build modern applications. IBM Developer is a community of developers learning how to build entire applications with AI, containers, blockchains, serverless functions and anything else you might want to learn about.

Go to softwareengineeringdaily.com/ibm and join the IBM Developer community. Get ideas for how to solve a problem at your job. Get help with side projects or think about how new technology can be used to build a business. At softwareengineeringdaily.com/ibm you will find the resources and community who can help you level up and build whatever you imagine.

Thanks to IBM Developer for being a sponsor of Software Engineering Daily and for putting together a useful set of resources about building new technology. I'm always a fan of people who build new technology, who build new applications and this is a great resource for finding some design patterns and some new ways to build and leverage these technologies, like serverless, and containers, and artificial intelligence APIs.

Thanks again to IBM Developer.

[INTERVIEW]

[00:03:19] JM: Edith Harbaugh, you are the CEO and cofounder of LaunchDarkly. Welcome to Software Engineering Daily.

[00:03:24] EH: Hey, thanks for having me.

[00:03:25] JM: I want to talk to you today about building feature flagging, but I also want to talk about the business, because LaunchDarkly is a pretty interesting case study of a successful developer tooling company and it's not under a major cloud provider. It's not based on an open source project. So there's a lot of lessons to take away from the product itself. We should start by just grounding the conversation in a discussion of what feature flagging is. Can you explain what a feature flag is and how it fits into a developer's workflow?

[00:03:59] EH: Yeah. Feature flagging is this concept of basically separate out deployment, which is pushing out all the code to release. With a feature flag what you could do is you can ship a feature off and then turn it on for selective people. For example, a pattern I use when I was at TripIt was we would turn it on for our internal users , like myself as a product manager, our UE designer, make sure everything works to our satisfaction basically doing a UAT, a user acceptance testing. Then we would turn it on for selective outside users, have to run a beta basically, but a beta where they're using actual beta instead of having to look at a beta server. This was really successful and that we could get real-world feedback, but kind of control the blast zone as it was by having only go out to certain people and having it be real-world data.

Then you can, if it's successful, start to do stuff like a percentage rollout and rollout to more and more of our population. That's what a feature flag does and it is incredibly powerful, because the mind shift that happens of, "I don't have to push everything to everybody at the same time," actually allows you to be a lot more deliberate and move faster at the same time, because you know that you could get feedback from people.

The other half of it is that if something is going wrong, like for example if the feature has a performance issue or it's just not what people really wanted, you could turn it off in seconds and instead of having to do a full-release to roll it back. That's what feature flagging is and it's incredibly powerful.

[00:05:27] JM: It's useful for things like releasing a complex feature in a way that allows you to test it with a super small segment of your product's audience. If it's a big change, then maybe you want to roll it out really slowly and you have this way of releasing it as to a super small subset of your audience and gradually ramping up that subset, or you could use it for A-B testing between different audiences with a similar feature, all kinds of use cases for it.

It sounds like a small problem to solve, but it's actually a bigger problem. It's actually a more complex problem. Why is that? Why is feature flagging as an engineering problem as a business? Why is it more complicated than it looks?

[00:06:13] EH: The joke is that it's something that sounds really small until you start to think about it. The joke is that we – Our nickname used to be bullions as a service, because people would say, "Hey, I've been feature flagging for years. It's just a flag. It's just a bullion and database on or off."

When you start to think about it though, it's a lot more complicated. The example I'd receive of different populations getting features at different times, "That's not just a bullion." That's something that you want to have as really a first-class object. That's something that you want to have visible to nontechnical users also, and you want to have it separated from a release. So then you get into having an AUI to manage all these features. You get into access controls about who could touch a feature when, and then you get into [inaudible 00:06:55] of who turned it on or off. My joke has always been that. A feature flag is really easy. Feature flag management is hard.

[00:07:02] JM: Right. I was talking to somebody about this yesterday and we're just talking about what I was working on and I was like, "Yeah, I'm preparing for a show about a feature flagging company," and they were like, "How do you have a company around feature flagging? Isn't that just config files and isn't that just a feature itself in like a continuous delivery platform or a cloud provider? How is that an entire company?" I was like, "Well, I guess I'm going to find out."

[00:07:28] EH: We actually love it if people have a config file or some homegrown solution, because that means that they've bought into the feature flags are good. Then they get to this breaking point where like, "Wow! We really depend on this for releases, and it's this half big thing that's really brittle and is always breaking. Maybe we should go to a provider for this."

[00:07:46] JM: This is often something that's associated with continuous delivery and there's a multitude of continuous delivery tooling providers out there, like CircleCI, or GoCD. There's a bunch of others. Why didn't the CD-tooling providers solved this problem?

[00:08:02] EH: It's interesting, because CircleCI was actually I think one of our first 20 customers. I actually do a podcast with their founder, Paul Biggar. What Paul said was they thought about this, but it's actually a much larger problem when you start to get into it.

At LaunchDarkly, we actually have client libraries for all the major languages, for Ruby, Python, Go, Android, iOS even C++. That's a lot of it. Then what we do is actually massive. We serve depending on the day between 30 to 50 billion, billion with a B, flags a day in real-time. If you want to turn a flag on or off, we do that in microseconds. That's a big, big, big, big task. It's not just a mini-project that you can whip out on a weekend. It's a whole company.

[00:08:44] JM: Developer tooling markets, the ones that I've talked to at least, you have things like continuous delivery, and monitoring, and logging, and I think feature flagging falls into this category as well. They're often times not winner-take-all. You have a multitude of competitors that are slugging it out, like the logging space. There's just so many logging providers and they're just slugging it out and it's like a war of attrition for pricing and go-to-market strategy and sales teams. It's both good and bad. It's good because it's pretty easy to build a business that has some customers, but it's bad because it's hard to win over customers who already have a logging provider, because like, "What are you doing? That's 10X better than the previous logging provider I have." Why is that? Why aren't developer tooling companies winner-take-all?

[00:09:40] EH: I don't really feel qualified to answer that to be honest. I don't know. I guess I'll just say I don't know.

[00:09:46] JM: Interesting. Do investors ever kind of assess or approach that question? What do you do about the multitude of competitors? Do you have to differentiate yourself sharply?

[00:09:59] EH: I guess my answer for us, for LaunchDarkly in particular, we're the first to do what we were doing. In the beginning people thought my cofounder, John, and I were just kind of flat-out crazy. My cofounder John has been out of Atlassian, but we were both really determined that we both been in software for decades and we saw this problem very acutely and we just were convinced that if we built this it would become a really valuable tool.

We had this faith and determination that we could create a category, which I think not everybody had. To your log example, it's easier to enter an existing category, unless you're completely determined and crazy like us that we're like, "We're just going to create this new thing."

Then as we went along we just started getting more and more momentum that our customers love us. I've been in software my whole life. When we go onsite, people say that we have changed our lives, that they are happier and more productive and that other customers are better. That's created a ton of momentum for us. We're at this nice point now where we've been in business for almost five years where customers who use just one job, and as you know developers switch job. They go to a new job and they say the first thing that I need here is LaunchDarkly.

I was just in New York and someone had left a customer that have been using us for a while moved somewhere and said, “I need LaunchDarkly immediately.” What is really helping LaunchDarkly right now is that we have become the de facto industry standard of not only is this the best way to do it. It’s something that you really want, that you feel like you need. The customer I talked to said they had – When he moved to a new job, dumped an existing Sprint because he said, “Until we get LaunchDarkly, and it doesn’t matter what we built.”

[00:11:35] JM: My impression of looking at the dashboard and looking at how you position LaunchDarkly relative to other competitors I’ve seen is it seems like it captures a very specific narrative, and that narrative is that developers will implement this. But it will be crucial to product managers, and managers of managers, and releases, and it’s kind of built in a way where the UI, which is what’s replacing – Like the LaunchDarkly UI which is what’s replacing the config files that people might be using as an alternative is accessible to people who are non-developers. It’s really oriented towards not only developer-specific category.

[00:12:25] EH: Yeah. I mean we’re developer first. I mean, I’ve been a developer and I’ve been a product manager. I could say if your developer does not want to do it, it’s usually not going to get done. We’re very laser-focused on making the developer extremely successful and happy and comfortable, comfortable that we’re reliable, comfortable that we’re scalable, comfortable that we can handle their load. After that, once they get really comfortable with us, then there’s this aha moment of, “Hey, I could focus on developing and then I can give the controller released to rest of that company.” That’s very liberating to the developer. It’s like, “Hey, if marketing wants to do a 6 a.m. launch, because that’s 9 a.m. in New York,” I could actually release this the night before and say, “Hey, marketing, you wake up early and switch us on.” Opposed alternative, which I’ve been through, which is the engineering team gets up at 5 a.m. to do a release and hope that nothing breaks.

[00:13:14] JM: I’ve done that.

[00:13:14] EH: You’ve been there, right?

[00:13:16] JM: Yeah.

[00:13:17] EH: Did something break?

[00:13:18] JM: Something did not break, but I have to sit there for a while and make sure it didn't break. Then it's 5 a.m. and like I'm not going to get back to sleep. My whole day is disrupted because I'm tired now and this is not good. It doesn't surprise me that if it leads to more harmonious teams, then you have people who go from one company to another and say, "I need this. I need this here."

In terms of a go-to-market strategy, it gives you something resembling kind of a network effect sort of thing. But I saw you at a RedPiont event and the topic of the conversation was – I think it was how you compete in a market for cloud software when you're not a major cloud provider, and how you exist in this market where a major cloud provider can constantly – At any time release some new offering to their bevy of preexisting offerings and their gigantic customer base that already exists and can really compete with you because they have such a big channel of people that they can just up-sell to on their current cloud products.

What had been your learnings from existing in that market? Do you feel like you have to act in a way where you insulate yourself from competition from these cloud providers? Do the cloud providers even understand how big of a market feature flagging is?

[00:14:46] EH: It's funny because we – Microsoft has been one of our biggest and most helpful partners. So we have been a Microsoft partner for about three years now. We built an integration with Visual Studio and presented it two-and-a-half years ago now at Microsoft Build, their big developer conference. They have been huge fans and supporters of us. They blog about feature flags. They blog about us. Their MVPs give demos. It's been a really successful partnership, because Microsoft is very incented right now to show that they're not the old Microsoft. The old Microsoft was the, "We rule everything." I eventually found out that they've created a walled garden where nobody knew they existed anymore. They isolate themselves. They've been extremely helpful to us and it's been a very helpful relationship for us too. All in all, Microsoft has been wonderful, wonderful to us.

[00:15:35] JM: Do you feel like that if you – If another major cloud provider released a competing feature flagging service – I mean I've actually seen the developer tooling companies that are not major cloud providers generally do really well when they're up against these major cloud providers, because I think that the developers know that if they're going to go with a vendor, in an ideal case the vendor would be existentially tied to providing that feature. With LaunchDarkly you're saying like, "We are focused on feature flags and we're going to do it better than anybody else and it's existential to us." Whereas if AWS comes out with AWS Feature Flagging, it's not existential and it might take a little more work to convince the customers.

[00:16:25] EH: Yeah. AWS for a little bit had something a little bit similar and then they actually pulled it. I think, again, what's really helping us is just, again, our entire company wakes up every day hearing a lot about feature flagging and making our customers successful. That's what we do.

[00:16:38] JM: Yeah. As you're building it, as you seeing other opportunities for product adjacencies? How do you see the product growing and expanding overtime?

[00:16:48] EH: I think we'll continue to be, like I said, just fanatical about feature flags. There are a lot of interesting use cases that come up originally. We thought it was mainly going to be around launches, hence our name; LaunchDarkly. It was from this idea that we would help people with a dark launch. What we found overtime is that feature flags are just so powerful. We have customers like IBM who use us to manage microservices.

They did a webinar with us a few months ago about how they actually use us to manage their own Kubernetes as a service, which is very meta if you think about it. We have other customers who use us to manage their own subscription plans. If you think of a subscription as just a grouping of features, they use us to manage that.

We have other customers who use us for really pretty deep backend use cases. Features are not just for users, like end users. You can use them for gaining different functionality. I was just meeting with – Do you know fubo.tv?

[00:17:43] JM: I know the company FUBU. The apparel company?

[00:17:46] EH: No. Fubu.tv is a sports app. They do a lot of video streaming of sports events. Depending on the day, the number one sports app, even ahead of ESPN. They use us for a lot of really interesting backend use cases in terms of if one video feed has a quality issue for whatever reason, it's quickly switched to another one without having to do a rerelease. They do that via feature flags.

[00:18:06] JM: They're doing load balancing via feature flags?

[00:18:09] EH: Quality balancing too [inaudible 00:18:11]. If something is cutting out, they could quickly switch. They also use us, there are devices all over the world and some devices just don't do well with new features. Android Honeycomb is notorious. You can use feature flags to say, "Hey, this cool new feature, just never show it to somebody at this device level."

[SPONSOR MESSAGE]

[00:18:36] JM: Logi Analytics is an embedded business intelligence tool. It allows you to make dashboards and reports embedded in your application. Create, deploy and constantly improve your analytic applications that engage users and drive revenue.

You focus on building at the best applications for your users while Logi gets you there faster and keeps you competitive. Logi Analytics is used by over 1,800 teams, including Verizon, Cisco, GoDaddy and J.P. Morgan Chase. Check it out by going to logianalytics.com/datascience. That's logianalytics.com/datascience.

Logi can be used to maintain your brand while keeping a consistent familiar and branded user interface so that your users don't feel like they're out of place. It's an embedded analytics tool. You can extend your application with advanced APIs. You can create custom experiences for all your users and you can deliver a platform that's tailored to meet specific customer needs and you could do all that with Logi Analytics.

Logianalytics.com/datascience to find out more, and thank you to Logi Analytics.

[INTERVIEW CONTINUED]

[00:20:04] JM: There's a preponderance of Android devices, and I can imagine you obviously don't want to include a bunch of code in your actual application to have some really long switch statement where you check the user agent and then you have a switch statement that's like 80 lines long and you're switching on all the different types of android. It's much easier to just have some kind of statement or import that allows you to defer this to the feature flag. What is the developer experience for using your product at least, or what's the ideal developer experience for integrating with feature flagging?

[00:20:48] EH: It is delightfully simple. It takes about 15 to 30 minutes to integrate us. You add our SDK, and then the tricky part is really figuring out what you want to pass us in terms of user attributes, because you can – It's most basic. All you need to do is pass us just a unique identifier. If it's anonymous traffic, this can be as simple as a cookie you're good.

You can get really sophisticated though. You could pass us IP addresses. For example, if you wanted to block some IPs, you could do that. You can pass us countries. For example, if you have rights to content in some countries but not to others, you could do that. You could pass us attributes from business analytics. For example, you could say this person has bought from us in the past 7 days a thousand dollars' worth of merchandize. They're going to get a different experience than somebody who has not bought in the past 3 years.

You can get really, really sophisticated with what you pass us, and that's the part where you take a little bit more time. Then after you have that available, if you're writing an individual flag you just say, "Hey, wrap the flag. Push it to LaunchDarkly," and then at the LaunchDarkly dashboard you could say, "Hey, here's who has rights to this."

[00:21:55] JM: The examples of frontend clients are pretty intuitive to me. If you wanted to do feature flag testing across different customer segments in different geos for example and try different pricing models across them, pricing is a very frontend-y kind of thing that you can just display to people different pricing pages.

But the IBM example you gave where they have a Kubernetes as a service product, that's a very backend-y kind of product. What are they feature flagging? What kinds of things are they offering to different subsets of users?

[00:22:31] EH: Let me walk you through a more simple example first. One thing that customers use us for is just database migrations. A customer was moving from MongoDB to – What's the other one? They're moving from one database provider to another, and that's a pretty risky operation when you think about it. Moving databases is extremely air-prone and you want to make sure that everything carries over.

What they did is they used a feature flag for a while to actually send all data to both systems and then verify on the new one that it was getting read in and then do kind of a slow cutover where they're making sure that everything was still going to both systems and then gradually ramping down and ramping up. That's was a case where they had kind of two parallel systems running and a feature flag to manage that.

[00:23:14] JM: They do that manually. Were they just looking at manually in the different – How the feature flags were manifesting or did they have some kind of automated check where they tested a bunch of different test cases?

[00:23:26] EH: Both, because they wanted to make sure that everything ran on both systems. So they had suites that they're running on both systems and they use the feature flag to push traffic to both.

The old way before you had feature flags, which I suffered through, was just to do a hard cutover with databases, which was actually pretty horrifying because there's really no return.

[00:23:44] JM: Now if you're doing feature flagging at the UI layer, you can have pretty lightweight ways of testing different copy on pricing, for example. That's very few variables. But as you've illustrated, something like a database migration or a Kubernetes as a service product is much deeper in the stack. It's much lower in the stack.

How do you handle different feature flagging cases? How do you split traffic in different ways when you have to handle these cases from very deep in the stack to much higher-level?

[00:24:22] EH: Yeah. Let me walk you through another example of how customers use us. They were testing out a new backend system and they weren't sure how it could work for them under load. This is something that without feature flags you could try to do on a staging server. What I've found was that staging servers never really replicate the real-world.

With LaunchDarkly what they did is they had their old service still running and they started pushing 5%, at 10%, out of traffic to the new one. We allow you just to say I want 5% or 10% of traffic. How we under the hood do that is you're passing us the unique I.D., we hash it. We could just do a percent rollout.

What they found is that the new system works fine until it got up to about 50% of load and then it just kind of fell over, which was fine, because we need to put it back down to zero and made some improvements to the new backend system and then retest it until that could maintain the same load as a new system. I mean it's the old one.

[00:25:15] JM: Yeah. It's more – You function more as a router. You're not actually spinning up infrastructure for these companies.

[00:25:21] EH: No. No. No. No. One of the initial ideas before LaunchDarkly was to do more spinning up new services, but that world is just so complex. We assume that you have infrastructure.

[00:25:33] JM: We have discussed the basics of feature flagging and how an engineer can implement it, how an engineer might use it. How does feature flagging affect the overall product strategy of a team and the direction of a team? How should PMs be using feature flagging?

[00:25:52] EH: Yeah. What feature flagging really does is change how you approach the software development lifecycle from being very waterfall to a much more agile. A typical use case might be a feature is in an early alpha state. Now a developer could say, "This is ready. Product manager, you pick who gets this." That's really liberating for both sides because there used to be a real battle about as a former PM and as a former engineer is, “Well, it’s a little bit shaky, and then a PM asks for it anyway and it goes out to too many customers,” and then they go back and basically yell at the engineer, “Why did you ship buggy code?”

With feature flagging, a PM could give it to a couple of people. If the feedback is really dire, like, “Hey, this just has some critical errors.” They could just turn it off without even running a new release, and that alone is very liberating.

[00:26:39] JM: I want to talk more about the broader experience that you’ve had in running a company, because there’s a lot of people out there that are interested in starting developer tooling software companies. What have you learned about managing a startup, in particular, the process of managing a startup that is selling a product to developers?

[00:27:01] EH: It’s very meta. How we change the way we run the startup has definitely changed as we’ve gotten bigger. When we’re a two-person company, it was John, my cofounder, and I and we just talked all the time. Even up to about eight people, we could still run it as a team. We had an all-team standup every day.

Breakpoint, it’s about when we got past 10 people we said, “Okay, we can’t do a daily standup every day for everybody. That’s just not manageable.” We started to have to put in place different management structures or we’d have managers and all-hands and then even we moved to a quarterly OKR system. Whereas, as I said, in the super early days there’s just like, “Hey, here’s our weekly sprint. Everybody start sprinting.”

[00:27:39] JM: Do you have a well-defined process for getting feedback from people who are using the product and maybe filtering that through customer success people and then getting the information to product managers, and the product managers turning it into features for the engineers. Have you defined that process or do you just kind of let it run and it works itself out?

[00:28:02] EH: It’s definitely changed as we’ve gotten bigger. In the super early days, John, my cofounder, and I would actually Slack channels with our early customers and we’d go onsite, because John was the one building stuff, we just both knew everything.

Now as we've gotten bigger we've had to put into place more process where we have a customer could request a feature. It goes into, use Airtable. The PM can sort and filter and look at all the requests. Then we have a feedback loop where once a feature is ready, we notify customers. It's definitely gotten more formalized as we've gotten bigger.

[00:28:33] JM: You use Airtable as the source of truth for these feature requests.

[00:28:38] EH: Yeah, it's been working well for us.

[00:28:41] JM: Why is that useful? I hear a lot about that tool and I'm not really sure how to use it myself or what it's useful for. I know where a spreadsheet is useful. Airtable, I think they don't even really know how to market it other than to tell people to try to use it, and people seem to love it, but I don't know what it's for.

[00:29:01] EH: It's one of those things where my theory is the person closest to the tool should pick and RPMs Airtable, and that's great with me.

[00:29:09] JM: Okay. Maybe I need to just try it. You have spent a lot of time in the technology industry before starting a company, and I was looking at your background and I wonder, you seem like you are really poised to run a company and you enjoy being entrepreneurial. Why didn't you start a company earlier in your life?

[00:29:32] EH: The joke is I wanted to start a company before, but I just didn't think I had any good ideas. I had been a software engineer. I had been a product manager and event marketing and I just didn't think I had any brilliant insights. The joke was on me, because what I really knew a lot about was how to make software. I had run projects. I had been a PM. I had marketed software.

We started a company, my cofounder and I, to help everybody make software, because that was what we knew. John had been at Coverity and Atlassian and he basically said the Venn diagram was – He knew how to make developer tools and so did I. Why not do that?

[00:30:06] JM: Yeah. Do you wish you would have done it sooner? Was there some inflection point where you received some piece of advice from somebody that caused you to finally do it or was it just a gradual process of internal recollection?

[00:30:21] EH: It was just a really tricky step. I mean John and I were both midcareer. We're in our 30s. I was a director. He was a manager. We saw this one path which was just basically, "Hey, keep doing what we're doing."

I think for both of us there's this keen interest of we both have this itch of, "I wanted to start a company." I finally kind of talked to him and myself into it by saying, "Hey, if this doesn't work out, we can go back to kind of the old way." I don't want to be 55 or 65 or 75 and say, "Hey, what if I tried that startup?"

[00:30:51] JM: Yeah. Yeah, fair enough. As we begin to wrap up, I'd love to know about what the future is for the business. How do you see LaunchDarkly evolving in the next 5 to 10 years?

[00:31:03] EH: I think what I touched on earlier, which is our base is absolutely making developers' lives easier. I read all Net Promoter surveys and a quote I saw which I loved was, "LaunchDarkly is catastrophe savior," because we provide a kill switch where if something goes bad in our release, you could just turn it off. That's definitely our initial strategy.

Beyond that, there're so many ways which we enable the entire enterprise. The things I talked about about product managers being able to manage feature separate from release. Then even marketing, being able to manage features. For example, if they want to do an 8 a.m. launch in New Zealand, they have the power to do that without having to have a developer wake up to turn something on.

Even more into the enterprise with stuff like if you have sales people out in the field that they could turn features on and off the customers. Customer success can turn something off if a customer complains. Even finance, if somebody is non-paying, they can turn a feature off, or if somebody signs a contract, they could turn it on. There are all sorts of use cases about how software really helps the entire enterprise, not just the developer.

[00:32:03] JM: Right. It sounds like it's turning into a dashboard where people throughout the organization can turn on and off different aspects of this piece of software whatever they're using LaunchDarkly to build and you don't have to have a ticketing system where the engineer has to manually create this sort of thing. You just put all the power in the hands of customer success people, or PMs, or whoever is not the engineer, because the engineers have really scarce time.

[00:32:33] EH: Yeah, I think it's the liberation of code where developers can focus on developing and building and the rest of the org can be able to entitle and enable the software.

[00:32:43] JM: Actually, this is reminding me, I should have asked you about this, but how has the internal infrastructure at LaunchDarkly, what you're running on, how has that evolved since you started the company?

[00:32:55] EH: Oh, a ton. I mean, we serve over 50 billion feature flags a day. We actually run LaunchDarkly on LaunchDarkly. We have a separate instance of LaunchDarkly to run ourselves. The architecture has definitely evolved just to keep up with the immense volume that we do.

[00:33:09] JM: Which cloud provider do you use?

[00:33:10] EH: We use AWS.

[00:33:11] JM: Okay. Any particular services, like newer managed services that have stood out to you as being particularly useful? Can you tell me more about the backend infrastructure management?

[00:33:23] EH: We wrote a post that really gets into it. It's pretty complicated, so I would point you on that one. It's on StackShare.

[00:33:29] JM: Okay. All right. Cool. Well, we will put that in the show notes. Edith Harbaugh, thank you for coming on Software Engineering Daily. It's been great talking.

[00:33:35] EH: Yeah, thanks for having me.

[END OF INTERVIEW]

[00:33:39] JM: Fission is an open source Kubernetes native serverless framework. Fission allows you to easily code serverless functions in any language and have them run wherever you have a Kubernetes cluster, be it in the public cloud, in your own data center or even on your own laptop. Fission automatically manages the infrastructure for you, so there’s no need for any in-depth knowledge of Kubernetes, no containers to build or registries to be managed.

With Fission, you can essentially run your own Lambda-like service benefiting from the speed and the cost savings of serverless on any environment. Go to fission.io to get started. You can use Fission to create serverless API backends, implement webhooks, create event handlers and more all without having to bother with managing infrastructure, let alone Kubernetes at scale.

To learn more about Fission and to try it out, go to fission.io.

[END]