EPISODE 1113

[INTRODUCTION]

[00:00:00] JM: Dev.to has become one of the most popular places for developers to write about engineering, programming languages and everyday life. For those who have not seen it, Dev is like a cross between Twitter and Medium, but targeted at developers. The content on Dev ranges from serious, to humorous, to technically useful. Dev contains a set of features which appeal to a developer community such as the ability to embed code snippets in a post. But for the most part, the entire app is generalizable to other types of communities. Hence, the motivation for Forem. Forem is an open source project to make it possible to spin up instances of communities that are like Dev, but for other communities, such as mixed martial arts, or doctors.

Ben Halpern is the creator of Dev and Forem, and je joins the show to talk about the Dev community and his long-term goals for what the Dev team is building. Full disclosure, I am an investor in Dev.

[SPONSOR MESSAGE]

[00:01:01] JM: Today's show is sponsored by StrongDM. Managing your remote team as they work from home can be difficult. You might be managing a gazillion SSH keys and database passwords and Kubernetes certs. So meet StrongDM. Manage and audit access to servers, databases and Kubernetes clusters no matter where your employees are. With StrongDM, you can easily extend your identity provider to manage infrastructure access. Automate onboarding, off-boarding and moving people within roles. These are annoying problems. You can grant temporary access that automatically expires to your on-call teams. Admins get full auditability into anything anyone does. When they connect, what queries they run, what commands are typed? It's full visibility into everything. For SSH and RDP and Kubernetes, that means video replays. For databases, it's a single unified query log across all database management systems. StrongDM is used by companies like Hurst, Peloton, Betterment, Greenhouse and SoFi to manage access. It's more control and less hassle. StrongDM allows you to manage and audit

remote access to infrastructure. Start your free 14-day trial today at strongdm.com/sedaily. That's strongdm.com/sedaily to start your free 14-day trial.

[INTERVIEW]

[00:02:31] JM: Ben, welcome back to the show.

[00:02:33] BH: Yeah, thanks for having me.

[00:02:34] JM: You are the founder of Dev, and that's the place we should start for people who haven't heard of it. Explain what the Dev community is.

[00:02:42] BH: Yeah, the Dev community is a social network for software developers. We seek to both be inclusive in valuing of network-based education. Everyone is teaching each other, because this job is hard to do in a silo. But, also, just sort of pushing social networking forward, and that's kind of been the journey we've been on with any platform. It's really human-based. It's really about making connections, friendships and ultimately following your passion.

[00:03:14] JM: Why do people write on Dev? And explain what the writing interface is.

[00:03:20] BH: Yeah. Dev is definitely a blogging-first platform. We also have lots of discussions in other sorts of format. But we basically take this one type of post sort of like as if it's a tweet, but longer, kind of just what blogging has always been. And folks blog on the platform because it's designed to reach people who might care about similar things. It's not going into a silo and then maybe playing the Hacker News lottery, which might be sort of tough to do if you want eyeballs. But also tough to do if you want moderation, the way we do it, which is definitely a different way than, say, a Hacker News or if you're trying to do Twitter. Really, the community is there. So people bring what they want to say to the community.

[00:04:10] JM: So in some ways, it's a niche that is not filled by Medium. It's not filled by Reddit. It's not filled by WordPress, which is kind of surprising. It's kind of surprising that there is still an existent niche for people to express themselves, people to communicate. Why is there an existing niche? What is the niche that was previously unfilled by these other platforms?

[00:04:38] BH: It's been quite the journey, but originally a big reason why I wanted to take this on is sort of the realization that there's a few things at play. One, maintaining any software including a blog is really hard. And I remember a few moments where I would just occasionally visit like an older blog with really great new content. And as a software developer feel for why the person is not updating the actual software, maybe it doesn't have HTTPS because that was not a thing and it's not so important, but it's just kind of like maintenance is difficult.

So then that's where things like Medium come into play. It's a really smooth, solid interface. But there were problems from the beginning, and that Medium didn't have syntax highlighting. And regardless of how you feel about syntax highlighting, it's sort of table stakes if you want to do software development blogging, and not that that was the be all end all, but it's sort of this idea that, "Oh! Medium actually doesn't care that much about developers as a cohort. They're an important technical audience," and developers kind of got to liking Medium at one point. But then Medium, the publishing platform, just sees developers kind of as an edge case. And any passion really is not the edge case for the people who are part of that passion. Whether it's literally your passion or just something you need to do a lot of for work is learning about software development.

And Medium just inherently has a hard time doing that for everyone. And then ultimately, Medium had to question their own business model. They had to make a lot of changes. It was just always a rocky journey that it was hard to be a part of if you really cared about the craft.

[00:06:24] JM: What's your philosophy around design? The site is well-designed. I want to get into the architecture, the engineering. But just to stay on the top level a little bit more, tell me about your perspective on design.

[00:06:36] BH: Yeah. Our design has gone through a lot of iterations. Originally I was definitely the chief designer. Now we have two designers on staff who've really taken hopefully some of the good parts, but also really lately polished things. It has really come a long way. But my influence is definitely one to sort of think that the design should be expressive of the idea.

I was just talking to one of our designers today about the current redesign of the comments area, which is trying to bring some uniformity and polish to the design. And just into the little things like the way we do the border, which currently sort of frames the comment, which I think gives it a certain element of importance in like the way it shows off the content. And we really try to let people express themselves and their work through the design. But then also have the platform express our values, inclusion, creativity and things like that. And it has to be subtle. It has to be understated. You don't want the design screaming at you, which is kind of I tended to do a little bit more of when I was like chiefly in charge of that. But it's really a journey and it's really speaking to people and you have to be thoughtful about it.

[00:07:50] JM: All right. Well, describe the software architecture of Dev. We talked about that a bit in a previous episode and it was basically a Rails application. It was my understanding. But I imagine it has become a little bit more involved. So, tell me the software architecture.

[00:08:05] BH: Yeah. I would say that these days, it's still mostly a Rails application. We've actually tried to, in some ways, pair things down to make it even more of just a Rails application. And a big part of that is because of where we want to go with the journey. We've really tried to keep things simple so that we don't have to have too many different services running in one container, anything like that.

The core applications still is as much as we can think about it a Rails application. The one wrinkle, which has always existed, is that we rely heavily on edge caching. So that's the new hotness. I mean, we have been heavy users of some of the coolest features that Fastly offers. And Fastly's stock is up about six times in the last few months. So I can tell that like it's popular, it's evolved. It has also really perfect infrastructure.

So we cache content at the edge for optimal performance, and we really want the site to be performant for everyone. The one user experience consideration that you really can't do without is performance for a site that is based mostly on consuming content. And then otherwise, creating content, but it's not like we're doing anything that requires extensive computation to load the page. So let's actually do this little computation as possible in sort of raw HTML.

And that is the core design. And then we now have mobile apps. We have iOS and Android and we actually continue to mostly just run those as web apps in a light shell overtop of the native app. And that's sometimes a frustrating existence to live in, but it's also really, really awesome to be able to just know that you're shipping one web app and it's the same everywhere. And there're definitely a lot of things to think about in those terms. But it's mostly just been a journey that we're trying to make the most of each opportunity.

[00:10:06] JM: Yes. The architecture is really well-reasoned, from top to bottom. I did a bunch of these shows on Facebook a while ago, and this funny period of time where Facebook tried to do everything in HTML5 when they were getting their mobile apps ready. The time was not good to go all-in on web for your mobile applications in the early days of Facebook. But today, they probably could do that. They could probably go all-in on web technology similar to what you've done just because the edge layer has gotten so good. The browsers have gotten so good. This is kind of a random question, but do you think Facebook could have the same architecture that you have today? Let's just go totally counterfactual.

[00:10:54] BH: In a simplified way. I'm sure they do things a little differently given their resources. But to this point, yeah, absolutely, I think. Luckily for me, Facebook is making these choices while I was in college. And for whatever reason, I was keeping up-to-date on it. That's kind of like interesting. I remember reading about this as it was happening. I remember thinking about it. And that's sometimes a good way to see history is when you really have the perspective of what was going on. And that was because I was like interested in social media as a genre and I always have been. I've been doing Forem since I was a kid and things like that. Without realizing it totally explicitly, I was always into this area.

And Facebook's choices early on were because of the same reasons we kind of want to make these choices. We really value simplicity of our shipping process so that we can move fast and hopefully not break things. But we try to push to production as often as we can. We try to create a scenario which enables creativity and fluidity and the capacity to constantly evolve. But between now and when Facebook was going through growing pains in this area, iPhones and many Androids have become more powerful than their desktop equivalent in a lot of ways especially in rendering. And this is backed up by pretty much every benchmark you can talk about.

And then it ultimately comes down to network conditions and performance, which you're going to struggle with no matter what your stack is. And it's frustrating to live in an iOS land where they still are pretty inconsistent with their rolling out of web features. And that's just kind of frustrating across the board. If Apple would just provide a more consistent experience with WKWebView and some of what they're doing, it would really close the loop. I'd say that's kind of the last piece that has to fall for this to be a little bit more common. But, yeah, the idea is good. The path was clear also because this is what Basecamp does. And Basecamp are the chief core team behind Ruby on Rails, which is our stack, and we really feel like there's a sense of camaraderie between the different users of the technology. And Rails, if it ever went away, it's certainly having a renaissance now with some extremely popular Rails apps these days. Shopify's growth has really buoyed the whole ecosystem. Pay.com is making us feel like Basecamp's success is not going away anytime soon. There's just a lot of good that stuff happening in Rails end. So anytime we can swim with the current there, we're feeling pretty good about things.

[SPONSOR MESSAGE]

[00:13:50] JM: Join GitLab on August 26th, 2020 for GitLab Virtual Commit. It's an immersive 24-hour day of practical DevOps strategies shared by developers, operations professionals, engineers, managers and leaders. The attendees will hear from the US Air Force and Army. The GNOME foundation, State Farm, Northwestern Mutual, Google, more companies, many more companies about problems solved, cultures changed, release times that have been reduced. You can come and be part of the community that is just as passionate as you are about DevOps. You can register today at softwareengineeringdaily.com/gitlabcommit. Register for GitLab Virtual Commit by going to softwareengineeringdaily.com/gitlabcommit. Thanks for listening and thank you to GitLab.

[INTERVIEW CONTINUED]

[00:14:46] JM: The architecture, basically you do everything you can to pre-calculate static HTML and just serve static HTLM from anywhere. And that works pretty well for a blogging platform, for a text-based platform. Are there types of dynamic content that is tricky to do that

way? I don't know. Animations, or video, or some kind of messaging system? Is there anything that's hard to do with static HTML?

[00:15:21] BH: Well, it's only static HTML to the extent that the server cases. Once it got shipped to the browser by any means, it's pretty much fine. We have a few constraints, but over the years, this is the fourth or fifth time I've talked to you about this on the show. It's been a pretty fun journey that we've been in discussion sort of throughout this because so much has changed. But early on, the constraints were of bigger importance, and there was more stuff we felt like we couldn't do because there is just fewer people working on the team. It was harder to accomplish things.

But these days, these constraints, you have to be aware of them. And when we have new people, sometimes it takes them a while to figure out or remember that you can't do certain things you're accustomed to just based on other inventions. But we can pretty much do anything we want. We can ship really any type of content. It's just we just can't do it based on certain specific user information, but then we can do that asynchronously if we need to. And then, with regards to animations and stuff in the native context, there's going to be some constraints about how you do animations in native or whatever. Like certain native gestures are not going to be as beautiful as they would be. But you sort of live with us constraints and learn what you do well and what you don't do well. And have native bridges for our native apps to where we do audio and video, because we just felt like audio and video is something that the native – The phones were just always going to have a leg up on. So we sort of have a communication bridge so we can do a handful of native sub-views or native interactions with the APIs. And we do that where we can achieve the most leverage. But we stick to web technologies most of the time.

The web team, which is like 90% of the tech people on our team, or 95, we basically just have a couple people working on native right now. Of course, we'd like to expand that in the future. But the web team mostly barely even knows what's going on with the native side of things, and that's good that they don't really have to worry about that and we just kind of make it work a little bit better with some glue and some important APIs. But it's designed so that fewer people have to be super-duper concerned about the way we deploy and the way we roll out features and things like that and mostly maximize for thinking about the user experience and thinking about

like what we can do the most of right now. And a lot of the times that's like just shipping small improvements on the website.

[00:18:07] JM: It's amazing how you could just focus on performance when the web is now in a place where scalability is not really a difficult problem. Scalability is kind of taken care of by the primitives at this point, whatever primitive you're deploying to. Are you still using Heroku?

[00:18:26] BH: We are still using Heroku for the core Dev app. But for the stuff we're going to talk about later in terms of what's next in our journey. We're shifting to some other infrastructure choices and stuff, which are still sort of a work in progress. But, yup, Heroku still serves Dev and it's been great.

[00:18:43] JM: Right. Yeah. I can't imagine you have too many issues, whatever. You just turn up the dial, scale up the dinos, pay the money. Not a big deal there. But I guess we could spend a little more time on Dev, then we'll get to Forem. But are there sources of technical debt? Where is the technical debt with Dev?

[00:18:59] BH: So, we haven't really talked about this yet. But for anyone who doesn't know, Dev, the codebase, is entirely open source. The technical debt is certainly there. But compared to any other project I've worked on that's lasted this long or had these types of issues, our technical debt is somewhat limited. I can't think of anything that is a major, major area of debt right now because we've asked people to look at the code and to put their eyes on things and to help us work things out. So a lot of areas that would otherwise get ignored forever like just get worked out a little bit faster. People can point out specific issues. And that was a big part of going open source. A big reason why we decided to put all our code out there is so that more people could have an eye on things. Question what we're doing. Point out specific bugs.

And so there's technical debt, but it's less so than I think a lot of organizations in our position. I think we've always put an emphasis on giving our team the opportunity to re-factor pretty consistently. To not constantly rebuilding features. To try to be thoughtful about these things. Nothing is perfect, but I think that's not a big problem for us compared to a lot of our peers.

[00:20:15] JM: And with that, I think we can talk about Forem. Because we've talked about Dev a bunch of previous episodes, and I bet most of the listeners are going to be familiar with Dev. Forem is the generalization of dev. Basically, the idea that Dev is this social network for developers. You've kept the functionality pretty paired down and minimalist in the sense that you don't have anything that's like super specific to engineers. I guess you have marked down like the most forums probably would not need markdown, but then again there's not much – I'm sorry, not markdown. Syntax, like syntax structuring. In a forum for mixed martial arts, you would not need syntax highlighting unless it's – Well, maybe it's a very specific – Maybe you can have a forum for mixed martial artists that also are software engineers. Why not? But explain what Forem is.

[00:21:12] BH: Yeah. First of all, Forem is spelled F-O-R-E-M, and we have the.com, forem.com. And the word Forem stands for for empowering community, which was actually just what we settled on as our mission statement before coming up with the name. So we decided that our product was for empowering community, and that was because we had this idea that we could be a little bit broader than just software development. And not like Dev itself isn't a great business to fall back on if this whole Forem thing doesn't work. GitHub and Stack Overflow have proven that developer attention is pretty valuable, but we felt like the fundamentals of what we're doing, our emphasis on moderation, our emphasis on allowing people to go deep on their passions and our emphasis on, honestly, like the technical craftsmanship that goes into building an open source version of this really had legs in various domains. We felt like there is an idea around a sort of decentralized version of Reddit, which has a lot of the niceties of Medium and just has a lot of stuff that really caught on with the Dev community, and we really felt like myself and my two founders, Jess and Peter, are not like pure all we think about is software development type of people, and that really helped us make Dev into a really human-friendly platform in the first place.

Yeah. So this was vaguely part of our journey for quite a long time. We had these ideas for a long time, but the clarity came gradually. Of course, the name is – That came last of all. So Forem is sort of just what we've been working on for a while in some ways, but it's kind for the clear picture of what we've been working on without even really quite realizing it.

By the time this comes out, we will have changed over to github.com/forum/forem, as the organization and repo behind the project, because it think we'll try to get that done this week. Yeah, we just announced a couple weeks ago that Forem is the name of the projects.

And you alluded to mixed martial arts. We have one community called this mmalife.com, which is alive and starting to be kind of active. And we have a couple other communities out there. Right now, we have not made it very easy to get up and running, because the hosting story is a little bit convoluted right now. We haven't even really published details around how to even do it best on Heroku. And we want to really emphasize the best way to do it. Period. Whether it's Heroku or any other cloud provider or anything like that. We want to provide it deployable as a service.

We have these early adapters who have dealt with the pain and just gotten up and running, because they couldn't wait. But the true story is still hopefully just a few weeks away from us getting a few more people up and running. But then within months, or in six months, I really suspect we'll be stable and supporting thousands of communities.

[00:24:33] JM: I'm sorry. The deployment, you just might want to make it possible to deploy easily to Heroku, and render.com, and GKE, and whatever. You want to have instructions for all these things. But then you also want to have some kind of basically a SaaS offering, right?

[00:24:50] BH: Yeah. We plan to provide something maybe called Forem Cloud or whatever we name it, which is going to be a SaaS offering. And we suspect we'll be really good at hosting forums and we want to do that, and mostly we want to be able to create a situation where we can continue to deploy quickly to all of the forums and create a story where we can keep forums up-to-date really well.

But if other people want to deploy on whatever infrastructure they choose, we really want to make that easy. In a lot of ways, we're just our first customers in getting this right. But ultimately, we want our technical community to have as much from that this is possible. I want at create a forum for my family, because I think it's actually a nice use case, because we have these like text message channels, which I find hard to keep up with and hard to kind of store pictures like I might. And it's highly private. And it's one of the few things I'd really want to host myself if

possible, and we want to give our technical community the tools to really be community leaders, whether or not that's in software development, or even just within their own family. We're excited about what the future holds. That picture is still a little blurry in terms of what we're going to come up with.

[00:26:12] JM: Just drilling on specific technical questions. So if you fast-forward to this future where you've got communities that people are deploying on GKE and Heroku and whatever else. And then you've got the Forem Cloud and you've got all these different deployment mediums, and then you want to have the optionality of adapting certain things. Let's say a new reaction. You have these different reactions, like a heart, or claps, or unicorn icon, and there's some updates that you might want to push out to all of the instances. There're some updates you want to push out to some subset of the instances. There're different ways you can do these kinds of updates to the overall system. You'd like to give people the option of updating, but you'd also like to have the potential to push it out. How do you solve that problem? Or have you studied how – The closest analogy I could think of is like a WordPress solves that problem.

[00:27:10] BH: Yeah. I would say we thoroughly have not solved that problem, but we've been avoiding the pitfalls of just doing it the exact same way as WordPress. We're too early to say that we fully know the answer, but we're sort of vaguely shaping that out. And that's why we're being pretty slow with this part of things, because we do really want to make it easy to get updates and for updates to get rolled out with great consistency.

WordPress grew up in a world where software updates happened a certain way. And seeing the way browsers get updated these days, the way certain updates happen and the way our software really is pretty opinionated about how things work and how we don't enable like, literally, any website the way WordPress tries to work for. WordPress does blogs, but it also does landing pages. It also does social media via BuddyPress and things like that. We really feel like social media and Forem is enough of a staple of the web that we don't have to kind of be all things for everyone. And hopefully that simplifies our deployment outlook, but these are thoroughly unfinished problems.

Our worst case scenario is that we do have like a quarterly deployment to a lot of the stack, and it's not as magical as we'd hope. That's probably still kind of table stakes and not the end of the

world. But we're gradually kind trying to stop these problems out one at a time and trying to really emphasize doing it ourselves at first and supporting our own Forem Cloud until we really, really figure out the story. But early on, folks have been updating pretty consistently along with us and we just want to make that as much a part of the culture as anything, like download the latest daily and just keep rolling with the updates.

[00:29:06] JM: Are you starting to architect the Forem Cloud yet?

[00:29:10] BH: Yeah, we are, but it's still somewhat trial and error at this point. Again, by the time this comes out, we'll probably have some stuff posted on it, but in such a way that it can be like it can in break and it's not critical. But a huge part of our plan is to keep our offering pretty simple. I said like we are a Rails app. We try to bundle everything into that concept. We don't have such a convoluted architecture that the cloud portion is going to have to be like this huge giant hairball. We try to keep things pretty simple.

So, I think ultimately our offering is going to kind of keep it simple. Like we're a furom, it's like not the most complicated software in the world. Some parts about what we – The challenge ahead are inherently complicated, but we want to try to keep the software simple wherever we possibly can. We just have one main database, that's Postgres. We do some things in Redis and Elasticsearch, but we really try to keep things in the Rails kind of way they do it, which is just not so typical. A lot of people tend to have their architecture sprawl, and we've been pretty slow and steady with that part of things as much as we can, because we do want to keep it simple so that the complicated stuff only has to be this stuff that's actually complicated.

[00:30:34] JM: Do you envision Kubernetes playing a role in the standard deployment?

[00:30:40] BH: Probably. This year, we feel like our main offering is simple enough that we probably don't need Kubernetes. We can use basic containers or VMs and stuff like that. I can probably comment more on that, but I'm really not sure exactly where we're going to land. We're trying a ton of different things, and none of it is so complicated that we can't tear down and go in a different directions. So we're sort of in a bake-off right now in terms of like different little ideas that might – In a few weeks, we'll have a really good idea what we're actually doing long-term.

[00:31:14] JM: Well, give me the gritty. Give me the dirty, undecided experiments you've been – So you're trying to play it via VM. You're trying the play via whatever, like container instance. Tell me a little about those experiments.

[00:31:27] BH: I'm trying to talk about it in a way that I'm – It's enough of something I'm not really worrying about in the details that I'm not even like sure what our latest is. This is actually like we're a big enough company that I'm not actually sure what – I have an idea about some of the choices we've been making in terms of just like Linux stack and stuff, but it's efficiently sort of outside of what I care about that I don't even really feel like I have all that much intelligent to say.

We've been making a bunch of sort of little experiments about how the deployment should work. We've been trying to pair-down how we do things and put them into a box. But it's just like a Linux machine. And the difference between the options is kind of like either above my pay grade or just below what I tend to care about. I've been trying to focus on making it as simple as possible for our systems team to get this right. But it's kind of cool. It's a little bit like outside of my domain to really provide great answers with, which I'm kidn of happy about forever. It was just like I knew everything.

[00:32:35] JM: That's fine.

[00:32:37] BH: Yeah. I mean, just genuinely, like I know kind of what we're doing, but I don't even really think I have anything smart to say about it. I have smarter people working on this stuff.

[00:32:46] JM: Yeah. It's good that you're in a position we can actually delegate. So how big is the team at this point?

[00:32:50] BH: Including some part-time people here and there, because we're a globally distributed team. We have different people in different structures of work and stuff. But we're about 27 people.

[00:33:01] JM: Gotcha. And the revenue streams, we could discuss these from the business point of view. Original version of Dev has advertisements, because it's a social media website. So it's easy to imagine each of these instances having some kind of advertising system. But the core Dev platform also has good ad revenue potential. Are you thinking much about revenue streams at this point?

[00:33:31] BH: Yeah. So we want empower each individual community to monetize in the most effective way possible, and some of that was sponsorships and ads. Some of them might be a membership model depending on the community. And we're going to be mostly be focused on making that technically possible and then have the communities figure that out.

And then we're going to charge based on usage and how big their database needs are, and we're pretty excited that our revenue model is basically selling a SaaS. We hope we grow like a social media company, but we monetize like a SaaS, and then hopefully provide trust and value like an open source company. And we suspect that within ecommerce, within membership models, within sponsorships, that individual communities are going to have a little bit more creativity over how this is done.

And we've even experienced this with Dev. So our best monetization model right now are these campaigns we run, which are essentially online hackathons, which the community gets incredibly excited about. Organizations are pretty blown away by the engagement in something, which otherwise if it was just a sponsorship or something, really, people wouldn't pay attention to or notice. But we get thousands and thousands of users engaging. Hundreds of people creating fully-fledged submissions, and it's such as a cool Dev way to monetize, because everybody really loves it and we make it pretty easy to sort of just click out if you don't think it's very interesting.

And then we hope that this – We call it campaign mode. Ultimately, any individual community can make really interesting fun use of this, which maybe monetizes and maybe it's just part of the contribution mechanic for the community. And it kind of speaks to the idea that, say, a medium does not have the organizational capacity to get super creative within a domain. Medium.com cares about publishing. They don't have the institutional creativity to treat each domain as something they care deeply about and then they can monetize in a creative, user-

friendly way. But by a giving a lot of powerful software to community leaders, we hope that they will be able to monetize through whatever is the most effective and interesting channel they have as long as we can help them gain membership, help them roll out the features that are sticky and that people really love engaging with, and then giving them as many tools as we can to help them monetize.

[00:36:17] JM: And so it's worth pointing out that there must be some design constraints that you've imposed on the core site, because you want to have a set of features that every single community needs, but you don't – Again, it's very easy to imagine hackathons being useful for Dev, but may or may not bear fruit in other instances. Can you imagine ways to pick and choose the functionality for the other communities? Has there been anything from Dev that you've had to strip away because you had to restrict it for those other communities?

[00:36:54] BH: Yeah. Dev, since we began, have been doing lots of different interesting things within the community to promote our values, to fund situations, to educate. So an example is She Coded, which is a campaign we've done for the last four years, I think, and it's all about celebrating women in tech and we do it on International Women's day. We feel like it is less driven by simple platitudes and really is a community-first event that people really have loved since we launched it.

In the first few years we ran it, we mostly like hardcoded whatever the new design was. We really made She Coded a concept within the app. But this year, in a forward-looking direction, we decided this is a year we're going to strip out all She Coded code and really develop up campaign mode, which is more of a generic way to spin up different types of functionality. So big campaign banners, but the functionality doesn't really care exactly how they work or what goes in them. Highlighted tags imposed so you can use the features, like tags and like posts and those sorts of things as building blocks for all sorts of interesting fun features.

So after we built campaign mode, we used that for every different type of campaign thereafter. So, hackathons, we did Earth Day celebration where we had an AMA with an animal sanctuary person. Not that that had anything to do with coding, but we just thought it was a fun idea and it was a little bit of a little break and people had fun with it. And all of these things were part of the same infrastructure of this idea of a campaign.

And we are hosting an online conference later this month, July 23rd and 24th, which is also making use of campaign mode. So Code Land is going to be this really cool custom use of the site for an online conference. But because we've built in enough tooling, it kind of works without any new code. There's code running, but it's running kind of as a plug-in and not is actually part of the application code. And this is our idea of turning the app into more of a kernel for a lot of interesting functionality and trying to strip away stuff that is hyper-specific and really get that out of the core app and enable as a platform people to build in in more different and interesting ways.

But because each app is its own instance, people can kind of do this with enough freedom that they're not having to support 100,000 different types of use cases all at once. I imagine if Facebook builds a custom coronavirus dashboard, that's kind of just built on that one Facebook instance and we get to kind of think about things a little differently.

[00:39:57] JM: So, you've had first-hand insight into the changing Internet consumption habits due to coronavirus. What have you seen? I mean, I've seen my own interesting stuff from just how people are consuming podcast content. It's kind of been up and down and up and down and there seems to be waves of how people are responding to Internet content consumption. What have you seen in your own business?

[00:40:27] BH: Yeah. It's hard to see strictly within our business any radical changes, except that we do podcasts. And I think that has changed a little bit. We brought in CodeNewbie be as part of our organization last year. CodeNewbie does podcasts. They do this conference, and that's where we – They've been doing an in-person conference called Code Land for a few years, and we thought it was best conference the world.

Our first attempt at doing that conference got shut down. That sort of was weird. And then podcasts also have been sort of like up and down in weird ways. And then we launch our own Dev Discuss podcast. So if you search Dev Discuss in whatever podcast you subscribe to and you can kind of follow along with some of our interesting discussions with community members and stuff like that.

But the core application has kind of trotted along like normal. It's hard to really plot coronavirus that much. The biggest impact on traffic, like or user behavior or anything, was like in January, Google rolled out an algorithm adjustment, which hit our traffic briefly. But then things rolled back to normal and we're like higher than ever. But it doesn't seem like that's a big coronavirus thing.

I think if our main usership was not software developers, things might be different. But I think people have been consuming our site the same way they always have, like while they're at work, whether it's at home or the office, they're going to go off to Dev to sort of just keep up with stuff and to read some things here and there. They're going to heavily use Google to find their answers. They're going to land on content that people created in that way. But coronavirus hasn't been, I think, a big factor one way or another in our traffic. It did make us want to roll out Forem a lot faster, but we quickly realized like if we're going to do this right, we can't just like rush it out like crazy. That also, maybe if we roll this out in the next month or so to broader consumption, coronavirus will play an impact in how people consume their online communities. Of course, it's important as ever now that people have these online communities. But I just think in the software development world, it hasn't been a huge difference.

[SPONSOR MESSAGE]

[00:43:04] JM: Open source tooling is generally preferable to closed source tooling, because with an open source tool, you're going to know what code is running. You're going to know what the community is saying about that code and you're going to have flexibility. But scaling open source tools is not that easy. You're going to have to spend a lot of time managing and maintaining that open source software. And the alternative is to use closed source software, which will be scalable, but you won't know exactly what code is running and you won't have an easy migration path.

Logz.io is a scalable and fully-managed observability platform for monitoring, troubleshooting and security, and it's all based on open source tools, like the ELK Stack and Grafana. Logs.io is open source software at the scale that you probably will need if you are a growing company. Sign up for Logs.io today by going to Logz.io/sedaily and you can get a free T-shirt. You can go to L-O-G-Z.io/sedaily, create an account for Logz.io and make a dashboard. Once you make

that dashboard, you will get a free Logz.io O T-shirt. Thanks for listening to Software Engineering Daily, and I hope you check out Logz.io. That's L-O-G-Z.io/sedaily.

[INTERVIEW CONTINUED]

[00:44:42] JM: So the content that people write on dev, I think earlier on, it was more about frontend stuff or maybe like more beginner related issues. Over time, there had been a lot of experience developers that have glommed on to the platform. It's really become a place where people publish, really, any content about software engineering. And I wonder what your barometer is for what people care about. Water are people focused on when you see the kind of things that people are posting? Is there anything that has surprised you in terms of what people care about the most? What people seem to be writing about the most? What Dev discussed subjects have been the most popular? What are you seeing these days?

[00:45:27] BH: Yeah, it's really interesting to see the adaption of software go from a fringe idea to just something everyone's interested in and talking about. The GraphQL ecosystem broadly has just become this like enough, almost since I started this project, being an consistent evolution. It still seems sort of new, but it's also sort of table stakes in a lot of ways. So the whole GraphQL ecosystem has been really interesting to see evolve in terms of its web dev, but it's also like kind of touches frontend and backend and it's this kind of new idea for how people want to develop the web.

I think Rust stands out as the one that, across really everyone's radar, it has something someone's interested in. So people are interested in WebAssembly. They're interested in going from web development to something maybe a little bit more systems and low level. They're interested in evolving to the newer version of what they're already doing if they're more of a C++ developer. And so those are a couple things. The Flutter ecosystem, I think people have been pretty excited about in ebbs and flows for a little while. That comes to mind. Yeah, just kind of like in observing it all, there're just a lot of repeat patterns too. There is an excitement. There's a period of time where it seems like there's way too much content. People are way too excited about it. I think Typescript fits that for me. I think, for a while, people thought it was going to change everything. And maybe it did get a lot of adaption, but nothing like dramatically changed

in the programming world because Typescript became more popular. And that's kind of how things go.

It's been exciting to see people take to Dev to talk about new things a lot more. For a while, I felt like that was strictly the realm of Twitter and maybe Hacker News for like what the latest and greatest is. But more and more, I think people have been wanting to take to Dev when they first think of a new idea, they first have questions about a new thing. It's been really exciting to see the creators of different ideas and paradigms and libraries posting on Dev.

Rich Harris, the creators of Svelte, has made a few posts on Dev, each of which became incredibly popular within the community just because they're a stance on web development in general for the most part. And it's been pretty exciting to see. But software development, it seems like the more things change, the more they stay the same. People mostly just need to talk a lot about stuff to gain an idea about what's actually going on and what the real trends are.

[00:48:13] JM: Have there been any unexpected engineering challenges as you have built out Dev?

[00:48:19] BH: Well, I can certainly say that I'm really happy that the product market fit has continued to evolve to allow us to make really important team expansions. I feel like we were at a phase of total mayhem in a lot of ways before we hired Molly Struve, our lead server reliability engineer. She came on like right when we needed her. And even though we've grown a lot since she came on, things have become much more stable. And that's mostly because we had these challenges of just like as much as our architecture is scalable, as much as Heroku makes certain things easy, there is like problems that if you don't have someone like really dedicated to dealing with it, you're going to encounter problems.

Molly coming on was like helps us avoid some major problems. So we were having like constant different types of memory issues. Ruby is a tough language to manage memory. And that was a big problem. But we brought in some great people and then we dealt with that. We have several folks within our organization who came in as from the open source side of things and then joined our team full-time. And that's how we've hired several key people. Obviously, like if

people are already interested enough just to be part of the process that are going to be great team members as long as they're good people as well.

Luckily, I'd say like we haven't run into anything that's been anything more than a somewhat of a tech, short-term technical issue. Although our future plans are more ambitious than anything we've done yet. So we have these issues of moving from a Heroku-centric infrastructure idea to something a little bit more generalized. But, of course, we're doing that across the board in terms of taking the word Dev out of the codebase and turning it into a variable. And then I suspect the technical challenge of deploying new code across the different networks all at once is going to be a difficult and exciting one. I'm excited to build out certain types of Chaos Monkey-esque features that have a better idea of just how things are running across the network as things get more complicated and distributed.

Of course, we're going to have to overcome challenges of authentication and privacy and how the network is going to allow people to be private between networks, but also have the user experience where they can maybe scroll between the different forums in a pretty easy way. We have a lot of stuff to figure out, and we're taking on the idea that we're betting the company that we're going to figure out some of these stuff, mostly because we've had unique access to some of the most creative and driven engineers in the world because we're a community for software developers and we can really put our values out there and see who wants to be part of this.

So far, like we've thankfully been able to overcome every little tiny challenge along the way, and it's mostly because we have this broad community of excited engineers just looking at stuff with us and a great pipeline of developers joining the team whenever we need them.

[00:51:46] JM: The memory management issue with Ruby, there was a period of time where people solved that with JRuby, where they transpiled it to Java byte code and then run it on the JVM. I imagine, that does not fit your simplicity ethos.

[00:51:58] BH: Yeah, that's the kind of thing like we'd rather, to some extent, just do things the most generic way and let the ecosystem solve some of our problems. Luckily, we run the same stack as Shopify, and GitHub, and Airbnb and a lot of other companies. And even Shopify, as far as we can tell, like also tries to keep things simple in a lot of ways. Just run Rails.

So a lot of these things get worked on over time, and we suspect that the problems are going to get better, but then also like this is just kind of part of the deal, like Ruby is kind of memory inefficient. It's kind of a pain. It's going to be a problem for us as we try to deploy all these different instances and try to keep our costs down, because now we need to try and make a profit on Forem Cloud or whatever we end up calling that. And it's going to be challenging, but we're not trying to like get too far ahead of that challenge. We could go JRuby or something. But, yeah, I think you hit the nail on the head. We try to keep things conventional and simple, and there are a ton of optimizations that I know exists within our app that we're just not doing in terms of application logic. I know that our basic queries for the feed bring in and store objects in-memory that are way too big, and that can be solved with application logic and we don't need to rush for some snazzy new runtime.

[00:53:38] JM: Well, speaking of the feed, I know we're up against time, but I do want to ask what the process has been for designing a good newsfeed and how you have updated it over time or how much engineering work goes into making a good feed.

[00:53:59] BH: Yeah. So, for a while, the feed was this convoluted mishmash of like different ideas, and the feed really was like our best attempt at not being totally useless. That was good enough for a while, because maybe our early adapters were willing to deal a feed which never seemed quite right for anybody. But it really is a difficult challenge, because, A, like just the queries involved. It's hard to like organize a good query for a feed because it's so unique. It's hard to like spread that across many users. But then there's also the challenge of like jut you don't get eyes into all the different situations. You want to give people stuff they like. And you also don't want to give people like a million knobs so that they have to like tune their feed to themselves, because it's like really complicated. You want to kind of give people something that mostly works for them, but they have some options to adjust it.

Generally, we've settled on giving people more control over their feed than a typical social network. So like on Twitter, you don't really get to see how you might tune your feed. On Dev, we allow people to set different weights for tag so that really helps them just tell us like, "Hey, I want to follow Ruby and JavaScript, but I care way more about Ruby. Don't give me so much JavaScript. There's way too much JavaScript here. Let me see more Ruby or Linux or whatever

I care about." That's one thing that really helps people like just get the experience they want, but we don't want to like overdo that. We don't want to create a million knobs. We don't want to create a situation on each forum that has maintain like this really complicated thing.

So we try make it possible for people to have a little bit more choice, but to try to keep it simple. And then after that, we just like AB test to see what people seem to care about a little bit more. We're early on enough that I don't think we have this success to say that we're building anything overly like addictive. And as long as we're feeding people vegetables and not junk food within the content, we tend to be able to focus on just general engagement and try to just make observations about what's working and what isn't and then iterate from there. But we don't have infinite engineering time. And right now, we're like making no efforts to improve the feed. We're occasionally taking some PR's, which is really nice as an open source project. We get to like accept contributions on things that are not primary concerns of hours.

But right now, 100% of our resources are just going into this form idea. We're really betting the company on this. So for the time being, like the feed is just what we provide. But once Forem is launched and we're really up and running, we're going to take another deep stab at what is it mean to serve a feed to a social media company, or to a social media community. What are the concerns around addiction? Around serving vegetables and not junk food, and what that means in the long run?

It's all about like what the most important things in the moment are. And we've thankfully made it possible for our users to really give themselves a great feed through control and through some simple decisions on our end. And in the long run, it's just going to be kind of an evolution of that, ship, iterate, ship, iterate, pay attention to behavior, ship, iterate. Give people control, but try to keep it simple. Ship, iterate. And that's the deal.

[00:57:31] JM: All right, Ben. Well, thanks for coming on the show and, really, congratulations on all the success. It's pretty impressive. And having seen it grow from basically a Twitter account to a social networking platform is pretty impressive. Really, nice work.

[00:57:46] BH: Yeah. Thanks, Jeff. And it's been really fun to touch in with you like at every step of the way. It's been really exciting and motivating to have you as part of the journey.

[END OF INTERVIEW]

[00:58:08] JM: Operations teams can find themselves choosing between two options, either take full control over the infrastructure yourself or give all developers permission to access production. The first approach increases the operations teams workload, which often results in overwhelming situations and ops becoming a bottleneck. And the second approach allows for rapid deployment of changes to production, but it causes a serious risk to infrastructure uptime.

With Octopus Deploy, operations teams have a third option. The Octopus platform can be authorized to run approved steps and play an intermediary role. Octopus delivers self-service without sacrificing control over production, and it also provides a comprehensive audit log of the changes that are being made. Developers can enable self-service with automation. By automating the processes that are forming a bottleneck, developers can free themselves from the waiting game. You can learn more about run book automation at octopus.com/runbooks. Octopus can help you with your run book automation, and just go to octopus.com/runbooks to learn how.

[END]