EPISODE 683

[INTRODUCTION]

[0:00:00.3] JM: Open source software is key to our software infrastructure. Closed source enterprises rely on open source software, but the development processes for closed source and open source software are often different in their approach to continuous integration and delivery.

Oren Novotny is a chief architect of DevOps and modern software and Blue Metal Architects where he works with a variety of clients to build products and internal applications. Oren spends considerable time developing open source software both within his job as well as during his spare time. He's been in the software industry for more than 15 years and has a breadth of insights from different businesses in how they apply software.

We started the conversation talking about electronic trading companies, which in some ways operate like large enterprises, and in other ways operate like startups. Oren described his time working in the financial industry through the 2008 crisis, which was a scary time. Then he switched industries to work at Microsoft before coming to blue metal architects. Oren and I then discussed the process of setting up continuous integration for an open source project, including the difficulties and the large benefits for adding continuous integration to an open source project as well as the differences between an open source project and a closed source project in how they should approach continuous integration. But there's clear value to both environments of integration.

I hope you enjoy this episode with Oren.

[SPONSOR MESSAGE]

[00:01:39] JM: When a bug occurs on your website, LogRocket captures the user behavior and allows you to do an instant replay to see how the user responded to the bug. LogRocket lets you replay what users do on your site helping to reproduce bugs and fix issues faster. See issues as if they happened in your own browser with a full video replay and get those issues fixed fast to maintain the health of your application and keep customers happy.

LogRocket works regardless of your applications language or framework and it provides SDKs for specific technologies. You can easily integrate with the tools that you already use. You can check out a demo at logrocket.com/sedaily, which would also support Software Engineering Daily.

LogRocket also records console logs, JavaScript errors, stack traces, network requests and responses with headers and bodies, browser metadata and custom logs. If you're triaging back into errors, it can be unclear why the frontend made an unexpected request. LogRocket integrates with backend logging and error reporting tools to show you the corresponding frontend session logs for every backend error and log entry.

Quickly understand your bugs and fix them with LogRocket. Go to logrocket.com/sedaily to find out more to see a demo and to support Software Engineering Daily. Thank you to LogRocket for being a sponsor.

[INTERVIEW]

[00:03:22] JM: Oren Novotny, you are a Chief Architect for DevOps and Modern Software at Blue Metal. Welcome to Software Engineering Daily.

[00:03:29] ON: Thank you. Thank you for having me.

[00:03:31] JM: I want to start with your background in trading applications. You worked in trading applications from 2004 to 2011. What are the engineering requirements for trading applications?

[00:03:45] ON: It's going to vary by specific desk, but one of the things I found is that there are generally complex data models and you're dealing with huge amounts of data that needs to be updated in real time with extremely low latency. Also, they tend to have heavy UI visualization, and I was on the frontend teams for the most part at that point in my career. It was all about creating visualizations and chart and order entry systems that could update and reflect the market data and the order data coming in real time. This is a time, like this is Windows forms,

Win32, and it was hard now, it was harder then in terms of being able to paint those displays fast enough.

[00:04:32] JM: What are those difficulties? What makes it hard?

[00:04:35] ON: You have data coming in off of different streams, market data, order data. It all has to come in. It has to get correlated, cross-joined, filtered sorted, painted in a grid. The typical threading issues and race conditions, deadlocks, live blocks all come into play. It's a very challenging environment to start with.

[00:04:56] JM: When I was just leaving school, I worked at a trading company for about five months back in 2013. Thinking back to those same problems that you just articulated, it's in some ways very similar to some of the challenges that companies are dealing with today in terms of streaming data. So today you have click streams of data, or fast incoming data about sensors from a self-driving car, something like, and you need to update your machine learning models, or your visualization systems quickly in response to those large streams of data. I find that reflective of this trend that in some ways trading technology is cutting-edge and it's ahead of the curve and they deal with challenges that are ahead of the curve, ahead of the consumer software market. But there are other ways in which when I compare the trading technology of companies that I saw when I was looking at this more closely, there's somewhere where they seem to lag behind the state-of-the-art of Silicon Valley companies or product- centric companies. Is that a fair assessment? Do you see that as a strange attribute of trading companies?

[00:06:16] ON: It is, and I would agree with your assessment. For instance, we were doing real-time pivot tables at one of the firms I worked for, this was the kind of thing where you look at it and say, "Look, why don't we just do this in Excel?" You have a master, order data, apparent data, child data, you could cross join this thing with all the different columns and attribute to roll up at arbitrary levels the summaries and averages and all that. You can right-click it and turn that into a chart, a heat map. It was basically self-describing data coming in which could include all of the aggregates that you could do on it, and you had effectively a dumb grid. I mean, I said dumb, but there're tons of smarts in there that could read the metadata coming in off of the data

type and just provide all these functionality. It had to do this with 10,000 or 20,000 rows of data all updating in real time.

I remember one thing that stuck out at me when I was at PDC in 2005 in Los Angeles, I showed some of these to some folks at the Birds of a Feather Session from the Excel team and their jaws dropped. I mean, the fact that we could do this in real time, and I'm thinking to myself like, “Why do we have to build this? Why can't we use a third-party component like Excel to visualize some of these?” But there was a huge gap in what was commercially available versus what we had to do ourselves.

But to the other end then, it's also very restricted and locked down environment, and that's ultimately why I left Wall Street, was you’re using older technology, at least when I left. It may have changed now. But they're doing cutting-edge problems, but you’re limited by what they can approve and install on the desktops, because anything that they deploy would have effect on all of the applications on there. So they were naturally very cautious, because you have huge sums of money flowing through these systems. You can’t afford a mistake, and many of these things were just very highly regulated, even pre-2008. You need industry regulators coming in to say, “We don't want to lose $100 million because of a typo.”

[00:08:43] JM: Was these all trading companies or was this just particular ones that had like a banking element, like a J.P. Morgan?

[00:08:49] ON: I would say the banks that I worked at tended to be more locked down. I did work at one firm that was developing a trading system using Microsoft technologies actually on the backend, as well as the frontend, and that was unique. Most of the backends in Wall Street were Linux, Java-focused, where the desktop was Windows.

[00:09:12] JM: Yeah. That was the architecture of the company I worked at actually. That exact architecture.

[00:09:17] ON: This firm I was working at, it was owned by Citigroup, but it was operated as a subsidiary, developed of client and server entirely in .net. That we had a lot more freedom over,

because we were in our own building, we didn't have to dress in Wall Street dress, we had cubes instead of a trading floor layout. It definitely felt much more of a software company feel.

[00:09:41] JM: What was it like being involved in that industry through the 2008 financial crisis?

[00:09:48] ON: It was scary, that was for sure. I was at that firm I just mentioned that one that was operating more like a software company, and we were just watching in silence as it were of like when Lehman crashed and we're like, "What's going to happen? Is that going to ripple with the domino? What's going to happen?" Nobody knew. There was a time when we were – Nobody knew what was going to happen to our group either, because were we selling this product to other firms? Was our parent company using it? We were just lucky –

[00:10:21] JM: Selling that product. You mean like CDO's or –

[00:10:24] ON: Well, in this case, this was the order management system.

[00:10:26] JM: Oh, okay.

[00:10:27] ON: It could do the equities order futures and FX were the three options, multi-asset, multi-broker order flow system, but nobody was using it yet. It's like, "Well, what's going to happen to it and do we still have jobs." We were lucky, we did. We showed up every day for a while, and then at some point it was a work from home. We'll come in if we need anything, but we were getting paid. Then one day they told us to that we were being spun off and acquired by another trading software manufacturer. It could have been a lot worse, and certainly it was for a lot of people.

[00:11:03] JM: Yeah. The financial technology world, more broadly, how did it change in the fallout of that 2008 crisis?

[00:11:14] ON: That's a tough one, because I'd never really saw it from the regulation perspective. Certainly, it became a lot harder to get a job. I mean –

[00:11:22] JM: Even as an engineer.

[00:11:24] ON: Oh yeah. I mean, there were fewer banks. No one was spending money. The purse strings were all tightened up. I recall that firm that acquired the software firm switched from .net to Java. They were thinking, "Oh! We'll just take a bunch of C# developers, and they're smart people. They understand the system and the business. We'll just make them do Java instead, because that was how they written their trading platform." I tried it. I gave it several months. It was not for me, and that's where I had left and gone on to another bank at that point.

[00:12:00] JM: Yeah. When I was at that trading company, what was so cool about it was we were in one of these companies where you have the traders on the same floor as the developers, and it's sort of like the developers are building the product that the traders are using. So you can have really easy communications and makes for a really interesting environment. It makes you – Since you're the developer and you're on the same floor as the traders, you're subject to the same psychological gyrations that the traders are going through. On a day like – I think I was there during the Facebook IPO, and the Facebook IPO was botched technologically. I'm not saying it was botched by the company I was working for. It was botched the IPO itself. There were some kind of issue with it. I don't remember exactly what it was. But there was another –

[00:12:50] ON: Sounds familiar.

[00:12:51] JM: You remember that?

[00:12:53] ON: It rings a bell. It was not until you mentioned it, but, yeah.

[00:12:54] JM: Yeah. So that kind of thing, or there was another – I think there was another day where there was kind of like a mini-flash crash – No. The market got halted. There was some reason why that market got halted because of some glitch in NASDAQ or something. It's just like crazy emotional gyrations. Of course, that happens in startups to some degree, but there's such a financial – Directly financial at a trading company. It's just a very interesting world. I haven't done as much coverage over it as I have of like startups and enterprise companies, but it's completely unique.

[00:13:25] ON: Oh, definitely. For better – Or as many will say, for worse, and I'm of the latter opinion, I was never a huge fan of that trading floor layout. I think that there's certainly benefits to being embedded and being close to your user. It was kind of Scrum and Agile at the early part of the Agile manifesto and Agile movement, or even predating that some degree and it's really valuable to have that constant input and feedback cycle with your end user.

But at the same time, I think that there is the developers and need a quiet space to get stuff done and –

[00:14:06] JM: I agree with that.

[00:14:07] ON: Or at least some developers. Nobody's the same. Some people can work in a café and put the headphones on. I'm not one of them. But I think that it's been – Everyone looked to Wall Street and said, "Oh! We should all do that. We should all have trading floor layout, and it's cheaper, you get more people in there." I mean, Microsoft, among others, has been tearing down their offices and putting everyone in open floor plans. Good luck with that.

[00:14:30] JM: I think it's insanity.

[00:14:32] ON: I agree, and that's where it's one of those things where I'm waiting for the pendulum to swing back the other direction, and you have all – The reports come out that says, "You know, maybe the productivity gains weren't all we thought," but there's still a lot of inertia that, "Open office, it's great."

[00:14:48] JM: I mean, to some degree it does work if you have enough phone booth space that people can escape into. But I remember like the chief architect, the person who wrote the most code basically would escape into silent office space and – Or not escape, but would go into private office space and be like, "Why do you think you want private office space, and why are we out here exposed to the vicissitudes of the traders?" To some degree I understood, because the we were kind of on-call all the time, because s if something went wrong, then that has serious ramifications. It's like direct financial ramifications.

I mean, if you're building a product for project management and there's an issue with your project management software and you're on-call for that. Yeah, you've got to fix it quickly, but it's not as directly financial as, "Oh, the trading software is broken and it's executing 10 x the number of trades that we wanted to. We need this fixed right now."

In some sense, I guess it makes sense that you're on the floor with them, but in many ways like your software is going to be working just fine, and the traders are just going to be having a bad day because of, you know, [inaudible 00:15:58] is down or something goes wrong in the markets and it has nothing to do with your software. In the meantime you're sitting there typing away in the other side of the room and the traders are just shouting and yelling and it's super distracting. It doesn't even have to be that extreme. If you think about somebody working – I've worked in shared office space, I know contrasted with me working at home, I am way more productive working at home, because I can get into my like frame of mind. Nobody comes by to bother me. I work very well with my coworkers who are also located remotely and it's just – I don't know.

[00:16:35] ON: I agree completely. One, I've been lucky since joining Blue Metal, and it's been six years now actually. It was two days passed my 6th anniversary. Never thought I'd be anywhere that long. But by and large, I've worked remotely from a home office where I have a powerful desktop, giant 4k, 24- inch monitors and I have my set up. People leave me alone. It's quiet, and I just get my stuff done.

Of course, it doesn't mean I don't see people. I go to the office and I go to meetings when necessary, when it's important, because face-to-face is important, but we try and schedule that, or good part of what I do is talking to customers and clients. I will fly and meet them wherever they are, because there's nothing better than that in-person meeting. But it's that balance. It's like, okay, at some point you just need to get stuff done and having the privacy and quietness for me anyway of a home office has been hugely helpful.

[SPONSOR MESSAGE]

[00:17:48] JM: I have learned a ton from QCon. QCon San Francisco takes place this year, November 5th through 9th, 2018, and I will be going for my fourth year in a row. I always love going and seeing the talks, and in between the talks I hang out and eat some mixed nuts, chat

with other engineering leadership about the latest talks and stuff that they're seeing, the 50 different stream processing systems that we're seeing, the different databases we're seeing, and QCon is a place where senior software developers and team leaders and managers and senior leaders, they all gather together, and you have fantastic conversations. You have fantastic presentations, and it's extremely high quality. Its technical. A lot of it is also cultural and about management, and you can get $100 by using the code SED100.

QCon is a great learning experience. The presentations this year include 18 editorial tracks with more than 140 speakers from places like Uber, Google, Dropbox, Slack, Twitter. They are curated high quality talks, and you can get $100 off if you use code SED100 at checkout for your ticket. Again, QCon San Francisco takes place November 5th through 9th, 2018, and the conference is the 4th through 7th, November 8th through 9th are the workshops. You can go to qconsf.com to find out more.

Thank you to QCon. If you are going to buy a ticket, please use code SED100, and we really appreciate the sponsorship of QCon.

[INTERVIEW CONTINUED]

[00:19:45] JM: You made a kind of a career transition, because you were in trading world for about seven years and then you went to Microsoft. I think I'm not exactly sure, but I think at Microsoft you did some work that was kind of related, like you might have had clients or the people that you worked with or the products that you're working on were probably tangentially related to the trading applications that you were working on. But that still must've been somewhat of a career transition. What was that career transition like?

[00:20:13] ON: Yeah. So it was always a dream opportunity for me to go work at Microsoft. It was one of these – I use Microsoft tools .net and Visual Studio and TFS for the most part, because I'm a lazy developer. I don't want to spend tons of time getting eclipse up and running. I always looked at, "Well, I need to have a startup script to launch this Java GUI," or whatever. It seems like lots of yak shaving. Whereas – And nothing's perfect by any means, but by and large, Microsoft tools just "worked" and I was able to get to do the parts that I needed to do and solve the real problems, not just spending times messing around with tools.

So it was always a dream work for Microsoft. Unfortunately, Microsoft doesn't have a lot of product development outside of Redmond, and I am still and have been tied to New York City. What I found was an opening in the technical sales team of the Visual Studio and TFS. My role was a TFP in visual – The New York Metro EPG enterprise customers dev tools sales, where my role was to try and help customers adopt Visual Studio Premium and Ultimate as well as team foundation server, and these are tools that I knew very, very well inside and out, and that was the transition. Taking that experience and knowledge in the product and trying to see how I could help Microsoft customers adopt it and solve their challenges using these tools.

[00:22:01] JM: From there you went to Blue Metal Architects where you are now. What do you do at Blue Metal?

[00:22:06] ON: So then an evolution. At Microsoft I was in sales, and I learned a huge amount about the sales process. I also learned that not necessarily interested in directly being a salesperson, and that's why I looked for – I wanted to do more dev, more hands-on architecture and I was connected with Blue Metal where at the time it was still a small company. I think I was the fourth or fifth person in the New York office at the time where I started as a software or senior software engineer. So I was a consultant writing code for clients. At time in 2012 it was Windows 8. There were apps for – At least Microsoft was pushing it. So they were paying customers and partners to do these things. A lot of mobile, we were early with Xamarin, and then some of the web technologies.

Since then I've grown to be more of that architect and chief architect role, where I oversee the strategic direction of how we go to market in areas like DevOps. What do we consider modern software? What's our view on where the direction is going, helping customers solve their challenges using technology. Looking to say, "Well, we're trying to –" I can give you an example. There's a manufacturing company that has some robots, and they're trying to get some help with how do you automate the deployment of the firmware to the device? How does this connect securely to the cloud? How does the cloud scale?

They spent tons and tons of time making this stuff work, and there's image recognition and cognitive services involved. They'd made it work and just have a really awesome product. Now,

they need to take it to that next level, and that's where they've engaged Blue Metal and Insight help do that.

[00:24:02] JM: Now, on the side, you work with many open source projects, or maybe for all I know that fits into your working life. How does open source contribution fit into your life?

[00:24:12] ON: Open source is really a core to what I do. Whenever I work with customers and client projects, I always try to leverage tools that already exist. So instead of reinventing the wheel each time trying to see, "Well, is there a tool or a library or starting point to go off of?" and that's been really beneficial with open source, because we can go in and not start from scratch. Of course, these engagements aiming to contribute back. If there's some feature we needed of a date time conversion framework, well, is this logic that goes as part of the core client deliverable or is this an open-source library that we're using and that the changes go there? We're delivering the value, the application, the business, the important stuff to the client, where some of the underlying pieces that are not business specific and just have no proprietary value are just generally useful to the community. So trying to balance that wherever possible, that was certainly a big part of it.

[00:25:20] JM: How long have you been contributing to open source?

[00:25:22] ON: I would say that the big – It took off around 2012 for me, and that's really where I started getting back into development. When I was at some of the banks, they certainly used many library, but there wasn't much appetite to contribute back at least at the time. It was also much earlier in the .net Microsoft ecosystem around open-source. Lots of commercial tool vendors, lots of commercial libraries, but this was before the new get package manager became a thing. I think it was around 2010, 2011.

So over that time since I had left Wall Street and since I joined Blue Metal, the ecosystem evolved to have ways of distributing packages, which started leading towards being able to get them and consume them. You also had GitHub rising as a place to collaborate on these things. So it was a combination of things that really started taking off around 2012.

[00:26:27] JM: How have you seen that world of open source contribution change over time?

[00:26:34] ON: I think there's been a dramatic shift to being open by default. I think there's a separation between what the core business value and where are you – Where is your expertise? What's the thing you're trying to do? What are you better at than anyone else and what's all the plumbing and glue that you need to get there?

All that plumbing and glue and other super important bits can be shared. I think there's become more of an understanding that there is value in doing that, because you're building off of others and others will leverage that. I think that that shift has definitely been occurring within the Microsoft ecosystem. Early in 2012 or even before, I think if you looked at Microsoft and open source, it was kind of a joke. Now, it's effectively de facto standard.

I mean, I think pretty much everything is built in the open unless – Or by default, unless there's a compelling reason why it shouldn't be.

[00:27:37] JM: How has that it changed the interaction between the consumers of the Microsoft stack and the people who are making that software stack in Redmond?

[00:27:49] ON: I think it's brought everyone closer together. GitHub has been phenomenal for that, where if you have a development team that's truly embracing collaborative development, it's not just code dump that's there. It's not just, "Okay, here's the code and an MIT license or some other open source license. Party on."

But if you are truly engaged with the community as Microsoft teams are, and some teams more than others. Again, it's a large company, but the .net team, for example, has huge amounts, and I don't remember specific numbers, but of community contributions regularly interact with GitHub issues. Certain features that are core to the platform had been contributed by non-Microsoft people, and they're all vetted and they work with the community to vet review and ensure that the standards and quality are met.

But the level of transparency has been – I can't say unparalleled. That's too hyperbola, but it's been amazing. You have like the C# team has language design meeting they stream and post the notes to, or they broadcast the framework design meeting. Different groups even there have

different levels, but these are meetings that you can tend in live or sometimes just delayed, but you can see the recording. It's hard to imagine that 10 years ago. Can you imagine Microsoft development team recording and broadcasting their meeting deliberations?

[00:29:26] JM: No. I think it really makes – I mean, me as an industry coverage person intrigued with this kind of the cultural change and there's like changing of the guard within the company. It's been around for 20 years or something, maybe longer than that, or – No. 30? 25 years? Something like that. So there's like just new people who are coming in who – So the institutional memory of that time is changing, or at least it's maintained in a way of like, "Okay, we understand the value of that institutional memory, but we also understand the costs of that institutional memory," and we can see the value of the more open model.

[00:30:10] ON: Absolutely. I mean, it's changed. To be fair, there are many great people there that have been in there a long time and. There are certain people that are no longer there, and it's an ebb and flow, but I think there's been that push to get things more open that has been certainly pressure from the community, that lots of people who have been there and passionate about this for so long were able to get the buy-in and grow that up the chain. That's just been amazing. Don't get me wrong. But this process started long before he was head of the company. This was all in the works for a while and it's been a gradual change.

[00:30:55] JM: You have worked at enterprises on premises at the enterprises. You've also done plenty of consulting work where – Well, I guess with Blue Metal, where you are working from home, which is remote, which is I think somewhere in between open source and working at an enterprise, because it's like working with the enterprise, because you are working in this closed source environment, or mostly closed source. I mean, most enterprises are not open source software companies. Maybe that's changing, but most enterprises are not open-source.

But open source, like open source contribution is a mostly remote. Well, actually even that might be changing. I mean, there're a lot of people that work at companies that are just full-time open source maintainers. But the model of contributing code to an open source repo is by default something that somebody could do from a remote computer sitting in their office at home. So you really have the full picture of the advantages of working in an enterprise software environment from a contributor perspective versus the advantages of working on an open-

source project that you can work on remotely from home. You see that the costs of both of those working environments. How does the process of contributing to an open-source project compare to contributing to code within an enterprise?

[00:32:19] ON: I think it's all – There are similarities and there are differences. Hopefully, in an enterprise, you have people – They'd like to make a distinction maybe, or even add one extra category of there're open-source that are small side projects or hobby project, maybe a small team of a few people or one person that are done on the side and moonlighting day job, or not as part of a day job, but that sort of thing, labor of love, versus enterprise open source where you have big companies that pay employees to write software that just happened to be open.

I think that's where I view some of the bigger differences there, like some of the things like project management. If there's somebody to do some of the – I don't want to say grunt work, but just the stuff that needs to get done and you're getting paid, or somebody's getting paid. That would get done. Whereas if it's in a hobby open-source project, people want to tend to focus in on the hard problems, or the interesting problems, or things that's perhaps a niche, whereas it can be harder to ensure that things like accessibility, or localization, or documentation are sufficiently covered. Because ask yourself, do you want to sit there? If you have a limited number of hours in the evening after work and after your kids go to bed or whatever, are you going to sit there and write documentations? Are you going to try and noodle on that problem of why wasn't this thing working the way I thought it was, or should? Whereas, "Hey, look. If you're doing this for your day job, well, today I got to – Or I'm going to spend this morning writing documentations. What I got to do?" right. That's where I see some of the different. More about that aspect of it than necessarily remote, or home, or enterprise.

[00:34:24] JM: More on the open-source side of things, what are the problems with management of open source projects and repos?

[00:34:34] ON: I'd say that there can be finding people to contribute is always an ongoing thing. Trying to scale the project beyond that hobby into something that has – To have people to review poll requests. Do you have people contributing? You need people to file an issue? But if you're doing six other things, like, "Okay, I'll get to whenever I feel like it," or "do you have

anybody stepping in that might be interested in fixing that?" It can be the balance trying to ensure that users of your project are not blocked with bugs or issues.

At the same time, again, it's a labor of love. For the most part, for a lot of open-source, nobody is paying for this, and that can lead to a friction around unrealistic expectations, or, "I filed an issue, why hasn't this been fixed already?" You're saying, "Nobody is paying for this." You want to support – You want something fixed in a certain time or in a certain way, maybe there should be a model with a support contract or some other way where you have guarantees. I think there can be unrealistic expectations around support for projects that nobody is paying for.

[00:35:51] JM: Well, the PR requests challenge is something that I'm intimately familiar with, because we have this open-source project, this Software Daily. It's on GitHub. We have with mobile apps. We basically have a podcast player that's specifically for Software Engineering Daily. So we have some power listeners that really like to use that app. The iOS app is open-source. The android app is open-source. We've got an open-source web version. We have an open-source backend, and figuring out who is in charge of accepting the poll requests, of reviewing them, of doing the testing, like making sure the tests have run and that there's no breaking changes. That's a single point of failure. If you only have one person who was in charge of that and then they're bottlenecked, and if you have a bunch of contributors who are issuing poll requests, you just have this open-source contributor who – Or this open-source poll requests arbitrator who is in charge of that, and that can be problematic. Then that can lead to burnout, open-source burnout. If you're feeling burned out, it's not hard to discard your open source contribution as part of your life, because it's like, "Well, I'm not getting paid for that. So I'm just going to like discard that."

Then it's like, "Okay, now my single point of failure has left and now the project just completely bottlenecks and it falls apart. You just see these kind of systemic issues crop up in open-source projects that have any amount of traction.

[00:37:22] ON: Yeah. I would say for those projects, there's also a lot of project where having not enough poll requests reviewers is a problem we wish we could have. I'll take it to that next level. By example, I maintain several open source projects, some of which I fell into simply because the maintainer wanted to do something else. I said, "Look, I can keep it alive, or I'll get

out a new release. I'll review the PR that comes in here and there, but I'm not doing huge active dev."

There's a number of those that I'll say, "Look, I'd love to have any active contributors come in and submit poll requests, because then if they do that, I'll review a bunch, but then I'd be happy to delegate some of that to some people that are – To some new blood." What I've done is set up some of the CICD, the maintenance. I can get out a new release without much friction, but how much time do I have to dedicate to individual libraries? It's alive. Somebody comes in, we can look at it, but you need that next level of finding people to just have the interest to submit poll requests. I'm sure that that's not a unique issue.

[SPONSOR MESSAGE]

[00:38:50] JM: DigitalOcean is a reliable, easy to use cloud provider. I've used DigitalOcean for years whenever I want to get an application off the ground quickly, and I've always loved the focus on user experience, the great documentation and the simple user interface. More and more people are finding out about DigitalOcean and realizing that DigitalOcean is perfect for their application workloads.

This year, DigitalOcean is making that even easier with new node types. A $15 flexible droplet that can mix and match different configurations of CPU and RAM to get the perfect amount of resources for your application. There are also CPU optimized droplets, perfect for highly active frontend servers or CICD workloads, and running on the cloud can get expensive, which is why DigitalOcean makes it easy to choose the right size instance. The prices on standard instances have gone down too. You can check out all their new deals by going to do.co/sedaily, and as a bonus to our listeners, you will get $100 in credit to use over 60 days. That's a lot of money to experiment with. You can make a hundred dollars go pretty far on DigitalOcean. You can use the credit for hosting, or infrastructure, and that includes load balancers, object storage. DigitalOcean Spaces is a great new product that provides object storage, of course, computation.

Get your free $100 credit at do.co/sedaily, and thanks to DigitalOcean for being a sponsor. The cofounder of DigitalOcean, Moisey Uretsky, was one of the first people I interviewed, and his

interview was really inspirational for me. So I've always thought of DigitalOcean as a pretty inspirational company. So thank you, DigitalOcean.

[INTERVIEW CONTINUED]

[00:40:57] JM: So these are kind of the level 1 issues of open source. These have been issues around open source for a long time. They certainly are talked about more these days. There're some great articles about the problems of open source burnout and whatnot. But there's also a suite of problems that exist for projects that have significantly more traction. So like Kubernetes is not going anywhere. That project is not going anywhere. Node.js is not going anywhere. These are open-source projects that have massive traction. They have tons of contributors with varying interests, and these kinds of projects, they start to look in some ways like enterprises. We've done lots of shows on DevOps in enterprises, and the release process in enterprises, and the maintenance of agility that can come with a DevOps. Does the bops idea map effectively to open source projects.

[00:42:01] ON: I think DevOps is key to open-source project especially to help mitigate burnout and maintain agility. It really comes down to that idea that you should be able to get release out easily. You should be able to have high confidence in poll requests. You should be able to quickly triage them and ensure that you're not going to have breaking changes and do all these things. In my opinion, the number one way to do that is to automate as much as you can, which is very much at the heart of many of the DevOps practices. That's everything from validation builds in a poll request, which are then you have the question of what's — build is build, fine. But are you checking things like code coverage? Are you scanning the API surface area?

There are tools to do this on .net. I assume there are in others as well. But you'll compare the public surface area to say, "Well, did you add a new method," or if you did, it gives you a place where you can review that and make sure it wasn't a breaking change. Is there sufficient test coverage and whether the tests pass or fail?

So as much as you can automate on that side, and then similarly on the release management side, are you signing your code with authentic code, which is really important on the Microsoft platform. Are you doing that in a secure and automated way? Are you deploying to CI or nightly

feeds of any kind? What kind of channel mechanism might you be using? Are you promoting artifacts into data or – No one uses beta anymore, right? It's forever preview, or finally eventually stable. But being able to orchestrate all the things in a way that doesn't require human intervention I think is really important to preventing that burnout, preventing that single cause of failure.

[00:44:03] JM: What's the process of setting up continuous delivery for an open source project?

[00:44:10] ON: These days, I would say once upon a time the reason many open-source projects may not have started with it, you had to find a machine. Somebody had to build a virtual machine as a build box. Someone had to install some build tools, whether it was Jenkins, whether it was any number of different build agents that would be running there. Then they required lots of – To maintain it, for security patches. Nowadays, there's a number of hosted providers, like CircleCI, [inaudible 00:44:44], and a recent addition to the open-source community is Azure pipelines, and these are offers and services that will allow open-source projects to build their projects often for free and take care of hosting the build agents and recycling them for you. That becomes extremely important. In fact, table stakes, if you want to be able to build a poll requests coming from a fork. So it's one thing to say, "Well, I trust my maintainers," or if I'm just building, was already committed. Okay, fine. But if you consider that builds are executing code on a machine, and if you're going to build what's effectively an un-trusted code from a poll requests coming from the public, you cannot trust that machine to do anything afterwards.

You have to assume compromise, because they could run a crypto miner in a poll request. I wouldn't put a pass to anyone. You don't want that on your VM. You certainly don't want that running on a machine behind your firewall in your network. That's perfect place for hosted services to run these in the public cloud where they will – Where each build agent, each build, get home clean VM that's only good for the life of the build and gets recycled each time. The Azure pipelines process offer product, whatever, allows you to do this for Linux, Mac and Windows build agents. It's the only hosted provider right now today that allows you to build on all of the platforms in a single YAML file and a single pipeline at the same time.

[00:46:36] JM: So when PR gets submitted to an open-source project with continuous delivery or continuous integration, how does that compare to the PR submission process without continuous integration?

[00:46:53] ON: Let's take it to the extreme, shall we? So some open-source projects, like the Linux kernel, operate by submitting patches via email to some maintainer of some subsystem. Eventually, they may or may not look at it, in which case you submit changes. Effectively, you're submitting diff and patches via email. You can't test or build this thing locally. You can't get feedback instantly real-time. At the whim of some subsystem maintainer, looking at it eventually deciding if they can build it, or is it build with whatever changes that have landed since then. It's a very low asynchronous process.

Also, from the maintainer standpoint, I can review the code, I want for style, but will it actually compile? You can't really know that until you try. Oftentimes, poll requests are just so big that despite best efforts of skimming code, you just got to build it and run the tests. Compare that to automated PR validation builds where all poll requests get built, and the test suite or run and you can eliminate a huge source of issues before they even crop up. I mean, if it doesn't build or the test fail, stop right there. Fix your tests, or fixed the build. It definitely takes that first layer away from having to prove you that. Then you can focus on things like if they added a new feature, did they add sufficient test coverage? Etc.

[00:48:29] JM: I've only work at companies where the continuous integration process, we only have to run it on Linux machines, because we're deploying our application to Linux machines and that's all I need to test on, because I've just been a backend developer. But if you're talking about an open source project like Node.js, for example, if I want submit a PR to Node.js, I am assuming that there is a CV pipeline that includes testing in a variety of environments. I'd probably tests on Windows, and Apple, and Linux and – Whatever, 55 different versions of Linux. Is that cross-compatibility like testing layer, is that an important phase of the continuous integration pipeline at least for open-source project? Because open source projects are often times these infrastructure things that need to run in all the different environments.

[00:49:28] ON: I would say it's really important for sure. Being able to test the code in as many different configurations as possible is really important, and having the broad variety of build agents and test agents being able to run that on a single PR is invaluable.

One thing I saw, libgit2, which is maintained by Ed Thompson.

[00:49:52] JM: Yes.

[00:49:53] ON: He just submitted a PR. He was able to get quemu. If I pronounced that right, right? Q-U-emu, the queme emulator. He had that running ARM on a x86 Docker agent. So even though it was using the Linux build agent on Azure pipelines, but he was able build and run his tests using an ARM configuration inside of that container on quemo – Or I'm sure mispronouncing that. I'm like, "That was pretty cool." I think he was only half joking when he said that there's like an S390 emulator in there. Are you doing CI and CD and testing of your mainframe code? Because you can now. What extreme do you want to go to? Because some things that have become much more possible lately.

[00:50:43] JM: What are the other types of tooling that contributors and open-source projects need or what other kinds of tooling are they using it to get code as well automated, or I don't know if you can automate the creation of documentation. What are the other suites of tools that – Because from what I heard, it's like GitHub is great, but GitHub has to do too much. GitHub has a very big job, and there's not enough tooling just in GitHub, because if they tried to put all the right tooling in GitHub, it would just be like bloated and you wouldn't be able to do anything with it, or just the UI would be too crowded. What are the other toolsets that successful open source projects use?

[00:51:28] ON: I think, documentation, you mentioned is one of them, and there's a few out there. DoxFX is one that is based on GitHub marked down to HTML. That's what the Microsoft doc system is based on loosely. There are others that certainly being able to take mark down to certainly the flavor digital or these days of documentation and render that somewhere is super important. But I'd say there's also products like – Do you want to check for security vulnerabilities or open source license compliance? Many things like WhiteSource will offer open source projects of free license. So granted their goal is certainly to sell commercial licenses for

the rest of the product. Do they make their money? But many of the toolsets have offers for open-source projects to help with those things.

[00:52:19] JM: As we begin to wrap up, I want to talk a little bit about the other projects you're involved. What are the open-source projects and I guess non-open source projects that you are spending your time in?

[00:52:32] ON: There's a few ones that come top of mind. One is a code signing service that I maintain on behalf of the .net foundation and really is available for any organization to deploy into their Azure environment. It supports authentic code, code signing, as well as you NuGet package signing and V6 signing, click once signing. All of the popular signing formats where the certificate is stored in Azure key vaults hardware security module for safety and security. That is a project that I've done, because I've found that as gap in the toolchain. We offer that as a service to the .net foundation member projects, but certainly any organization can use the underlying software.

Another one is Humanizer, which is you may have used in some places or another, but the idea is to take things like numbers and take the number thousand and turn it into the word a thousand, or a date, or a time span and say it was 30 minutes ago, or something like that.

xUnit is a unit testing toolset that I help maintain the device spanner for. So when you want to use unit tests on Xamarin for iOS or android or UWP, you can do that using some of the device runners that I've helped write. Those are a few that are top of mind. In general though, I also try and help a large number of open source projects implement CI and CD and really help unblock them. So if they're having trouble building something, I'll try and make sure that they can build it and do it on an automated way. I chip in a lot of places.

[00:54:13] JM: What kinds of blockers. Do you see?

[00:54:15] ON: Sometimes it's just people haven't been done it. So they're not sure where to start. Other times, you're trying – The project tools evolve so quickly, and then there are easier ways to do things now. So trying to get from the way things were happening to the new way, which once you're there, it's simpler. You still need to get there and unstick certain things. So

trying to find where those bumps are and provide workarounds in repeatable ways so that people don't have to get stuck there.

[00:54:51] JM: If there's somebody out there that's listening and they have an open source project and they're trying to get continuous delivery, continuous integration going, can they reach out to you for some help or some guidance?

[00:55:04] ON: Oh, absolutely. I'd say Twitter is by and far the easiest way to get a hold of me. My handle is onovotny.

[00:55:12] JM: Okay. Cool. Oren, it's been really great talking to you. We covered a lot of ground.

[00:55:16] ON: Oh, it's a pleasure. Thank you for having me.

[END OF INTERVIEW]

[00:55:21] JM: Your audience is most likely global. Your customers are everywhere. They're in different countries speaking different languages. For your product or service to reach these new markets, you'll need a reliable solution to localize your digital content quickly. Transifex is a SaaS based localization and translation platform that easily integrates with your Agile development process.

Your software, your websites, your games, apps, video subtitles and more can all be translated with Transifex. You can use Transifex with in-house translation teams, language service providers. You can even crowd source your translations. If you're a developer who is ready to reach a global audience, check out Transifex. You can visit transifex.com/sedaily and sign up for a free 15-day trial.

With Transifex, source content and translations are automatically synced to a global content repository that's accessible at any time. Translators work on live content within the development cycle, eliminating the need for freezes or batched translations. Whether you are translating a

website, a game, a mobile app or even video subtitles, Transifex gives developers the powerful tools needed to manage the software localization process.

Sign up for a free 15 day trial and support Software Engineering Daily by going to transifex.com/sedaily. That's transifex.com/sedaily.

[END]