EPISODE 1228

[INTRODUCTION]

[00:00:00] JM: Many startups today begin their life as an open source project. Open source projects allow early adopters of technology to experiment, to contribute code and feedback and to shape the evolution of the project in its early stages. When a community maintainer company emerges to provide service offerings based on that project, its early customer base often consists of contributors to the open source project who have already had experience with the technology but want a more fully-featured offering. Orbit is a community experience platform that provides insights on activity for open source and proprietary projects. Orbit is founded on the Orbit Model, which it describes as a framework for building high gravity companies. That is developer communities with passionate and engaged contributors. The Orbit model is offered as an alternative to traditional sales and marketing funnels that gives a focus on cultivating early adopters for a project.

Patrick Woods and Josh Dzielak are the co-founders of Orbit. They join the show to talk about the importance of an active and passionate community for projects and how Orbit helps maintainers grow and shape their users experience as well as how they see Orbit evolving in the future.

[INTERVIEW]

[00:01:06] JM: Guys, welcome to the show.

[00:01:06] PW: Thanks so much. Really excited to be here.

[00:01:08] JM: I want to start off with a simple question. The role of developer relations, this was kind of a subtle throwaway role for a while I think. In the early days of developer relations it was somebody who just like managed the documentation or fielded random questions from the

community, but developer relations has really become a pivotal role within companies. Tell me about how developer relations has evolved.

[00:01:35] JD: Yeah. Great question. Happy to speak to that. Josh here. I think developer relations has come a long way, and the most recent wave that started I would say around the Twilio and Stripe right after 2010, ’11, ’12, ’13. I think those are some of the first companies that started to take developer relations really seriously. I put Firebase in that mix too. Where all the early success of those companies in the early adoption could be tracked back to the kind of activities that their developer advocates or founders or all of the early employees were doing. So as we saw that big uptake in API-driven companies, again, at the start of the last decade, that's where I think we really started to see developer relations become more – A serious team inside of these developer-facing businesses.

[00:02:21] JM: And tell me about the problems that developer relations members deal with, or I guess take me through a day in the life of a developer relations person. What are they doing? What kinds of tools are they working with and what kinds of challenges are they encountering?

[00:02:38] JD: Yeah, absolutely. As mentioned before, documentation and developer experience is a big part of what developer advocate or developer evangelist is thinking about. And that might mean they're building those assets. A lot of developer advocates are actually the ones building the documentation site, maintaining it, writing the code to do that, keeping it up to date. But then a lot of the tools of the trade of a developer evangelist are the ones to create awareness for the products. So that can be anything from the tools that they use to create talks. These days it's a lot of streaming. So video services like YouTube and Twitch, and as well as a lot of the tools that the rest of the company uses in order to stay in touch with customers and community members. Sometimes that can be spreadsheets. Sometimes it's more marketing tools. But when I was a developer advocate, I had to be familiar with almost every system in the company, both the engineering systems, because I was contributing to code base especially around the documentation and then also the customer-facing systems because I was constantly meeting customers, meeting community members out in the field or on GitHub issues and pull requests.

So the tools for developer advocate in the day in the life really varies between proactive, “I’m building stuff. I’m putting it out there,” and reactive, “Here's a community member. They're trying to build a thing. How can I help them build that thing? They're stuck. Can I do support? Do I need to get in touch with one of our engineers to help?” So it's really the balance of the proactive and the reactive between building and evangelizing, building and speaking, writing that character as the role I would say.

[00:04:06] PW: Yeah. And I would just add to that. There's this classification of thinking about the overlap between code, content and community. And sort of to just zoom out of what Josh was saying there. I think many folks characterize their role as a sort of distribution of their time and effort across those three areas and it can depend on the company strategy and the individual's goals for the quarter in terms of how they spend that time, but I think about that code, content, community breakdown pretty frequently when helping people understand the role and its impact.

[00:04:34] JM: There's a lot of disparate tools is what I’m hearing from you guys. And I think what you're working on is bringing together some of that spread out tool heterogeneity. Tell me about how the disparate tooling creates problems or bottlenecks and workflows.

[00:04:55] JD: One thing we like to say is that community is distributed, and so we observe that, and we see that happening in a lot of developer communities first and foremost. We have GitHub mailing lists, Twitter, Twitch, YouTube, Discourse, maybe GitHub discussions, the new thing from GitHub. The community is taking place across a lot of different channels, and the problems that presents are measuring it, attributing what parts of the community. What channels are actually helping it grow? And just getting – As a practitioner, as someone doing it day-to-day, being able to keep tabs on that many different services, which may also include a Discord or a Slack or both of them. We see people having that too. So where the developers want to go is where we have to meet them, and that can be very hard if it's in a lot of different places. So that's one of the challenges. When you have that data in one place, it's a lot easier

to be able to measure the progress that you're making and manage the workload when you have to bounce around everywhere, then that's obviously harder.

[00:05:53] PW: Yeah. It's been pretty interesting to have in-depth conversations with hundreds and hundreds of developer relations, leaders, community builders, things like that, and really to dig into their challenges. And you might be surprised to learn that the sort of simple questions of who is in a community and what are they doing is a non-trivial question to answer for most companies and projects of any substantial size. And it comes back to the distribution of that data across different channels. And the vast majority of our customers, before they started using Orbit or a tool like us, the spreadsheet was the state of the art in terms of understanding the impact of their efforts. And pretty much everyone we've spoken with has had a Google Sheet with one tab for tracking the number of pull requests another, another tab just counting the number of Twitter followers. And pull requests and Twitter follows and discussions on the forums, those are very different data types and it's kind of like apples to oranges. And so we found that the frontline folks, the DevRels, the community team, just didn't have the data to make decisions about how to work with the community, how to prioritize, how to deliver a great experience. And then that meant that their bosses, whether that's a VP of DevRel or marketing or what have you, didn't have the data to tell the story about the impact of those organizations. Yeah, you're spot on with your question. The distributed nature of the community is a big driver of both the challenges of these roles, but also the opportunity of understanding what's going on at scale. And it's a lot of what we think about trying to solve for.

[00:07:22] JM: You guys work on Orbit, which is a community experience developer relations platform. Can you describe what Orbit does?

[00:07:32] JD: Sure, happy to describe a bit of that. And then, Patrick, you can pick it up from there. So Orbit as a tool connects to all the different platforms where developer activity takes place like GitHub, Twitter, Discourse, Slack, soon, Discord. Many other services through Zapier integration connects to all these different places and it solves some of those key questions or helps teams answer them, the ones that Patrick was mentioning like, “Who is in my community? What are they doing? How is my community trending over time?” How's our

gravity as we like to call it in the Orbit model. Are we bringing more people in than we're losing? And so Orbit really serves as a single pane of glass into the developer community where someone working in developer relations or an open source maintainer or a founder of a developer pacing startup can see in one place everything that's happening across their ecosystem.

[00:08:20] PW: Yeah, I think that's right. I think getting the data into the platform is one part of the story. And then the sort of what next question that comes after that. So Orbit, the platform, provides tools like lots of charts and reports to query the data and understand the impact of different initiatives. So if you did a meet up last month, did that contribute to an uptick in Twitter conversations or forum discussions or more pull requests? Has anybody joined the community lately that has a lot of Twitter followers? Because maybe you just want to know that. But for many of our users it's questions like, “Who's been the most active over the past day, week or month? Who's doing a lot of activities such that we can recognize and celebrate those folks internally and out in the community as well to help further the aims of the community?” whether it's more contributions or higher quality contributions or whatever the specific goals and strategy are for that particular time.

[00:09:11] JD: One thing we like to say is manage relationships, not spreadsheets. And we hope that using Orbit gives a lot of that relationship time back to the community builder because they're not spending their day in spreadsheets trying to come up with spending hours a week reporting results to leadership or minutes at a time tracking down contact information.

[00:09:31] PW: Yeah, it's a great point. So many of our users were essentially human ETL pipelines before Orbit where they were like copying and pasting data between various things and trying to normalize data from various platforms. And you just spend hours per week doing that stuff and it's not terribly rewarding work for most folks and it distracts from what community builders are great at, which is actually building those communities and relationships. So that's a big thing we think about a lot is how can we reduce the manual input and the manual labor associated with the self-actualization that comes with community building. Can we get people out of spreadsheets and into more conversations?

[00:10:10] JM: Could you give an example of how a company is using Orbit? Just kind of comprehensively how it assists the company from a number of different angles.

[00:10:22] PW: Sure. Take a stab at an example. Here we've got a few. One of my favorites is the team at Typeform uses Orbit to understand what's happening with their developer community and deliver a great experience to new developers coming into the platform. So many folks know Typeform as the forum company. Sort of a b2b or consumery solution, but they do have a developer platform. And a guy named Nicolas Grenié runs that program and one of our early community members and users of Orbit. So he uses Orbit to give him some leverage in terms of managing that part of the business.

So one specific example is when someone registers for an API key with Typeform, that event, the sort of request event is logged as an activity inside of the Orbit platform, which means he can see quickly in one place everyone who's done that. But he also has instrumented his workflow using Zapier and some other tools to send an email out to those folks that have gotten the API key saying, “Hey, Nico from Typeform here. I’m the DevRel lead and would love to have a conversation with you about what you're planning on building. Here's the Calendly link. Grab 15 minutes.” And it's a pretty interesting way for him to orchestrate that workflow. And basically on the backend he uses a sort of essentially – And this is getting in the weeds, but I guess that's the spirit of the question. He essentially looks up in Orbit, “Is this a new member to the community? And if so, send this email. If not, don't send that email because I already know this person. They don't want to get spammed for me or whatever.”

And so what he's found is that those initial outreach conversations or emails have driven lots of sort of “conversion” to actual phone calls with developers getting started for the first time. And I think for the most part you know most developers aren't interested just jumping on to a Zoom call with someone arbitrarily, but I think he's found that the right message at the right time to the right person has really made a difference in terms of building that relationship and converting those people into advocates. And so for him it means he's not having to manually monitor the queue essentially of new API keys requested and kind of look up, “Is this new? Is

this person an existing member or whatever?" That all happens automatically. And so he can actually focus his time on just jumping on those Calendly meetings. Those meetings when Calendly schedules them to deliver that experience.

Now on the backend he's got Orbit as sort of a CRM style tool where he can see who that email's been sent to. Who's jumped on the call? He uses Orbit to take notes and add context from those conversations. So he has all of those touch points in a single place. And so that's for me a pretty great example of using the data inside of Orbit to trigger lots of activities and really use it as a central, almost a central point of inspiration, which might be true. But what I meant to say was the single point of orchestration across those different tools. Josh, I don't know if you have any others that really stand out for you as great examples.

[00:13:07] JD: The only other one I'd add is around reporting. We see a lot of users who download the reports that we produce directly in Orbit or members activities or metrics related to our own Orbit model, and those are going directly into their slide decks to share with their boss or to the executive team. Yeah, that's another thing that we try to make really easy is that reporting about the community.

[00:13:31] PW: Yeah, that's a great point. That's not a specific company necessarily, but it is maybe the most common use case, which is the impact of community building isn't quite as direct as perhaps like a revenue metric where you run an ad and somebody purchases something and it's a very close coupling of input to output. And so what we try to do with the reporting suite in Orbit is help folks have a combination of essentially leading indicators about what's working and what's not across whatever integration they have plugged in. So how are things looking on GitHub? How are they looking in the Slack community? How are they looking on Discourse?

Yeah, to Josh's point, we've heard tons of stories about folks putting those in board decks, putting those in pitch decks even, because for many early stage open source companies, the path to revenue is not super clear yet, and yet you still want to measure how are things going directionally with the community. And so that's I think a niche Orbit's been able to fill, is

providing those founders of open source, open core companies a lot of telemetry around what's trending. How things are going overall? And then having more meaningful conversations about how the community is being built versus just counting number of stars for example.

[00:14:36] JM: This is kind of an adjunct question to questions about software engineering, but I think it's relevant to your tool. What's the relationship between developer relations and sales departments?

[00:14:53] JD: That's a great question. I think that developer relations and sales, even though their objectives are often quite different, they are, tend to be two of the teams inside of an organization that have the best pulse on customers and community members who are out there in the field. So when I've done developer relations in the past to work closely with the sales team, my goals were to build the community. They weren't to move leads and opportunities down the funnel, but there's a lot of interplay and back and forth that can happen between the teams that's helpful to the goals of both.

Developer relations can provide warm introductions to community members at interesting companies. Maybe five people from a fortune 500 company have started contributing to the open source effort of a company. Developer advocate is a good position to know that that's happening to know who those people are and to facilitate the right kind of introduction to a sales team. That is more likely to lead to a good result than just finding someone's email and emailing them cold.

In the other direction, through the stream of new customers, sales are able to recommend a lot of developers over in the direction of the community. They're able to point a lot of the community members in that direction. So the developer relations team really benefits there. Building enterprise community is a really great example of that where sales after a deal is closed. The question is, "How are these developers getting up to speed on the new technology? What's their training path? Are they getting to know people? Do they know where everything is? And developer relations is uniquely suited to answer a lot of those questions. So the short answer is that there're great handoffs that can happen in both directions. The thing to

watch out for of course is the data, finding itself in the wrong hands of someone however good their intention. We don't recommend that community data be just put in Salesforce and then put into the same outreach cues as the rest of the other data, because usually community members haven't opted into that kind of communication.

So one thing we think about at Orbit quite a bit is what's the right data that should be shared between the DevRel and the sales team so that each can do their jobs? And so that, organizationally, when it's a fast pace, things are flying around, that community members don't end up being the result of outreach that wasn't really appropriate for them.

[00:17:03] PW: At a high-level we think a lot about teams inside the company. Some teams are oriented around capturing value. Optimizing the sales funnel for example and extracting terms of a deal, whatever. And some teams in the organization are reliant around creating value for the community, for users, for other folks like that. And so I think the DevRel team, community team, folks like that, are usually oriented around value creation in the form of training docs, community connections, things like that, and sales teams are often more around the value capture side of things. I think understanding who's doing what is really important in that relationship because those are both useful orientations in a business setting. But Josh had some excellent points around orchestrating that handoff and really asking yourself what is the appropriate experience for every, every person that's involved in this relationship.

[00:17:53] JM: Orbit is built around something called the Orbit Model. Can you describe what the Orbit Model is?

[00:17:59] JD: Sure. Yeah, happy to talk about that. And, Patrick, feel free to jump in at any time or after. So the Orbit Model is a visual canvas for community building. It places community members at different levels from the center. If you can imagine, a solar system that's laid in a two-dimensional flat piece of paper, you've got the sun at the center, and that's the team or the project. That's the thing that the community is coalescing around. And at varying distances from that center you have community members. These community members are Orbiting the community as they do various things as they participate in various ways. And

we break the Orbits down into four concentric circles that represent the degree of engagement the community member has. Starting with the inner circle, the inside, we call those advocates. The next members out we call contributors, participants and observers. And the labels aren't perfect. And depending on the community, their labels might be different. But it gets at the idea that the member is at a different part in their journey as they go through these different Orbits and as they go around the system. And it's okay sometimes for a member to go from the third Orbit level to the first Orbit level as their engagement increases. And then maybe they change a job or they start using a different technology and they come back out for a little while.

One of the differences with the funnel and the Orbit model is that in the Orbit model it's okay to go backwards. It's okay to go out for a little while before you come back in. Just in the funnel world, as soon as you're out of the funnel, it's kind of like, “Well, we have other leads. We're not worried about you right now.” But in the longer term, kind of non-binary, non-linear version of the model that the Orbit has, there's a difference there. So I think those are kind of the summary concepts. We love the space analogy and we are commonly sharing assets internally of our own community members flying around like they're on rocket ships. But that's how we think about their member experience of the overall member journey that we're trying to give them. And people, our community members, who are practicing the Orbit model, that's how they're thinking about it too.

[00:20:00] PW: Yeah. The whole idea for the Orbit model was really born out of a frustration we were sensing with our customers in our community with the sort of sales and marketing funnel being the only mental model anyone has inside of a business. If anybody's ever been in any type of meeting in a “business”, then like the funnel is a thing that people are going to talk about. And it's kind of like that old aphorism. When the only tool you have is a hammer, everything looks like a nail. And so we found that inside of companies. This led to questions like, “How many leads did we get from the forum last week or how many leads came from the meet-up?” And for those that have experience with community building, you know that's not quite exactly the appropriate question to ask for a lot of reasons, but it's hard to describe exactly why.

And so the Orbit model was really – Our attempt to model from first principles, how communities are actually shaped and how they grow? And it's not linear and binary like the funnel metaphor, but it is concentric. And the goals, it's less about extracting value and more about increasing the love and reach of the members of your community. So we found that by providing different lever, love and reach specifically. It changes the conversation around what you're trying to build for and measure and optimize. So to use the funnel as kind of the counter metaphor, it's like in the funnel world it's how do we optimize every phase of this, every stage of this funnel, to get more and more out of each step. But in Orbit model land it's more about how can we increase the love of our community or how can we increase the reach of the individuals in it. And we found that those questions orient teams and companies around much more meaningful conversations around how to actually build and measure the community. So to Josh's point, both a visual canvas to help have these conversations, but the model also comes with some ideas around how to measure love and reach believe it or not. We do think love is measurable over time. So that's the Orbit model.

[00:21:50] JD: That's right. And I'll add that it's open source. It lives on GitHub. So anyone wishing to learn or to have a conversation about it, it's an ongoing conversation. And the mathematics and some of the formulas are actually in there being debated on GitHub. So that can be kind of fun to check out for the mathematically-inclined.

[00:22:09] JM: There's a wide array of integrations that you have between Orbit and other platforms like Discord or Discourse or Twitter, GitHub. Tell me about the integration surface between Orbit and all these other platforms.

[00:22:24] JD: Yeah. I'd say the two core pieces of data that are flowing between Orbit and these platforms are the community members and then the activities that they're doing. So I could take GitHub as an example. GitHub is the first integration that we built because it was the most in-demand by our developer community builders. And so from the GitHub integration we're able to create profiles for each member who's done an issue, a pull request, a star. Basically there's a set of event types that happen on GitHub and we're able to import a subset of those that we think represent real community activity and then create profiles for those

members and also show an activity history inside of Orbit. So the activity stream that's coming in allows us to calculate that love and allow us to show that in the user interface. But the main surface area from the integrations or our API, the same code that we build the integrations on top of is available for anyone to call via the API that. The main things flowing back and forth of the members and the activities, which is the essentially anything that a member does.

[00:23:29] PW: Yeah, an activity is basically just a verb that you want to keep track of. So start the repo, submit it to PR, followed on Twitter, joined the forum, replied on Slack, submitted a forum. Whatever is meaningful for you as a company, as a community, we make it really easy to send those via the integrations or the API. And the Zapier app, lots of folks use this Zapier app to do some fun stuff too.

[00:23:53] JM: Tell me about some of the biggest engineering challenges you've had when building Orbit.

[00:23:58] JD: The biggest one has been the extent of the third-party APIs that we work with and that we try to normalize into one kind of homogeneous single pane of glass view for our customers. So the GitHub API is one example where we have a lot of different types of data that are coming in. They're like the pull requests, the issues, the GitHub stars, things like that. And we have to – We need a data model that can support all of those, but at the same time also discourse topics, discourse posts, Twitter mentions, Twitter follows, lots of different types of activities. And then custom activities, which is anyone that someone can create through the API.

So a lot of the engineering challenge has been designing that in a way where it's powerful but it's also approachable and where we can support in the future hundreds of integrations to lots of different community tools and still have those create a sensible end experience for the user who's looking at member profiles, looking at activity timelines. And then I think for the third party APIs too, that's just the question of learning all of the quirks. And there's a lot you can learn about an API from its documentation. But calling that API several times a minute every day for a year is going to tell you a lot more about it. So we've uncovered some of the quirks of

the GitHub API. We found that very interesting certain undocumented limits, like you can only get 400 pages worth of results for some repositories for performance reasons. That's a good one that if anyone else has used the GitHub API, maybe you've run into that. But there are things that are not documented. And so we've just had to build some things around there.

But I think by and large, really, both in the engineering and the product design side, it's taking these disparate types of communities, these different disparate channels where community happens and making that into a seamless experience both for developers working on our code base and also for the end user who's sitting in front of the tool. The only thing I would add to that is that some communities are very large. Kubernetes is an example of one community that we work with, and that's a lot of activity coming from a place like that. So just in terms of scalability has been an interesting thing engineering team-wise in the last six months as the company and some of the larger projects, open source or commercial that we have grown talking to potentially tens of thousands of activities a month coming from some of those places.

[00:26:30] JM: So give a little more detail about how the Orbit platform can be used to build custom workflows. And what would be an example of a custom workflow?

[00:26:41] PW: Yeah. So I think there's a couple ways to answer that. There's what's going on today and then there's some pretty exciting stuff on the roadmap as well that we actually demoed this morning in all hands. But we found that the community builders all have their own workflows and playbooks. And so it's been really fun for us to learn what everyone's doing. And so some common ones are, sort of I alluded to earlier, a lot of times you just want to know who's been the most active contributor for the past month. And so a common workflow is once you've integrated all your platforms into Orbit, your GitHub, your Twitter, your forum, what have you. It's really easy to see who's been the most active and you can see that across all those channels or at the channel level as well.

And so some of our users will actually highlight and shout out their most active members from those different channels. And so it's like, “Hey, here's the three people that were most active across the whole community this past month. Shout out to them for being amazing.” Grabbing

their information and dropping into the newsletter essentially to highlight them or even mentioning them on Twitter to say, “Thanks.” Recognizing, rewarding and acknowledging community members is such a big part of the workflow. And so Orbit makes it really easy to see those folks and quickly mention them and celebrate them. Today the sort of last mile to that is off platform essentially. So someone would see that list of people inside of Orbit and quickly grab their Twitter handles, for example, and then compose a tweet.

What's coming soon is some automation or I guess some semi-automation at the end of that step essentially to help people actually close that loop if you will. And so actually sending the tweet from Orbit or sending a web hook to another system based on a rule set that's been instrumented. We see people using the concept of tags pretty heavily to build workflows. So inside of Orbit, you can tag members with different labels, any arbitrary label that you'd like. So common use cases are using labels, using tags to keep track of champions program members or users of specific products or members who have contributed a lot in the past. And so we see people even using tags to follow up on things. So adding a tag called follow-up and then being able to quickly just go to the members table, for example, and seeing everyone that will follow up to you.

And so the specific workflows are pretty unique and specific to the community and what they're trying to do. Coming up soon is some pretty interesting orchestration around sending alerts and web hooks to other systems based on rule sets inside of Orbit. So you could say that anytime someone with a tag VIP has a pull request merged or submitted, then send me an alert to the VIP Slack channel so we can make sure that that was a great experience. So it's for us becoming a pretty interesting conversation of using the data that's inside the platform to then trigger cool stuff and useful things across other tools as well.

I would say another workflow that is a little less workflow and more just common task is hanging out in the activities feed inside of Orbit. So the activities feed basically aggregates activity from any channel that's been integrated as well as the API. And so it's essentially like the one tab in the browser that shows you stuff that's happening in your GitHub, on Twitter, in your forum, whatever. And so we know that a lot of our members just hang out there and lots

of folks just leave that tab open and quickly respond to issues as they emerge. If anybody with a lot of reach shows up, they usually will reach out to them and say hello. Welcome to them to the community manually. If they notice multiple comments on a particular issue, that may be some reason for them to go and investigate more deeply. So at some point boils down to just visibility about what's happening across all the various channels and giving the team leverage because you only have so many eyeballs that can look at so many tabs at once. By aggregating all that data in one place, it seems to be a useful way to help folks just quickly triage and react where appropriate just in service of delivering a better experience overall.

[00:30:32] JM: So today you've got a system that ingests lots of information from lots of these different content and community platforms. You can use it as a system for defining workflows. A system for storing your catalog of different members, of different community members. How does that compare to the overall vision of what you're trying to do with Orbit? What is the long-term vision and what are you building towards?

[00:31:00] JD: I'd say the long-term vision is to make community builders, community professionals, developer advocates successful in their roles and give them the kind of capabilities and a seat at the table inside of the companies that they work in along with their colleagues from other teams like marketing, sales, product and engineering. So one of the things that we test all of our product feature ideas against is is this helping our users successfully build their communities and build their teams? Get resources and budget where it's appropriate? And so I think that that's usually one of the litmus tests that we use when thinking about new product features. Is this giving time back? Is this is rolling up to the vision of making it easier to build communities and making the people who do it more successful in their roles?

[00:31:51] PW: Yeah. We've seen firsthand the impact of a thriving community on a business both on the business itself as well as the self-actualization of the community members as well. So we're excited to provide the infrastructure and the tooling to help operationalize all the things that we know work well and then give folks the data to prove that it's working. And so I think this is how we mature the field and ensure that DevRel teams and community teams and

people that are focused on creating value that they have the resources, the credibility and really the influence inside the organization the same way that a customer success person does or a sales person does. Those folks can all just push a few buttons and have a lot of reports to show the impact. And community hasn't really had that. It's been a manual game to this point as we mentioned related to spreadsheets. And so we're hoping we can provide the level of tooling and infrastructure to help this whole new way of thinking about creating value and going to market in the same way that other CRMs and marketing tools have kind of figured out what we would say the linear funnel-based world view of today.

[00:32:50] JM: How does the community building process compare between or a company based on open source software and a company based on closed source software?

[00:33:01] JD: I think one example would be what you're hoping for community members to do or to make and how you guide them toward that. So in an open source project, very often it's the contributions themselves. You're managing a contributor community that has a core team. Some casual contributors, maybe a long tail, and it's about improving that project there. In the case of a closed source, like an API or a SaaS solution, it's not necessarily the core that's – It's not the core that's trying to be improved. So the question is what kind of contributions are we trying to encourage developers to make? And a lot of times that's building and sharing the things they're building on top of the technology. A lot of times it's content, writing about their experiences, or sharing their code, their integration code, their projects so that other developers can pick them up and be inspired from them. So a lot of the things are the same I would say, but instead of guiding members toward making contributions to the project directly in the code, it's more about showcasing their experience, connecting with other community members. This is how we're using the tool. This is what we're building. But I think a lot of the things are fundamentally similar. Even products that are not open source tend to have other open source projects around them, SDKs, for example. The companies that I've worked at, even if they were closed sourced to the core, there was SDKs, other projects, things that were open source. And those are things that I would encourage developers of our community to contribute to or just to use because it maybe solved the problem that they had.

[00:34:39] PW: Yeah. It seems like in open source companies, much of the focus is on the building, the project itself, to Josh's point. And then what we see for companies for whom open source isn't as big a part of their effort, there's a lot of emphasis on skilling up and leveling up and the community members helping each other get more effective, sharing ideas, becoming power users. We see this in our own community. It's kind of meta, Orbit on Orbit, but our community members are very active and channels on our Discord, such as the show and tell channel where people talk about the different flows they're building on top of the API or stuff they're doing with Zapier. Some folks have built connectors to Orbit and other automation services like n8n. The DevRel there built an Orbit node for that. And that's a conversation that happens in show and tell. And people asking questions like, “How can I better run reports to show the impact?” Just sort of like practical application of the tool itself. And so I think for communities that aren't specifically around open source, that's where you see a lot of the emphasis, is kind of co-education, if you will, and sharing best practices, ideas, hacks and things like that.

[00:35:47] JM: Do you use Orbit to manage your own community?

[00:35:54] PW: Yes. I love this question. It's where I spend most of my day. And that also means that our product team in Orbit probably really loathes me because I tend to find all the edge cases and all of the weird issues. Yeah, it's been fun to build a couple of things at once. And what I mean by that is we've got the Orbit model concept that we're constantly refining. We have Orbit the platform. The software we've been talking about today that we're building and learning about. And then we have our own community of early power users essentially. We've kind of been building all three at once, and it's been really, really fun for me because I’ve seen the overlap in those areas. And so the community really started a couple of years ago when Josh and I were consulting in the DevRel space before starting Orbit and the Orbit model started its life as a blog post. And we put it on a repo and people started committing. And the community kind of started there. And then we started a company and built a product and we've learned from our community every step of the way around the high-level concepts like the Orbit model, but also like specifics of their day-to-day and understanding how we can use software to alleviate a lot of the boring manual stuff and give them super powers.

And so I've been user number one of Orbit essentially to help identify the folks in our community who were contributing a lot or people who were active a few months ago and who we haven't heard from in a while. Maybe I should reach out to them. We've instrumented things like our product milestones into Orbit as well. So when someone creates an Orbit account, I can see that in our Orbit workspace when they start inviting users. That's a pretty neat milestone. So I can see that in our workspace as well. So at the individual level, it's pretty powerful to be able to see a year ago. Someone started the repo and then followed us on Twitter. And now today they're using us and they just invited five people to their workspace. So that for me is a great reason to reach out and say hello. And so those are some – I've got lots of workflows that I use. Specifically tagging and notes pretty extensively. We have notes just like many CRMs would, but two cool things about notes in Orbit is that they show up on that activity timeline I mentioned earlier. So if I drop meeting notes into a profile, everyone in the company can see those notes in the flow of the activities which is great for context sharing. But also the index of notes are – Content notes or index research. So if we're searching for specific ideas later we can see that inside of Orbit as well. So that's a dangerous question to ask me because I have lots to say about it. It is quite meta, I will admit, but it's been very fun to essentially co-build our solution alongside the community as we've learned and I've learned alongside of them as well.

[00:38:25] JM: Well, as we begin to draw to a close, I'd love to get your guys perspective on the future of software in general. You sit at a very unique position building a pretty unique product. I haven't seen anything like Orbit. So tell me just how you see the world of software evolving in the next five to ten years.

[00:38:44] JD: I think I continue to see the increase of open source and in open core companies in particular, companies that are founded around open source projects. We're able to see from our customer base just how fast that trend is increasing. So I think there's some interesting things that will happen just to the business landscape in general when we have these companies built around open source projects that have had to get good at building a community, because if that's a big part of what a company is doing, that's just so important.

And so what I what I think we're going to see is the rise of more of these companies where there's some community that's very early on in their DNA or they think about their customer base through that lens and that kind of shapes the rest of the organization. So I'm really interesting to see how is the structure of organizations that build, create, maintain market and sell software. How is that going to change in the next five to ten years? And I think it's going to become more about bottoms up adoption. More about full life cycle, member management of communities, more individualistic in that sense and more contributors, hopefully more maintainable software as we kind of put the burden of contribution or the opportunity of contribution on top of more people rather than just the core. So these are some of the areas that I'm excited about.

[00:40:03] PW: Yeah. And I would add the big shift that I've seen in the past really year has been the pressure from an early stage investors for founders to have a community strategy. This is something that even, I don't know, two or three years ago we really didn't see. And there's there was a time that you sort of alluded to at the beginning of the conversation where ideas like DevRel were kind of like, I don't know, a nice to have or some something, some adjacent to the business sort of functional area. But, today, early stage founders are being asked like before they start the company, "What's your community strategy? How are you going to drive bottom-up adoption?" And that's a big shift from where we even saw it even a year or maybe 18 months ago. So that's I think a pretty big shift and how maybe less about how software is built, but how software companies are built. And I think that's really quite exciting for me to see.

[00:40:50] JD: Yeah. I think we're seeing that first developer relations or advocacy higher for these companies move within the first five and definitely the first 10 people, whereas more of two, three, five years ago was a question of 20 people, 30 people. Some of the more forward-thinking companies I think we're doing it earlier, but now that's moving up. And that just represents a shift in those priorities around building the community. And ultimately I hope it leads to better software because we're getting more feedback into these companies from the community and from real users as things are unfolding. The way we get great products is by having a lot of early adopters banging on it and testing it and giving feedback, and building a

community is amazing way to make sure you've got real feedback coming in. So I'm optimistic around that for how that plays into the field of creating software because we know that sometimes software isn't good because there just wasn't enough early users to really shape it the right way and give it the features that it needs. So I'm optimistic for the feedback coming in from the community to help improve the software that gets created.

[00:41:52] JM: All right, guys. Well, thank you so much for coming on the show. Is there anything else you'd like to add?

[00:41:57] JD: Thanks a lot for having us. It was really fun.

[00:42:00] PW: Yeah, thanks so much.

[END]