Developer Ecosystems: Build yours for projects large and small

I'm here to help you build your developer ecosystem, large or small. Um, I have been doing this building tools and technologies and developer ecosystems. For an embarrassing long time. Uh, currently I'm a product manager, uh, for Meta, uh, where I work on our augmented reality, uh, tools and frameworks.

I'm not here for meta today. I'm here representing that kind of, that long string, uh, of ecosystems, uh, that I've worked on in the past. And so what you see here is my, um, personal email address as well as how to get a hold on of me on Twitter or GitHub. Uh, or LinkedIn if you'd like to. Okay. So I have been to Poland before.

I have been to, uh, Warsaw before, uh, a few years ago. I came, um, uh, to give a conference talk much like this one. And I learned some, uh, interesting and important things actually. Can we switch these slides to present mode? Actually, I think I can switch these slides to present mode. No, I can't. Switch these slides.

I am view only if someone from uh, uh, is there a button? Oh, look at that. There it is. This is the audience participation portion of the lecture. Thank you very much. Helping me find these slideshow mode. I'm embarrassed to say I worked at Google for seven years and I didn't look at the for that button. We got it.

Great. So, uh, last time I was here in Poland, I learned, uh, a couple of very important things. One thing I learned is the, um, the people here, uh, in Poland are smart and fun and friendly and very, uh, welcoming. So thank you for that. I also learned about the huge number, uh, the abundant number of green spaces, which unfortunately this time of year has triggered my allergies.

So if you, um, see me sneeze, I'm, I'm, I'm not carrying the plague and I am definitely not allergic, uh, to this conference. It is, it is the, it is the green spaces. So sorry about that. But the most important thing I learned was that Poland creates the, has, uh, created the world's most dangerous alcohol. Um, and uh, I don't really remember much about the evening.

I discovered, uh, knick, uh, but I do remember a great deal about the morning after and mostly the most important thing I learned the last time I was in Poland is avoid the knick. So I will pass that along to you. Okay. Now let's talk about what you came for, which is learning about ecosystems. So these are some example ecosystems.

These are, these are the names of technologies. That's true. But each one of them has an ecosystem behind it. And I'm gonna ask you one at a time to, you know, just raise your hand so I can get a sense of where we are. Um, raise your hand as I read out these technologies, if you've written an actual application, not if you've heard of it, not if you've read some docs, but if you've actually written a thing.

That actually uses these technology. Raise your hands and that'll give me an idea. So win 32. Some old Windows programmers in the audience. Windows Telephony framework, tappy, couple of folks. Nice. Um, mfc, C plus plus, I'm sure a lot of you have done c plus plus for sure. Common htl, couple of folks. Sorry about that by the way.

Net C sharp. Yeah, lots of net. Uh, that's a much more pleasant environment for sure. Uh, WinForms, WPF and of course Visual Studio. Right. Okay. So all of these technologies, as I say, represent ecosystems. All of these

ecosystems. I was a contributing member, and we'll talk about what that means. But in this particular case, that means I wrote a book, a magazine article, a five day short course.

Gave a talk much like this one I actually gave back to the community, and we'll see why that's something you, uh, will really want to encourage in your own users as we move along here. Now, the next set of technologies where I was actually, uh, a product manager, I was actually helping invent and push these technologies, and I'm just curious how far they made it into the world.

Oslo is anyone? No, not the city. In Scandinavia. Yeah. Okay. Sad. Um, win js. Oh yes. Sorry again. Sorry about that. Yep. Google Cloud platform folks have written some G GCP apps. Great. Android, of course. J Pack flutter. Great. I'm a big fan of Flutter. And then the tool, uh, and technology I work on today is called Meadow Spark Studio.

Any. Any meta spark studio, well, that's, that's for 3D creators. You're not really our audience, so I won't take that personally. And then on the bottom I've listed, uh, a set of an arbitrary selection of, of open source projects. I've, I've been a contributor on, so,

Here we go. Now, what is an ecosystem, right? I just said these are a bunch of ecosystems. What the heck is one? Well, it's really everything and it's everything that has to do with the technology. Obviously it's the team building, the technology, whether it's an individual, um, open, uh, uh, directed open source project, or, you know, uh, a team of folks building a framework or an operating system or a.

Important library or technology. It includes the releases that team builds. It includes the applications and extensions that your users build with your technology. It includes all the docs and samples and videos and blog posts and Facebook posts and tweets and events and all of the things, right? That's all part of your ecosystem.

And most importantly, and when we talk about. Building developer ecosystems. This is turns out to be the most important thing. It's your users, your people using, um, your technology. Okay? So when you're building, um, uh, an ecosystem, you have, uh, a, a choice, right? What kind of an ecosystem do you want to build?

Now, I, I, I, under, I'm gathering from the people raising their hands, this is a room full of engineers. That's great. You're my people, right? So there are a small handful of, uh, ecosystems that were built almost completely with the engineering alone. We, it was just, Hey, we're gonna build some tech. We're gonna put that tech out and, and then we're just gonna build the best tech we can and the, and the ecosystem will catch fire.

And I've listed some successful, largely what we call unmanaged ecosystems. Up here in the slide, turns out it's pretty difficult to come up with large, well-known, uh, ecosystems that aren't actively managed, uh, by some team or organization to make them successful. That's why it's a small list. Um, however, you know, 99% of.

Of open source projects fall into this category, they're unmanaged. Somebody releases some source code. They list, you know, in the release notes that list of bugs that are fixed, and then they wonder why they don't have users. Right. On the other hand, we have the idea of a managed ecosystem and I, I have a small smattering of examples there, but there we can come up with a bunch of actively managed, um, Uh, ecosystems, and when I say managed, it's somebody actually is working to make sure that not only is the technology there, but all the rest of the ecosystem is there too to make it successful.

And a tiny percentage of open source projects get this right as well. And I'm here today to tell you what those techniques are. So no matter if you're, you know, building a framework or an operating system or an important

library at a, a large, you know, well-known organization, or if you're working on an open source project, you can use these same techniques to make your project successful.

Great. Now, as I mentioned, engineering is not sufficient. But it is necessary, right? You have to be solving a problem that people actually care about the solution for, right? So that solid engineering, solving a real problem in a way that people can actually consume. That's, that's, you know, zero number zero on the list.

You have to do that first. Okay? If you don't have that, nothing else you do will matter. Um, but if you take that solid engineering and you add that user focus that I'll talk about, That's when you can have a successful ecosystem. Okay.

All right. So what kind of users are we talking about here? We said we gotta focus on users. What are we talking about? Well, there's really two kinds of users that we're worried about. The bulk of your users as you mature over time are going to be consumers, right? They're gonna be the ones that show up at 9:01 AM and they grab themselves a coffee and they talk O with other people over the water cooler, and they read some email and eventually they wander over to their computer and they start their day using whatever technology somebody has told them to use.

Right. That is going to be the bulk of users of whatever your technology is, somebody has already decided for them. The technology that they're using. These, like I say, ultimately are the bulk of your users over time. But these are not the ones that you wanna focus your efforts on because these are not the ones, none of you by the way fit this category cuz you're here, right?

You're giving back. You're here at this conference. You're reading the blog post, you're asking questions, you're posting bugs to your favorite libraries, right? The, the people that you want to focus on are the ones that contribute back to your ecosystem over time. And so we'll talk about the kinds of things that we want contributing members, uh, of our ecosystems to do, and how to focus on them so it makes them want to do more.

And it makes, um, your ecosystem pull in more of these contributing users. Now, I draw it as a, um, as a pyramid because your con contributing users are always gonna be a fraction of your total user base. Over time as your, your, uh, your technology, uh, becomes, you know, more mature, it'll be a smaller percentage.

We, we like to think of it as like the top 10%. Okay, that's what you want to focus on. But in the beginning of your technology, there might be nearly a hundred percent right? Early adopters, folks that are willing to take a chance on a new technology. That sounds pretty cool, right? All of the people using L L M today, all early adopters, right?

All people, right? Trying to think of all the crazy things they can do with LLMs, cram them into applications, build extensions, do all these crazy things. All early adopters, all contributing members, all giving back to the community. Okay? But regardless, remember I said to consumers, somebody made the decision.

What technology do you use? Right? The contributors, these are the ones that make that decision. They're the ones that attend the conference talks, and read the blog posts, and watch the videos. They're the ones that, you know, rewrite your application over a weekend with this new technology because you think they think it can be just better, right?

Those are the ones you wanna focus on, and the reason you wanna focus on them is because ultimately they become your champions inside of their team or organization or company, right? You want them to bring your tech into what they're doing. That's why you focus on them. Okay, great. Now, how do you do that?

How do you focus on them? Well, it's really simple.

United States, we have something we call a walkie-talkie. You guys, what do you, what do you, what do you call 'em here in Poland? Do you have those? We call 'em walkie-talkies. It's perfect. Great. So the idea of a walkie-talkie is, you know, uh, you know, I had a, when I was a kid, you just barely worked. Right?

Somebody was standing a hundred feet away, right? And that was as far as it could work. And the deal is you push the button and you say your thing, and you take your finger off the button and you listen carefully over this, the static, the noise to hear what that other person is saying. Right. That is the perfect analogy for what we're talking about here, right?

You wanna say your thing, you wanna say whatever it is you wanna say about your technology, and we'll talk about how to think about that, right? How do you say it? What do you say? But then you take your finger off the button and you listen. Okay.

So how do we do that? How do we speak effectively about how do we, what do we say when we have our finger on the button and how do we say it first and foremost? How many people have heard Guys? Guy Kawasaki, couple of folks. Okay. Guy Kawasaki, um, he was on the Macintosh team back in the eighties. Right.

Remember back in the eighties, the IBM PC was everything in computing, right? It was dos, it was one application at a time. It was all text mode. We look at that and we, and we think how silly that was, but at the time that was everything, right? And then the guys at Macintosh, at Apple, Steve Jobs was busy building this crazy computer with folders and icons and graphics and a mouse of all things, right?

And so Guy Kawasaki and his team had to go and talk all of these IBM PC developers into building applications for this brand new world cuz they couldn't be successful without their users. They couldn't build all the applications that they needed. They couldn't build, you know, the spreadsheet and the, and the slides and the, and the word processing and the calculators, and they couldn't build all the things you needed.

They needed external developers. So they invented this idea that they called at the time software evangelism, which we call developer evangelism or developer relations today. And what he said was, when you're talking to engineers, fundamentally it's marketing for sure, but you cannot lie. The thing you deliver has to do what you say it can do.

Right. And the reason is, is because you are talking to engineers, you cannot sell an engineer, right? An engineer will hear a sales pitch from a, from a mile away. They're like a dog with that dog whistle, right? They hear that high pitched wine as soon as someone is trying to sell them on something, right?

And they'll, they're out. They're gone. Right? You cannot do that. Whatever. Wherever you are in the stage of your product, whatever it can do, there's also gonna be things that it can't do. There are also gonna be things that you're thinking of it doing in the future and things that you aren't, and there's gonna be bugs and problems and you have to be super transparent about all of that, cuz if you aren't, you don't feel to the engineers like you're selling them and they are gone.

They are out. Right. First and foremost, you must be sincere and speak the truth to engineers. Excellent. Now, what are you doing? This is kind of product managery, but what you're really doing is you're building a funnel, right? Because what you're trying to do is attract new people to even think about using your technology.

And then they'll move through various parts of a funnel and we'll talk about what those are. But the reason it's a funnel is it gets smaller and smaller number of users until out comes at the bottom. Actual people with applications or extensions to your technology that they ship out into the world, right?

That's the job. Get as many new people in as you can. So you have applications that come out the bottom. Great.

Now, how do you do that? How do you get them into the top of the funnel? Well, the phrase I like to use is launches, not releases. What's a release? A release is the engineering team has put a bunch of stuff together and they have a release notes that says, here are all the bugs we fixed. And some of them are features that people have requested and some of them are just bugs.

And we, here's the list. Be happy. Dad does not get anyone excited. I can't tell you how many times that I've been following us, uh, open source project that I like. That is cool. That makes me happy. And then I go, they have a new release. That's awesome. And then I see this long list and I'm like, what? Which, when, what in here am I supposed to care about?

Right? What is here supposed to excite me? Right. What's supposed to make me happy? A launch, oh, and by the way, releases, they should be, you know, at this day and age, continuous, right? We should just be having point releases and point, point releases and whatever. Something's fixed or whatever. Almost continuous, right?

The phrase I like to use is ship early and often, however, launches have to be more measured. But more importantly, they have to come with a reason to care, or, or, or in fact, ideally, as many potential reasons for different users to care, as you can put in there, right, a launch is a release plus a story or a set of stories about why you care.

Now, of course, you wanna make sure that the existing users have new features and new bug fixes. And that you are making them happy always, for sure, but you also want to be continually supporting new capabilities that expand the user scenario, that expand the potential user base, right? Because what you're trying to do, remember it's a funnel.

We'll talk about the stages of the funnel, but the bigger you make the top of the funnel, the more folks are gonna come out the bottom producing things with your technology. Okay. So you want as many new users, uh, in the top of the funnel as you can make, which means those launches have to have reasons for new users, uh, to care as well.

And then this drives me crazy when companies do this. When's the last time you saw like this cool demo, right? Maybe this conference show. I mean, you know, Google io, we just had that big conference and Microsoft build. Right. We just had that big conference and it drives me crazy when at the end of this super compelling demo, what they say is, and you will be able to use this in six weeks, six months, whatever.

Well, who knows where I'm gonna be in six weeks or six months, what I'll be doing, you have my attention right now. I want to go and download and try your thing right now. I'm excited right now. Right. You want to give your users, when you do a launch, a call to action that they, when they, at the peak of their excitement, they will

never be more excited about their tech, your technology than after they've just seen the most amazing demo ever.

Take that energy and pour it into your ecosystem by making those releases available right now. Right. Okay.

Great. So what are the reasons to care? What are the things that you can say to your users that will make them go, that's amazing. I need that right now. Well, for sure new, new features. Um, ideally new scenarios. So you widen the funnel for sure. Polish right, reducing the number of bugs, um, improving the performance users.

Always love to hear about that. New productivity. Hey, you've included a new tool. You've made the existing tooling better or faster or stronger. Users love to hear about that. And one thing that users, um, love to hear about is version numbers. From an engineering point of view, that seems silly, right?

Because a version number is just, it's a string of numbers X, Y, Z. Right, and depending on somebody in product probably is what, which one of those numbers you increment and you don't care. It doesn't matter to you, right? By definition, any version that's that's higher than the previous version is better, right?

So what difference does it make? Turns out, makes a difference. Makes a difference to corporate. It makes a difference to CTOs, makes a difference. Uh, to CEOs, right? Engineering leads. It makes, essentially, it makes a difference to anyone who's not in the actual guts of your technology, which is gonna be most people, right?

And the reason it makes a difference for them is because they're trying to help their team make a decision about whether to adopt this technology and when to adopt, uh, this technology. Those, those higher level decision makers, um, they have to make a choice, right? When to use your technology, whether to use your technology. If it's, if your version starts with a zero, that means to them pre-release, right? It's a beta, it's not ready for production. Great. If your version number starts with a one, uh, that means it's production ready.

As far as. The people building the technology. So if you're really, you know, excited about it, one oh is probably safe, but it probably doesn't do all the things. Two Oh aha. Two oh is twice as good as one. Oh, right. So we're ready. Okay. Now we're all set. Right. Those version numbers matter and so I can can hear the engineering brains in all of you going, oh, that's easy.

Right. The first alpha, that's one. Oh, right. In the second alpha, that's two. Oh, right. We'll just game the system. Bad idea, right? This is the classic boy who cried wolf. Right? You have to be clear about the, what those version numbers mean because people are gonna be making decisions. They make the wrong one.

That's it. You've lost users forever, right? So turns out version versions matter. Another thing you wanna, um, Uh, take your time when you're doing launches, and by the way, you know, a launch can be packaged however you want. And maybe it's a blog post. Maybe it's a keynote and an event like this one. Maybe it's a talk at an event like this one.

Maybe it's a video. However, however you do your launches, this is the kind of information you need, and one of the pieces of information is you want to celebrate your contributors. The people that are taking your technology and doing amazing things, as you move from beta to one, oh, you wanna celebrate those users that were there with you at beta doing amazing things.

Show off their stuff, do co-market, right? Show how cool their stuff is because not only do you generally feel, you know, happy that they gave you that feedback and contributed, and you want to give back to them, but

also you want to encourage that behavior. Right. You want other contributors to go, wow, what a great ecosystem.

I wanna be part of that ecosystem because look at what they do for people that are part of that ecosystem. That's really cool. I want to be part of that. Great. So that first set of reasons to care is, hey, this is all about, we care about you, right? We're doing work because we've heard back from you. We've heard what you've asked for.

We've fixed the bugs you've asked for, we've added the features we've asked for. Another part of any launch has got to be, Hey, by the way, it's just, it's, it's not just you, other, other folks care about us as well. For example, you know the numbers of your user base, right? Maybe you've gone from 5,000 to 10,000, maybe you've gone from a hundred thousand to 200,000, whatever, whatever the number is, right?

Whatever the numbers are, right? Let people know things are growing, things are getting better. Nobody wants to hear about an ecosystem that's flat. Nobody wants to hear about an ecosystem that's losing users, right? They all wanna be part of, you know, a rocket ship heading to space, or at least a train leaving the station forward motion.

That's what we all want be part of. Okay? So you wanna be talking about your user base. You also want to, in addition to recognizing your contributors, you wanna have, if you can, well-known brand name customers. Someone from Acmecorp, that's a made up company, but everyone knows, Hey, somebody from Acmecorp has done a thing.

Let me tell you about it. Let let them, in fact, even better let them tell their story of your technology. Cuz it's always way more powerful to have somebody else telling your story than it is you telling your own, you're telling your own story, you're tooting your own horn, right? Of course you're gonna say nice things about yourself.

Right, but when somebody else says it, that's even more powerful. I'll give you an example. The other day I was reading that Microsoft is replacing 180,000 lines of c and c plus plus code with rust. The rust developer relation folks were dancing the jig when they saw that announcement. They can take the rest of the day off.

Their job is done because Microsoft saying how amazing Rust is, and not just saying it, but voting with their feet to replace hundreds of thousands of lines of goat in Windows, one of the most important popular operating system on the planet has just said this thing about rust that is so much more powerful than anything.

That the rust team could say, right? Put your lighthouse customers up front. Let them tell your story again, whether it's a quote, a video, you know, a blog, post, something. Let them tell your story. And also, it's very important that you can tell your story about using your own technology. In the US we call this eating your own dog food, which is a particularly gross way to think about it.

Do you have, do you have that here? You use that same phrase. I'm sorry about that. You've infected your language that way. But the idea is if you are using your own technology, and we saw this uh, in Flutter, the Google team, it was always very powerful to, to talk about teams at Google who have a choice.

Choosing to use flutter because of X, Y, and Z, right? If you can't, and by the way, here's the other problem. If you can't come up with other groups or teams or whatever that are part of your organization using your own technology, that's a problem, right? Because it means not only can you not tell this part of the story, but it might not be building a thing that enough people care about to be able to tell this story.

Okay, so that's important. And then, you know, there, hey, we just got awarded the most visionary product in Forbes, top 1000 Visionary Products or whatever. Right? Um, you know, the number of, uh, questions people ask about your technology on Stack Overflow, the number of, um, Uh, oh. GitHub stars, right? That you have, you know, various things like that, right?

If you can, you know, keep track of those kinds of things. Um, those are, uh, super useful to report as well. Now, if I think about, you know, the, the things on this launch, I'll give you an example. The Flutter team, the other day, they released a, uh, a launch where they said, Hey, we're doing more for iOS developers with this launch.

Uh, that we have in the, in the past we're doing, we're, we're, if you're an iOS developer, come to Flutter. That was essentially their message. But they started with, um, Hey, by the way, did you know there's a million apps built using Flutter? Right. Which means, you know, millions of devs, right? And this is up from, I don't know, whatever the last number is.

They said, okay. And then they said, Hey, by the way, Did you know that Newbank is using this? Did you know that PUBG is using this? Did you know that Google is using this? Right? So they started talking about all of the other folks that you've heard of that are using it, this technology, long before they ever started talking about the iOS features.

They started by talking about, Hey, this is a technology you should care about because other people care about. And then they talked about, oh, and we have some new features and we've done some performance work and we fix, we've listened to you and fixed some bugs and we've made the tooling better for iOS developers, et cetera, et cetera.

Textbook, right? Just like this. Exactly what was on this slide. Okay. And I think if you look, if you watch, um, from now on, uh, launches, you'll see that the good ones look like this.

Okay, so as I mentioned, there's various stages through this funnel, and the important thing to realize is this funnel gets smaller and smaller as you keep going down, right? It's your job to make this funnel be as wide as possible at the top, and to make it look as much like a tube as possible, okay? But you're gonna lose folks every step of the way, and it's your job to figure out how not to.

So first and foremost is, hey, somebody has discovered, uh, your technology and you bring 'em to the website and they want to see that the, they want to hear about your technology in a way that fits their problems. They don't want a list of features. They want a list of problems that this helps them solve.

Right Features is how we think about it. Problems and the fact that we provide solutions. That's how they think about it. Okay. Then you want to get 'em to the next stage, which is a ha. You're here on our website, you're looking at our stuff. Get started now. Big button that says, download the thing, whatever.

It's right. How do you get started? And you wanna make that super easy to download and install and upgrade and, and you wanna make it super easy for them to see the samples and use the tools. You want to think about how long does it take them for them to go from. I now want to try this to, I'm actually writing that first line of code against whatever, right?

Firebase was the master of this before Firebase was acquired by Google. I remember going to their website and I'm like, oh, this is pretty cool. And then they had right there a code editor inside the website and 20 minutes later I'd realized I'd spend the last 20 minutes writing the Firebase code. Wow.

Right. Their time to code was like 60 seconds flat. It was amazing. Okay. That's the kind of thing you want to do is help them get started super quick and then you wanna get them into development. Okay, great. I, I like it. I think it'll solve my problem. It seems easy to use. I can get started quickly now. I want to commit.

I want to develop a thing. Great. So now you want. Great docs, great samples. You know, you wanna, again, your documentation to be solution oriented. I know you have these problems. Let me help you solve these problems with this technology, okay? Um, and then deploy. What does it mean to deploy, Hey, you're targeting my operating system.

How do you package your thing so that users, your users, Can install their thing on this operating system and, and use it. You're, you're building a framework, Hey, how do you deploy? You're an app built with this framework to whatever set of devices it supports. Hey, you're building a library. Is there anything I need to know when I'm deploying an instance of my app into the field using your library, right?

Does it have analytics? You don't have to worry about it crashing. Does it give off signals, logs? What do I need to deploy it? And then over time, of course, once they're hooked, once they've used your thing, they've made a decision. And that decision generally lasts for a while. They're gonna expect you to be there to help them from then on for the lifetime of that application.

And by the way, they know this. And so that's part of what they're looking for. When they first make that decision to use your technology, are you gonna be around to support them over time? Okay, great. So that was, how do you speak effectively? What do you do when you've got your finger on the button? Now what do you do when you take your finger off?

You listen. Great. So. Just to be clear, remember what we're fo, what we're focusing on, we're focusing on contributors. These are the folks that want to give back to you. They want to talk to you. Consumers. They don't want to talk to you. They don't want to hear from you. They just wanna use your stuff and go home.

Right? Contributors though, the ones you're targeting, they want to talk to you. They want to talk to you on social media, either directly or as replies to whatever tweets and posts that you make. They want to comment on your YouTube videos. They want to comment on your blog post, whatever, be there for them, okay?

They want to ask questions in whatever q and a form. Maybe using Stack Overflow. Maybe you're using a Google group. Who knows? Maybe you're using Facebook discussion. They wanna make bug reports, right? They wanna complain. Let's be clear. Hey, I tried to do this and I could, it didn't work. Or it didn't work, as well as whatever they wanna complain.

Great. They wanna make feature requests, they want more, Hey, this is great, but I needed to do X, Y, and Z. Well, when are you gonna do that? You know, it is not useful at all unless I have, you know, it can be blue on Tuesdays, whatever, whatever. It's right, they want to talk to you. That's great. First and foremost, make sure you have every single one of these channels.

Open and available for them to talk to. You see many, many technologies, they just don't, they don't have these channels open. There's just no way to have these conversations with the people who are providing this technology, which right away sends anyone who's a potential con, uh, contributor away. Cause they're like, Hey, I want to talk to you in this way and you're not letting me second, make sure.

We called it, we, in the US we say operators are standing by. Make sure there's somebody listening when they say this. Whatever the channels are that you provide, that there's somebody listening and they can respond that there to talk back, doesn't it do you any good? In fact, it's worse if you have one of these channels open.

For people to ask questions or report issues or whatever, and there is nobody there to pick up the phone. That is terrible because when's the last time you as a person said, Hey, I have some feedback to give, and you heard nothing that you were excited to go and give the second piece of feedback. Now we're humans.

We got only so much time in the world. Why would we waste it talking to you if you're not gonna at least let us know that you heard from us, let alone, and, and by the way, the answers, well, we'll talk about the, the answers in a minute, but unless there is, uh, somebody there, And by the way, the somebody, it could be you.

It could be, Hey, I'm a member of the team that builds this technology. Big or small, maybe you're the only one. Maybe you replied all the questions and you, you know, it's your open source project. Maybe you're part of a team. Even better is when you get contributors to your communities that are further down the road that start answering the questions for you.

This is when you start making your community, the phrase we use is either scalable or self-sustaining. Right? This is when it can grow on its own. Right? When they start helping each other because they're so happy to be part of it. Right? Okay, so first and foremost. You have to be there to answer the, to reply to say something back.

Second, be nice. Thank them for their feedback. They're, they may be asking for the dumbest thing in the world from your point of view, but it's not dumb the way th they think about it. I'll give you an example. I was in the hotel room with my partner the other day and it had this fancy little Android tablet.

And you could turn the lights on and check in and check out and order room service. And it was pretty great. And when she said, Hey, grab the iPad so we can make a, an, uh, turn off the lights when we go to bed. Great. Now it's not an iPad. And my engineer brain in my head is like, no way. It's an Android tablet.

Android is much more open. Uh, I doubt if you could even build an application like this on an iPad because Apple wouldn't allow it. Right? Cause it took up the whole screen and there was no, you know, apple software in your face. By the way, the Android tablets are much cheaper. So you could do these in bulk.

And all I could think about was all the engineering responses to bring the iPad over here. Right? And, you know, I was visibly like, wait. She doesn't care about any of that, right? For her iPad equals tablet. That's all she meant, right? So when you are replying to these folks, meet them where they are, be nice.

Doesn't mean you can't correct them, but when you do correct them, be nice about it, right? Thank them. Be pleasant. Because if you're not, Who is gonna want to come back and contribute to community where people are mean to them, right? Be nice. And I say that to a room full of engineers because God love you.

I'm one of you. You are the meanest ones, right? Because you were gonna say, pat, are you kidding me? Let me tell you all the reasons you're dumb. No, that is not helpful. Don't do that. Now, that's not to say that every time someone's ask you for something, the answer is yes. Right? Just like Santa Claus.

Sometimes the answer is no. You're not having a pony. Right. Great. So it's okay to say no. Be nice. It's okay to ask clarifying questions. Hey, you just reported a bug. But you haven't told me how to reproduce the bug, and I can't make it reproduce in my machine, which means I can't fix it. Can you help me?

Right. Great. Also, you want to do this, all of these channels you want to do, in a way, I think I mentioned this, where everyone other contributing members of their community can jump in and say, oh yeah, I ran into this bug too. Here's a workaround. I know they're looking at it. It's actually a duplicate of this bug over here that they've had it open for a while.

But in the meantime, et cetera, right? Let other users participate as well, or let other users go, thumbs up, right? I had this problem too, or I ha I want this feature too. Because what you're doing at that point is you're crowdsourcing your product development priorities, right? The more people participating on a thread about a bug or a feature request, the more thumbs up.

The easier it is for you to know what the priorities are going to be for the next release of your thing, you will now know what's important to them and what you should consider for future releases. Let them participate many of the peoples that want to in those bugs and questions and feature requests.

And then as I said, follow up as a rule of thumb. You or somebody from your team should be the last. Person on every one of those threads, somebody says something, you should be there to say something. Okay? Now, in addition to contributors wanting to talk to you, they also want to talk for you, and this is what you want to enable.

This is when you get scalability. This is when you get a self-sustaining ecosystem, an ecosystem that can reach far beyond what you or your team. Could ever hope to do so. Um, you, as I mentioned, you wanna let them respond to questions and bugs and prs and, and feature requests. You wanna help them to promote your launches when you tweet, right, you want them to retweet.

When you post a video or a demo, you want them to go, wow, take a look at this. Um, you want them to, uh, take a look and as I mentioned, be your champion. Hey, I just saw this amazing thing. We should think about this in our org, right? Um, you want them to do blog posts or blog series. You want them to set up a YouTube channel.

You want them to set up, um, Podcast, right? You want to enable them to do all. No, no matter how many developer relations people you have on your team, and most for, to be honest, for most it's zero, right? So you're gonna have to do this yourself. You want to enable people to do that evangelism for you and encourage them, uh, to do so.

And I already mentioned about how important it is to have your individual customers telling. Their story, which is really your story right out to the world and how powerful that is. And then you want to reward that behavior because this is what gives you scale for your ecosystem. You want to call them out, Hey, look at this amazing customer did with their technology.

Hey, look at what this amazing user did with their technology. Hey, go check out this blog series that they built. Go check out this extension on our technology that they built. Go watch these YouTube videos. They're doing amazing things. Thank you for your contributions and please everyone else go take a look, right?

Let, um, encourage the behavior you want. And then, you know, for a certain of them, the ones that are technically savvy, the ones that really you'd love to hire if you could, you know, make 'em an mvp, right? Give 'em special status, give 'em early access, get early feedback from them. Make 'em an admin for your, um, web forums, right?

To make sure that they can make, make sure all the answers questions are answered. Make them, uh, maintain it on their project, right? Give them special rights and access, right? Reward them. Give them time with the teams. Send them swag. Give talks to their user groups. Whatever they want, give it to them because they are helping to scale your ecosystem.

They are working for free for you to take your ecosystem to 10 or a hundred x more people than you could ever hope to on your own.

Okay, so where are we? Uh, are we at the stage where, oh, great, I built this amazing ecosystem and now I can roll in the dough. This is not that talk. I have told you all about, you know, how to focus on your users to actually turn your engineering into an ecosystem. And there's all kinds of reasons to build an ecosystem, right?

Some of them might be money, some of them might be, Hey, this is just more fun. Some of it might be there's, um, uh, fame, right? Uh, I'll give you an example. I mean, most of the, the tooling, uh, in ecosystems I've worked with, Have been means to an end, right? When I build really great tools for the Google Cloud platform, it's not so we can make money on the tools and so that the cloud itself is so easy to use that you'll pick GCP over other cloud solutions and then of course you're paying for the cloud, right?

Tools free. You pay for the service. There's all kinds of ways to turn it into, uh, revenue direct or indirect. And there's all kinds of reasons for people to build ecosystems, but what I can give you is happy developers.