EPISODE 1030

[INTRODUCTION]

[00:00:00] JM: Over the last 15 years, there has been a massive increase in the number of new software tools. This is true with the infrastructure layer. There are more databases, more cloud providers and more open source projects, but it's also true at a higher level. There are more APIs, more project management systems and more productivity tools than ever.

ClickUp is a project management and productivity system for organizations and individuals. The goal of ClickUp is to create a system that integrates closely with other project management systems, popular SaaS tools and the Google suite of docs and spreadsheets. ClickUp was started in 2016, and despite raising zero outside capital, it has grown as rapidly as many venture-backed companies.

Zeb Evans and Yurkowski are the founders of ClickUp. They join the show to talk about their experience building the company, and we talk through their process of scaling the infrastructure as well as their philosophy of moving very, very fast. This episode has some useful strategic advice for anyone who is looking to take a product to market and iterate quickly even if that product is bootstrapped.

Full disclosure; ClickUp is a sponsor of Software Engineering Daily.

[SPONSOR MESSAGE]

[00:01:25] JM: If you are selling enterprise software, you want to be able to deliver that software to every kind of customer. Some enterprises are hosted on-prem. Some enterprises are on AWS. There might be a different cloud provider they use entirely, and you want to be able to deliver to all of these kinds of enterprises.

Gravity is a product for delivering software to any of these kinds of potential environments or data centers that your customers might want to run applications in. You can think of Gravity as something that you use to copy+paste entire production environments across clouds and data

centers. It puts a bubble of consistency around your applications so that you can write it once and deploy it anywhere. Gravity is open source so you can look into the code and understand how it works.

Gravity is trusted by leading companies, including MuleSoft, Splunk and Anaconda. You can go to gravitational.com/sedaily to try Gravity Enterprise free for 60 days. That's gravitational.com/sedaily to find out how applications can run the way that your customers expect in their preferred data center. That's gravitational.com/sedaily.

Thanks to the team behind Gravity, the company Gravitational, for being a sponsor of Software Engineering Daily.

Some enterprises are on cloud providers that you've never heard of and every cloud provider works differently.

[INTERVIEW]

[00:02:51] JM: Zeb Evans and Alex Yurkowski, welcome to Software Engineering Daily.

[00:02:54] ZE: Thanks for having us, Jeff.

[00:02:56] JM: Over the last 10 years, there's been an explosion of new software tools, and this is great because there are lots of options, but the downside is there are lots of options. How does the proliferation of new tools create problems for an organization?

[00:03:14] ZE: Yeah. I mean, great question. What generally happens if you have a fragmented ecosystem in organizations, you'll have an engineering team using one tool, marketing team using another, operations using another one, and sales obviously using their own CRM. The biggest problem you have is this fragmentation that's created and the inefficiency around that.

This was what happened at our previous company. The primary problem we were solving was putting all of the teams together in one ecosystem, and that's how ClickUp was created and that was our first vision early on.

[00:03:48] JM: There are all these tools that said a lot of them have integrations already. I can integrate Slack. I can integrate Google docs and Google Drive and all these different things. Most of them have native integrations. What's the problem with the typical integration approach?

[00:04:07] ZE: The integration approach, generally, even in native integrations, it feels more like you're halfway connecting something rather than a full connection. Even in tools like Slack where you paste a link and something's unfurled, it's great to have that link to an object. But to really be able to work inside that same tool is something entirely different.

An example with ClickUp, if you paste a link to a Google Doc, you can actually open it in an embed view and you edit the doc right from the program without leaving it. That's kind of our thesis, is saving people time through not leaving the tool.

[00:04:42] JM: You gave the example of the Google Docs embedded view, and that's a great example of pulling functionality into this singular tooling rather than having to go out to Google Docs, and I've had the experience many times of clicking a Google Docs link, opening a new tab, going into that tab. But the idea of bringing all these stuff into one tool, what are some other examples of doing that and how comprehensive can you get? Because, I mean, most of the time with all these different integrated tools, you have to go outside of the app itself.

[00:05:13] ZE: We handle a lot of the direct connections between things that you would normally have to externally go to anyway. In another example, that would be Figma. When you paste a link to Figma, you can actually open the embed from within inside of ClickUp. You can also do things like linking your Google Slides presentation, that you can edit your Google Slides inside of ClickUp without leaving that. So those are all just integrations where we can have an embedded version awesome with iframe.

But when we're speaking about something else like GitHub or GitLab, we feel that most other tools just scratch the surface on the integration. You could link a pull request or link a branch to a task. But inside of ClickUp, you can actually can go a step further and you can create new branches from tasks. You can close branches or merge pull requests from within inside the tool

and look at the difference in the code from within inside the tool as well. We're really trying to make these tools these separate tools feel like it's one integrated tool.

[00:06:15] JM: When you push all that functionality into one single tool, does that lead to any kind of bloat in the feeling of the application or in the engineering process? Does it just get to be too much?

[00:06:29] ZE: Yeah. It's something we always struggle with, for sure. From the beginning, we've always thought about the bloat and we've tried to build a tool that can be very simple for a two-person use case or extremely complex for the 2000-person enterprise. In ClickUp, many of the tools, we call them ClickUp Apps, can be turned on and off. At its core, you can have something very simple, and at the end of that cycle, you're going to have something extremely complex with all of the engineering features, scrum points and any type of custom field we could imagine. That's what we're always thinking about, for sure, and we're not 100% there yet, but we do think that it's an extremely flexible tool that doesn't necessarily have to have the bloat if you don't want it.

[00:07:13] JM: Give me a little bit more context on how the company got started. That was in 2016. What was the motivation for finding the company?

[00:07:21] ZE: It truly was a problem that I had myself at my other company that I started. I've been an entrepreneur my whole life. In my other company, we went from 2 to 100 people. We started with Basecamp and then we ended up with Jira for engineering, marketing for Asana, Trello for operations. We had Todoist for our personal reminders. We had Google Docs, Google Sheets. We had Slack. All of these tools that were supposed to make you more productive felt like they were just decreasing our productivity.

Ultimately, we built this tool as an internal tool for ourselves, and that was where we started. Solving our own use case was really just for us and then realized that this is something that other people want as well. We were hitting on a much bigger problem set.

Keep in mind, three years ago, when we first had this idea, project management software was very opinionated. They really, really wanted you to work a certain way. Asana was not flexible.

Jira certainly wasn't as flexible even as it is today and there really wasn't this cross-team collaboration that we're seeing more of a proliferation right now.

[00:08:26] JM: That parent company of ClickUp is called Mango Technologies. As I understand, you've built a number of different companies and iterated, failed, retried, failed, iterated. Give me a little bit of color into the failure and retry process of eventually getting to where you are with ClickUp.

[00:08:48] ZE: Everybody says fail fast, and that's certainly valid. What we do even at ClickUp on a day-to-day basis is iterate every single week. We have a very close feedback cycle with all of our user base through our feedback platform. We use Canny. There's a ton of them out there through Facebook groups, and still every single piece of feedback I read. Every single piece of feedback today I read. We're all just focused on shipping as fast as possible at our core so that we can fail fast and then listen to users where we fail and maybe we can hit it right on the head too and we'll know that as soon as we ship it and then we iterate really, really quickly, meaning every single week.

[00:09:30] JM: It sounds like that philosophy came before you even had a project management company.

[00:09:37] ZE: Yeah. At my previous company, I went through the learning lessons of I was trying to be perfect. I was trying to do all of the development principles and methodologies that people preach today, and ultimately that led to really slow development cycles. Great code and great test coverage, but extremely slow development cycles. That was something that if we, from early on, knew we had to be faster than our competition in this industry. It's the most competitive software market in the world. Period.

[00:10:08] JM: Project management?

[00:10:09] ZE: Productivity. Yeah, collaboration productivity. If you look on G2 crowd, those matrixes are really funny, because the next closest one, CRM, we're about three times more competitive. It's very competitive. There're new verticals every day that are being challenged with different tools and we always wanted to build a horizontal product. No vertical focus built for

everyone, and doing that is even a little bit more challenging in a competitive industry especially when you're going up against firms that have hundreds of million dollars to spend on marketing.

[00:10:40] JM: How aggressively or how willing are people to switch between the project management or productivity tools? I know new companies get started all the time, new teams get spun up within individual companies. Maybe just greenfield projects are enough of a market, but do you have to get people to switch also in order to build a big enough business?

[00:11:03] ZE: Yes and no. We find that the people that are willing to switch are generally small teams. It's under five people. There that threshold at five that kind of changes churn, really, for us. All of our churn happens from 1 to 4 person teams. It's smaller teams and it's also personal use cases that crossover, and a notion would be a good personal use case team that would use that.

There is a lot of new tools out there. I mean, there is. There're new ones every single day. For us, what we find is we aggregate your project management software. Most people that are coming to us are replacing several tools. They're usually using at least a few different project management or productivity tools. So once they come to us and find that sweet spot and start using us, then we don't see that churn happen.

[00:11:50] JM: What's been the division of labor between you two? You've iterated through a bunch of different companies and eventually got to where you are. It sounds like you two have a pretty strong synergistic partnership in how you work together. What is that partnership look like?

[00:12:08] ZE: Yeah. I'm product. I'm 100% product-focused and Alex is 100% technical-focused. It's been a great relationship, because he trust me with product and I trust him with tech. Ultimately, we're just continually moving fast based on that.

[00:12:25] JM: Nice. How does product stuff get transformed into engineering tasks? Are you doing actual designs, or who's doing the design? Because that's something that's kind of often product gets filtered through the lens of design into engineering.

[00:12:42] ZE: I am a designer by trade. I've been a designer before and I was originally with our product. Now we have four designers. So I kind of work with our product team and our design team and creating this, taking this vision and turning it into designs and then we pass it on to the technical side for Alex.

[00:12:58] AY: We'll generally have like a product meeting where we try to take the engineering side, take the product side, see what's possible in a reasonable amount of time. That's how we continue to develop quickly weekly and just those bills out.

[00:13:14] JM: The idea of moving really fast when building a project management tool, it seems like there will be some inherent risk because this is stuff that is mission-critical software for people. Have you found that to be a downside at all when you're trying to move really quickly? Is there any – I mean, obviously you get to ship features faster and you also get to fix problems faster. It's seems like there's maybe a tradeoff, but what are the risks and benefits of moving fast when you're building project management software?

[00:13:47] ZE: I mean, the risk certainly is alienating a user base that doesn't like change , right? That's always a risk. We feel that it's outweighing that by providing better value and better features for a product in general. I think our general user base would agree. We've kind of created this natural product market fit where we haven't spent any money on marketing or sales until about a month ago. It's just our user base that's marketing for us. Our user base has gotten us millions of users for free, and I think we can attribute a large portion of it to really listening to them and iterating every single week.

[00:14:29] JM: How do you get the first set of customers?

[00:14:31] ZE: Organically through SEO. SEO, I can't preach enough still. It definitely matters. Everybody says content is king, and its king and it's easy. Just start a blog post. Start a blog when you're free product before you've even created your product, because it takes time. It takes several months to do. But start hitting for those terms, and by the time you have a product, you'll have some users there. It's not as hard as everyone thinks it is.

[00:14:56] JM: I want to get to talking a little bit about the engineering, but first I think we should evaluate what the product actually does a little bit more in order to get into what the actual engineering problems are. If you could just explain in more detail why is this any different than project management tools that people have seen before? I'm sure most the audience has seen Jira, or Asana, or Basecamp. What's different with ClickUp? Give more product evaluation.

[00:15:27] ZE: Yeah. A great way to look at ClickUp is more like a no-code project management tool or productivity tool in general where you really come to us and you're building your own productivity software. It's customized for the way that you work and it also can be customized and segmented for different parts of your company.

We have these things called spaces where you can work and have an extremely complex area for engineering and you can also have one that's very simple for marketing. It's generally aggregating your project management tools and bringing them together. But in general, we're replacing other tools as well. Lots of people even replace Slack with ClickUp. We still use Slack personally. We find that direct message one-to-one approach hasn't really evaporated yet. But there're lots of people that replace Slack and they replace Google Docs and Google Sheets with ClickUp. It's best to look at us more of like a dock platform, kind of like Notion plus a database platform like Airtable plus your standard productivity tool like Asana, or Trello, or Jira.

[00:16:28] JM: That is a lot of functionality. There must be some gaps in functionality. If you're delivering like spreadsheets and like a Trello kind of board and Gantt charts and all these integrations with all these other stuff. What are the places where you feel like maybe there're some gaps or it's not as like feature-rich. It's like an Airtable or something. Are there any areas where you deliberately have made choices to just keep it simpler?

[00:16:59] ZE: Absolutely. There's 10% to 20% use case in each of those tools that you've mentioned that we view is more of the edge use case, even in some of our competition. More advanced use cases in Airtable, we certainly don't hit. More advanced use cases in something like Notion, we don't hit. But we don't feel that people are sacrificing at least using our software or missing those missing pieces.

We even look at large competitors like Monday and you see certain things like automation that we don't do as deep as they do. But from the data, it really doesn't look like those tools are used as much as you would think that they are, and it's certainly not something that our users complain about.

[SPONSOR MESSAGE]

[00:17:53] JM: DigitalOcean makes infrastructure simple. I continue to use DigitalOcean because of the low friction and attention to user experience. DigitalOcean has kept the experience simple and I can spin up a server in less than a minute and get high quality performance for a low price. For an application that needs to scale, DigitalOcean has CPU-optimized droplets, memory-optimized droplets, managed databases, managed Kubernetes and many more products. DigitalOcean has the flexibility to choose the right instance for the right workload and he could mix-and-match different configurations of CPU and RAM.

If you get stuck, DigitalOcean has thousands of high-quality tutorials, responsive Q&A forums and a customer team who treats customers respectfully. DigitalOcean lets developers focus on what they are building. Visit do.co/sedaily and receive $100 in credit over 60 days. That $100 can be put towards hosting or infrastructure and that includes managed databases, a managed Kubernetes service and more.

If you want to get started with Kubernetes, DigitalOcean is a great place to go. You can use your $100 to start building your distributed system and you can get that $100 in credit for free at do.co/sedaily.

Thank you to DigitalOcean for being a sponsor of Software Engineering Daily.

[INTERVIEW CONTINUED]

[00:19:28] JM: Again, before we get into the engineering, something that's strategically interesting about the company is the fact that you haven’t raised any money. The product looks really, really feature-rich. It looks like the kind of product that you would get after throwing a ton of venture capital into a product and just watching it grow and grow and grow and get more and

more and more features. How did you build so much functionality in such a short period of time without raising any money?

[00:19:59] ZE: Well, there are several answers so that. I mean, it certainly starts with the people. It sounds like a copout answer, but it definitely does. You can find 5X or 10X engineers and find the people that also want to work hard. If you find 5X or 10X engineers that want to work twice as hard, then you're looking at 10 to 20X engineers. When you have a group of them that work together, it's this feedback cycle that's pretty exponential in growth.

There is all the little tricks that we do too. One thing I always preach about is separating CSS role. A dedicated CSS person so that your frontend engineers are going in and coding functionality really quickly and not worrying what it looks like and then just passing is straight off to a CSS person. Another thing that we do that is somewhat unconventional is we don't write automated testing on the frontend because of things that are changing so quickly. This is what we did at my previous company where we had 80%, 90% test coverage on everything, and what we found was when you're changing something quickly, especially when you have a larger team, we're just going repeatedly in this feedback cycle of fixing our tests. It wasn't necessary breaking in functionality. It would just be fixing the test.

We find that we can move a lot faster without automated testing. The reality is you're always going to have to manually test everything anyway before you ship it, right? Automated test isn't going to catch everything. There is no downside in manually testing from our side, and those are a few things that we do to ship really, really quickly.

[00:21:26] JM: And what I like about that is there's a lot of things that you have already described that are pretty heretical to how you build software. I did this pretty long series of interviews with Facebook engineers over the last year, last couple years, like 25, 30 engineers that I interviewed, and they emphasized a lot of the same things, like get something built first, and then maybe if you really need to test it, you can write some integration tests or unit tests or something. The emphasis on 10X engineers, 5X, 10X, 20X engineers and the fact that it compounds.

Also, something that you haven't said but I find to be true is just mistakes are going to be inevitable. Whether you make tests or not, you are going to make mistakes and you would rather have a development process that affords those kinds of mistakes and can fix them quickly rather than trying to test them beforehand. You're actually going to go slower and perhaps maybe even end up with more bugs if you take the slow deliberate approach.

Now, we're not building heart rate monitor software. So maybe with heart rate monitor software, you should take a different approach. But it's also, I assume, you believe just more fun to not write unit tests.

[00:22:43] ZE: Oh! Absolutely. I mean, every everybody does, right? You'll certainly find when we're hiring some people that argue with that, but as soon as you kind of explain our reasoning behind it, they're on board. I mean, Alex can speak more about this as well, but we design things – I think people overthink the future as far as infrastructure and tech debt goes, and there's too much conversation around tech debt that happens when the reality is it just doesn't really matter until it happens.

We're more of a reactive company [inaudible 00:23:13] on that. Of course we get for early warning signs using many of the tools that you guys have on this show. but at the end of the day we're more reactive when it comes to tech debt. When things aren't scaling appropriately, then we can fix them at that point.

I don't think that there's ever been a company that is successful that hasn't been able to scale, and there's been many of them, a graveyard of them, that haven't shipped product. But at the end of the day, even companies that are notorious for having scaling troubles like Instagram, you can come in there and you hire a team and you fix it and you get it done. You may have a little bit of trouble for an intermittent period of time, but at the end of the day, we figure it out.

[00:23:50] JM: Alex, what steps do you take to ensure that there's some kind of safety net in place? Like you could do rolling backups, you could do extra aggressive rolling backups. You could do –

[00:24:03] AY: Do you mean like on the data side or more of when we're pushing features and stuff? Making sure that it's good to go?

[00:24:09] JM: I think the data side is probably more important, but obviously one integrity in the UI also. So maybe both.

[00:24:16] AY: If we're on the data side, we have point in time backups. We can do point in time resolution to any point for the past like week or so. We do that by having both a backup where we can restore to and then the log files to be able to go to any point in time. If there was an issue, we'd able to restore to a specific point in time and we'd be okay. Granted you'd lose maybe things between that point and what people are trying to do now. But it would be a very small window. As far as the UI goes, we do extensive testing before we build. We do have a team that does our testing and make sure everybody can log in. Everybody can do everything they need to do. Then we extensively test our new features. So we make sure those look good.

[00:24:58] JM: The point in time recovery system, how granular is that? Is that like you can go to a point time across the entire app or is it on a per company basis, on a per user basis?

[00:25:08] AY: It'd be for the whole database. We can go to a specific transaction point in time. So, say you made one change to delete an entire table, we could go immediately before that and have the state of the database at that point.

[00:25:23] ZE: Which has happened. Don't tell anybody.

[00:25:25] JM: So when that happened, like triaging that, fixing it, was that a matter of going back in time across the entire database and then just making a fix for one specific dataset? How do you triage an issue like that? Like if data gets corrupted for one specific company or for one specific user?

[00:25:47] ZE: If it's for one specific user, we'll try to just fix whatever they need, depending on the size of the user. If they deleted all their stuff intentionally, we can't go grab everything for them. It takes time. But if we needed to go point in time resolution for one user, we can spin up a backup DB point in time for that. Do the change to our actual production database. For the

instance where Zeb was talking about, where say we lost an entire table of data. What we can do is go to a point in time, grab that table and put that back into our production database. That way, that app still runs fine. We never have a real downtime and just continue from there, because we can grab that point in time, put it back in our production database, make any changes we really need to and continue. Does that make sense?

[00:26:37] JM: Absolutely. Are there any other safety nets you need or like – I mean, that to me is the main thing, is like data corruption or people make a – Either you ship something to production that accidentally has some blast radius of accidentally deleting data that you didn't mean to or users are confused by a UI thing and they accidently delete some data and you have to actually do some manual intervention. Other than that, a database change log basically gives you all the safety net that you need, right?

[00:27:09] AY: Yeah, for the backend, that's pretty much all we need. For instances, that would really be our fault, right? If we accidentally had a security hole or some bad code that wiped something or it made bad changes. Having a complete change log for the past week, we should catch it within the week and we should be able to roll back. As far as user base problems, the most common thing is they deleted something they didn't want to delete. The way we handle that is it's built into the product. We have a trashcan. Everything soft deletes. So it deletes in the database. There's a column that says it's deleted and its soft deletes, and then we clean those up over time. I think they stay for a month or so and then we clean them up.

If a user does make a mistake where they – Or it has a malicious user that goes in and deletes everything, we can always get those things back without having to go do a backup or anything like that. It's still there.

[00:28:00] JM: Describe the stack when you started ClickUp.

[00:28:04] AY: It's pretty simple. We just had a NodeJS backend with a Postgres database and an Angular frontend and we host everything on AWS. It's expanded a bit since then. When we wanted to scale globally, we switched and we got a multi-master Postgres database. That lives in each region that's in each continent. That helped us scale globally for reads and writes. It also will help us scale horizontally if we ever wanted more than one master in one region. But

otherwise, we just have backup recovery as far as failover. Then we added Elasticsearch for searching. What else? Redis for some caching and so on, but mostly we've been sticking to our main infrastructure that we've had.

[00:28:47] JM: Is it all the AWS versions of those services?

[00:28:51] AY: No. We try not to simply because we get more control in things. We can't use our bidirectional replication. Postgres BDR is the extension we use for Postgres to make it multi-master. With AWS, they actually recently came out with something that allows you to have a multi-master Postgres database. But I'm not sure if it still does, but for the longest time, you're not allowed to have them in separate regions. You wouldn't be able to do like geo-scaling [inaudible 00:29:18] in the US, in Europe with a master. That might've changed, but it didn't the last I checked.

[00:29:25] JM: So you just don't use a managed service. You just spin it up on AWS.

[00:29:27] AY: We just use EC2 instances. Yeah.

[00:29:30] JM: EC2 instances. Right. Okay. So you just manage your own Postgrse on EC2.

[00:29:34] AY: Yup.

[00:29:35] JM: Wow! That's surprising. Did you consider like CockroachDB or one of these transactionally-consistent things?

[00:29:42] ZE: So, we have, but – Well, especially with transactionally-consistent, if you're not eventually consistent, there's going to be some lag. There's going to be a delay. But BDR works for us.

[00:29:52] JM: I like that response. That's the honest response.

[00:29:54] AY: We were already using Postgres. I don't think we were in production at the time with Postgres, but some of our team is in Europe and they were complaining about right speed,

and then it was like, “Oh, we can have a read replica so that it's fast for you to read. But oh. You're right. Speed is still going to not be great. So we need some sort of multi-master. That’s where we just decided what's the best Postgres multi-master solution? That's when actually the AWS solution for multi-master was in beta. They wouldn't give it to us, but then we found out it's not multi-region anyway. So it didn't work for us.

[00:30:26] JM: It’s so interesting. I've talked to a lot of different companies about this globally-consistent SQL transactional database problem, and it's funny because you can't – It seems almost impossible to get the full Venn diagram intersection of all the features that you want out of any particular managed service. If you look at Spanner and Cockroach and whatever managed instances versions, Aurora I think is the AWS version, and I think YugaByte, YubaByteDB, they have a lot of this stuff covered. Oh! And there's FaunaDB. But the dust is really not settled on that category.

[00:31:08] AY: It hasn’t, and I think it also became a lot more common with all – What 5, 7 years ago or whatever, all the NoSQL database had popped up and they were all eventually consistent. People love that you could write all of them, but then people didn't – Fell off the NoSQL bandwagon and they wanted their SQL databases back, but they wanted eventually consistent. They wanted that geo-scaling and horizontal scaling of writes. I think they're slowly getting that together with SQL, but there's not great solutions.

I think we’re getting there, but with like AWS supporting Aurora with multi-master, SQL databases is a huge thing, like showing that that's the direction we want to be going. Yeah, we chose BDR, Postgres and we had our troubles with it to begin with, but we've worked with the folks who make it and we’ve gotten pretty far with it.

[00:32:04] JM: Well, what about Citus Data? Do you looked at Citus Data?

[00:32:07] AY: Yeah. That’s an interesting question, because we’ve considered using that as well for some of our larger datasets, but that will still be in a single region. It wouldn't geo-scale.

[00:32:18] JM: Really?

[00:32:19] AY: Yeah. What we've thought about doing is having a CitusDB in each region and then figuring out how to make them consistent, eventually consistent. So you could either build that yourself, or since Citus is Postgres or support Postgres, which I've talked to their team and they said, "Well, you might be able to get BDR to work with the Citus master. BDR is just a Postgres like plug-in or extension. So you might be able to get it to work, but it won't work out-of-the-box. So you're going to have to figure out how to get those to work together. But it might be something we look at one day.

[00:32:54] JM: What about the Elasticsearch and Redis stuff?

[00:32:56] AY: What about them?

[00:32:57] JM: Well, are you using managed services? AWS-managed services are Elastic-managed services or –

[00:33:03] AY: For Elasticsearch, we're using Elastic. So they're managing it. Then Redis, we are using the AWS Redis' managed services.

[00:33:12] JM: Walk me through each of those product selections.

[00:33:15] AY: They just manage the servers. It's not much.

[00:33:17] JM: No. But I mean, so you could have used – Oh! I guess you probably took Elastic off-the-shelf before AWS had Elasticsearch service.

[00:33:26] AY: No, but we just – As far as choosing the Elastic on AWS versus Elastic with Elastic, I mean, Elastic also hosts those servers on AWS.

[00:33:35] JM: Right.

[00:33:36] AY: So all you're getting there is a little additional support from Elastic.

[00:33:39] JM: Oh! I see.

[00:33:41] AY: That's about it.

[00:33:41] JM: I see, and like a better UI, right?

[00:33:43] AY: Yeah, the UI is decent, but how often do you go out in the UI for Elasticsearch? Unless we have issues, we're not.

[00:33:50] JM: So you're basically paying Elastic for support is the main thing you're getting.

[00:33:53] AY: Yeah. I mean, we haven't really needed it, but it's there if we do.

[00:33:57] JM: Right. Do you know what the cost – This is get into the weeds. But do you know what the cost differential is if you go with Elastic versus Elasticsearch service?

[00:34:06] AY: I'm not entirely sure. We definitely discussed it. It wasn't that far off.

[00:34:13] ZE: I thought it was cheaper going with Elastic directly.

[00:34:15] AY: It can't be. It might actually be, but I don't think it is just because they are hosted on AWS. They give you a whole dashboard of what in AWS they're using. I think they also give you the option to not use AWS. I think you can use Google Cloud as well.

[00:34:31] JM: Right. The reason I kind of dive into this is just I don't know if you've seen all the like licensing wars between Elastic and AWS. Do you know what I'm talking about at all?

[00:34:42] AY: No.

[00:34:42] JM: Okay, that's great. Nevermind. It's all just like Hacker News drama. So it doesn't matter. Redis, you're using Elastic hash, you said.

[00:34:50] AY: Yes.

[00:34:51] JM: Okay. Cool. You started with Angular. Did you move to React eventually?

[00:34:56] AY: No. We're still on Angular.

[00:34:57] JM: Nice! Did you consider moving to React or it just doesn't matter?

[00:35:04] AY: We had a conversation like maybe once, but no. I don't think it really matters. We just want to keep moving forward.

[00:35:10] JM: I love it. You guys are total pragmatists.

[00:35:13] ZE: There was definitely the thought of that, especially when we would ran into some challenges early on that I think people hadn't really solved yet. We've got an incredible team and they have been able to solve a lot of these scaling and performance problems that Angular is just inherently notorious for. We've been able to fix those and we're very confident in it now and we're happy with it now, for sure. But 2-1/2 years ago, there wasn't really a clear winner in the path of this. When we were deciding it was really not a clear decision as it would have been today.

[00:35:46] JM: Now, the thing with React – And I could be totally wrong about this, but the thing that I understand about the gravity that's pulling more and more people to React is a lot of it is the you can just take a lot of stuff off-the-shelf, because the ecosystem is so big, because there are so many people using React and you could take things off-the-shelf. Is that ever a shortcoming in the Angular ecosystem where you want to take some kind of component off-the-shelf and it's like, "Well, actually, we got to write it from scratch because nobody's built this thing."

[00:36:11] ZE: Believe it or not, there is a large ecosystem for Angular as well. There really is, especially when you look worldwide. An example of that is we found the most popular Angular calendar library, and I reached out to the guy that created that and then we hired him. We did the same thing with the most popular timeline library, reached out to that guy and we hired him also. We've been fortunate in finding those great people that have built those components in those libraries, but I think it's still readily available in Angular as well.

[00:36:44] JM: It's actually pretty sweet, because like if the market is moving in the direction of React, everybody is going to React. You can just snap up the Angular developers.

[00:36:51] ZE: Yeah. Low-hanging fruit for us, honestly.

[00:36:52] JM: Amazing. Is there anything significantly different between building an Electron app and building a desktop web application?

[00:36:59] AY: You're going to have like hooking in some external integrations can be really annoying in the Electron app if it's going to have a pop-up or something like that you want to integrate with, and those can typically have some things that just don't work. Push notifications to it work a little differently, things like that. But, otherwise, the core functionality? No. It's pretty much just crosses over.

We've had to do some tricks too with Electron for – We mentioned embedding items earlier. For instanc embedding a Google Doc into the Electron app is a little bit more challenging if it's not public, because you don't have those cookies that are available in the browser that the I-frame would normally be looking at. We've kind of just fool Google into thinking that we are a web browser and then we can store those cookies.

[00:37:43] JM: Nice. How popular is that feature where if I click on a doc and it opens in an iframe, or a lot of people – That sounds like pretty useful.

[00:37:51] AY: Very popular. Yeah, extremely popular. I mean, it's kind of the first platform where it's not just linking your tools. You're actually able to work from it, and that's just one of the examples on there, but that is an extremely powerful feature for people that don't want to have to go to other tools to work.

[00:38:10] JM: One particular the engineering problem I'd like discuss is collaboration. So like you have real-time connections where people are collaborating. What's the architecture for your real-time collaboration system?

[00:38:24] AY: It's an interesting issue. It's all WebSocket-based. The key there is just scaling it. The big issue is just figuring out how to scale those WebSocket messages so they're quick. Me and Zeb actually have had an argument early on about whether or not to call things, like chat or like a conversation, because the way you word it is definitely going to be the expectation of how quickly people get those messages.

We've actually been able to scale it really well though. If you do open up a couple of browsers or open up one of those chats and start typing, you'll see this person is typing message, this person is viewing, and they come fairly instantly. Then once you comment, that comments pops right up. So it really has become a chat. I was little worried in beginning about making sure it scaled well. Basically, we scale that by using queues. It's just a queue of WebSocket messages that a certain service handles and pulls those off and makes sure that it has the data it needs and sends them to who needs them.

[00:39:19] JM: Cool. There's a single queue and all the WebSocket messages just get queued up in that single queue and then pulled off. Then you have – What? Like the client is just continually scanning the queue?

[00:39:32] AY: Users have their WebSocket connection, which connects to a certain server. Their WebSocket servers in that region. Those handle subscriptions. You subscribe to what you want to listen to. Say, you open a task, you subscribe to the task, and then actions – Say, someone else does an action like a comment. That comment will get sent to our WebSocket queue. Then once it gets processed in that queue, it'll find the subscribers who need to hear about it and send it out to those WebSocket servers, which then send it down to the clients.

[00:40:03] JM: That's a Redis queue?

[00:40:04] AY: It is a Redis queue.

[00:40:05] JM: Can you tell me about a particular scalability problem that you've encountered as ClickUp has grown?

[00:40:11] AY: It's not an issue per se as far as how to scale it. It's just a matter of scaling it, because that queue, every time we get more users, that queue starts to fill more. We haven't split the queue yet. Right now, the queue is all WebSocket messages. But what we do plan on doing in the future is splitting the queue between things that need to be more instantaneous and things that can come in a few seconds. If you're on our web app and you see the notification indicator, a little dot shows up that you have a new notifications. Right now that's in that same queue, but it's okay if you get that five seconds after you need to know you have notification, right? Whereas a comment, you want to come up very quickly. So we'll probably split those out and then have two queues or three queues and slowly expand upon how quickly do we need these messages to go out. Also, we can just horizontally scale those databases if we don't want to do it and just read that queue really fast. Because we have more than one worker pulling things off that queue already. So, just scaling that infrastructure.

[00:41:13] JM: Any particular mistakes you made and how you architected ClickUp from the beginning, or has everything scaled pretty nicely?

[00:41:22] AY: We'll always have some ways where like, "Oh, maybe we should've formatted this data a little differently," and we can get to it when we do a migration or something like that. But otherwise, everything scaled pretty decently. From the start, it's been a clustered node app. So it's always built to scale horizontally. Then the database has always been, what we were talking about earlier, like a horizontally-scaled database. Those being the two big pieces on the backend, it's always pretty much been built to scale horizontally, which should solve most problems. Otherwise, yeah, just doing the normal, like de-normalization of our data to make sure that it scales.

[00:41:59] JM: This node app, is it entirely monolithic or do you have a bunch of services? What does that look like?

[00:42:07] AY: We started completely monolithic. The codebase actually is completely monolithic, but it's all in one repo, but it splits out into services now. We started as monoliths just because that's way faster to just move. If you start with microservices, you're going to be slower and that's harder to scale as well and the infrastructure is more complicated and whatnot. We

start it with monolith. We split out things for imports, and notifications, and they slowly become their own services.

We originally just had it one because it was faster to develop. We've slowly had more services come off that for integrations, like our Google Calendar integration is pretty heavy. That has its own service, things like that.

[SPONSOR MESSAGE]

[00:42:52] JM: ClickUp is a productivity app to replace all your other tools. ClickUp has all the functionality that you need out of a productivity tool. Project management, wikis, spreadsheets, email, chat, it integrates with your existing tools like Google Docs, Slack and GitHub and gives you development workflows together with your productivity apps. ClickUp is trusted by companies like Google, Airbnb, Uber and Tesla. All of them have made the switch to ClickUp. ClickUp is free to use with unlimited users and unlimited tasks. Try it today by going to clickup.com/sedaily. That's clickup.com/sedaily.

Thank you to ClickUp for being a sponsor of Software Engineering Daily.

[INTERVIEW CONTINUED]

[00:43:43] JM: Tell me more about the integrations. So you have a ton of integrations and you have to have some systematic way of updating them, upgrading them. Do you have particular integration engineers? How do you manage the boatload of integrations?

[00:43:59] AY: Yeah. We have a few people specifically working on integrations, but typically they don't really need updates unless we make a change in our product that we want to do across all integrations, or generally people will contact you if you have to update an integration because they're deprecating an API or something. They don't always, which is great. Yeah, we go just manage those. It hasn't been a problem yet.

[00:44:25] ZE: Google Calendar was a particularly tricky one. One that you would think would be relatively easy to develop, but ended up being much more complicated than you would think.

[00:44:33] AY: Google Calendar's web hooks are terrible. They don't give you enough information. Trying to sync two-way on a Google Calendar is pretty tricky when changes are made. There is always this build versus buy scenario, this kind of decision that you make. When we built this, there really wasn't another option. Now, Nihilist built an entire business out of this.

[00:44:53] JM: Nihilist, yeah. That's what they do.

[00:44:55] AY: Yeah. We realized, "Oh! Well, we probably could've waited several months and then use Nihilist for this." But at the end of the day, we figured it out. It was certainly very challenging, but it works for all now.

[00:45:08] JM: Nihilist, that business is entirely around calendar and email integrations, right?

[00:45:13] AY: Correct.

[00:45:14] JM: How big do you think that market is?

[00:45:16] AY: It's bigger than you would think, for sure, because it goes through this horizontal path of tools from project management, to CRM, to personal calendar applications. As the economy scales for these tools, there is more demand, and they're a micro-transactional tool. You're paying for usage. I think there's a big demand for it. I think that they will have a ton of competition, probably not, but at the end of the day we actually are using Nihilist for our new email integrations that are coming out soon.

[00:45:49] JM: Any other APIs or infrastructure products that have been surprisingly useful?

[00:45:57] ZE: We don't use that many infrastructure as a service products. We certainly consider them especially for WebSockets early on, but to build what we were building in a very flexible way, we decided to do it ourselves for the most part.

[00:46:12] AY: And that decision is going to actually payoff a lot when we do. We don't currently offer on-prem, but when we do, that will be of much easier transition because we're not relying on any service that we can't put on-prem because it's just our own services.

[00:46:28] JM: Right. What would you have used otherwise? Like PubNub or –

[00:46:32] AY: Yeah, Twilio, PubNub. Handling our chats was one conversation that we had around this, and I think that building versus buying is certainly – We would generally advocate for more of the buying approach early on and then building it your selves when you need to. But from the beginning, we were more of the building type of people. We built our own billing system and we built our own CRM.

[00:46:58] JM: Why would you do that?

[00:47:00] ZE: Yeah. The customer always comes first. It's our number one core value, and there was a lot of these billing challenges, billing kind of cases that we had to handle where at the time even the Stripes of the world were pretty inflexible when it came to billing. We wanted to roll up people's transactions on a 24-hour periods so that you don't get billed several times if you add a few different users. At the time, that wasn't possible. So we ended up building our own billing system for that.

[00:47:29] JM: How long did that take?

[00:47:31] AY: The original one really wasn't so bad. It just takes a lot of maintenance as far as, "Oh! Let's add promo codes that can do this," or we now want to build in this scenario. Those are the like –Especially as you start to get enterprise contracts that – You could probably speak more to this, have some interesting – Like, "Oh! You pay this much for this many seeds, but then your price, your pricing tiers," and so that stuff starts to get a little complicated.

[00:47:53] JM: I still don't understand. I need to know all these without raising any money. Is it just like heard insanity that's causing people to raise so much money and not even thinking about like what's a trajectory where we can bootstrap or – I mean, it's just so hard to do the bootstrapping route. I don't know. What are the characteristics that have made you successful?

[00:48:19] ZE: I mean, look, the reality is that when you look at today versus 10 years ago, development costs have actually gone down and pretty much every way with new libraries, infrastructure as a service, and certainly the availability of cloud technology has made the development costs go down. If you look 10 years ago, the average seed round was about a million. Today, it's $6 million. Where is all of that extra money going? It's not necessarily going to development costs. It's going to marketing. It's going to buying your market and sales.

We've always preached our company natural product market fit. Meaning, let your users tell you what they want. Let you users promote your product for you. Whereas you have this artificial and product market fit, which is generally created by buying customers, by buying users, and that's what has become proliferated, really, in fundraising. It's your SaaS metrics. Your typical SaaS metrics, right? Quick ratios, your CAC, LTV and the ratios between those, that's what you're optimizing for. When you can hit those, you raise more money. Where do you spend that money? It's usually not on development costs or development in general, or even product, frankly. It's more for buying more customers and growing as fast as possible. For us, it's more been about product focus, and that's where all the money has gone. We're highly profitable today. So we continue reinvesting that money into product rather than marketing and sales.

[00:49:42] JM: How has it impacted hiring? I mean, you don't have equity to – You could give away equity. I guess you can give a dividend. Do you do anything like that or you just like pay people good salaries?

[00:49:53] ZE: We still give equity, for sure, and it's not to say that we'll never have an exit, because we will, but we see more the path to an IPO. Doing that more of a very nontraditional way. It's still there and it's, again, not to say that we'll never raise funds, but when we do, it will be a sustainable way and more so spending this money on marketing spend, whereas everything else becomes cash flow positive or everything else remains cash flow positive still. So that when you have a market downturn, like we're pretty much experiencing now and funding dries up, we don't have to lay people off. We have a very sustainable business where we can continue growing and you remove that stress from your life of saying, "Hey, where's our next check coming from? Which metrics do we have to optimize for?" Instead let's just focus on more

fun. It's what do we care about doing. What do we want to build next? What are some cooler things that we can do?

[00:50:47] JM: Do you guys mostly hire in-person or you higher overseas? What's your process of finding engineers?

[00:50:55] ZE: Ideally in-person when we can. We don't limit ourselves, because some of the best engineers are overseas and we have some of them, but our new headquarters is in San Diego, and that's pretty much where we say, “Hey, you kind of got to be in the office at this point.”

[00:51:11] JM: Yeah, but San Diego is probably a pretty good place to build a company these days. Like, not incredible competition for – What are your main competition for hiring? Was is it like Google? Is it Google [inaudible 00:51:22] has an office there?

[00:51:24] ZE: There isn't much competition for hiring, to be honest with you. I don't know if I want to tell everybody this, because San Diego is an awesome place. We were in San Francisco for two years, and I wanted to come out here because I've been an entrepreneur my whole life and we've always love technology and I thought this was the place to be. To some extent, it still is, of course. But the reality is, especially as a bootstrap startup, you can't compete with salaries here, and there's competition on every doorstep.

In San Diego, we still got the best of both worlds and we got the California quality of life, but we also have many people there that were working, either they were telecommuting to San Francisco or actually commuting to San Francisco, or they’re working for, let's say, not as exciting startups. There're a lot of older technologies, that Qualcomm. That's more of a San Diego company. So when you're competing with that, believe it or not, it's actually easier to compete with that because those people are looking for a change. They’re looking for something more exciting, a higher growth potential, and it's been a phenomenal hiring in San Diego.

[00:52:26] JM: What’s the build process for the app? It's so big, right? You get so much stuff going on in there.

[00:52:31] AY: Our backend is pretty simple. It's a Node.JS app that deploys to Elastic Beanstalk. So we just have a Travis build that zips it up, puts it up to Elastic Beanstalk and it does have to go to all of those regions and all those services. That takes about 20 minutes to deploy and check health, because it does a rolling deployment. But the backend is pretty fairly simple. Our frontend after we get everything tested and we're good to go, the actual build process, it makes the Angular app. That does take a while. It is pretty big. So it takes about – I don't know, maybe 15 minutes, and then it deploys it. It deploys up to S3 and then just deploy it. I mean, that invalidates the cache, the CDN, and then it should be good to go.

[00:53:14] JM: As you said, the manual testing. When you ship a new build, the manual testing takes place in like a staging environment and then eventually you just manually promote it from staging to production.

[00:53:26] AY: Yeah. We have a three-step. We have a staging, and that's where actually we even do our own work. So that's kind of where we dog food our product and new features. Once it's ready there, so some breaks happen there, because we deploy every Friday with new features. We push every day or so with all the bug fixes and updates, but our actual big feature builds is every Friday. So it'll go through manual testing on staging. With the new things, it will go to our stage I environment, which is a production database, but the new frontend. It'll get tested there to make sure everything is backwards compatible. Everything works, there're no regressions. Then from there, we'll push it to production.

[00:54:05] JM: And this is more a product question. But as you said, there are some users who switch from Slack to ClickUp. How would you compare division for ClickUp to Slack in the limit?

[00:54:19] ZE: It's more so putting all of your work in one place. Slack is simply a connector and that ecosystem, whereas we really want to be that single source of truth for all of your work, that platform for your work. We have a ton of tools that you can either leave on the table or you can take with you and use them every day. Some of our users come for just the tasks, but many of them come for our OKR software only. Then lots of people, they come for one thing and then they stay for many of these things.

If OKR software, doc software, spreadsheets, tasks is our core product. We still have chat and then all of the integrations and the tools that you have. We find that people usually – Again, come for one thing, and they kind of stay for everything else. Now, are we trying to replace Slack? No. That's not certainly in in our roadmap to do so or even product vision to do so. It's, again, more creating this fundamental workplace for all of your work collaboration to happen and to extend from.

[00:55:21] JM: You mentioned do you think of ClickUp as a no-code tool?

[00:55:24] ZE: Yeah, no-code productivity tool.

[00:55:27] JM: And can you tell me what you think this ecosystem of no-code stuff is going to evolve into? It's pretty hard to tell at this point.

[00:55:36] ZE: It's very hard to tell, and I think that it's a little bit overblown to some extent, but the reality is, what our vision, is that software that has a certain functionality. An intended functionality is generally going to be can be better than software that doesn't have that functionality. Ours, we don't just say no-code tool in general. It's a no-code productivity tool. What that means is that there are productivity tools, like tasks that can be highly customizable, highly flexible, which is where the no-code parts comes in. We remove the opinion from everything.

Our perspective is just building this toolbox of things that you can use your tools to customize yourself and not necessarily building something that's completely flexible and starts with a blank canvas so to speak. The blank canvas thing would be something like Airtable or Coda, whereas ours is more of here's a foundation of productivity, and then you can kind of code or no-code your tools on top of that.

[00:56:41] JM: How heavily have you seen people using no-code? I don't know how much you talk to your companies, the companies that ClickUp, but like how they're using these newer tools, the Retools and the WebFlows and stuff. I'm just kind of trying to track what's going on in that space and how heavily those are being adapted by companies.

[00:57:02] ZE: There's a huge buzz around it. We don't necessarily see the adaption there, especially when you're thinking about something like Coda or Airtable replacing. It's more of a different market. Sometimes you hear people comparing Coda or Airtable to a ClickUp, to a productivity tool, a project management tool, when really it's a far stretch to really replace project management with those tools. In some respects, it's too flexible. It works for a couple person team to go build out a project management tool on something like coda. But for a large organization, it's just never going to work. It's too flexible to use a dock for that. That's, again, goes back to our perspective of building this toolbox for people that does have some intended functionality and then allowing them to kind of customize it themselves.

[00:57:47] JM: What does it take for people to switch from Google Sheets or Google Docs to Notion or Airtable or whatever? Is it novelty and curiosity, and then eventually they find something in that product that they like? What causes people to actually switch to those things?

[00:58:04] ZE: I think it's the user experience folks. It's the people that can see an order of magnitude and better user experience, and that's what those tools provide, for sure. I wouldn't say that it's that much different functionality. It's relatively similar functionality with a much better and vastly different user experience that you can build other things on top of. Whereas, some tools, like Google Docs and Google Sheets, are somewhat limited in the way that you can kind of view your information. Something like Airtable gives you an extremely flexible way of viewing it.

[00:58:35] JM: How do engineering workflows change when people are using ClickUp?

[00:58:42] ZE: Generally, you have engineering workflows separate from everything else in your company. From a high-level, it's putting your engineering with everything else. It's putting it with your marketing, your sales, your product folks, usually people that are switching to ClickUp. It's their whole company rather than just a certain team, or at the bottoms up approach where it's just their engineering team or just their marketing team. Then they start working together. That is a fundamental switch in how people are working. You can have complex tickets like Jira inside of ClickUp. Yet you can also have very simple things for sales and for marketing. There is an inherent kind of efficiency that's involved with that.

On the other side of things, even when you're using Jira, certainly owned by Atlassian, obviously, and Bitbucket is as well, but the reality is those integrations are not as deep as you would think that they are, and our integration is just as deep as Jira's is with Bitbucket. There isn't much of a bottleneck to you changing to ClickUp, really, other than everybody knows Jira. You know how to configure it. You have an admin for it. But at the end of the day, we don't find people having problems with switching. It's more so the efficiency involved in having everybody in one place.

[00:59:53] JM: What are your goals for the next few years with the company?

[00:59:56] ZE: We only look a year forward. Everything is just too impossible to predict when you start strategizing more than a year or so in the future. We have our kind of quantitative goals, of course. But at the end of the day, our mission is saving people time. It's making people more productive. That's all we care about as a team. It's moving that, we'll work forward and building our product in a way that we are making people more productive and we are saving people time. But some quantitative goals we have is getting to 4 million users this year. That's certainly the topline focus that we all care about, and we'll be at 150 team members by the end of the year.

[01:00:30] JM: Wow! What's been the hardest part of scaling the workforce?

[01:00:34] ZE: I mean, everybody says hiring, and hiring is certainly a bottleneck. For, again, our number one core value is providing exceptional customer service. I've had so many horrible customer experiences in my lifetime, and from being an entrepreneur since I was born, I had always cared about the customer first. Finding people that really care about customers for scaling customer service has been challenging. It was very challenging in San Francisco. In San Diego, we've have been able to scale that up tremendously and

[01:01:04] JM: Because you try to keep them on-prem.

[01:01:05] ZE: Yup. Exactly. You're trying to keep everybody on-prem, and then scaling that out for not normal business hours. We learned the international product. We need to keep customer

service going 24 hours a day. I would say that's actually the hardest part of kind of scaling up the people side of things.

[01:01:21] JM: Last question, what would each of you be building if you aren't building ClickUp?

[01:01:26] ZE: We've got a thousand ideas. But the funny answer is we were actually intending to build a Craigslist competitor before ClickUp. That was our plan, was building a new Craigslist where you could pay in-app and basically remove the sketchiness from Craigslist. So you'd have review profiles.

[01:01:44] JM: That's a pretty good idea.

[01:01:45] ZE: Yeah. Well, we got into it a little bit and we saw that there was some competition that was doing this. Actually, there's Close5, a competitor that never really kind of took off and offer up, which is still around. For some reason, still hasn't done the in-app payments part of it. But we saw Close5, and eBay was an investor. We inspected some of their source code and found PayPal attributes that were kind of hidden in there. We also saw that they're during the Super Bowl ad that year. That was the decision made, we're like, "All right, let's just take this internal tool, ClickUp, that we were developing at the time to be able to create the Craigslist competitor. That's when we took ClickUp and launched it as a product.

[01:02:23] JM: Same thing for you? You've been building Craigslist?

[01:02:26] AY: Yeah. Just anything that like makes people just more productive, happier, easier lives. I mean, that's what we're doing now and just something else that continues to do that.

[01:02:37] ZE: It's all about your own frustrations at the end of the day. I think too many people try to start a company hypothesizing about where is the product market fit? Where is this market? But it's much easier to solve your own problems. That's what I would recommend all day, is do something that you're solving your own use case, your own problem. You're going to have much more ideas when you get started into it and you're only scratching the surface on it.

[01:02:58] JM: Guys, thanks for coming on the show. It's been great talking.

[01:03:00] ZE: Thanks a lot, Jeff.

[END OF INTERVIEW]

[01:03:10] JM: When I'm building a new product, G2i is the company that I call on to help me find a developer who can build the first version of my product. G2i is a hiring platform run by engineers that matches you with React, React Native, GraphQL and mobile engineers who you can trust. Whether you are a new company building your first product, like me, or an established company that wants additional engineering help, G2i has the talent that you need to accomplish your goals.

Go to softwareengineeringdaily.com/g2i to learn more about what G2i has to offer. We've also done several shows with the people who run G2i, Gabe Greenberg, and the rest of his team. These are engineers who know about the React ecosystem, about the mobile ecosystem, about GraphQL, React Native. They know their stuff and they run a great organization.

In my personal experience, G2i has linked me up with experienced engineers that can fit my budget, and the G2i staff are friendly and easy to work with. They know how product development works. They can help you find the perfect engineer for your stack, and you can go to softwareengineeringdaily.com/g2i to learn more about G2i.

Thank you to G2i for being a great supporter of Software Engineering Daily both as listeners and also as people who have contributed code that have helped me out in my projects. So if you want to get some additional help for your engineering projects, go to softwareengineeringdaily.com/g2i.

[END]