SACRA

EXPERT INTERVIEW

UPDATED
08/21/2023

Zachary Kirby, co-founder of Vessel, on building the Vercel for integrations

TEAM

Jan-Erik Asplund
Co-Founder
jan@sacra.com

SACRA

Published on **Aug 21st, 2023**

Zachary Kirby, co-founder of Vessel, on building the Vercel for integrations

By **Jan-Erik Asplund**

Background

Zachary Kirby is co-founder of Vessel, the developer-first, native integration platform for GTM tools.

Interview

You're a B2B SaaS company in 2023. How do you build integrations and how has that process changed over the last 5 years? What will it look like over the next 5 years?

Really understanding the answer to that requires understanding two other pieces of information about B2B SaaS, in general.

The first piece is there's a lot more B2B SaaS out there. The amount has exponentially grown year over year.

There's a company called Productiv that tracks essentially the average number of B2B SaaS apps that companies use and they came up with a report two years ago. 254 was the average number, which is already absurd.

They came up with another report and the number was 310. There's just a hell of a lot more SaaS out there being used, being produced, being worked on.

The fundamentals of having your application connect to another application and push data hasn't changed. Integrations have been around for almost as long as B2B SaaS has been, but the requirements for building integrations have become extremely difficult to manage with this proliferation of SaaS. Developers have a lot more demands on them to go and build integrations and those integrations are more complex to build.

The second piece is because of that competition, it's no longer sufficient to just have a Zapier integration or some sort of lightweight integration that only pushes your contacts from your CRM to your sales engagement platform. Companies are competing on the quality of the integration that they have.

There used to be this world—and I'm going to talk a lot more about this later because this is our whole thesis—where internal integrations were treated the same way as external integrations. What people are starting to see, and why these integration platforms are taking off right now, is that that's no longer the case.

Now, I meet companies all the time where they say that they won a deal because their Zendesk or Salesforce integration was better. You're seeing customer-facing integrations get treated in the same genre as any other product features, and so it's no longer sufficient to just have that same thing. Those two together have combined to raise the priority of customer-facing integrations. I think internal integrations—stuff that you just make so that your marketing team can publish things from their Twitter account to their Medium account or something like that—largely haven't seen too much innovation, but customer-facing integrations have heated up tremendously because of those two facts.

What does all of this mean?

The world nowadays, If you're a B2B SaaS company, building in-house is looking way, way less appealing. We're starting to see the shift that we saw in authentication a while back. People used to build authentication in-house. Then, there came these demands: "I want to be able to authenticate to your app through Google. I want to be able to use a magic link."

Now, by and large, people don't ever even consider building authentication themselves in-house. They have OAuth, PropelAuth, and all kinds of different providers—there are hundreds of them out there.

We're now seeing the same thing happen in customer-facing integrations,where this whole idea of should we build in-house or should we outsource is becoming a lot more about which outsourced tools we should use, not should we build it ourselves.

In the past, it would have been, “Our customers use Salesforce and HubSpot. That's it. Let's just build those integrations, ourselves.” That’s no longer the case. You're starting to see these providers take a much bigger stage because essentially that's what they're built for—helping you build these at scale.

Are integrations table stakes or do companies have a competitive advantage in how they're implemented and/or in how they enable partnerships?

It really depends on the type of company and especially the founder and their experience working with integrations before.

We've had certain companies come to us before they built a single integration because they know they're a sales tool and that they're going to need a Salesforce hub.

This founder, or this company, or this integration leader has built a Salesforce integration before and they're just thinking, "Hell, no. I'm not building this again myself. This was such a horrible experience. I will use anything that will help me with this." So they start looking right away and integrate with us because that's essentially what we provide.

However, we've also seen on the whole, most companies wait to be asked for these integrations. It's less of a push and more

of a pull. They'll say they have these integrations on their website because they know it gets them into the conversations, but you won't believe how many companies out there say they integrate with 100 tools and they have 0 to 1 of those actually built out.
They wait—which I think is an intelligent strategy—for a company to come to them and say, "Hey! I want to use your tool. I want to improve my rep's close rates, but we use Close or Pipedrive or whatever, and so I need that integration in order to use you." That's when they decide, "Okay. It's time to build that."

A lot of companies still start building one of these in-house, but once they've done that two or three or four times, they start to take a step back and say, "Hey! Wait a second. This is taking up a fifth of my week just fixing bugs in these integrations or just adding new integrations." They're not an integration provider. That isn't their main use case. That's when they begin to search online, ask around, and find out about integration platforms.

There are a lot of different approaches to building in-house/outsourcing/sort of a hybrid of the two. How do customers decide between buying iPaaS, embedded iPaaS, universal APIs and native integrations? How do we think about and make sense of these products?

The space is difficult to navigate right now. There are a lot of players in the space and many different ways to build integrations. You touched on a bunch of them. Let me just break it down for you the way that we think about it.

At the very top it's just building integrations. We split that off into two categories: customer-facing and internal integrations.

The reason that we split those off is because those are two very different types of integrations that you need to build. You don't really have to deal as much with OAuth or worry about your customer's rate limits or just general concerns that you have with customer-facing integration when you do something internally.

I won't touch on the internal integration space because that's not what we do, and that's a whole other beast. Those are your Workato, Zapier, Tray, etc. Internal integrations were what got

the integration space started, so all these companies mentioned above were built initially with internal use cases in mind or around them. What that means is these are no-code workflows tools. These are a lot of, let me drag and drop a couple of building blocks together that transform data. This is built for a product team. This is built for a non-technical user. Then, they realize, “Hey! This is great for internal use cases. We can also slap an embedded authentication modal into your product and make this work for external facing integrations.” There's a bunch of companies in that space and so, those internal integrations start becoming external integration use cases. We broadly categorize those as workflow solutions and that's the status quo for a while.

These are still the biggest companies out there—Zapier is 10 times bigger than any of the other companies I'm about to mention. Same for Workato, Tray, and Paragon and all these other companies that you've heard of.

There’s this new wave or genre of thinking that hinges on bringing all of this into your code. It’s the idea that it doesn’t really make sense to put all this stuff built for product engineers into a bunch of no-code building blocks.

If I'm competing with these other companies, I need my engineers working on how deep my integrations are. I need people in the code hacking away, figuring out how we can be 10 times better than our competitors. You can't do that with a workflow solution. You need that to be in your code. You need fine-grained control. We largely categorize that genre of company as a native integrations platform.

I'm biased here because we’re building a ”native integration platform” but I would put all of Vessel, Nango, Superglue, Merge, etc. into that category. What we share is that instead of using a workflow solution, you are actually writing this inside of your code. You're making either an API call to an ETL database or you're making an API call directly to the actual downstream integration platform, or they're doing a transformation on the data etc.

That's what's starting to gain traction and people are waking up to the idea of, "I can't be competitive using a workflow solution. I have to use this in my code and I need that for control, which means I need to use a native integration platform."

You touched on another thing—the universal API. I don't even categorize that as a type of integration platform or as an iPaaS in general.

To me, universal API versus ETL is a feature. That's not really difficult work. Anyone can go and figure out how to map these fields. There's difficulty in getting it right and it's definitely more of an art than a science in some regards. But by and large, that's not the difficult part.

The difficult part is the data extraction. I get frustrated sometimes when we get compared to a company like Finch, Rutter, and even Merge HRIS solution. That's a very different type of company than what we're building, which is about product integrations.

All of the integrations we work with for the most part don't require web scraping. A lot of them have APIs and so we can build a lot more of them really quickly. We focus entirely on our ETL pipeline so we have, I would consider a world-class ETL pipeline dedicated to specifically the types of integrations that we provide.

We can pull Salesforce data faster than anyone else on the market because we custom-built pipelines specifically for pulling Salesforce data and shooting that right into your database. Or we'll hook the database and you can hit our database with some dedicated APIs versus you going and looking at a Plaid or a Finch.

Some of these walled garden data sources are much harder to extract data from. There's a lot more value that they're retrieving—and not the breadth of the integrations—in just the fact that they can even offer you this banking data to begin with at all. When you start to talk about native integrations platforms, it's less about the feature set. It's like “Do you have a Universal API? Do you do ETL?” That's not necessarily the difficult part. I think you have to look really closely at what is the integration set that they provide and how are they actually extracting that data.

That's where it starts to differentiate and where I throw these verticalized solutions, like Plaid, Finch, Rutter into one category and Nango, Superglue, us into another category of product-based integrations.

What is Vessel in short, who are your customers and why do they choose Vessel? Are there any early signs of product-market fit that you can share?

For Vessel, we describe ourselves as a native, a developer-first native integrations platform.

We focus very heavily on selling to developers, integration development teams, and technical founders. Our pitch here is everything that I've just mentioned before. It's not sufficient anymore to just have a simple Salesforce integration. You're competing with everyone else building these deep integrations and we want to give you the tooling to do that.

We view ourselves as the Vercel or AWS of this space—that infrastructure for building world-class integrations. We provide unified APIs, but we view that as a tool in your tool belt. We also provide an action API which is a quirkless API wrapper over the downstream integrations APIs that provide typing. It provides a better interface over the documentation so you don't have to actually read Salesforce documentation if you want to use something that's specific to Salesforce. Those two in tandem allow you to get any data that you might need. When the unified API falls apart, you can use the action API if you need something specific and still want a first-class experience.

We also provide a bunch of features that you would also need around how you move that data. We have webhooks. We have a cache where we actually ETL that data on our side and keep it up to date using polling and webhooks and all kinds of techniques and and host that data in the database so you can go and just hit it from our API without worrying about hosting it yourself. We also have filtering on top of that—basically everything you would need to ever have on a data lake for that integration data.

The final piece is that we do the managed ETL. We actually have an option to ETL that data right into your database. We've spent all of our time building out these infrastructure pieces and we view that much more as the value add that we provide rather than, "Okay, we have this unified API, that unified API, whatever."

If you look at our customer set, we started with GTM integration, so we had a lot of GTM-focused customers—up

and coming startups in the GTM vertical. But because we focused on building that infrastructure and not on very specific verticalized solutions to every single type of integration, it was very easy for us to add new integrations that just kind of slot into these pipelines and use any of these features.
Just recently we went from about 20 to close to 100 integrations in the span of about two weeks because we built this general framework and all we needed to do was set up the authentication and hook in some endpoints. There's a little bit of AI sprinkled in there to help build that out. You can just use that for any of the different types of infrastructure.

What did the process of building integrations look like before Vessel? What does it look like now?

For a lot of customers, it depends on what stage they're at, what their cycle is, where they're at with their integrations.

Let's just take a generic average customer that hasn't built an integration yet. They suddenly get some demand. They get a customer that’s come to them asking for a HubSpot integration, and they don’t want to build it themselves.

They come to us for a specific integration, and they start with a unified API because that allows them to create the most generic workflow. Over time what we see is that their customers are usually the ones that push them for different demands.

A customer will come to them and say, "Hey! I don't want you to touch this piece of data. I don't want you to use deals in your product even though your product allows me to, I don't know, filter over all my deals or have deal insights or something like that." What they do then is use Vessel to customize the scopes for that specific customer or not ETL data for that specific customer.

Another customer will come to them and say, "Hey! I want to use Salesforce cases, which is something that doesn't really get unified." They might use the action API for this one-off thing for Salesforce cases.

Taking a step back, we started as a unified CRM API. All of these insights come from experience and seeing how people used our unified API. Over time there would just be debt that

would be built up because the unified API falls apart in specific places. Even a time span of six months sees them all of a sudden making pass-through requests. Then, they have to go and read the Salesforce documentation. Now they start to step back and question, "Why am I even using Vessel? I've become a Salesforce expert myself. Why am I paying for this?"
With our product over time, they just continue to use either the action API or the unified API. We have everything they could possibly need. Essentially it gives them a lot more comfort in knowing that no matter what their customer comes to ask, they feel equipped to actually go and handle it and not have to build out anything themselves in-house.

We often hear from developers that universal APIs sound great in theory, but (1) they only support a narrow set of use cases and (2) they tend to serve the lowest common denominator. You end up having to write custom code to create something differentiated and/or to handle corner cases, and then you're back to where you started. How does Vessel address this developer pushback with its model?

To touch on the backlash to begin with, in our experience, a well-designed universal API will actually probably work in more use cases than you're thinking.

There are a lot of companies out there that need to do one or two very specific things and then, a universal API will actually work fine. You won't believe how many companies there are that only want to read in contacts from the downstream system or just read in the deals because they need that data. The hard part is the unification. It's like, "Well, I don't want to build out an ETL pipeline. I don't want to build out, figure out the authentication, all the ops, everything else that goes with that."

But you're exactly right. When the unified API does fall apart, it falls apart badly—to the point where you have to go and basically understand how Salesforce works yourself and start figuring out, "Okay. What are all the exact requests and everything like that?" It actually becomes really dangerous because you don't know exactly the implementation details.

For something like Vessel that's open source, you will. But for something like the unified APIs that other competitors out there build, you won't know how exactly that unification works. You

don't know, “I need this specific thing with a lead. Is the unified API actually already returning that? What is this?” I don't really know.
It can become dangerous. You can end up showing incorrect data to the user. You can end up building a wrong use case. This was something we discovered firsthand when we started with the unified CRM API and that was happening to us. That's what prompted us to take a step back and be, "Okay. When the unified API falls apart, it falls apart badly. How can we provide a different way to do this?"

We actually turned to workflow solutions. They actually have a really great use case and like an elegant way around it. It just happens to be a no-code UI that it's the way that it's exposed. They have these things called actions, and these basically are just building blocks that you can use that are specific to Salesforce, HubSpot. When I go to pull in a HubSpot action, I don't need to go to the HubSpot API docs, it's all there for me. I see a dropdown of all my options.

You can figure it out just from looking at the actual workflow solution. We were like, "Hey! That's actually a decent idea. The problem is that it's a no-code UI meant for a product person. Let's just take that and turn that into something that's meant for a developer."

That's what our action API is. It’s essentially that you can go to our docs and you look at these options for Salesforce. We'll kind of massage this to work with our unified API because it’s built for unified API. We'll be able to point out certain use cases that might be better suited to the unified API or overlap, or anything like that. That creates this really nice elegant balance where the majority of your customers or the vast majority of your use cases all look the same. It's all this unified API. Then, it's very specific spots. You go deep and you dive into Salesforce. You dive into HubSpot for those specific customers. It reduces the complexity of the code tremendously if you had had to build that out for each use case.

What we've seen is a lot of customers kind of use that as an upsell, right? It's like you want this specific thing for Salesforce, for a specific thing. With HubSpot, it's like you can use the unified API, which was easy for us to build out these cases supported by the unified API. But if you want the more specific thing, then that's an upsell or that's an upcharge. You

don't have to do it for every single customer. It's just when you need it, it's there.

Vessel started out focused on go-to-market (GTM) integrations and now is horizontal. What did you learn about the opportunity that resulted in this change?

That's our position on the matter. Of course, we're going to see over time that that plays out because there are players trying to thread the needle and do both. I get frustrated when people make comparisons to Vessel as the Plaid for CRMs, which is admittedly how we sold ourselves at the beginning because it's easy to conceptualize. But really, if you start to go into “Where is the data extraction? What is the actual value that we're delivering?”, the unified API tends to actually kind of be the easy part.

You can take any junior developer, give them three different API docs and say, "Hey! Find a way to match all of these up." Where is the difficulty? Why are these providers giving so much value? It comes down to how that data is actually extracted. That's what goes into really fully deeply understanding these APIs and doing that at scale.

If we take a company like Plaid, what are the kinds of products that they would build? They're very different from the kinds of products that we would build. Yes, we are exposing an API on top of ultimately some data in a different system, but they're much more concerned about circumventing—working with these providers to circumvent issues around web scraping.

A good example here would be, Plaid had issues extracting data at one point, and so they had a closet full of android phones they were using to log into. When a customer would enter their credentials, they would log into these Android phones and then, do the scraping from there because they couldn't do the scraping on the desktop.

That's never a problem we would have, and we would never have to dedicate time to doing that because we work with these open APIs. What does that mean? That means that we focus all of our time on building these infrastructure layers that I was talking about. We spend all of our time figuring out how to actually scale the breadth of integrations. It’d be really, really difficult for a company like Plaid or a company building HRIS to

go from 20 to 80 integrations in two weeks. But it's been pretty easy for us to do this.
A lot of companies get grouped under this universal umbrella, but really that's just the GTM skin on top. The actual unit you need to look at is the actual technology, the actual way that this data is being processed, the way that this data is extracted because ultimately that tells you what these companies are spending all of their time on.

For example, you can look at certain companies that seem like they have this similar feature set to us who will give their customers six-hour refresh windows for this data that they're sending to their data lake via ETL from a tool like Salesforce or HubSpot. That doesn't make any sense for those kinds of tools—HubSpot, for example, has rolling window rate limits where you could be refreshing data continuously every hour faster for a smaller account. Why would they have six-hour SLAs?

It makes sense because if you look at the other types of data that they support and the other types of integrations, some of them involve web scraping. Some involve companies with much more rigorous rate limits because of the data that they tend to house. So you end up building a pipeline or infrastructure that isn't really optimal for the integration set that you have. That's where I kind of make this distinction in my mind, like unified API, that's not a mode. There's no mode there. Everyone can go and build a unified API. The mode is around how you actually pull this data and process it.

How does the proliferation of integrations companies like Vessel affect CDP companies like Segment, ETL companies like Fivetran and reverse ETL companies like Census and Hightouch, a.k.a. third parties wholly dedicated to helping data move between apps?

The answer to this really clarifies our unique perspective on the market and our feelings about the integration space in general. You've mentioned a couple of companies there—Fivetran, Hightouch, Airbyte.

Let's talk about Airbyte, who are a fellow YC company. They've started to push an embedded solution. There are other companies out there like trigger.dev that started as workflows that are starting to push embedded solutions. This goes back to my entire point in my entire thesis, which is that the

technology itself isn't the differentiator. It's not about you supporting an ETL solution. Building this infrastructure and getting it right is really tricky. But our entire perspective is around where and how these actual integrations are being used, which is, is it a customer-facing integration or is it an internal facing integration?
To us, that matters so much more than the way that data from that downstream system is actually extracted, the type of integration this is going to be and the type of data. Is this something you have to do a lot of web scraping for? Is it a walled garden or is this just product-facing? Is it just a product integration?

That's where the actual type of company definition is coming from. It's completely within our wheelhouse to support something that looks like Fivetran and Airbyte where we're ETL'ing that data directly to you because that's a use case you might have for a customer-facing Salesforce integration.

The only requirement for us is, this is a use case you have for a customer-facing product integration. In a way, it is a different perspective on how to actually build an integration platform and less so about this focus on the technology. “Is it a unified API company? Is it an ETL company?”

No. It's a developer-first native integration platform for customer-facing product integrations. That's the definition of the company. It’s not about doing ETL or having a unified API or an actions API.

We do whatever you need us to do because we're building that infrastructure. The integrations are the easy part for us to add on. In a way, I've heard other companies use the term for rippling, for example, compound startup. I think, in a way, we sort of fall under that. Maybe that will help you conceptualize exactly what we're doing and who we are.

In an easier way to say compound startup, I don't necessarily view it that way because we're not intending to go into internal integrations. But in a way we are compounding all of this technology where it's like a company might have an Airbyte solution. They might have a workload solution. They might have a unified API. But we have all of that already for you. So if you need product integrations where the buck ends here—

we have customer-facing product integrations— there's no other solutions out there that you need to take on ever. **Would you say then you would never build what you call a workflow integrations platform because even if it's customer-facing, it would not be developer-first? It would be more for a no-code end user who's not a developer— would that be accurate?**

The only thing that's certain in life is death and taxes, so I won't ever say "never", but in terms of what our actual strategy is and where we view ourselves in the market, it would take a tremendous change for us to actually consider building an internal integrations platform or anything that faces internal integrations.

If you're building something that needs to compete with your competitors' integrations, it needs to be maintained and built by your developers. It needs to provide powerful APIs. So an internal workflow solution is something we could build, but if we did step into that, we're talking a decade from now, or however long for now.

There's so much more work to be done in the customer-facing product integration space that we're going to be busy with this for a while.

There's been a recent boom in universal API companies. In addition to Merge, Finch and Vessel, we have Rutter, Alloy, Recall and a number of others. How do you explain the boom? Why is it an attractive market to be in? How do you see it shaking out?

One of the big differentiators between the two types of unified APIs I was talking about is that one is about unlocking HRIS and finance data, which wasn't accessible before. That creates a different type of company where you can go to a lot of companies that have probably never built this out and say, "Look at this new thing that I have for you."

We're playing a totally different game over here. A lot of companies have already built out a lot of the integrations that we support at least before we had launched that new set of integrations. So, our sell is just completely different. We come to these companies and say,

"Hey! Look, you have an integration team with five people, but you're not an integrations company. Integrations are just a table stakes thing that you need. What if we just replace half of that team? What if we supercharge it? What if we help you make your integrations a more competitive advantage?”

That kind of thing ends up really changing the type of company that you can go after.

Ultimately, the only reason that I think now is the time to build a native integrations platform besides the rise in SaaS is also this changing perspective on what it actually meant to build integrations.

When we started a year ago, we were competing much more heavily for a team that's like, "Should I build this? Should I use you guys? I see the value of this, but also I can just build this myself." That's not something you really have to deal with if you're Plaid, unless it's someone who's really dedicated to building these banking integrations. But that was a really common problem for us.

What's really interesting is over the course of a year, we've noticed that building in-house has become way less often. I don't have specific metrics on this, it's just pure anecdotal, but I'm sure if you ask a lot of other players in the space, they'll say the same thing. Building in-house is just way less on the table, way less common. It's not brought up as much. Now we're just competing more often with companies that are providing the same value as we are.

They're an iPaaS solution or a customer-facing iPaaS solution. I think that those two things work in tandem. I don't have any exact scientific evidence about correlation and causation, but the rise in the number of SaaS applications. There's more competition, people are being more competitive now and that means you need more competitive features, including your integrations. Integrations are just more important now because of the number of tools that you have. It's like, "I need this data to flow really seamlessly between if I'm ever going to consider using your product."

The fact that there are a lot of companies now solving this problem to the point where it feels a lot like auth did a couple years ago where it's like, "Well, I'm not even going to consider

building auth in-house. I'm just going to find which solution is right for me."
It feels the same way right now for customer-facing integrations platforms. All of those points are coming together all at once right now to signal this wave of new integration platforms. I really truly believe that maybe two, three years from now, maybe even sooner, there's going to be a lot less companies out there where you see raw API calls being made from within their code base to a Salesforce unless it's really important to them. You're going to see a Vessel in there and that's the tool that they'd use the same way that you would for an auth provider.

Function calling enables GPT to make API calls using natural language. How will AI's ability to use APIs change the nature of demand for APIs and products like Vessel?

In terms of products like Vessel, it will only increase the demand. If I was building an internal-facing integrations platform, I would probably be a little worried about OpenAI's ability. The reason for that is, when you're executing a workflow internally, you have a lot more leeway around errors. You have the subject matter experts around what is actually supposed to happen there, in your company, in the room ready to go and debug, and figure things out.

For customer-facing integrations, you need that to work a hundred percent of the time or else your customer is going to be like, “What the heck is going on here? I'm just going to go to someone else with a better integration.” So I don't see anything like that ever being used for external customer-facing integrations.

However, it is interesting because like you said, if you want your data to be accessible within OpenAI, within some of these bots for OpenAI, you need to have a public-facing API. So, I totally see that there's a real chance this is going to drive more demand for companies to start having external-facing APIs. That's already naturally happened.

The number of companies themselves that just support external-facing APIs, and there's a whole other conversation we could have on this new genre of a company coming up called developer like infrastructure experience or so. It’s essentially around this idea of companies like Speakeasy and

Firm that are helping you build APIs, just public APIs in general.
So the counterpoint to the rise in integrations is the rise in public-facing APIs, which is fundamentally how customer-facing integrations are built. How integrations are built is through these APIs.

So I totally see that there can be a rise in demand there, and a lot of companies have started to focus more on their API for this very reason where it's like, "I need access to that raw data programmatically because finally I can do something useful with it for the first time instead of having an internal team that has to go and wrangle that and build something with it."

If everything goes right with Vessel over the next five years, what does it become? How has the world changed?

The dream here would be to be in the same genre as those other developer infrastructure products that you always hear of.

No one ever considers building AWS unless they need their own server racks or something like that. People are going to use a cloud platform provider for their hosting solution. People are going to use OAuth or some variant of that for their authentication. A lot of people use Vercel for hosting.

The long-term vision for Vessel is that when you go to build your first customer-facing integration, it's not about, "Well, let me go pull up the Salesforce docs and figure out how to do that." It's just as simple as using Vessel.

In the span of a couple minutes, you have it all set up. It’s not like, "Oh! Let's take two weeks to go and build a Salesforce integration." It just comes out of the box when you set up the application to begin with. It’s free to sign up—you can go ahead and just play around with the API in a couple minutes.

The other thing that is really interesting about this space is, we are a public-facing API. We work with public-facing APIs, but at the end of the day, we're also a public-facing API ourselves. One thing that's really interesting working with these native integrations platforms is, they're some of the least pleasant APIs to work with. Because we work with so many APIs, we

just know which are nice to work with, which are bad to work with.
If you're worried about the complexity of the API or setting things up, you'd be very pleasantly surprised how quick it is to set up something like Vessel. It takes about an hour at most, on average, sometimes faster, and the APIs have been built over the past couple of years to really be developer-heavy.

Disclaimers