SACRA

EXPERT INTERVIEW

UPDATED
10/02/2024

Ayan Barua, CEO of Ampersand, on going upmarket with deep native product integrations

TEAM

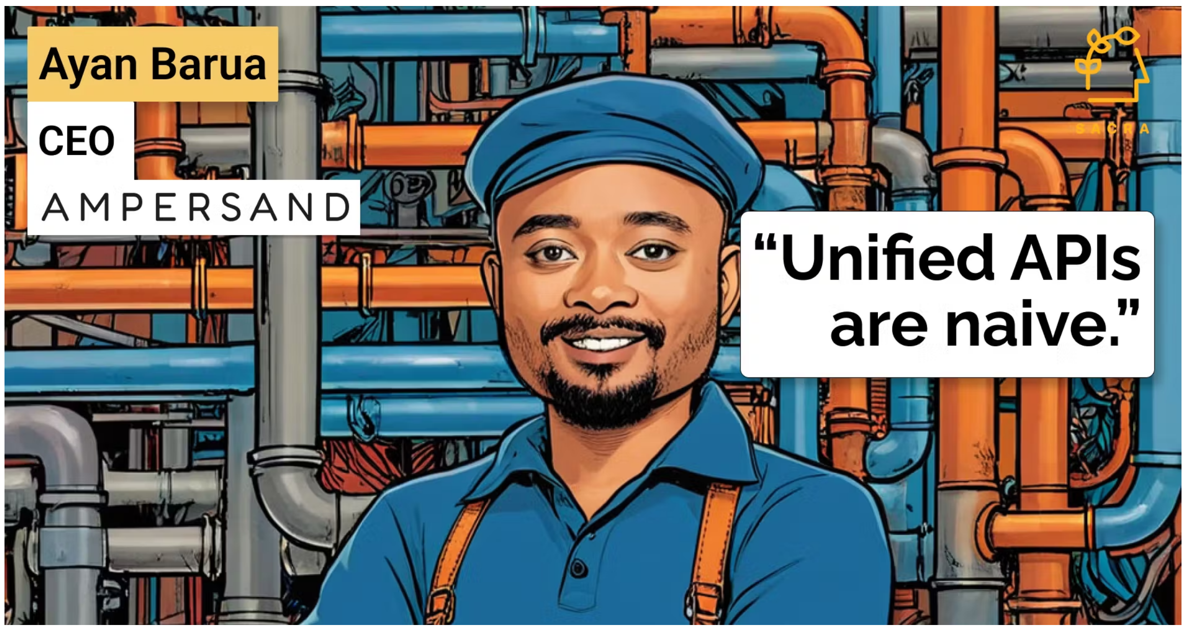
Jan-Erik Asplund
Co-Founder
jan@sacra.com

DISCLAIMERS

Published on **Oct 02nd, 2024**

Ayan Barua, CEO of Ampersand, on going upmarket with deep native product integrations

By **Jan-Erik Asplund**



Background

After speaking to Jeremy Zhang at Finch (General Catalyst, $62M raised), Peter Zhou at Rutter (A16Z, $28.5M raised), and Sara Du at Alloy Automation (A16Z, $25M raised) about unified APIs, we reached out to Ayan Barua, co-founder and CEO at Ampersand (Matrix, $4.7M raised), who is coming at the problem of SaaS integrations with a focus on depth rather than breadth.

Key points from our conversation via Sacra AI:

- **Before the rise of SaaS and open APIs, circa 2003-2006, MuleSoft built integration middleware that could ingest data from any protocol (SOAP, CORBA, RPC, XMPP, etc) and convert it into XML, a standardized format readable by any app, to allow for deep, app-by-app configuration—an architecture that would define the iPaaS category.** "Oracle [would] come in and says, 'Here's a box, do something with it. If you want data in and out, that's not my problem. You figure it out.’. . . MuleSoft came in in a big way where they said, ‘Hey, I can connect disparate data sources between your systems and centralize them so that you can act on it.’ This is the premise of iPaaS, and there are many companies in this space. It's making a lot of money, but it's still a previous generation construct where moving data around was not the vendor's responsibility, it was the buyer's responsibility.”

- **300 public APIs in 2006 exploded to 20,000 by 2019, shifting the responsibility for integration from buyer to vendor and creating the opportunity for “unified APIs” like Merge (founded in 2020, $74.5M raised) and Finch (founded in 2020, $62M raised) that consolidate the most common fields from third-party APIs into a single model, enabling vendors to integrate once and connect to every platform they need.** “Along came unified APIs, which promised to get to faster integration very quickly with common models. Unified APIs actually scale much better in the HRIS category because that's where you're trying to standardize everything… Unified APIs pushed the boundary because now the product and engineering teams are using a category of products built for them, with SDKs and logs.”
- **Positioning against the breadth-first appeal of unified APIs, new native product integration infra companies like Ampersand (Matrix, $4.7M seed) enable vendors to build deep custom integrations with the handful of highest revenue generating, enterprise-tier, “fat head” systems of record like Salesforce and HubSpot.** “We are more like Firebase than we are like dbt. . . [Enterprise integrations] might involve syncing 50 million contacts, writing to many custom objects, or sending large amounts of data into their customer's CRM. . . Developers need observability baked into this… at the granularity of an object, at the granularity of a field. What data came in? Did it conform with the data you were expecting? If something broke, was it a permission error or an API error? Is there a problem with rate limits?... You can't just pull 20,000,000 contacts in 10 minutes. Salesforce will gate you.”

Interview

Before starting Ampersand, you were VP of Engineering at G2, and you got there via the acquisition of your startup Siftery. Can you talk about building Siftery, your

experience at G2, and what inspired you to start Ampersand?

I think the meta here is that SaaS really exploded in 2010-2011. I built my first company, Credii, as a way to select better software. It became more of a content play, so it didn't really go anywhere. My second company, Siftery, was basically a data play instead of a content play. We gathered technology stacks for 15 million companies. We crawled the internet, gathered interesting datasets from different sources, and also had a large community of people giving us data and letting us know what we got right or wrong. It was a very interesting product in 2016-2017 when we launched.

The interesting part is that this type of data can make its way into the hands of both buyers and sellers, because who's using what is very valuable information. You can sell or buy based on that. We chose the buy side and didn't choose the sell side because everybody was doing the sell side. We thought we could use this data to build a product, which was essentially spend management on steroids. The buyers of this product were finance and procurement, and it became like an internal introspection product focused on what we're spending and why.

We created that category of SaaS spend management. There are a bunch of companies in the space right now. One of the key insights was that the data needed to do good spend and SaaS management was distributed across multiple sources - in ERP, banking systems, and different SaaS applications. Looking back, I think 40-50% of our roadmap was basically integrations - getting the data together and then shipping it back into some of these systems of record.

When I was looking around, I realized that 40-50% of entire SaaS is probably integrations. Every SaaS is like a workflow intelligence system of action, and now you can replace that whole thing with agents, which do the same thing but are a little more autonomous.

We looked at all types of platforms - unified APIs, embedded iPaaS, iPaaS. The problem with the mid-market and enterprise is that at that layer, every company is a unique snowflake. Every customer has customized their ERPs or CRMs extensively, and we were having challenges with tenancy management. If you deconstruct the problem, it's like: we just

built a Salesforce integration, great, engineering says it's done, we move on. But every few weeks, there'd be an extension we needed to build or a customer with a unique use case. Product would say we don't support this, so does it go back to the roadmap? Is it a big company? These conversations would happen, and no engineer wakes up in the morning thinking they'll do the 70th version of Salesforce.
We looked at all the products, and kind of all of them are shallow because they're catering to the lowest common denominator. This was the whole premise of unified APIs when they came out - you just need 10 fields from these systems, we consolidated it into a common model. The challenge with that premise in the enterprise and mid-market is that companies want to govern their own schema and have their own model. They want others to map to that model, so their schema is precious and needs to extend.

Our positioning has been: if you want to go very deep natively, whatever you can do on top of the API, you can do with Ampersand, cutting down from 6 months to 6 days. Then it's maintenance on autopilot and steroids. Everybody can build, but nobody wants to maintain in this world, and this maintenance is often tedious. A lot of times, the maintenance work isn't even your problem because you may have lost access to an object in your customer's tenant - you just don't have the permissions. Observability around that is going to save the day for you because you can tell your customer that it's not even your problem, it's something happening in their environment.

I saw this problem of enterprise use cases not being supported by any vendor, and every company is spending millions of dollars on this. So the opportunity is there to make a dent in that space.

Could you sum up how you describe Ampersand? Who are your customers today? Why did they choose Ampersand? And do you have any signs of initial product-market fit?

Ampersand is a developer platform designed for deep product integrations, especially in the go-to-market stack. The go-to-market stack is represented by the CRM at its center, with UX enhancements built on top of it. These enhancements include sales engagement and sales enablement tools. You can think of the CRM as the sun, with various categories of tools orbiting

around it like planets. People are constantly moving data between the CRM, engagement tools, enablement tools, and sometimes data warehouses.
Ampersand helps developers of go-to-market stack companies facilitate this bidirectional data movement. We can support anything that can be done on top of the API because we mirror the API and provide a declarative interface. While we're not exactly a unified API, we are unifying the patterns of interaction. We have read, write, and subscribe actions, similar to how HTTP has GET, PUT, and POST methods. We're abstracting at that layer.

Companies using our platform today include some impressive businesses selling to the mid-market and enterprise segments. There have been cases where companies have built integrations themselves but are now selling to enterprise clients with unique use cases. These might involve syncing 50 million contacts, writing to many custom objects, or sending large amounts of data into their customer's CRM. In such cases, they're using Ampersand to orchestrate Salesforce's bulk APIs, handling volume infrastructure use cases like updating 300,000 records daily. In all these scenarios, the company is dealing with a mid-market or enterprise customer.

How has the process of writing an individual integration changed over the past 5 years? If we zoom in a little, what did that process look like before, and how has that pattern changed now that you're using Ampersand?

Yes, I think they're individual integrations. Let's take the example of Salesforce. You have a base integration that you set up, and as you gain more customers on that integration, you're tweaking it with customer-specific changes in code. You're writing custom code for each customer. This is a common problem across many companies that I've seen.

In the Ampersand world, you are moving that customization that you have to deal with per customer into configuration, and that allows you to scale across your customer base much better. The implication of this is that you are having to do less customization per customer. It also cuts down your implementation time - you go from about 6 months to maybe a few weeks. A lot of that 6 months is project management in enterprise and people-related issues, but purely from an object-to-field mapping perspective - what to bring in, what to

avoid, what permissions to sync, what not to do - all of that is now productized.
So, Ampersand's customer's customer is coming into that product and configuring the integrations themselves, because they are the ones who know all the nitty-gritty details. This is usually a RevOps person or a salesperson who's configuring. All of that is productized now, so in the Ampersand world, the whole adoption curve is really cut down. But you're not losing out on the flexibility of a native integration; you're not losing out on the customizability. You keep all of that while cutting down the implementation time.

Then, nobody really does - and I know you have touched upon the DevOps aspect - you're not going to build sophisticated tooling just to see how logs are syncing, what's breaking, what's not. That's almost like you're building a Datadog for that sync infrastructure. Even with a lot of budget, I don't think you're going to build some sync infrastructure there. So we do that as well. The build, the onboarding, and the maintenance are hypercharged, and it feels like we still have all the control we needed to build this.

Without belaboring the point too much, could you give a concrete example to illustrate this concept?

Let's say a company is selling to GE. It's a sales analytics tool, and GE is interested because it increases their sales velocity. So when you're selling to GE, they have 190 tenants. Every country has a Salesforce org, and within those orgs, there are also regions. Compensation management is happening around that quota, and all of the sales performance is happening on that.

If I'm a SaaS company selling to GE, it's like a 7-figure deal. Now I have to understand which location, which geography, which tenant to read from, and what is the frequency of that read. That might change in a different arc in a different location. So there are 170 tenants, and I'm now writing or configuring per tenant a bunch of rules. That's a lot of custom work.

In the Ampersand world, it's actually just a YAML file that you're writing the configuration around. So you went from a 6-month implementation to maybe a 6-day initial deploy. That's an enabler that doesn't exist today.

And that means deep right then. Your engineering team is telling our product and engineering team, "Hey, we've got this. We can do it."

We are not monkey-patching. This is not a hack job - we can support it. Our go-to-market strategy feels much more enabled now that we can close this seven-figure deal.

I feel the clarity that example brings is the distinction you were drawing between integrations in the enterprise world versus integrations in the SMB world. It also highlights the need for depth and customization versus a more superficial type of integration.

This phenomenon manifests itself not only in CRM but also in ERP and EDI. If you look at SAP, for example, you'll find that not only do you have custom tenants, but you also have custom code per tenant that you have to reverse engineer. It's not all configuration. As you move into the enterprise space, this level of customization becomes the name of the game. I think that Ampersand is trying to unlock this potential. Our initial customer base consists of companies that are growing fast. They're trying to sell to customers, particularly in the enterprise or mid-market sectors, and we serve as an enterprise enabler or unlock solution for them.

The real IPO business here is essentially enterprises dealing with enterprises and having that data exchange on steroids. This is because there's a lot of productivity loss that happens today due to all of that inefficiency.

Can you talk about your focus on CRM and go-to-market strategy? Is that where you think the 100% focus will be over the next 3 to 5 years? Or do you intend to start with CRM and go-to-market, then move vertically as some of these companies have done?

I think the way we view this is that ERP might be a little more complex than CRM. The customization in ERP is far more extensive. From my personal integration experience, which is primarily with NetSuite and Salesforce, I would say 60% of my bandwidth went to NetSuite and 40% to Salesforce. Salesforce is an easier product to start with, but it has a multi-tenancy problem as I mentioned to you earlier. You have 70-71 orgs, so how do you move data around these orgs?

We're starting there. We want to prove that when good engineering teams who can build for this multi-tenancy look at Ampersand, they see that they can build this over two years and then maintain it for the rest of their lifetime. That represents a lot of accrual cost and productivity loss. Integration isn't particularly aspirational, meaning it doesn't progress your career as much as if you work on more core LLM or AI product stuff.

Instead of building for two years and managing for the next ten, our customers look at Ampersand and think, "This is like AWS. This is our integration infrastructure - read, write, subscribe. They have the right architecture, the right parts are built the right way. We can trust it to cut our two years into two months, and then we don't have to maintain it." That's the story we need to repeat vertical by vertical.

We are starting with the go-to-market stack. It's interesting because there are companies doing sales compensation management, like Spiff, which was actually acquired by Salesforce. These companies are marrying CRM data with performance management data, which is in NetSuite. Our customers have already asked whether we can extend Ampersand's architecture into ERP as well as the data stack, because some of this data is in Snowflake. One of our customers is thinking through how they would approach data connectors and data sharing.

I think our customers are bringing us into different parts of the stack, but we want to do this methodically. We aim to prove that we can do this with CRM, then with ERP, and go upmarket into areas where there's a lot of legacy software. You'd be surprised to know how much legacy software is trying to make its way into Salesforce itself. There are companies using legacy CRMs considering Salesforce after denying it for the last 10 years. That's also an opportunity where we can help them migrate.

Another important aspect is communication. Nylas is not doing a great job with email APIs; they are costly and seem like previous generation technology. Many of our customers have asked us, "Can you also give us a communication abstraction on top of Gmail and Calendar?"

For example, if I'm building a sales engagement tool, CRM and communication are at the core of it. We're looking at multiple systems of record: communication is one, customer data is another, customer analytics is a third, and ERP is a fourth. Our hypothesis is that SaaS businesses or agent-based businesses are at the center of many systems of record, and they're stitching this data together. There is a lot of compounded value as you connect different systems, so we want to unlock as many as possible for these companies.

Why is devops so crucial to be part of the platform and a core component of what you're offering?

I think that when you're building a SaaS business, you're not thinking about whether you should read or write; you're just thinking about the customer problem. Let's say the customer problem is that they are not doing the best follow-up that they possibly can, which means the CRM may not be updated based on certain signals, and that triggers a bunch of other things. For many companies, it's essentially reading from the CRM, reading from sales engagement data, and then based on certain signals, writing back into this custom object for the customer.

They are thinking customer problem first; they are not thinking, "Oh, I'll use ETL." That's a solution part. The product or business is not thinking about this. Our team is for the product stack, not for the analytics stack. There are really good products like Fivetran for ETL, and for reverse ETL, you have Hightouch and Census. Airbyte is there too. Those are for the analytics stack, where you're basically getting your data from your CRM to your warehouse. At the end of the day, it is essentially analytical workloads being run on top of your warehouse. That is the use case that these ETL tools help with. Reverse ETL is like, "Okay, you got this analytics now you have to go back to your own Salesforce and update these things."

Our case is a product use case. That's why we think read, write, and subscribe – subscribe being real-time events emanating from these systems. It's for the product use case. You can use an Airbyte for the product use case, but it's not meant to be. Essentially, that's the centrality around it: we are for transactional workloads. In fact, Matrix led our seed at Ampersand, and they also led the Series A of Fivetran. When I was speaking with the partnership there, I asked how they view us, and they said, "Oh, that's for analytical workload. You guys are for transactional workload. This is more Firebase." My co-founder is from Firebase; she built extensions there. Firebase founder is an investor. So we are more like Firebase than we are like dbt. There are different ICPs essentially. Hence, because we are like Firebase, developers need observability baked into this, and granular observability. When we say maintenance on steroids or maintenance on autopilot, and you can build for 2 years, you will probably manage for 10 years given that your company's successful. So that maintenance pitch basically means that the logs we are gathering for this synchronization happening between your infrastructure and all your customer tenants are extremely granular.

In the unified API world, you're just logging those 10 fields. You're just scratching the surface around the logs because it talks about a common data model. If you're building a native integration, and we help you with that, it's full-featured. It's at the granularity of an object, at the granularity of a field. What data came in? Did it conform with the data you were expecting? If something broke, was it a permission error or an API error? Is there a problem with rate limits?

Engineering has to build a bunch of these rate-limiting infrastructures because it's not free-form. You can't just pull 20,000,000 contacts in 10 minutes. Salesforce will gate you. The interesting part is a lot of this quota is per tenant. You may be trying to write to your customer's tenant, but you're being rate limited. You might think, "Oh, maybe I was the one who blew through our customer's quota." Actually, you weren't. Gong is being used by the same tenant, and you had 1,000,000 API calls. Gong has already eaten up 900K of those. So there is a noisy neighbor problem.

Now your engineering team has to figure out – two days later, we're like, "Oh, we are not even the guilty party." But we are getting rate limited by Salesforce because it's a shared API rate limit. We can save on that effort. We are enabling our customers' engineering team to say, "Here's the quota, here is what we ended up calling, and I don't think we're the guilty party here."

That granular level completely changes the way you maintain this customer-facing infrastructure. I think that is only possible when you actually go very deep. Otherwise, you'll not have those logs. That is also the centrality of the platform.

Can you talk about the pricing component, specifically pricing based on data?

The pricing based on data is a caveat, as we are early in the process. As you know from working with early-stage startups, we've heard repeatedly in the market, whether from big companies or smaller startups, that connection-based pricing is not aligned with their needs. I'm talking about unified APIs here – there are iPaaS, embedded iPaaS, and unified APIs trying to solve some of these problems. For unified APIs, connection-based pricing means you create a sync connection and pay for that, which doesn't really scale with product usage or the value our customers are creating for their customers.

There could be a case where a bunch of people sign up on their platform and connect their ads or CRMs. They're not on a contract and are just trying out the product. They may never become a valid customer, but you are now paying $25-30,000 just to support that fee for your customer. This issue came up repeatedly.

The problem isn't the $25-30,000 or $50-80,000 – enterprise companies have enough money. The issue is that unified API products are shallow, yet I'm paying top-tier dollars for what is essentially just an authentication and token management service that holds user tokens and provides a pass-through. Many end teams have said, "We tried Merge. It was a good pitch and made sense, but one year later, we're paying $50,000 for a pass-through API with authentication features. It doesn't make sense."

I think much of the visceral feedback we got on pricing was because the product isn't deep enough, yet we are paying per connection. Our thinking is that we won't charge per connection if you're driving a lot of usage through us. If you're doing something special with that customer, it means you're deriving value. So we're trying to align pricing with data delivered, and to date, we've received a lot of good feedback. We'll continue to learn from the market.

No product integration company has tried data delivery as a pricing metric, so it's also a business model innovation that we are trying to iterate on.

Can you talk about Ampersand as a revenue generator versus Ampersand as a painkiller or vitamin when it comes to making developers' lives easier? Which do you think is more important: Ampersand helping you generate revenue or Ampersand making your developers happier?

I think both of them are very interconnected, but it is a revenue generator if you are trying to sell to an energy company. That might be your first six-figure contract or even seven-figure contract because energy companies have a lot of money. What Ampersand gives you is this almost infinite scale, so you're not worried about whether you can handle it when they onboard. We're saying that we're like AWS - that's why the name Ampersand. We are extensions to your infrastructure and your team so that you can unlock the enterprise and mid-market much quicker, better, and faster.

I think there's a lot of promise around AI. I was at Dreamforce and SaaStr, and for the first time in many years, I met a bunch of buyers who said that for the first time in the last 20 years, they think they can move in weeks, not months. Enterprises typically think in terms of 3-5 quarters, planning to do things in Q4 and so on. But now they have so much pressure to adopt AI that they're trying these new tools because they don't want to miss the opportunity. Enterprise adoption expectations are changing, and speed is coming from the buyer. Ampersand is well-positioned to help companies who are trying to break into the enterprise market and get on the adoption curve much quicker.

Usually, when a company is thinking about this strategically, they're not thinking about developer experience as much, but it will be an engineering initiative. Engineering is asking whether they should build this themselves or rely on something like Ampersand. When we're in some of these conversations, it's purely a build versus buy decision. I don't see some of the competitors you've mentioned in certain deals because companies are looking for depth rather than shallow solutions. There's a lot of build versus buy journey that we have to go through.

I think, between you and me, the real challenge is gaining engineering trust - can we convince them that we'll be sticking around for a long time and doing this the right way? That's the trust we need to gain. It's not necessarily about the competition that's out there. In a way, this is very competitive, but it's almost like we don't see some of these companies at all in some of the conversations we're having.

We threw out a bunch of categories: iPaaS, embedded iPaaS, unified API, universal API—I don't know if you think of those as the same or different—native integrations, and then we put aside ETL, reverse ETLs, and maybe CDP. Within this iPaaS, embedded, and all this side of it, how do you see this market, and where do you put Ampersand in this map?

I think at the core of all of this is fragmentation, and if you layer personalization on top of that fragmentation, those are the two key elements. iPaaS is a previous generation construct. It's making a lot of money, but it's still a previous generation construct where moving data around was not the vendor's responsibility, it was the buyer's responsibility. Oracle comes in and says, "Here's a box, do something with it. If you want data in and out, that's not my problem. You figure it out."

MuleSoft came in in a big way where they said, "Hey, I can connect disparate data sources between your systems and centralize them so that you can act on it." This is the premise of iPaaS, and there are many companies in this space. There are some billion-dollar companies; it's quite significant. If you look at Gartner's Magic Quadrant for iPaaS, it's a heated one.

Categories like CRM, ERP, and iPaaS are among the first 5 or 6 that come to mind. It's a combination of categories because, at its core, it's about fragmentation and personalization. It's almost human DNA to want to have your own corner of the world or the internet and customize it to your personality.

Now, what has happened is that products have come out, and the responsibility for shifting the data is moving from the buyer to the vendor. For example, Outreach can't say, "You figure out how to connect your Salesforce; it's not my problem." Outreach has to build a great native Salesforce integration. Similarly, Ramp cannot say, "Connecting your NetSuite to my enterprise account is not my problem." Ramp will offer a great NetSuite

integration. This shift has been happening over the last 10 years.
Many of these iPaaS vendors thought they had a product use case and an expansion play. They wrapped around something, either OEM or a thin layer of an API, to incorporate that iPaaS into their product. However, engineering has strong opinions about it, questioning why there's an iframe and viewing it as an alien section of the product. They believe they should have a good product experience if they're trying to build it.

Along came unified APIs, which promised to get to faster integration very quickly with common models. Unified APIs actually scale much better in the HRIS category because that's where you're trying to standardize everything. There's a lot of fragmentation, but as you go from category to category, I can't say that there's one API that connects to all SaaS – that's naive in my opinion.

Unified APIs pushed the boundary because now the product and engineering teams are using a category of products built for them, with SDKs and logs. An engineer looks at a unified API and says, "This is built for us." Whether it gets the job done is a secondary question, but there's a form factor change from embedded iPaaS to unified API.

Now, all the buyers are aware that unified APIs exist. They're asking, "Should I build for 2 years and manage for 10 years?" They're trying to buy or evaluate, and that's where Ampersand comes in. As a buyer, I tried to evaluate a bunch of these tools and realized they weren't going deep enough, so I still had to build it myself and motivate my engineering team to build it.

I think you'll see a group of products like Ampersand which are squarely built for the application stack, for the product use case, for the B2B SaaS AI use case – essentially moving data around between a SaaS vendor and all its customers. That's why I call it native integration infrastructure for the product use case.

When I was talking to LV, I said I didn't want to create a category for the sake of it, but it makes sense to have integration infrastructure because we think of this as an infrastructure company. When I was trying to buy this product, I had no idea what iPaaS was because iPaaS sells to IT.

Engineers aren't typically aware of iPaaS until they get into this world and realize it's a thing.
I think a fourth category should exist, which is thinking of this as an infrastructure problem – configuration and scale for all your customer environments.

Why is this an attractive market, and why have you chosen not only to build a company but also to invest the next 10-plus years of your life into it?

I think it's a common developer challenge. While I can't speak to all other companies' founding stories, I've personally lived through this PTSD-like situation for 6-7 years. This is my third company, and we sold my last company to G2.

It's a common developer problem, and you want it solved not only because it's infrastructure, but also because there's a lot of domain knowledge involved that you don't know. You might be a good developer, but you have no clue how NetSuite works. This involves business logic and domain knowledge.

I think the feeling from developers and product people is a bit visceral. They wonder, "How did we land here? We were building compensation management. How did we end up building a NetSuite integration for 5 years? This is crazy!"

The problem manifests itself because SaaS is growing so much. All these systems of record are good databases, but they have poor UX. Have you been to Salesforce recently? They're our main partner, and I'd like them to take us very seriously, but sometimes you think, "Hey, it's almost 2025. The UX needs to be a little better." Many companies have built good tooling around usability, and Salesforce understands that.

The proliferation of SaaS has made this a very common problem, which is why many people have tried to address it in their own ways with their own vision. It's also a revenue generator for many companies. The ACV is decently high, so when you dig around, you see that this isn't just a developer tool that you monetize in ARR. Commercialization can happen quicker than in other categories, which makes it lucrative. Some people get in because it's easy money.

The reason I got into this, you have to look at my background. I tried to solve SaaS selection, usage, spending, and

management for 10 years. At G2, we talked about the marketplace, the buying and selling of SaaS. My hypothesis is that companies like Ampersand need to go deeper into the infrastructure layer to build connectivity so that adoption and the buying and selling process is actually faster.
What we're doing is essentially unlocking revenue in the enterprise. Enterprises are buying quicker, getting the data in, validating whether it works or not, and moving on. I personally feel that I'm still continuing my mission of SaaS proliferation, just attacking the same problem at the infrastructure layer. It looks like an integration company, but we're building general-purpose infrastructure to build connectivity for the next generation of SaaS. That's how I think about Ampersand and how we approach it.

One of the things you mentioned is HR as its own category. My understanding is that Finch operates in this space, and Merge's strongest product-market fit is in HR, even though they do go-to-market and other functions as well. Do you think HR is actually a very specific category with unique characteristics?

I think that companies need to verticalize first before they go horizontal. This is my take on entrepreneurial journeys because then you can solve problems of a particular small group of people better. We've chosen the go-to-market stack because at G2 we were solving for that use case. I've seen it quite a bit, and I also know a lot of go-to-market stack founders.

We chose the go-to-market stack and ancillary ERP stack. HR is like any other software category - it's pretty big and has a lot of fragmentation because it conforms to local governance. There's this physical component to HR based on geography. California as opposed to Texas, as opposed to India, as opposed to Ukraine - so natural fragmentation is higher.

I think the first product that Finch and Merge built actually makes sense to me. You may not have to go deep; you just want to run analytics on top of your ATS and connect it to your payroll. You want to understand how many people came to the door and how many people are on PIP - what is your conversion funnel? So that kind of use case is what Merge and Finch enable.

The product architecture is somewhat similar because if you are going into Workday, Workday will also have a lot of customization and interesting workflows running on top of their customization. The unified API is doing an interesting challenge, architecturally. I have an engineering background, so you basically first say, "Hey, shove this complexity under the carpet. Here are the 10 fields that you need." And then you're now like, "Where is the 11th field? Where is the 12th field? Where is the 13th field?" You know, where is the 23rd thing? Then you're bringing in selectively. It feels like a hack on top of a hack.

The architecture there will probably evolve, and I'm sure Merge and Finch, as they raise more capital and try to sell more to the enterprise, will have to adapt to that complexity and customization. In the enterprise, customization is the name of the game, so they have to adapt. I think that unified APIs were a good way to bring in this momentum, but I think you'll see other nomenclature around it. It's a good go-to-market play; it may not be a great product play, and so there's some technical debt that we have to clean up along the way.

Plaid is interesting. If nobody is building sales, marketing, customer support, or revenue operations tooling, I don't have a business. If nobody's building HR tech, SaaS companies like Merge and Finch will not have a business. I think that fintechs have slowed down a little bit, which is why Plaid is probably seeing a decline. We are all beholden to market patterns.

If there's a huge surge in sales and marketing tooling where AI is impacting that stack, you can rapidly experiment. We are seeing some organic pull, but I think it's very much oriented around market dynamics. Our growth is influenced by these factors.

I think what Merge has done is interesting, where they have attacked multiple categories. The success of a company will be governed by how many categories they can address with the right form factor, so that enterprises are capable of adopting their solutions across the stack. In that way, we're all very early in this space, because the next 10 years will shake out in interesting ways.

How does AI's ability to write code and use APIs change the nature of demand for products like Ampersand?

I think about that question quite a bit. Writing basic code is the easiest part; scaling that code and then scaling a maintenance product around it is pretty hard. Right now, all the code you're writing is basically to check off boxes. We have a Salesforce integration checkbox, for example. The AI can spit out something, so you use all of this code generation to largely unlock the first customer conversation - to show that we have something.

As you get into more sophisticated territories, I think a lot more reasoning and resilience is needed. AWS cannot tell me, "Oh, our database hallucinated and that's why we lost a bunch of your data." I think AI starts the job, and products like Ampersand finish it and solidify it. We are an edge case scaling product, and that is where a lot of human reasoning will be needed.

I think that AI is actually going to help us in areas where the edge cases have much higher value than the basic work that AI will do. For example, in the GE use case of tenancy management with a lot of configuration, I can see AI agents using Ampersand to solve some of those issues. I don't see AI doing this natively on its own - that's my take on it.

The end cases, the depth, the personalization - all of that is hard stuff, and humans are not getting it right. Humans are struggling with these aspects. I think that we are well-positioned in the AI world because we are depth-first.

Assuming everything goes right over the next 5 years, what does Ampersand become, and how has the world changed as a result?

I think if Ampersand is successful—and you should probably quote this—you know, I'm old enough where I want my company to be successful, but a bigger goal here is to increase global developer productivity by 30 to 40%. That's because so much time is being spent on integrations, and it's just lost—a bunch of people have lost their youth on this.

I think that when a company like Ampersand (hopefully it'll be Ampersand) is successful, it just increases velocity to innovation. You're not saying, "Hey, we can't read data from this custom ERP, and we don't know what's going on." We won't work with you because we have no way to talk to your

system that works for you but doesn't work for us. So that interoperability that is at the core of software, we want to solve for it.
In 5 years, I think that the long tail would not matter. You can do really deep edge cases; you practically don't have a problem with integrations at all. Good SaaS companies, which are essentially going to be all AI companies, are being valued by the workflows and the intelligence that they bring to that product. It's not like, "Hey, I have a better Salesforce integration so I can reach CRM data better, and that's why I can do these things better." That is a commodity.

I don't worry in the morning thinking, "Oh, today I'll run servers, and if it goes down, then what do I do?" It's commoditized now within three large platforms, which we call the public cloud. I hope that companies will come in and create these customer interoperability clouds where you're reading from and writing to these systems, and you don't bother about rate limits and retry logic. It's taken care of, and then you're just focused on the way you're acting on that data because it's high fidelity, it's fresh, it's in good shape. Your workflows, your UX, your data quality, and the models that you're running on top of that—success is defined by that, not what integrations you build.

One thing I didn't talk about is that a lot of legacy software still exists in the world. Over the next 10-20 years, there's going to be a huge migration from legacy systems to more modern systems, and I think products like Ampersand are going to help there too. It's about interoperability and also migration to the cloud. I think there's a lot of revenue potential there as well.

We are 12 months into this venture, so I think there's a long journey ahead in the next 10 years. Modernizing software in general is also something I'm passionate about. If Ampersand is successful, then we can instrument a future where software is more modern and being built quicker and better. That's kind of the mission around Ampersand.

Disclaimers