SACRA

EXPERT INTERVIEW

UPDATED
03/11/2022

Bucky Moore, Partner at Kleiner Perkins, on Jamstack's big upside case

TEAM

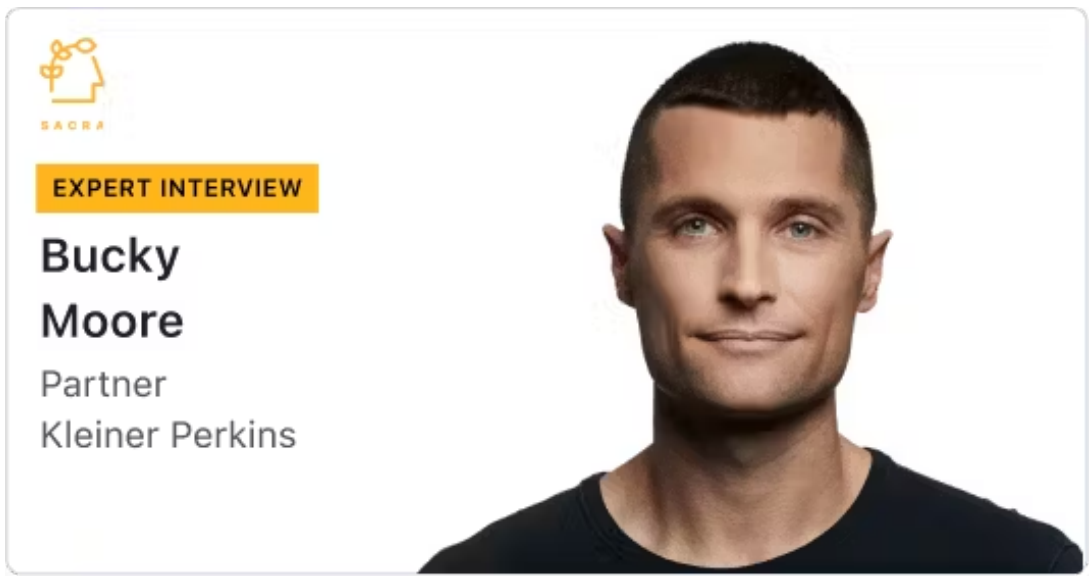
Jan-Erik Asplund
Co-Founder
jan@sacra.com

DISCLAIMERS

Published on **Mar 11th, 2022**

Bucky Moore, Partner at Kleiner Perkins, on Jamstack's big upside case

By **Jan-Erik Asplund**



Background

Bucky Moore is a partner at Kleiner Perkins. We talked to Bucky because he's a prolific investor in developer-focused tools, including Netlify and other Jamstack or Jamstack-adjacent tools, and we wanted to learn more about how he sees the market evolving and what the upside case looks like for the Jamstack ecosystem.

Interview

Could you start by painting for us how you think about the Jamstack space from a high level and what your investment thesis is?

If you think about the number of web applications and websites that exist on the internet today, the rate at which those websites are being evolved or rebuilt and, frankly, the net new type of web applications and websites that are being built, it's an enormous pool of activity. So is the amount of

infrastructure, tooling and services that need to be purchased, adopted and consumed to support that activity is also large, and the resulting opportunity for startups.
For most companies, web development is synonymous with software development. The web is the primary medium of interaction for businesses and their customers. Ultimately the web has won and will be relevant for a very long time. While there will always be a place for mobile apps, the web is a very, very strategic platform for any business owner that has ambitions to interact with their customers through digital channels. This makes web infrastructure a very exciting theme to index against as an investor.

With respect to the Jamstack, what's happening is that there is this new way of developing these applications that optimizes for performance, security, simplicity and -- probably most importantly -- the productivity of the developer and the ability to build these really cutting edge, engaging experiences. This new way of doing things is being pulled forward by the JavaScript community. Web developers want to work in the latest iterations of JavaScript and frameworks that enable you to choose between static and dynamic page rendering so that users have a much faster and more delightful experience with your products.

Ultimately, many developers would say it's just an objectively better way of building and delivering web experiences. This is what is driving the groundswell of commercial opportunity for companies like Netlify and others in the ecosystem.

Do you have a way that you taxonomize the space and categorize which companies are important? For instance, we've been thinking about Vercel and Netlify, on the one hand, and then all the various APIs that become important in a Jamstack approach, on the other, as two of the really big categories.

I think there's a way to talk about that today, and there's a way to talk about what I think it might look like five to ten years from now, probably closer to five. The way I think about it today is the companies that you mention are platforms that allow developers to deploy and host modern web applications. More specifically, the front-end components of your application, and how they interact with back-end components like databases and APIs. Now, technologies like Next.js and Remix, "some

pages should be statically rendered, and we'll give you the infrastructure to build, deploy and serve those static assets to your end users, but there are also going to be some things you're going to want to remder dynamically. And for those dynamic things, we think it's an objectively better solution to arm the developer tasked with designing the frontend to decide which part of that web experience renders dynamically versus statically." Next.js is known for introducing this kind of server-side rendering to the equation.

By way of the possibilities that that unlocks for the citizen front-end developer who isn't used to being able to do server-side rendering type things without having to go and spin up a bunch of infrastructure and be proficient in backend technologies, Next.js is this unbelievably unique experience for them. They can actually do that all in the same framework that they're building their frontend in. That's why I think Next.js really broke out -- it just unlocked a new set of possibilities for front-end developers that didn't exist before, and in a way that didn’t force them to take on all kinds of new complexity and learn new things.

When I made the comment about today versus five years from now, today you would say those platforms are the hosting infrastructure for the frontend of your application, where anything backend would land on something like AWS or a third-party service like Twilio or Algolia. But I think they are ultimately the clouds of the future. These are end-user-focused cloud computing platforms that treat the serverless, edge-first way of doing things as a first-class citizen, but are increasingly bringing full-stack possibilities to bear to the front-end developer.

What these platforms have succeeded at is giving front-end developers superpowers. They make it really easy for front-end developers to work fast and to get their stuff on the internet, without having to think about hosting and configuration and all the other stuff that they used to dread having to worry about and generally stay away from. Before Netlify existed, the only way you could do those things was to go to the AWS console, which is a very overwhelming experience for a developer who's used to working in React on their local environment and occasionally going into GitHub and checking code. I think of these products as the hosting platforms of this new era of serverless computing.

I think it starts at the front-end developer because the front-end developer is arguably going to be the largest population of engineers as time goes on, because more and more of the backend stuff is being automated by managed services. The stuff that people used to build and used to be paid to operate as DevOps people or SREs -- that surface area of concern, is asymptotically going to approach zero at some point. There will always be software vendors that have some point of view on how a part of their backend needs to be more custom, and they're going to go and take it on themselves. But even at that level, the ladders of abstraction just continue to climb upward. So I think the Jamstack movement speaks to empowering the front-end developer as the guiding light, and that's the right side of history to be on.

One thing we've heard is that Vercel and Netlify today look like they're reselling a bundle of AWS services, while AWS -- and maybe Azure to an extent -- are attempting to go after the same market with products like Amplify. I'm curious if part of your thesis is that it's going to be hard for AWS to put together a similar end user experience that Netlify or Vercel has because they're an infrastructure company?

Suffice it to say that history has shown us that AWS does not necessarily excel at building world-class user experiences. The reason they don't excel at that is because that's not the DNA of the company that they've built. It's not what people are incentivized to be great at, and frankly, it's not clear that it's a strategic priority for them to excel at that. What they want to excel at is providing underlying primitives that are the cheapest, most performant, most reliable and most scalable primitives that you can get from anywhere. That's what they focus on, that's what they do.

They build those primitives for the mass market. The implication of building primitives for the mass market is there will always be niches of developers that may appear niche today to a business of that size but that are actually quite interesting to a startup and will probably get larger over time, such that they become things that you would probably be stretched to call "niches." There's arbitrage there, which is to go and build things that help them do their work better and that AWS isn't going to go and touch..

In the case of Netlify, I think one thing you could say is, "Wow, they're really just building a layer on top of what would otherwise be services you could glue together in AWS." That is in fact true, but it's not about those different services. What it's about is the workflow on top of those services that makes everything really easy and low cognitive load to operate. AWS requires you to make all these decisions about how you configure, provision and set things. I think the future is a world of more end user-focused infrastructure, where you think about just an individual developer trying to get their work done and not having a desire to express preferences on those things. They expect the vendor to make those choices for them and just make it work.

That's where you get back to this serverless way of doing things. I think the term "serverless" is a bit loaded. On one hand, it means presenting a solution that precludes the developer from having to think about infrastructure whatsoever, so no clusters, no provisioning, no configuration, etc. On the other hand, serverless refers to pay-as-you-go billing, which is "I'm only paying for something I'm using, so I'm not paying for the cost of the long running server."

Netlify actually delivers on both of those objectives, as do other players in the space. I think it's enabled by the fact that everyone is really focused on delivering a workflow rather than the infrastructure. The workflow sits above the infrastructure and is basically there so that you don't have to think about it.

When we think about products that abstract away a lot of the backend and other things that developers -- especially front-end ones -- don't necessarily want to deal with, one product that comes to mind is Heroku. Does looking at Vercel and Netlify as a Heroku for the modern generation resonate with you? Are there lessons to be taken away from Heroku's successes and failures?

So my approximate understanding of both the success and failure of Heroku was that they were capable of making it really easy and convenient to get started with. But as your application became bigger and more important and required more specificity to it, forcing you to consume the database from it was limiting, and it drove people away. By focusing on the needs of the front-end developer and then extending to best-of-breed providers of back-end services like databases

and 3rd party APIs, you give the developer more flexibility, and can scale with them in more cases.
When Heroku was getting started, there was not this acceptance and proliferation of third-party cloud-native database services like PlanetScale and Cockroach. I think these are the future of what big customer-facing experiences are going to be anchored by. I can't speak for every company in this space, but my sense is that the Jamstack ecosystem envisions being a partner to those companies and pulling them into the ecosystem.

Disclaimers