EPISODE 656

[INTRODUCTION]

[0:00:00.3] JM: Building software today is much faster than it was just a few years ago. The tools are higher level and they abstract away tasks that would have required months of development in the past. Much of a developer's time used to be spent optimizing databases, load balancers and queuing systems in order to be able to handle the load created by just thousands of users.

Today, scalability is built into much of our infrastructure pieces by default. We've had several years of infrastructure with automatic scalability and some of the more recent advances in developer tooling are about convenience and faster development time. These are tools that are built, because we're no longer worrying so much about automatic scalability since that's built into the tools. Developers are spending less and less time with the ambiguous idea of a server and the problems of distributed systems that come out of those servers, and more time interacting with well-defined APIs and data sources. Developers have a lot of trust in the consistency and the reliability of the distributed systems applications that they can focus more on the higher level components of their application, and their own unique application data model.

A few examples of this higher level data source system are app sync from Amazon Web Services and Firebase from Google. These tools are databases with very rich interactive functionality. If you have done any high-level application prototyping recently, or built any new apps yourself, or built web apps, mobile apps, things that you're entirely building from scratch in the last couple years and you haven't looked at Firebase or AppSync, these things might be very useful to you, because they take so much of the difficulty out of application development, at least on the backend. Instead of having to create a server to listen to a database for changes and push notifications to your users in response to those changes, AppSync and Firebase are back-end systems that can be programmed to have this functionality built in.

They could be hard to explain what that means, unless you've actually dealt with it, or if you have tried to build a back-end service, but you really get a lot of value out of having these high-

level rich database-like systems with server capacity, server functionality built in; things like push notifications in response to a database update that used to be something that you would have to build an entire service to be listening to your database for, and now it's available in these super rich database-like products, like AppSync or Firebase.

There are many other examples of high level APIs, rich back-ends, developer productivity tools that lead to shorter development time. It's much faster to build applications today; consumer applications, business applications, and this gives you a lot of leverage as an engineer. How should you respond to that leverage? It means that you can have faster expectations for getting a product to market. We can prototype really quickly for low amounts of money without sacrificing quality. We can also keep our teams smaller. We can perhaps raise less money. We could spend more time focusing on design and user experience and business models, and less time focusing on keeping the application up and running.

Today, I talk to Nader Dabit, who is a developer advocate at Amazon Web Services, and I always enjoy talking to Nader, because he's entrepreneurial, but he is now working in Amazon, and there's a lot of conversation that we have in this episode about why would somebody as entrepreneurial as Nader, he did run a very successful business where he trained people how to use React Native, he has a podcast React Native – I think it's – what is it? React Native Radio. He has written books. He's done a lot of stuff on his own and yet, he still decided to go and join Amazon.

He joined Amazon Web Services as a developer advocate, where he works on developer tooling, developer outreach, and he explains why he did that, why he left working by himself to go work for Amazon. It makes complete sense to me. We also talked about more generally how developer tooling is changing and how that change in tooling changes the potential for high output, fast iteration among developers. It's a strategic philosophical discussion of how to build modern software, how to build businesses and how to position yourself when there's so much change going on, and there's so much tooling that you could take advantage of.

We also talked a little bit about AR and some of the potential applications there, other platforms that are nascent. I really enjoyed this conversation, because it is the same stuff that I'm always

thinking about. What can I build? How fast can I build it? Is there a durable business opportunity in something? We touched on all these things.

Before we get on with the show, I want to mention that I am hiring for a new company that I'm starting and I can't talk about the product quite yet. You just give me a month or two, but I'm very excited about it, and I'm looking for an engineer in the Bay Area with significant experience in React JS and some sort of cloud, whether that's Heroku, or Firebase, or Google Cloud, or AWS, just a back-end cloud technology.

If you have that experience and you think you would work well with me, e-mail me at jeff@softwareengineeringdaily.com, or you can check out the job posting for more details at softwareengineeringdaily.com/jobs

[SPONSOR MESSAGE]

[0:06:38.7] JM: Failure is unpredictable. You don't know when your system will break, but you know it will happen. Gremlin prepares for these outages. Gremlin provides resilience as a service using chaos engineering techniques pioneered at Netflix and Amazon.

Prepare your team for disaster by proactively testing failure scenarios. Max out CPU, black hole, or slow down network traffic to a dependency, terminate processes and hosts. Each of these shows how your system reacts, allowing you to harden things before a production incident.

Checkout Gremlin and get a free demo by going to gremlin.com/sedaily. That's gremlin.com/sedaily to get your free demo of how Gremlin can help you prepare with resilience as a service.

[INTERVIEW]

[0:07:37.5] JM: Nader Dabit, you are a developer advocate at AWS. Welcome back to Software Engineering Daily.

[0:07:42.5] ND: Hey, thanks for having me.

[0:07:43.8] JM: Last time you were on, we talked about React Native. Today, we're talking about something that's quite different. We might talk about React some, but there's a stark difference in how to build a consumer-facing application today, versus five to 10 years ago. That's really the thing that you and I want to talk about on the podcast today. Basically, the idea is that I don't think people realize just how much faster software development has gotten, and I think you would probably agree with this, that's why I'm excited to talk to you.

I want to start this conversation by just going through some examples of consumer applications that people think of as quite hard to build, but they're actually – there's actually a lot of newer tools that make these applications much easier to build. There's a set of applications, I think even a boot camp graduate, or a computer science university student could hack together in a few weekends with modern tools. That if we were talking about five years ago, or 10 years ago, these things would be much, much harder to build.

To go through some examples, I want to start with Instagram, because this is obviously an app that tons of people use. If you were to build Instagram today, how would you contrast that with how you would build it five years ago?

[0:09:06.5] ND: Well, I guess to start with you could think about the team that would be involved in building it, or you could just say the person involved in building it, if we're talking about how you might throw something together today and like you said a few weekends, at least get a prototype or something like that. Before, you'd have to have – you would have to have a full team of engineers probably. You'd need a DevOps team, or a DevOps person. Maybe you need someone that was understood, server-side development really well, you need a front-end developer. You might even need mobile developers for the different platforms that you're on, so iOS and Android. I guess, maybe a system administrator.

You're talking about a good five to 10 people and that's just – unless of course, the person understood all of this stuff, which is I guess is possible, but in reality, it's rare for someone to be specialized in all these different areas. Today, what we're going to be talking about is taking advantage of managed services and how you can use an authentication provider like Auth0, or AWS, we have our Cognito, or whatever authentication provider you'd like to use.

You can use manage database layer like either what we have is AWS app seek, or Firebase, or something like that to manage your data store. That would get you to the point where you could maybe leave off the server-side developer and be the frontend developer working with these services. Then S3 to store your images, or whatever media that you're working with, I guess with Instagram, would be images and videos.

The idea is taking advantage of managed services as a frontend developer, increasing your productivity and moving further up the stack. I think that's a general thing that we're going to be seeing a lot more of in the coming years. We at AWS and AWS mobile where I work, we're building a lot of stuff around it. I think a lot of other cloud providers are looking into this space and they're building a lot of tooling around it, because I think it's going to be the future of where you're going to be seeing a lot of software engineering in the future.

[0:11:08.4] JM: Completely agree. Five or 10 years ago, I would probably start with a rails monolith, or perhaps a NodeJS monolith. I would be running my own Mongo database. I would have to set up my own database node. Today, both the server aspect of software development and the database aspect have been rapidly – the process of building them has been rapidly accelerated. The process of maintaining them, the DevOps role that you might have needed to hire for in the past is abstracted into the position of the cloud provider. I think that will be one theme of the conversation today.

What are the specific tools that are making this faster? They're making the development process faster. We're talking about abstracting away our back-end, like replacing the rails monolith of the past, or the node monolith of the past, or replacing the self-hosted Mongo database of the past. What are the tools that are replacing those tools?

[0:12:11.6] ND: Specifically on the database and what we're going to be seeing in the future, we're going to see other companies that they come out with stuff like this, but at AWS, we have an abstraction, or a managed GraphQL service called AWS AppSync. GraphQL, you've probably heard about it. If you haven't, it's basically a different way of interacting with your resources than the traditional REST API.

The general idea behind it is you have more control over your payload size. You ask for what you need from the backend from your database and you get only that, so you end up with smaller payloads, you also end up with – they're just a different, completely different general idea of how you build out. It's a different philosophy of how you build out your APIs. To go into what AppSync is, it's basically a managed GraphQL service.

With AppSync, you can have a single GraphQL endpoint and you can then work with pretty much any data source, lambda function that you would like. You end up having more of like a consistent API working with multiple different resources. If you have for instance, a DynamoDB, table or DynamoDB database, you also have maybe Elasticsearch for searching, maybe you have different lambda functions working with microservices that have nothing to do with AWS, you can basically interact with all of these different resources using a consistent and single API endpoint.

Before, you would have to have maybe just this is more like a GraphQL thing. Say you would have a user's resource in your REST API and then you would have maybe users/ another endpoint, like students maybe, or maybe you would have users/students/semesterone, or whatever, you would have all these different endpoints to drill down and get different types of information.

With GraphQL, you would basically have your users as the main resource and then you could then ask for the different data from within those users. You could, instead of having multiple endpoints, or you're basically working with a single endpoint and having different asks, I guess you'd say from your database.

[0:14:24.6] JM: GraphQL, the challenge that I always used to hear about GraphQL was that you have to set up your own server, you have to set up a GraphQL server, because GraphQL is this piece of middleware that translates all of your GraphQL specific requests into specific requests for your different back-end. If you had Elasticsearch back-end, if you had a node back-end, if you had some other back-end, a lambda function, it would be able to federate the requests to those different back-ends, and that that setting up that middleware layer of the GraphQL server is annoying. How has that been simplified?

[0:15:05.4] ND: I think you hit the nail on the head with that. That's where a new technology like GraphQL, that's just what's going to happen at first. There's going to be a lot of innovation, a lot of changes and no real consistency across any best practices, or different tooling around it. I think we've seen a lot of, a lot of work done lately, especially with Apollo. They've released a new Apollo server that seems to work really well. Really at the end of the day, building a GraphQL server on top of your existing resource, maybe if you have an existing REST API, or if you just have an existing database, it is a lot of work. Dealing with best practices around something this new, and then thinking about things like security and authentication and authorization and just – and also just building it to be as fast as possible, it's not an easy task.

You have to think about not only learning how GraphQL works, but then you're also implementing the typical things that you would in your in your database fetching operations. I think where something like AppSync, or something like AppSync in the future, I'm sure there's going to be other things out there; what you're gaining there is you're having a competent group of engineers that have spent a lot of time thinking all of these best practices and building them for you.

Instead of having to learn all this stuff and understand all of it, you basically can just use a managed service where people have already done all this work and you can just depend on your managed service and the decisions made there, and you can just build your frontend and not really have to worry about building, maintaining and deploying your database – or I'm sorry, your server.

It's just like if you're subscribing to the idea of using something like Auth0 for authorization, you don't want to build out your entire authorization scheme and worry about encryption and different things around security. You can depend on them to have all that worked out for you, with user storage and things like that and you can then not build it, just pay them to manage that for you. It's the same idea, but on the side of a database.

[0:17:16.6] JM: Is the model here that I write my own front-end, like I write a mobile front-end in React Native, or in iOS and Android, or on the web and I communicate directly with my GraphQL server and that handles my entire back-end?

[0:17:31.9] ND: That's the idea. Yeah. With AppSync right now, we have a few different first-class databases that are just already out-of-the-box, configured to work really well without doing any extra work.

Right now, we have DynamoDB and Elasticsearch and we also have lambda functions. If you're not using DynamoDB, if you're not using Elasticsearch, if you have maybe some other resources within AWS like Aurora, or RDS, you can simply just use the lambda function to interact with those. We're working on possibly additional data stores that are first-class that are going to be part of what AppSync is.

Also, if you already have an existing MongoDB somewhere out there database, you can just use a lambda function to interact with it. Basically, the idea would be to use lambda functions as the entry points into your other data sources, and then have the GraphQL as the single source of your API URL.

[0:18:27.4] JM: Now first of all, a minor elephant that I want to get out of the room, so for people who are just hacking on apps, which I know is a lot of a – a large percentage of the audience is boot camp grads and college students and stuff. The idea of using managed services to hack together what you're building, in terms of the expense that you're going to spend on these things is almost always going to be super trivial. You might as well be – it's like 5 bucks a month, or 20 bucks a month, or something like that in exchange for saving you tons of time.

I just want to say upfront, I'm a big fan of the idea of using managed services to save time and to build faster. Now that said, I think there's actually a quite a big array of different managed services to choose from. For example, I have not used AppSnc. It sounds perfectly reasonable to me. I've used Firebase. Firebase has been an amazing tool for me. I've used it in two or three different applications and it's just been so simple, and I don't even write much software anymore, so the fact that I can work with Firebase quite easily makes me really optimistic.

How do people choose? How do you recommend people choose between these different back-endless models of development? Because I think, Firebase is the Google version of the back-endless development system and AppSync is AWS's back-endless system.

[0:19:54.0] ND: Yeah. It really depends on the type of application you're building and what you're comfortable with. When you're working with some of the stuff that we're building, I can't really speak to anything else, but with AWS and especially with AppSync, you're really using the service, because you k now that you're going to be dealing with something that is scalable, you're going to be dealing with something that is secure, you're going to be dealing with something that has already been battle tested and that's production-ready.

The trade-off and you talked about just when you're prototyping stuff how much easier it is and how much time you save when you use these managed services, and that's entirely true. When you think about, if you value your time at a certain rate, or if you charge hourly for your time to build stuff, you can think of it that way. If you're like a $100 an hour developer and you want to deploy a back-end, is it going to cost you 10, or 15 hours to build this? That's going to be a $1,000, $1,500, or do you want to spend 5, or 10 bucks, or whatever it's going to cost to just deploy it with one of these managed services?

Another benefit of using AppSync, I guess you would say, we've really, really focused on a couple of things that are really starting to become more and more prevalent and more mobile applications, especially with the prevalence of IoT and where companies are trying to ship applications all around the world, where their connectivity might not work as well as what we're used to. Those two things that we're really focused on at when we we're building AppSync and that we're continuing to focus on, or offline functionality that works out of the box and real-time functionality.

When you're using an application, when you go in the subway, so for instance if you have Twitter, you go in the subway, you lose connectivity, your liking tweets and stuff like that, you might not be able to get a new feed, but you can still interact with the application without it crashing. You come back online, you don't really get an error or anything. You expect the app to continue to work.

Now in reality, building that type of application is really hard. You have to think of a lot of different scenarios, you have to handle a lot of different use case. With our AppSync, GraphQL client and AppSync together, we have an offline functionality that's just built in out-of-the-box. It

works not only with JavaScript applications, but we also have Native clients. If you're an iOS developer and Android developer, we have first-class GraphQL clients that work in this fashion.

Then the second is subscriptions, which are the real-time functionality. This isn't really an AppSync specific thing, it's more of a GraphQL thing. You mentioned Firebase and they are really – they're really well-known not only for their manage services as a managed database, but also for its real-time capabilities. If you've worked with AngularFire, it's really a nice experience.

With GraphQL, you have this idea of subscriptions, which is the real-time functionality that you get with GraphQL. The idea is you have a data set that you've pulled down and your application is rendering it to the screen. You subscribe then to changes within that data set. If anything changes there, for instance you have an array of tweets, for example. When someone pushes a new tweet to that data source, your subscription fires and the new piece of data that you haven't already had comes down and gets pulled down into your application, and then you have that data in real-time.

The same thing I guess with messaging applications, or even dashboards, or if you're building something like a editing application, where you have multiple people editing a document for example, this is where that real-time and also the offline stuff makes a huge difference.

[SPONSOR MESSAGE]

[0:23:49.4] JM: Azure Container Service simplifies the deployment, management and operations of Kubernetes. Eliminate the complicated planning and deployment of fully orchestrated containerized applications with Kubernetes.

You can quickly provision clusters to be up and running in no time, while simplifying your monitoring and cluster management through auto upgrades and a built-in operations console. Avoid being locked-in to any one vendor or resource. You can continue to work with the tools that you already know, so just helm and move applications to any Kubernetes deployment.

Integrate with your choice of container registry, including Azure container registry. Also, quickly and efficiently scale to maximize your resource utilization without having to take your applications offline. Isolate your application from infrastructure failures and transparently scale the underlying infrastructure to meet growing demands, all while increasing the security, reliability and availability of critical business workloads with Azure.

To learn more about Azure Container Service and other Azure services, as well as receive a free e-book by Brendan Burns, go to aka.ms/sedaily. Brendan Burns is the creator of Kubernetes and his e-book is about some of the distributed systems design lessons that he has learned building Kubernetes.

That e-book is available at aka.ms/sedaily.

[INTERVIEW CONTINUED]

[0:25:24.7] JM: As we're talking about this stuff, I know there are people in the audience who are fresh boot camp grads, or they just graduated with a computer science degree, and they have been building applications in Java, or they've been building NodeJS applications, so they've been learning Ruby on Rails in their boot camp. Now they're hearing wait, we don't need those back-ends? We're just supposed to actually completely change our paradigm and move into a world of completely managed services? That's how we should be hacking together our greenfield apps?

If you're to have a conversation with somebody like that, how would you tell them to shift their mindset to picking from this palette of managed services, instead of that, I think the very tried and true, and in fact, a model of application development that's going to take a lot of time for people to overcome as the default, which is like, "Let's stand up a monolith and then let's start building a frontend to talk to the monolith and etc." With managed services, you really can throw a lot of that out the window, but I think people are having trouble letting go of that idea.

[0:26:44.3] ND: I don't feel it's a winner-takes-all thing, to be completely honest. This is just my view of everything, but the way I see it is we're going to see more and more front-end developers be able to take on a lot of the work of back-end developers, and I feel like back-end

developers are going to be of course still very much so in demand. I think they might move to instead of just building REST APIs where they're just throwing up endpoints, they might move on to more important tasks, or more specialized tasks.

You're seeing a lot more data science and stuff like that around the backend with it, with back-end developers, or people that understand these languages really well, like Python and whatever, C++ and stuff like that. I don't really think it's going to be oh, all of a sudden, these skills are worthless. Instead, I think we're going to see these skill sets be just changed in a way, where frontend developers can, to take on a lot of this extra work that was being done. Then the back-end developers are just going to be doing similar things, but maybe slightly different things.

You see this and how we've seen cloud computing change over the last few years with the move to more and more people using these serverless services, but just serverless in general has taken off, but you still see a huge demand for DevOps and people that understand how to be a network engineer and things like that and system administrator.

[0:28:13.1] JM: Completely agree. I was playing a bit of an extreme, taking extreme position there. Really the –

[0:28:20.0] ND: It's a good point though.

[0:28:21.9] JM: Well, so the thrust of what you're trying to say is if you're somebody who's a mobile developer, or you consider yourself a web frontend developer, you can leverage these tools to get a lot more mileage out of your position and probably build apps a lot faster than you would have had to in the past, especially if you would have in the past needed a back-end developer to help you with standing up some of those end points. Now you can focus a lot more on design, you can focus a lot more on user experience. You may not even need a back-end developer on your "team."

[0:29:01.9] ND: This is entirely true. I mean, I just published an application to my Github repo called Heard, and it's a TRD, like I heard something. It's me creating a clone of what Twitter would be. When you think about Twitter, like you see all these different Twitter apps that are in

tutorials and stuff like that and documentation, but a lot of times they don't really take into account things like following and followers, and you want to listen for new tweets, so when a tweet comes in, you want it to show up in your screen. Then you have to think about authorization and fine-grained access control. Then of course before that, you have to have authentication.

Imagine building all of that from scratch from the ground up as a new developer. It's just mind bogglingly hard in a real-world application. I threw all this stuff together using AWS Amplify, which is something I would like to talk about too in just a second. The idea is we have a bunch of different services that are geared towards this serverless, or managed service model, where frontend developers, once they've gotten to the point where they're competent with their frontend tooling, they can then move on and learn how to work in the ecosystem, or in the consoles of these managed services and then just be the glue that interacts with those managed services.

Yeah, ideally if you are in a startup scenario and you're just strapped for resources, this is the way they go in my opinion. You're going to hit a wall to the – you're going to probably get to a point where something isn't scaling well, or you need something that you just can't do. You're going to end up of course needing to bring on more and more people. The fact that you can build something as a single developer that not only works, but scales, because when you think about it we can all build something that works. It might take a little extra time, but in reality some of the issues that happen, or when you end up getting a bunch of users and you have to worry about that scalability, these manage services especially are well-suited for that. If you can spend the same amount of time, or less time building something that doesn't scale, versus building something that does scale, I think it's a no-brainer.

[0:31:12.2] JM: AWS Amplify, what is that?

[0:31:14.5] ND: Amplify is something, is a CLI that we've just released on July 16th. It's a JavaScript SDK that was released last year in 2017. It's two different things now. Before, we only had the JavaScript SDK, but now with the CLI, it's more of a workflow, I guess you'd say for frontend to developers to quickly scaffold cloud-enabled applications, or manage serverless applications from their command line.

If you're familiar with rails, you understand how it came out of nowhere a little bit and it revolutionized really the way a lot of developers were able to become all of a sudden extremely efficient in building real-world web applications, because from the command line, you could just spin up a lot of functionality just with a single command, that you have the controllers and models and you could do authorization and things like that.

Amplify, you install it as a – from your command line. It's a globally installed NPM package, and then you can scaffold different services. You can just say AWS – I'm sorry, you can just say Amplify INIT, which will create a new Amplify serverless application in your – the current folder that you're in, and then you can add things like authorization, you can add things a GraphQL API, you can add analytics, you can add chatbots, you can add AR and VR stuff, you can add all types of stuff just with these single commands. You can just add one thing at a time and then say that you're just playing around with something and you don't like it anymore, you can just remove it. It's pretty easy.

I think it's a really good entry point into working with these more sophisticated services for frontend developers, because we're so used to working on the command line. It's something we're used to. It's like your views create react up, you're working with NPM and stuff, now you can just spin up a cloud-enabled, or cloud-enabled application and start adding services from the command line. The command line is the Amplify CLI and then the JavaScript SDK is the Amplify library.

It’s a library to be the glue between the services in your frontend application. We again do all types of stuff there. We do authorization and analytics and push notifications, and a host of other things. I think we have 11 or 12 different things. Then we even have special components for Angular and for React and React Native, or we just have a vanilla JavaScript implementation.

[0:33:56.0] JM: AR example, I find really interesting. Let’s say you wanted to build some combination augmented reality and streaming application, something like, I don't know, I'm having trouble thinking of an application, but I'm sure we'll see things like this, where there is a high bandwidth video stream that is being run between your device and the cloud, and there's

all kinds of buffering and latency issues and codecs and all these complicated things that you don't want to think about as a developer.

These are solved problems. They've been solved by companies like AWS, or Mux. That's a sponsor of the show, but also friends of the show. They work on video streaming APIs. I know these are hard problems from talking to these people. Your idea with the AWS Amplify stuff is basically, if you wanted to include that in your app, it makes sense for this to be provided by a managed service in contrast to something like an NPM package, because an NPM package is not going to be able to manage, or it's not configured, auto-configured to manage cloud infrastructure for you.

Here, you're talking about something where it's a CLI that it's from AWS, so it's probably comes already connected to your cloud account. If you just wanted to declaratively say, “I want AR streaming capabilities in my app,” it makes sense to do that through a cloud provider CLI. Am I understanding the gist of amplify correctly?

[0:35:36.9] ND: Exactly. Exactly right on the on the head actually, because what ends up happening when you do initialize a new project, you have a copy of the config locally, and then you also have the actual application service that's been spun up in your account. Then you can go into the account and actually look in the console if you like to, I don't know, play around with the settings or whatever from there. Then say that you make a change in your settings there, you can just pull down the new configuration in your application, or if you make a change locally, you can just push up the new configuration. As you make changes, you're working from this flow of local to your actual managed service, and you're able to interact with it from the command line, which is really powerful.

[0:36:22.8] JM: I think this stuff is so exciting; the proliferation of managed services. Not to be a shill for the cloud industry, or for the type of companies that support Software Engineering Daily, but as a developer with as much objectivity as I can speak with, this stuff is super exciting, because – so we've been talking about – we started the conversation talking about crud apps, like a photo sharing app, or a social network like Twitter, or a dating website, and those are interesting, right? Crud apps are never going to become uninteresting.

We're at the frontier of some really exciting technologies, things like machine learning, augmented reality, video streaming, that building them with open source software is it sounds great. I mean, it is great, but there's a lot of challenges in the network and things that you just – it makes sense to have to pay for. It makes sense to have a cloud provider help you with these things. I want to get into a conversation around managed services and how to leverage them, because I think this an under-explored area, so people who are trying to think of entrepreneurial ideas, and I know there's a lot of listeners who are like that.

I think there's a brainstorming process to be built around looking at managed services. It's an artist. Artist sits down in front of their palette, or a pianist sits down in front of their piano and it's like, "Okay, what can I make out of these colors, or what can I make out of these piano keys?" With a developer, now you can look at all these managed services, like you can look at Algorithmia, you could look at AppSync, you could look at Google's video intelligence API, you could look at AR kit. AR kit is not a managed service, but something like that. Amazon has a zillion of these different services.

Do you have any ideas around the ideation process, the brainstorming process of a developer who is in this new world where there's this proliferation of managed services that are extremely high-leverage and have just not been explored very much?

[0:38:38.7] ND: Yes. I mean, I have a lot of opinions about this. I mean, it's one of the reasons I actually joined the team, because before working with AWS, I was running a really, really successful consulting company. I did not have a lack of business, that's not why I switched gears a little bit. It was because I saw some of the stuff they're working on, and I really I'm always interested in the next big thing.

If you think about cloud resources in general, or working in a cloud environment in general, it basically allows people to trade capital expense for variable expense. Instead of having to, when you come back to the general idea of cloud services, instead of having to build and manage your infrastructure, Bobby's servers and all this extra resources that you may need just in case, you're instead able to scale and then just pay for them as you go, as your actual application grows.

If you move, or if you take that general philosophy and apply that maybe to actual software engineering, this is where I see what this stuff is all about. What that to me means, say I'm a frontend developer, or just say I'm an app developer and I want to build something and I don't know how to build this server. Say I go get an estimate and the person that says it's going to cost to get this service and this service and this service built, it's going to be say $100,000. What if instead I could just subscribe to a managed version of that service for a few pennies a day, and then once I get enough users to where I actually am successful, “Okay, I have money coming in now, my build then does actually go up some,” but I can actually afford to pay it then.

You're able to innovate a little bit easier as well, because you're able to build these sophisticated things without having to invest a lot of money upfront. As a new developer coming into the ecosystem, or even existing developer, you have to think of efficiency. I think this is the ultimate form of efficiency, because you're able to take on a lot more work, but not really work, I would say. You're able to accomplish a lot more work. Then the typical developer, if you understand how to utilize these things and be efficient with them.

Instead of a company, so in a few years say for example, a company or someone wants to hire someone to build something, if you're able to come to the table and deliver 5X, and we always hear things like the 10x developer, I’m not talking about that. I'm saying if you're able to deliver 5X more capabilities of functionality than your competition, they're going to hire you. I believe in the future, we're going to see developers getting paid a lot more that can understand this entire idea and navigate their way through these different providers that offer these different services. Not just AWS, but just in general.

[0:41:34.3] JM: I completely agree. I think it's super exciting for me. Okay, so things like push notifications, SMS stuff, building chat applications, crud apps, scaling a database, scaling serverless functions, I think of these as solved problems, not fully solved, but there's a lot of really good solutions for these things. If you're struggling to build a crud app, or struggling to figure out how to deploy a database, there's a lot of solutions for these things. We talked about that at the beginning with the AppSync. I think of that as a frontier solution to solving your crud, solving your API management issue.

Not to say these are easy problems solved, there's best practices, there's well-defined tooling around it, and the tooling will always get better. I want to ask you about the things that are still a little bit too difficult, or the things that have recently gotten easy. I think of something that recently got easier, perhaps augmented reality, perhaps image recognition, or audio transcription, these are things that have recently gotten extremely easy, because they've been turned into API's. There are things that are still a little bit difficult. I think machine learning is one example there, where yes, there are frameworks for it, there are cloud services for it, like SageMaker from Amazon, I've heard really good things about. Algorithmia has something around machine learning deployments. Aside from machine learning, what are the things that are still a little bit too difficult, or the things that have recently gotten easy that are exciting to you? The frontier.

[0:43:15.3] ND: For sure, for sure, we're seeing – I mean, when I say we're like, I'm seeing a lot of demand for machine learning engineers. We're seeing that it's not easy to build real-world machine learning algorithms that actually work with data sets, because there's just a lot to learn there and there's also such a hot demand that any competent engineer that understands machine learning, or data science to a degree can make a lot of money, especially compared to just a typical developer.

To me, I see that machine learning is – I'm seeing glimpses of what could be a solved version of machine learning, and that's in managed services again, where you can have a API that you can interact with to do certain things. In reality, say that you have a specialized problem that needs to be solved, you have to actually hire someone that understands how to write this code, and also how to work with the proper data sets and do everything, and I think – I feel like that's not going to be a solved problem. I think we're going to see more and more frameworks and libraries that are built to make it easier. I don't think that that entire idea can be ever abstracted away.

What we are seeing that's become super-duper easy are things like authorization. It's almost crazy now in my opinion as a small company to medium – a small company, I guess you'd say, like especially to build your own authentication layer when you have to really – when you really are worried about security, when you can just work with one of these managed services. Or of course, that's just my opinion of course. Yeah, that's really easy now.

Working with GraphQL over the last few years has gotten much easier. Now with AppSync, I feel that's – among you say it's a solved problem, but you don't have to build in GraphQL API anymore, you can just spin one up and just hit an endpoint and it works. That's pretty cool. Yeah, I don't know. I mean, there's probably dozens of examples out there, but those are just the two that are off the top of my mind, because I've felt the pain of actually building these things. Now that I can just – I guess, the first time I ever used Auth0, that was when I was really like, "Wow, this is an amazing service," after building all this stuff myself and then using that. That was my first look, glimpse into these types of services. Now I work with Cognito, which to me is just awesome as well. I'm really sold on that stuff.

[SPONSOR MESSAGE]

[0:45:54.8] JM: Cloud computing can get expensive. If you're spending too much money on your cloud infrastructure, check out DoIT International. DoIT International helps startups optimize the cost of their workloads across Google Cloud and AWS, so that the startups can spend more time building their new software and less time reducing their cost.

DoIT International helps clients optimize their costs, and if your cloud bill is over $10,000 per month, you can get a free cost optimization assessment by going to doit-intl.com/sedaily. That's D-O-I-T-I-N-T-L.com/sedaily. This assessment will show you how you can save money on your cloud, and DoIT International is offering it to our listeners for free. They normally charge $5,000 for this assessment, but DoIT International is offering it free to listeners of the show with more than $10,000 in monthly spend.

If you don't know whether or not you're spending $10,000 if your company is that big, there's a good chance you're spending $10,000, so maybe go ask somebody else in the finance department.

DoIT International is a company that's made up of experts in cloud engineering and optimization. They can help you run your infrastructure more efficiently by helping you use commitments, spot instances, right sizing and unique purchasing techniques. This to me sounds extremely domain-specific, so it makes sense to me from that perspective to hire a team of

people who can help you figure out how to implement these techniques. DoIT International can help you write more efficient code, they can help you build more efficient infrastructure. They also have their own custom software that they've written, which is a complete cost optimization platform for Google Cloud, and that's available at reoptimize.io is a free service, if you want to check out what DoIT International is capable of building.

DoIT International are experts in cloud cost optimization. If you're spending more than $10,000, you can get a free assessment by going to doit-intl.com/sedaily and see how much money you can save on your cloud deployment.

[INTERVIEW CONTINUED]

[0:48:18.8] JM: We got to talk some about the frontend, because if we're talking about the whole picture of how application development is changing, we have to touch on the frontend, and you're an expert in this area. You were running your own business with React Native training and as you said, you did not have any shortage of business. You moved on to Amazon, because it sounds like you wanted, you were curious about the rest of the stack. I mean, you're going to come out of this Amazon experience, or maybe you'll stay there for decades, who knows, but in any case, I'm sure you're going to be brimming with ideas very shortly, because how fast people can build really impressive frontend applications at this point and how you're going to get a master class in what's available to rapidly develop at the back-end.

Since we talked about the back-end at this point, we've talked about these APIs, we've talked about building your back-endless back-end, I got to ask you about the frontend. There's a couple of interesting developments. First of all, you were working on React Native training and that was, I believe was you were helping businesses figure out their internal strategies, their internal processes around developing React Native applications, which is a cross-platform framework for developing mobile applications. You write what is it? Write once, write anywhere or something.

[0:49:41.0] ND: Yeah, it’s like learn once, write anywhere.

[0:49:42.8] JM: Learn once, write anywhere, right. Basically, the idea is you get to write mobile applications faster. A couple interesting data points, so recently we've done a couple shows about Flutter, which is Google's spin on the cross-platform mobile application development, and there was also this Airbnb post about how Airbnb had built a ton of code around React Native and then they announced, “This is not working out for us. We're moving off of React Native, slowly but steadily.”

I want to throw those two ideas at you and I want to get your perspective, especially because now you don’t have – really have a dog in the fight. Where is cross-platform mobile development today, and what are your thoughts on how it looks in the next two to five years?

[0:50:30.0] ND: Yeah. Me personally, I'm all about some efficiency. When I see something that can increase efficiency, I'm all over it. I feel like that was the reason that I got into React Native development in the first place. Not only was it really cool to build a mobile app with JavaScript, and then be able to go then build a website with JavaScript. Now, we're able to build VR and AR and desktop apps and Apple watch apps and whatever. You can build a dozen different targets with React Native, and I think that's where React really shines and React Native really shines. I still I'm actually super bullish on React Native, but definitely Flutter has been a newcomer in the mix. Actually, I gave a talk about the future of cross-platform application development exactly a year ago, and it was at React Native EU. Part of my talk was about Flutter. I feel like a lot of the predictions that, and a lot of the thoughts that I had bought in Flutter I've seen actually continue, and maybe even become more of a reality faster than I expected.

To me right now, of course, in that space there are two main of course, competitors, and in my opinion that's React Native and Flutter. I think that with – I'll talk about React Native first. React Native has been around now for over three years, I think. It was around 2015 in March, I believe it was released. It's been around for over three years and that's a short amount of time, but it's also a long amount of time compared to Flutter.

What we've seen is a lot of companies have adopted React Native at an early stage. I think that was Airbnb’s situation. They've had time to actually work with it and then found out that it's not right for them. If you actually read through the blog post, they had a lot of good points there. If you've kept up with the React Native ecosystem and the repo lately, you've noticed that there

hasn't been a lot of activity within the Facebook actual organization into the project. They released a blog post recently, kind of talked about why that was. That's because they're –

[0:52:33.2] JM: Facebook did?

[0:52:33.3] ND: - actually addressing a lot of the issues.

[0:52:35.9] JM: You said Facebook released a blog post recently talking about that?

[0:52:38.5] ND: Yes, yes. Facebook released – it's on the Facebook blog. I can give you a link to it and maybe we can share with the people listening.

[0:52:45.0] JM: Yeah, yeah. Sure, sure.

[0:52:47.1] ND: Yeah, and they went over three main points in the blog post. They're reengineering how React Native is built actually, how it works. They've spent some time over the last months actually doing that. A lot of the issues that Airbnb had were addressed in that blog post, and I think that once yeah, I think that once React Native releases this new version, we're going to see a lot of the pain points addressed there.

I will say this about React Native versus Flutter, I think in the couple of years, we're going to see an equal fight between the two. I wouldn't say fight, but we're going to see – I think both of them are going to be equally viable, because I think that Flutter has a lot of advantages that React Native doesn't have. Those have a lot to do with how the project is actually run. Yeah. I mean, I love Facebook's team and React Native. I know a lot of those guys personally. They're amazing.

I feel like Google has taken in Flutter and put it as a first-class citizen and they're listening to everything that people are saying, and they're very responsive. They have pristine documentation and examples, and they have a few things built in that React Native doesn't really support, that they let the community support like navigation, UI libraries and stuff like that, that Flutter actually maintains. I feel that's a huge plus.

For me, if I was going to pick up cross-platform development today, I would actually look at both and see what I like better. If you want to if you want to build web applications and you want to build VR and AR, you might want to look at React Native. If you're strictly worried about deploying to iOS and Android, I might choose Flutter at this point. That's just because I really like the way that the project's handled and I feel like the people running it are not only as competent as the Facebook people, but they're more engaged and actually being, I don't know, they're being more proactive about stuff.

[0:54:39.8] JM: Well, so I interviewed Eric who is the engineering manager on Flutter. I mean, he was talking about – First of all, Flutter directly interfaces with the GP – I guess, the – all right, was it a GP? Anyway, it interfaces with the operating system in the same way that Open GL does. Almost, like you're building a game. Games are cross-platform applications that work really well on both iOS and Android, and that's because games are often written, I guess in OpenGL, or in Unity or something.

These game engines that compile in a way that is very close to the hardware, and so it runs on both iOS and Android and it runs performantly, but the thing is it doesn't maintain the UI elements. They talked about re-engineering these UI developments in the context of this game engine-like runtime API. I'm sure Eric can explain it better than I can, but he talked about, I believe they would record the speed at which native UI components would perform. They would literally use a high-definition camera to record the physics. Then they would they would use the math from those recordings to recreate those in the context of Flutter. Anyway, I'm not going to butcher this story anymore. Really the precision and the approach, the highly disciplined approach to Flutter, that really impressed me.

[0:56:12.2] ND: Yeah, holy moly. That's interesting. That's super interesting. I'm going to have to actually listen to that episode.

[0:56:17.0] JM: Yeah. I mean, in terms of the future stuff, like have you seen the fuchsia, or I guess it's a rumor mill, but the fuchsia operating system coming out at Google?

[0:56:26.6] ND: I have, I have. I don't know a lot about it.

[0:56:28.7] JM: Nobody does.

[0:56:29.4] ND: I've read a few different threads. Yeah.

[0:56:32.2] JM: Nobody does, but it's interesting to think about like, "Oh, here we got a new operating system and here's a UI layer that runs anywhere," and it would be pretty cool to have a completely fresh start, completely new UIs, completely new operating system. Anyway, I'm getting us off topic. Anyway, coming back to application development today, so we've got nice cross-platform tools, whether you're talking about Flutter or React Native, I mean, we can pick Democrat or Republican. In any case, the world is improving. Things are getting quite easy on the frontend, things are getting quite plentiful on the back-end. What does this change about entrepreneurship, about product development?

[0:57:15.8] ND: Well, just like how the cloud in general just enabled more innovation, more experimentation and just more activity, I think we're going to see just more and more of that. We're going to see an acceleration of that. Because again, if you as a single developer can take your skillset and extend it to do all these other things, like why not? Once you have all these people that have taken their existing skill set and added all of these new things that they can do, they're going to be building more and more stuff. I think it's going to – you're going to see a lot of positive things happen as far as new ideas being fleshed out, where people before have an idea and not actually be able to do anything with it.

They may be able to actually get a prototype out now. Then maybe people will see it and they'll get funding. It's just more experimentation, more innovation and more opportunity to try things out that may, or may not work, because you have the ability now. I think if you're listening and you're a frontend developer and you think of yourself of a frontend developer and you're like, "Oh, should I learn – like which should I learn next?" I mean, seriously consider the idea of what maybe could be called a serverless developer, or serverless++ developer. I don't really have a name for it. A developer that can competently work with all these managed services to extend your capabilities to all these other things.

I think Amplify is a good place to start, but I wouldn't limit it to that. Look around, see what's out there and just think about this paradigm, because – and really notice as things progress in the

future, when you see these startups that are coming out with these microservices, we're going to see a lot. I think we're going to see a lot more of companies that are building services that will work in the same way that we're talking about.

[0:59:07.8] JM: You were running your own business with React Native training before this, and now you're at Amazon. Existentially, how do the two compare? I briefly worked at Amazon. That is a seriously entrepreneurial organization. In some sense, you get to be more entrepreneurial at Amazon than you would running your own thing, because you have access to these crazy resources. How does your lifestyle as an entrepreneur compare between running React Native and now working at Amazon?

[0:59:42.4] ND: This is my first time working at a large corporation company like this. It's been definitely a major change than just completely working on your own. My role there is actually a developer advocate. A developer advocate does a few things. First of all, we write blog posts explaining how to use a lot of the different tools that we have. We speak at conferences, talking about some of the new tooling that we have coming out and new services and stuff like that. Appear on podcasts of course, like the one we're listening to.

Really, the main bread and butter of my position is actually interacting with developers and talking to them and understanding the different issues that they're having and what they like, or what they don't like about just anything. Bringing that information back to the engineers and the project and product managers and fleshing out ways to improve things really in the developer ecosystem.

I work with AWS Mobile, which is the mobile team. We focus somewhat, but not a 100% on mobile. We're also focused on IoT and serverless stuff. I would say changing from my day-to-day, actually not a whole lot, believe it or not. I used to travel three weeks – two to three weeks out of the month. I work remotely now, so I do still travel but not as much. I travel maybe two weeks out of the month, and there aren't a lot of remote employees with AWS, but as a developer advocate, I'm actually on the road a lot anyway, so it doesn't matter as much. I'm doing a lot of videos and recordings and stuff, so it's good for me to have my own space. I think that's why it worked out for me.

I am, like as an individual just all, like a really hardworking person, or us, I work a lot of hours, I guess you'd say. I would say it hasn't been a lot of change. I don't feel like I have a lot of oversight as far as like, “Oh, you have to do this,” at my current position. It's more like, as long as we're out there bringing back good information from our customers and trying to improve the customer experience, and as long as I'm doing stuff, they're cool with it. It seems to be a somewhat more laid-back than I was expecting culture, because I haven't heard a lot of that from Amazon. Always hear that it might be a lot of work and stuff like that. It is a lot of work, but I feel it's not – it's a much better experience than I expected.

[1:02:09.3] JM: Well, that's so true. I used to work at Amazon very briefly, just eight months, and some people have gotten the perception from me saying that that I didn't like working at Amazon. I just want put it out there that was not at all the case. I loved Amazon and the organization really shaped how I think about the world and how I think about entrepreneurship.

I think if you're an entrepreneur, it's possible that if you don't know what you're doing and you don't have a reputation, or anything, you can't start your own business, it's quite possible that there's literally no better place to go, because it's like basic training for entrepreneurship. The reality is that the real world is difficult. It's not straightforward how to build your own business.

It takes a lot of work, but once you figure it out, you're like, “Oh, okay. You work hard and you rinse and repeat, and you you've got to work long hours,” and that's just the nature of reality. It doesn't surprise me that coming from the real world where you built your own business and you realized how much you had to do on your own, and then you went to Amazon you're like, “Oh, this actually is not at all like that New York Times article that I read.” Anyway, not to put words in your mouth, but I just – I really like Amazon and it was – I didn't feel I was strained as an individual. I know that there's a public perception that it does strain you.

[1:03:35.8] ND: Yeah, totally. Well, I'm curious what team were you working with there?

[1:03:39.5] JM: I was on a team, I think it was Amazon Global. I worked on logistics software.

[1:03:46.0] ND: Interesting. Yeah. I mean, like you mentioned, people don't realize how much of an entrepreneurial spirit there is within the company. I mean, this is an AWS of course. I haven't

worked in the Amazon, like the rest of Amazon, so I can't really speak to that. We have a lot of free rein to innovate and do new things, and as long as the idea is somewhat of a good idea, we have full ranger to pursue new things. That's how a lot of services in AWS and a lot of ideas in Amazon have actually come about. People are just thinking of something interesting and be like, “Hey, let's try this out.”

I mean, you would be surprised. These aren't ideas that come out of some thought leader at the top of the company. A lot of times, these are just engineers that'll throw together a white paper with an idea and it'll be pursued. Now you see millions of people use again. I think working in AWS and seeing how all the different services are connected is really beneficial, like you said. If you're looking to be in the future like an entrepreneur, seeing how things work together and how to work with these cloud services, it’s definitely a good place to be. I'm really enjoying it.

[1:04:58.6] JM: Yeah, and that culture. Man, that culture is something else.

[1:05:01.8] ND: Yeah, yeah. It's definitely – it fits me perfectly, so I like it a lot.

[1:05:07.7] JM: Okay. Well Nader, this has been great. I really enjoyed talking to you.

[1:05:10.9] ND: Yeah, man. Thanks Jeff for having me back on. I really enjoyed it. I hope that a lot of people got a lot out of this. If anyone ever wants to talk about this stuff, I'm readily available on Twitter. My Twitter handle is Dabit3, D-A-B-I-T-3, Github and Medium are the same handle, if everyone are reading in my stuff, or check out my code.

[END OF INTERVIEW]

With GoCD running on Kubernetes, you define your build workflow and let GoCD provision and scale your infrastructure on the fly. GoCD agents use Kubernetes to scale as needed. Check out gocd.org/sedaily and learn about how you can get started. GoCD was built with the learnings of the ThoughtWorks engineering team, who have talked about building the product in previous episodes of Software Engineering Daily, and it's great to see the continued progress on GoCD with the new Kubernetes integrations.

You can check it out for yourself at gocd.org/sedaily. Thank you so much to ThoughtWorks for being a long-time sponsor of Software Engineering Daily. We're proud to have ThoughtWorks and GoCD as sponsors of the show.

[END]