EPISODE 1392

[INTRODUCTION]

[00:00:00] KP: Writing your applications code is only half the battle. Getting it to run in your machine is a milestone, but it's far from your code running in a production environment. There are an increasing set of options application designers have for helping to manage deployment environments and CI/CD. Encore is a backend engine for the Go language. One of its core features is the ability to turn any function into an API endpoint with just an annotation.

André Eriksson is the founder of encore. In this episode, we discuss his experience as a developer and explore the features and functionality Encore has to offer.

[INTERVIEW]

[00:00:37] AS: Andre, welcome to Software Engineering Daily.

[00:00:40] AK: Thank you very much. It's a pleasure to be here.

[00:00:43] KP: Let's learn a little bit about you first. How did you get your start in software?

[00:00:47] AK: Oh, I started way back when I was 10. So it's like 22 years ago. And just built a bunch of websites. Eventually, as a young teenager, was very much into games. Started playing World of Warcraft. And you can make user interface modifications for the game. So I started doing that. Sharing it online. And it just took off. We had 300 million downloads over a couple of years. And a lot of the ideas I came up with as a 14-year-old ended up being incorporated into the game. I ended up consulting with Blizzard Entertainment on a bunch of stuff, and then moved into web development and eventually back in development. And then I kept doing a bunch of side projects growing up and eventually joined Spotify right after university.

[00:01:39] KP: What kind of things were you working on at Spotify?

[00:01:43] AK: Quite a bit of everything, I would say. I started fairly junior. But I had a lot of experience from all these side projects. So I ended up working across probably hundreds of different projects over the years. I was there for eight years. And towards my last four years, I was one of the most senior engineers of the company. So a lot of different projects across Spotify Premium, and royalties, and smart speakers, and very much all across the company. But in terms of discipline, I was very focused on backend engineering. That's always been a big passion of mine. But even outside of that, I've also done a lot of web development, and data, and machine learning, and mobile. So pretty all over the place, really.

[00:02:35] KP: Were you working on a Go stack at Spotify, or something else?

[00:02:40] AK: A little bit of Go. Spotify, primarily when I joined, was a Python shop. And then nowadays it’s primarily a Java shop. But there are some parts of it using Go around primarily Kubernetes infrastructure. And I built up the vast majority of the internal tooling for using Go as Spotify. So yes and no. But my day-to-day work was in other languages, I would say.

[00:03:09] KP: Well, if my understanding is correct, it's Encore that started as a side project around this time. Maybe let's get started with definitions. Tell the listeners what is Encore.

[00:03:20] AK: So we're calling it a backend development engine. And what we mean by that is it's like a game engine, but for backend development. So if you know anything about game engines, like Unity or Unreal Engine, the way they work is they make it really, really easy, and fun, and productive to build games. They let you focus on your game logic and how you want your game to work. And we're trying to bring the same concept over to backend development, where when you just want to build your product at the end of the day, which on the backend side is a bunch of API's and you're talking to databases and maybe microservices and that sort of thing. The reality is that you tend to spend a lot of time not really on building your product, but on a bunch of other things surrounding that. Lots of boilerplate. Lots of, I would say, unnecessary complexity, that isn't really part of your product, accidental complexity. And so that's really what we're trying to get away from, and create a developer experience that really lets you focus on building your product and nothing else.

[00:04:38] KP: So the analogy to a gaming engine is interesting. If I was going to go build a game, I would look for one with a good physics engine, maybe some scoring systems. Certainly has to do rendering and sound. There're all these services I get. What are some of the services that encore provides?

[00:04:53] AK: Yeah, exactly. So like you said, a game engine is all about how can we provide you with an opinionated way of expressing your game logic, and in exchange, you get all of these things for free? Like physics engine and 3D rendering engine and so on. And the analogy on the backend side are things like making it easy to set up API's, generating API documentation, integrating with third-party services, connecting to databases, provisioning databases, doing distributed tracing, or monitoring, or observability in general. All of these things really are the primitives we use when we're building backend applications. Integrating with the queues and Pub/Sub topics and all these sort of things. It's kind of been an evolving set of building blocks that you build backend services out of. And they really come into this world over the last 15 years since AWS launched and they introduced all these different services that today we kind of take for granted. But we forget that they've been very recent innovation, and our tooling haven't really caught up. So we're trying to take these new building blocks and creating a better experience for using them.

[00:06:31] KP: So in that regard, would I consider Encore a framework the way Java has Spring framework?

[00:06:38] AK: I think it's an interesting question. Yes and no. Yes, in the sense that it changes how you write code, because what our primary focus is, is to make you as a developer empower you and increase the impact you can have by providing much better tools. And as a developer, you want to spend your time writing code and creating amazing products. And so it stands to reason that to empower you and make you more impactful, we need to change and improve how you write code. So in that sense, it's like a framework, and that we provide a certain way of doing things. But on the other hand, it's very much not like a framework, or at least it's not like any other framework you've seen, because we also provide a lot of things that have to do with how your application integrates with the cloud. So things like automatically setting up infrastructure, setting up a test environment, setting up environments for each and every pull request, doing distributed tracing in production, gathering insights from production, and feeding

them back into the development process, and so on, and so on. So it's really about taking an end-to-end look at the developer process and figuring out how can we improve it in a very holistic sense?

[00:08:10] KP: Go is the primary language of Encore. Is there any other features? Or is it just a Go system?

[00:08:16] AK: We think of it as much like you're writing. When you choose to use Unity to build a game, you're using C#. And when you're using Unreal Engine, you write C++. So our experience is centered around Go for several reasons. We think that to make it a really productive experience, you really need to double down on a single language. And we've chosen Go just because it's built from the ground up for backend development. It's really performant. It's really scalable. It has really great support for highly concurrent backend services. But it also compiles really quickly, which is right in our wheelhouse in terms of thinking about developer productivity. And it has a really simple language. So a lot of what we do is based on static analysis, to understand how your application fits together. And that's really where a lot of our really unique value propositions come from is in this understanding. And so because Go is such a simple language, it's quite easy to do that static analysis that you otherwise wouldn't be able to do. So we're all focused on Go. It tends to be a really nice language for this sort of thing. And we have no immediate plans to change that. But maybe sometime in the future, we would add another language. But don't expect that we're going to add 5, 10 different languages. We're sticking with one and maybe two.

[00:09:54] KP: Well, let's put ourselves in the shoes of a listener who is a seasoned Go developer. They're coming to the table with a few options about you know which cloud they prefer and which database they like and they want to get a greenfield project up and running pretty quick. What is the Hello World path for Encore?

[00:10:12] AK: So we support all the major cloud providers, Amazon, Google and Microsoft Azure. And really, what you do is you can just download the product. It's all open source. So you install it in just one line in the terminal. You can brew install it. Then you create your application. It bootstraps a simple Git repository.

And in that, to define an API, it's really just a function. We have an annotation, which is how you communicate to Encore that this function should be exposed as an API. But really, conceptually, an API is just an API endpoint, or a REST API, or whatever you want to call it. Conceptually, it has a request and it has a response. And we're taking this, and that's what functions are too. So we can just model an API as a single function.

And what's really nice about this is it essentially just becomes plane Go. There is almost nothing that you have to learn. API's are functions. API calls to other services. Because we're modeling not just a single back end service, but we're modeling a whole distributed system is what you build with Encore, you can very easily define multiple services. So if you want to make an API call from one service to another, that is just a function call. So you get this very, very natural developer experience where everything just works with Go concepts that you already know. And the framework, if you want to call it that, kind of just blends into the background. And you're just writing plane Go with requests being function calls and the data that you're passing is just regular data structures.

And behind the scenes, we analyze all this code, and we handle the boring parts for you. We set up the HTTP handler. We generate documentation. We can generate a client for your frontend to use. And then we can set everything up in the cloud for you as well. So everything that you have to do is just focus on your application. What do you want your API to do? And that's it.

[00:12:34] KP: Very cool. So as I'm developing my application, I'll spend a lot of time focused on test-driven development. I'll probably mock that request and response. I'll get my API looking really good in my unit tests. And now I want to see it locally before I do any deployment. What do I do?

[00:12:52] AK: Well, so that's the beauty of Encore, is it's fully built for local development. So you run everything locally in your computer. It has great inbuilt support for testing. So you can both easily mock out calling other services. You can mock out databases. If you don't want to mock them out, it also supports really easily doing integration testing against an actual database. And you can just keep iterating on it locally. And every time we make a change, we'll live reload everything. We're working on a test-driven development mode, where every time you

save, we'll rerun all the tests for you and so on. So it has a very, very local first type experience. And it's only when you want to ship something that you really have to integrate with the cloud.

[00:13:49] KP: Live reload is something we all kind of have come to expect. But it has a lot of challenging engineering issues behind the scenes. In my experience developing Go, it was always very kind of test-driven. I never had a REPL with Go or anything like that. I don't know if that's baked in or if that's something you had to build. How do you manage that live reload feature that I'm used to from like, I guess, Webpack, and things like that in Node?

[00:14:15] AK: Yeah. So we don't yet provide a REPL in that sense. But what we do provide is we monitor the source code for changes. And every time it does, we recompile everything. And Go is really nice here because it has both really fast compile times. But it also has built-in incremental compiles. So usually every change is a couple milliseconds to recompile everything. But behind the scenes, we're running a local daemon, a background process that is not just your binary. And it's actually that daemon process that is running the listener, the HTTP listener, which means that when we recompile your application, we can just redirect that socket to talk to the new binary, which means that you never have to worry about losing requests if you're doing something in the background. You can just have something running against your backend nonstop. And as soon as you hit save, and your new version is recompiled and starts up, Encore will just immediately reroute request to that. So it becomes this really sort of seamless handover from the old version to the new version that works really nicely.

[00:15:38] KP: And what's the deployment story?

[00:15:40] AK: The easiest way to think about it is when it comes to deployment, what do you want to do is really two things. You want to make sure that if you're making changes to the infrastructure itself, and this is quite common, when you're building distributed systems, or you're building in a sort of cloud native style, where you're using all these cloud primitives, whether they're distributed queues, or Pub/Sub topics, or object storage, or whatnot.

What tends to happen is, as you're developing your application, you make changes to the infrastructure quite frequently both in terms of the products that you use, but also your internal infrastructure. You're spinning up new services. You're changing the contracts between services

and so on. So to deploy your application with Encore, you can either deploy through GitHub, where you integrate Encore's cloud platform with your GitHub account. And as soon as we detect a change, we will then take that commit on your main branch and orchestrate the deployment. Or you can push code directly to Encore's Cloud Platform. Either way, it works the same.

And the first thing we do is we really check all of your application for all of the infrastructure needs that you have. Have you introduced a new service? Do we need to set up a new database? Are you using a queue that wasn't there before? All of these things, we take care of first. And we make sure that we provision all of the necessary infrastructure. And by the way, all of this happens in your own cloud account. So you can just keep using AWS, or GCP, or Azure, whatever you're used to using. It all goes straight into your own cloud account.

And we set up all that infrastructure. And then once that's been set up and configured, then we actually take the new code and we deploy it. And we can deploy it in a microservices architecture if you like, where if you have a Kubernetes cluster we can – For each microservice, we'll deploy it separately to different containers and so on.

[00:18:01] KP: Interesting. Actually, I'm wondering if we could zoom in on some of that? Did I understand correctly that you're doing provisioning as well?

[00:18:09] AK: Yes. So all of this is based on us understanding how all of your backend architecture fits together. So all of the infrastructure needs, your databases, all of these different primitives. And that understanding enables us to provision all that infrastructure for you directly into your own cloud account.

[00:18:29] KP: So a common path I've taken in the past is let's use Terraform for this, right? I need queues, I need a Kafka instance, whatever it is, let me write up a big Terraform script, deploy it, and then put that in environment variables or something like that. Are you're able to, though, and somehow detect from the source code via Encore that I need a queue, and I need to stream, and you're building them for me? Is that correct?

[00:18:53] AK: Yes, exactly.

[00:18:54] KP: So how do you know they're there?

[00:18:57] AE: Yeah. So that's really where this framework aspect comes into play, right? Because software is an incredibly general idea. Like, in many ways, it's the most general-purpose thing we've come up with for expressing logic and behavior. So what our framework does is it provides a certain way of expressing that I want a cue. And it's a very Go native way of expressing that. So you just use the API's that Encore provide for expressing, like, “I want there to be a queue. It looks like this. These are the properties.” And we then analyze your code and we can pull all this out at compile time.

So in many ways, we're building a sort of language on top of Go that is all about distributed systems concepts, really. So the output of that is really a graph of all of your distributed systems, your services, and each and every services infrastructure needs. And that's what we then use to provision all of the infrastructure.

[00:20:09] KP: Well, I really like the idea of coupling provisioning more at the application level. I actually had some challenges with Terraform running it from my machine, and then how did I get it to somebody else? And they do have a cloud solution we started using. It seems like that's necessary to coordinate a lot of this. Can we talk more about what your cloud platform does?

[00:20:30] AK: Yeah. So it works quite similarly to Terraform’s cloud solution. The main difference is, as you say, in the development side, where we're trying to get you to not write your – When you use Terraform – I love Terraform, by the way. But you end up writing everything two or three times, where you write your Terraform definition, which is all of the infrastructure. And then you have your cloud provider, which knows about the current state of things. And then you have your source code, which has the expectation of all of this infrastructure. And we're just trying to unify all these together so you just have a single source of truth for what you need, which is the source code itself.

And what our cloud platform does, very similarly to Terraform, is we can set up all this for you in your own cloud account. You just integrate your Cloud account with our platform. And all of this is very straightforward. It depends a bit on the cloud provider. And then we can just track and

provision all the infrastructure for you and keep track of the lifecycle. So when your source code changes, we can detect that and figure out what are the changes that are needed in order to deploy the next release. And depending on the changes, we might ask you a few questions. Like if you're setting up a database, we need to know – Like if you set up a relational database, we need to know which region do you want this deployed if you have a multi-region application, because it can matter for latency reasons and so on. But in general, most of those things are only needed for production environments. And for test environments, and preview environments for a pull request, we can just take care of all of that for you.

[00:22:23] KP: Can you talk more about that process? You'd glanced across it earlier. And it sounded really like a major step that every PR could have an environment and be isolated for testing? What are some of the options there?

[00:22:35] AK: Yeah, exactly. So when it comes to – I think in many ways, software engineering, when you're building a cloud backend, or really a cloud-based application in general, we tend to not really collaborate between backend and frontend as much as we should. And by that, I mean that, oftentimes, we should be codeveloping at the same time. But what ends up happening in practice is it's a lot of hot potato, where a frontend developer might write the frontend and then they say, “Hey, backend developer, I need an API that looks like this.” And then they go off and do something else. And the backend developer takes over and does their thing. And then when they're done, they hand it back and say, “Oh, the backend is ready.” And it becomes this sort of very asynchronous collaboration. And usually, in every handover, you realize that you had misunderstood something.

So we're trying to create a much more collaborative experience, where it becomes much more interactive. And there are lots of different ways that we do that. We have a way of very easily live sharing your local development environment with a colleague of yours. And that means that you can just connect straight to their computer and you can live collaborate that way. The other way of doing it is with this preview environment. And what that means is, if you integrate encore with GitHub, then every pull request that you create, we will analyze the changes that you made. And then we will provision a very paired down environment specifically for that pull request. And we'll optimize it and bring everything into a single Docker container so that it's very,

very lightweight. Bring in all of the databases, run queues in memory or just on disk. Because it's not important that it be fully distributed and reliable and so on for a preview moment.

And then we spin that up. And every time you make a change, you will update that with the latest configuration. And this works really nicely with collaborating with frontend developers, because we can then enable you to disconnect Your frontend directly to the environment. And all you have to do is you have to just change environment name production to PR23. So it's a one line change to connect straight to a preview environment.

[00:25:17] KP: Very neat. Well, it sounds like there's a strong enterprise story. What about the broke developer with a side project who wants to maybe put something on Lambda so it's pay per use? How can they leverage Encore?

[00:25:30] AK: Yeah. All of this, I've grown up – I am that very person myself. So a lot of that is actually built in. By default, we provision everything in a serverless way. You don't have to do anything. We can set it up with Lambda. We can set it up with Google Cloud Run and so on. And we also provide serverless hosting for hobby use, which is completely for free. So you can use our cloud platform. You don't have to do anything. You don't have to connect any AWS account. And we'll take care of all the hosting for you. And all of this is you can do it for very generous use. I would expect that most hobby developers never have to move over to any sort of paid tier with their cloud provider. But it's also entirely open source. So you can take all of this stuff that we've been talking about and run it yourself if you'd like. We have lots of people who are doing that, and they really love the experience. So that's the idea.

[00:26:36] KP: Can you talk a little bit about adoption? That's probably a wide spectrum. But what are some examples or personas of the people who've picked up Encore and are running with it?

[00:26:45] AK: We have lots of different – I mean, it's really all over the place. Everything from like consumer, like software as a service, kind of like a content play for mental health, to developer tooling, and DevOps, to b2b, to like standard like b2c sort of – How should I put it? Like hardware sales. Quite all over the place. We're primarily focusing on building a developer experience for smaller companies right now, so startups in particular, because we believe that

the product – Like when you really want that extra acceleration is at the early stage. And that's what we're focusing on. But over time, we expect that this sort of developer experience will be invaluable to larger companies as well.

[00:27:43] KP: I'm thinking about the market. As you'd mentioned, you're very developer-focused. And I'm thinking about something like Heroku, which it's not like an apples to apples comparison. But it's also not apples to oranges. Where do you guys see yourself in relationship to a provider like Heroku?

[00:27:58] AK: I think what's been really interesting is we've had this sort of synonym between cloud providers and developer productivity. And I actually think it's quite strange, because cloud providers themselves, they actually sell infrastructure. And there's a real conflict of interest between trying to make you as impactful and successful as possible, versus selling you infrastructure that you might not need. So we're trying to take ourselves out of that conflict entirely by not doing hosting at all. And so I think we think of ourselves much more as a complement to the cloud, where the cloud providers really excel at providing extremely high quality infrastructure that's very, very reliable, and durable and solid, but maybe not the best developer experience.

So what we do is we work with the cloud providers, and we instead focus exclusively on the developer experience. And the only hosting that we do is, like we talked about earlier, with hobby developers. Where if you just want to do something at a small scale, you shouldn't have to worry about setting up an integration with a cloud provider and so on. So for everything, but the smallest of use cases, you should still use your cloud provider. But let us build and provide a much better developer experience.

[00:29:34] KP: Well, I that vision. And it's open source. You have the generous hobby layer and all that, but you need to fund the project. How are you guys going to make money?

[00:29:43] AK: Yeah. So for us, it all comes down to, if you're a larger company, you actually really see a lot of value from the product. Then we'll provide additional things on top that you can pay for. And that can be everything from additional concurrent builds, so that you can be productive at scale, to maybe more advanced collaboration features or testing features,

additional retention for tracing and so on. So there are a couple of those things that we think larger companies will want that aren't really critical when you're at a smaller scale.

[00:30:25] KP: I'm curious about what your aspirations are to exist within the GO ecosystem. I'm thinking of like the R statistical language. And it has these collection of libraries they call the tidyverse. And some would say like you don't even use R without using the tidyverse anymore. And sometimes, like maybe even spring, you could say has the same effect on Java that it's this prolific framework or way of coding. What will Encore look like in the pantheon of Go development a few years down the road?

[00:30:55] AK: I think, as a longtime Go developer, I really, really love how the language has been shaped over time. So we're actually trying to stay very close to the language itself. We're trying not to introduce a bunch of concepts. When you're writing Spring, like you say, in many ways, it feels like a different language altogether. You're not a Java developer. You're a spring developer.

But with Encore, we're trying to stick very, very close to the language. So that the framework blends into the background, where you're defining API's, they're just functions and API calls or function calls. We're using all of the standard library ecosystem for a lot of the integrations. So when you use a relational database, we provide a standard library interface for that. So we're really trying to stick very close to the Go philosophy in the standard library. And then just providing a few different API's for integrating with the cloud. Like if you need a queue, for example, that a package that we provide, because it turns out that's not available in the standard library. So we have to provide something either way. But the language experience itself, if you're familiar with Go, you can be productive with Encore in five minutes.

[00:32:20] KP: And what's next? Can you tell us a little bit about what's on the roadmap or exciting things you're currently working on?

[00:32:26] AK: Yeah. So the primary focus is really on extending the sort of cloud primitives that we support. So that's the short term. Including things like object storage, and caching, and additional types of databases, and so on. So that all of these building blocks that you're used to

using are just at your fingertips. And we'll support all them so that you can just focus on building your product.

And then slightly longer term, we're looking at much more interesting things around observability and debugging. And really, using this knowledge that we talked about that Encore has about your application to provide you with much more actionable and insightful ways of helping you do your job more effectively. So everything from like when you wake up in the middle of the night, because your backend service is on fire. And everything you get is an alert fired and the latency is too high. That's not very useful when you've just woken up.

So what we do is we can use all this understanding to gather all that data for you that you would normally spend an hour gleaning over graphs to figure out that, “Hey, there was a deployment two hours ago that caused this circuit breaker to trigger. And as a result, this downstream service started failing. And that's what caused your problem.” And we can do that by actually understanding all of these different pieces how your system fits together and all of your infrastructure where it is provisioned. So we're looking to use that knowledge to really help you in lots of different ways. That's maybe just a very specific example. But I think there are similar things we can do in almost every aspect of software engineering.

[00:34:30] KP: In that regard, do you see yourself potentially evolving into the observability space and application performance monitoring?

[00:34:36] AK: Yes, we already provide built-in distributed tracing, for example. And that works. You don't have to do anything. You can just write code like you would normally, and we'll instrument your application for you. And we even provide distributed tracing for local development out of the box. And we definitely want to do even more there. We're not going to be an observability provider ourselves. But we're looking to provide very easy integrations with all of the major ones. So that if you want to send all of your observability data to Grafana, or Datadog, or whatever, that's just click of a button. And we'll take care of all of the work of actually instrumenting and orchestrating that. But beyond just sending that observability data, we absolutely want to do much more with this sort of unique understanding that we have and provide you with more actionable things that you can do with that data to make you more effective.

[00:35:43] KP: So greenfield projects, you've got some nice demos and tours and things people can get up and running. What about the migration path, someone who has several years into a mature Go application? How can they get it integrated with Encore?

[00:35:58] AK: The way we think about it is both, really, for Go companies and other languages. The beauty of building distributed systems is that you don't need to use a single language for your whole backend. If you really like the experience that Encore provides, then it's very, very straightforward to use it for a single service that can run alongside all of your other services. So if you already have a backend running in, let's say, Google Cloud Platform, all you have to do is integrate Encore with your Cloud account and tell us where your application lives. And Encore can actually deploy the things you're building with encore alongside what you already have. And then they can integrate seamlessly with API's, like anything else, calling each other over the network. So that's really how we encourage companies to adopt it, is try it out for something on the side, like a small project. Wee what the experience is like. And if you like it, you can just deploy it alongside everything that you have and it will work seamlessly like any other service.

[00:37:15] KP: Well, if listeners want to dive right in, tell us the path to get started.

[00:37:19] AK: Yeah, so it's all open source. The easiest way to get started is go to encore.dev. And from there, it's just a few clicks to download it and set up a simple Hello World application and go from there.

[00:37:36] KP: Very cool. Well, André, thank you so much for taking the time to come on Software Engineering Daily.

[00:37:41] AK: Thank you very much for having me.

[END]