
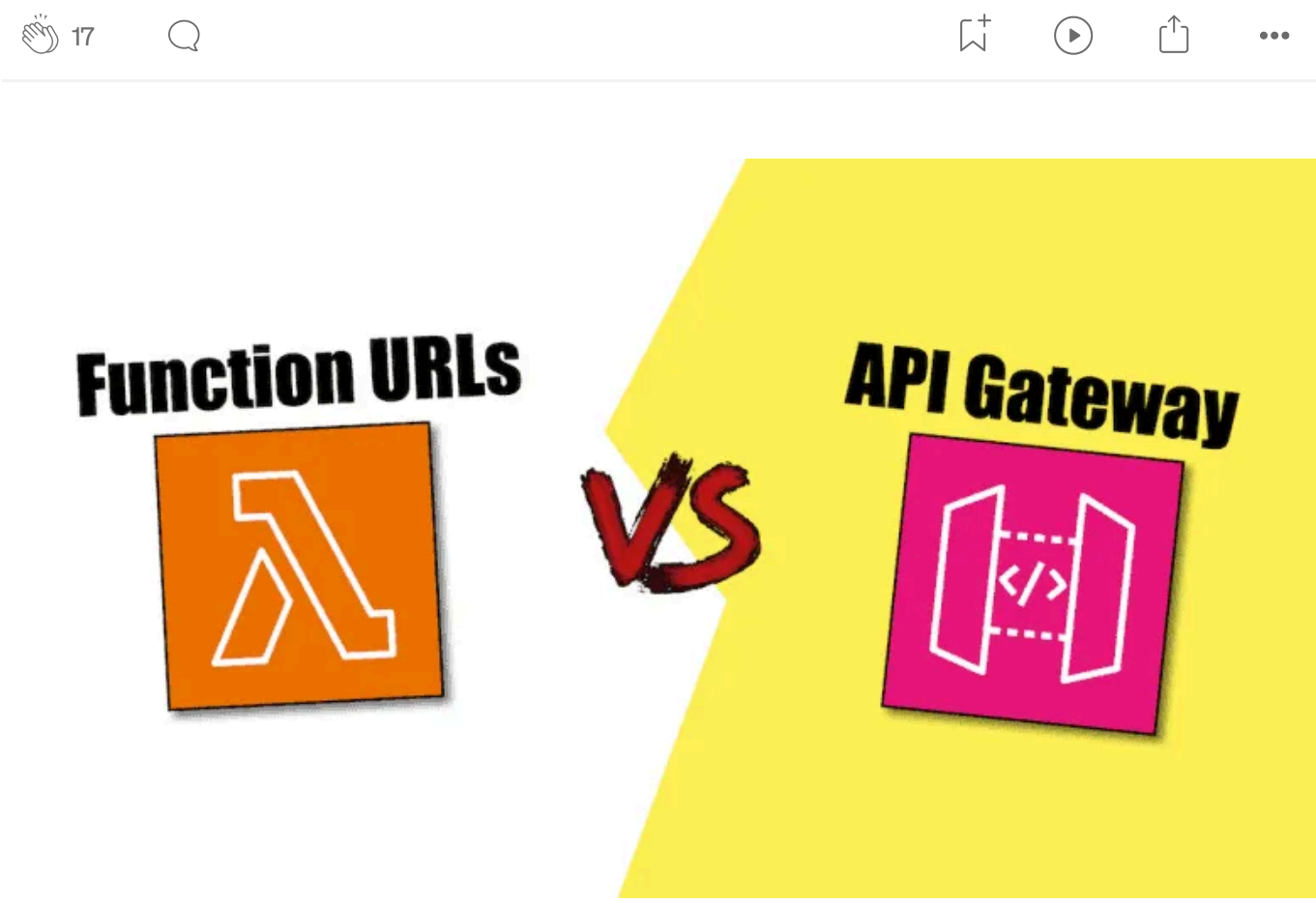


When to use API Gateway vs. Lambda Function URLs

 Yan Cui · [Follow](#)
Published in [theburningmonk.com](#) · 5 min read · Mar 3, 2024



“Lambdalith” is a monolithic approach to building serverless applications where a single Lambda function serves an entire API, instead of one function per endpoint.

It’s an increasingly popular approach.

It provides portability between Lambda functions and container applications. You can lift and shift an existing application into Lambda without rewriting it. You can use web frameworks you are already familiar with, and lean on the existing ecosystems of tools, ORMs and middleware. It also makes testing easier, because you can apply familiar testing methodologies.

Tools like the [AWS Lambda Web Adapter](#) [1] have made this approach more accessible. In addition, [Lambda Function URLs](#) [2] also work well with this pattern.

Which brings up a good question.

“In 2024, if you want to build a REST API using serverless technologies. Should you use Lambda Function URLs or API Gateway?”

Here are some trade-offs you should consider when making this decision.

Function URL pros

1. It works naturally with Lambdaliths.
2. No latency overhead from API Gateway.
3. No API Gateway-related costs.
4. It supports [response streaming](#) [3]. This is useful for returning large payloads in the HTTP response.
5. There are fewer things to configure.
6. Your API can run for up to 15 mins.

Function URL cons

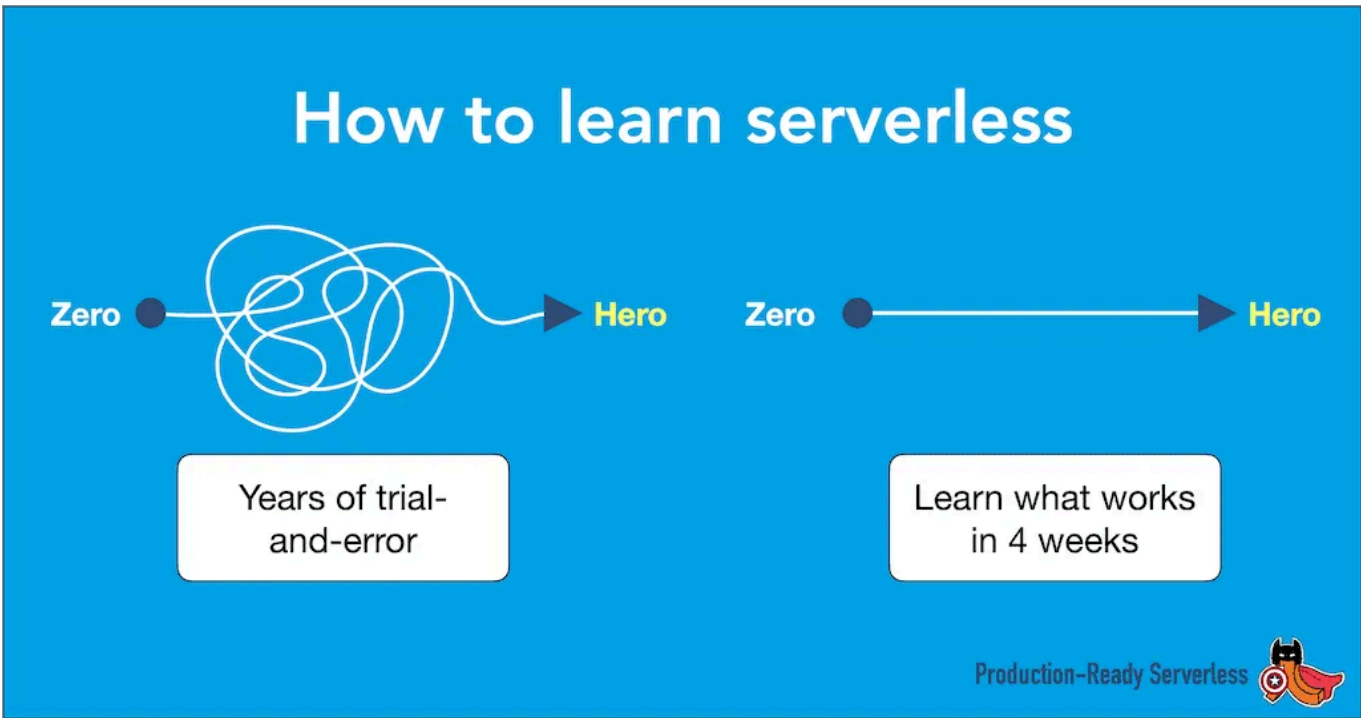
1. You have to use Lambdaliths and deal with all the shortcomings that come with that.
2. No per-endpoint metrics. It’s possible to make up for this by implementing custom metrics with [embedded metric format](#) (EMF) [4].
3. No direct integration with WAF. Although this is possible through CloudFront, the original function URL would still be unprotected.
4. It only supports AWS_IAM auth.
5. You cannot configure different auth methods per endpoint.

Where Function URL makes sense

If you want to (i.e. not forced into it by the choice to use Function URL) build a Lambdalith and you don’t need any of the additional features that API Gateway offers. Then Function URLs make sense? -it’s cheaper, faster and has fewer moving parts.

Similarly, if you need to return a large payload (> 10MB) or to run for more than 29s, then Function URL can also make sense. That is if you can’t refactor the client-server interaction.

Given the limited support for authentication & authorization, it’s not suitable for user-facing APIs. These APIs often require a Cognito authorizer or a custom Lambda authorizer.



This leaves function URLs as best suited for public APIs or internal APIs inside a microservices architecture.

By “internal API”, I refer to APIs that are used by other services but not by the frontend application directly. These APIs usually require AWS_IAM auth because the caller is another AWS resource — a Lambda function, an EC2 instance, an ECS task, etc.

API Gateway pros

1. It’s more flexible. It works with both Lambdaliths and the one function per endpoint approach.
2. It has direct integration with most AWS services. So in many cases, you don’t even need a Lambda function. If you’re curious about when and why you should consider this, check out [this video](#) [5] where I explain the rationale behind service proxies.
3. It can proxy requests to any HTTP APIs. This is very useful for integrating with 3rd party APIs.
4. It offers lots more features, including (but limited to) the following:
 - cognito authorizer
 - usage plans (great for SAAS applications that offer tiered pricing)
 - built-in request validation with request models
 - detailed per-endpoint metrics
 - mock endpoints (useful for endpoints that return static data)
 - request and response transformation (also useful for integrating with 3rd party APIs)

- and lots more...

API Gateway cons

1. Additional latency overhead.
2. Additional cost.
3. No response streaming.
4. 29s integration limit.
5. 10MB response limit.

When API Gateway makes sense

Given the vast array of features that API Gateway offers, it makes sense in most cases if you're OK with the additional cost that comes with the convenience.

The 29s and 10MB response limits can be problematic in some cases.

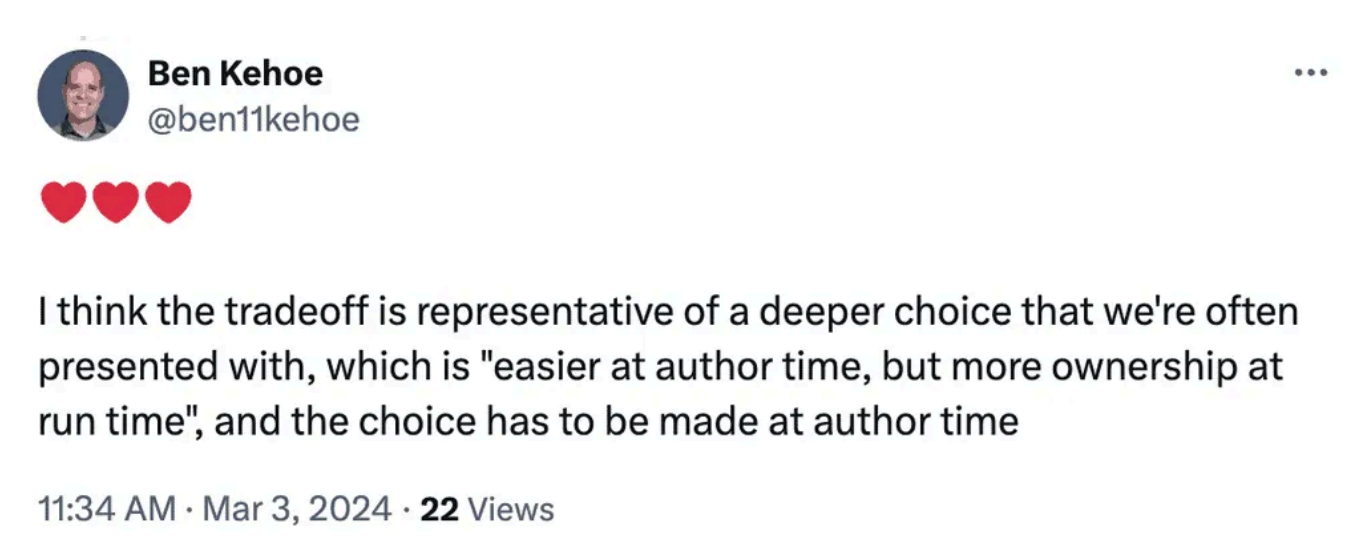
But they can be mitigated with patterns such as [Decoupled Invocations](#) [6] and [S3 presigned URLs](#) [7]. However, these workarounds require you to refactor the client-server interaction, so they are not always viable.

Summary

Because of its flexibility, I prefer API Gateway over Function URLs or ALBs. But Function URL is a useful tool, especially when cost and performance are your primary concerns.

It's also an important lift-and-shift option for people migrating from an existing EC2 or container-based application. It lets them enjoy the benefits of Lambda without rewriting their application.

Finally, as Ben eloquently [puts it](#), this tradeoff represents a deeper choice we often face.

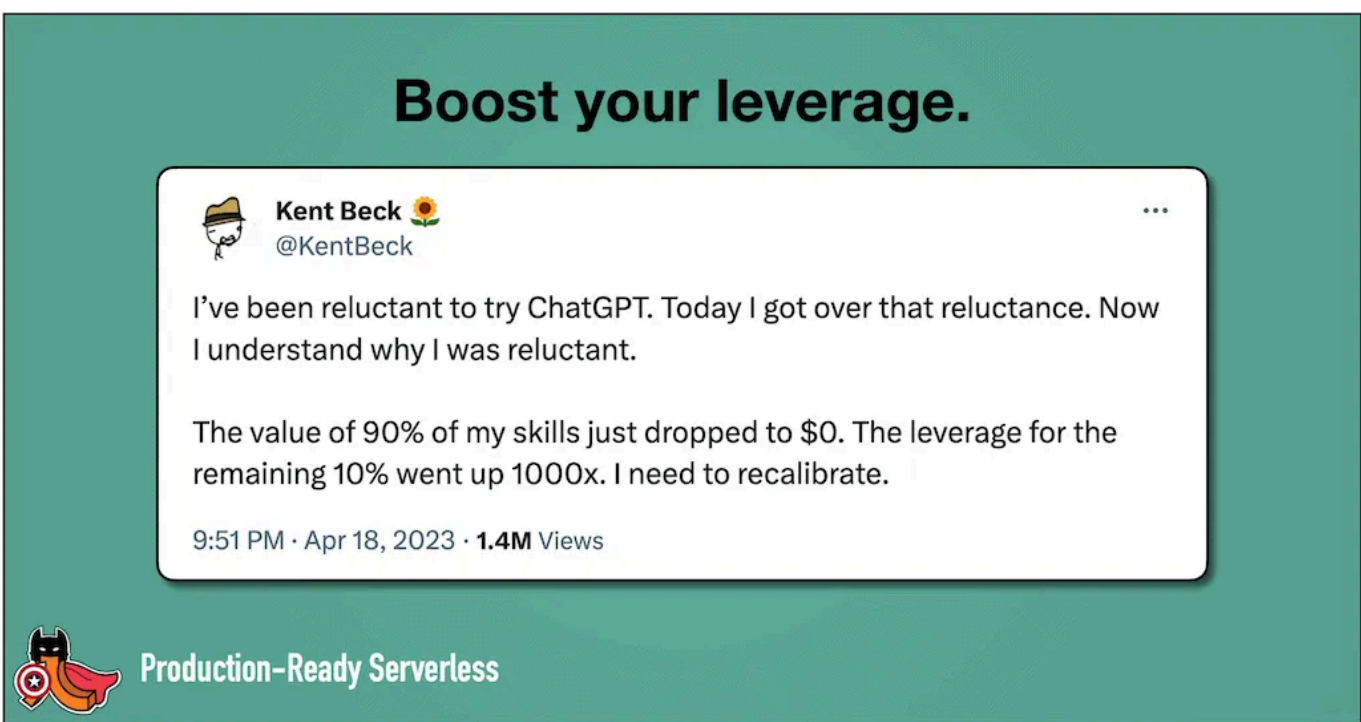


What's easier at author time might mean more ownership at runtime. For example, we don't get per-endpoint metrics from function URLs so we have to bring that capability to the table ourselves and be responsible for them.

Something for you to think about ;-)

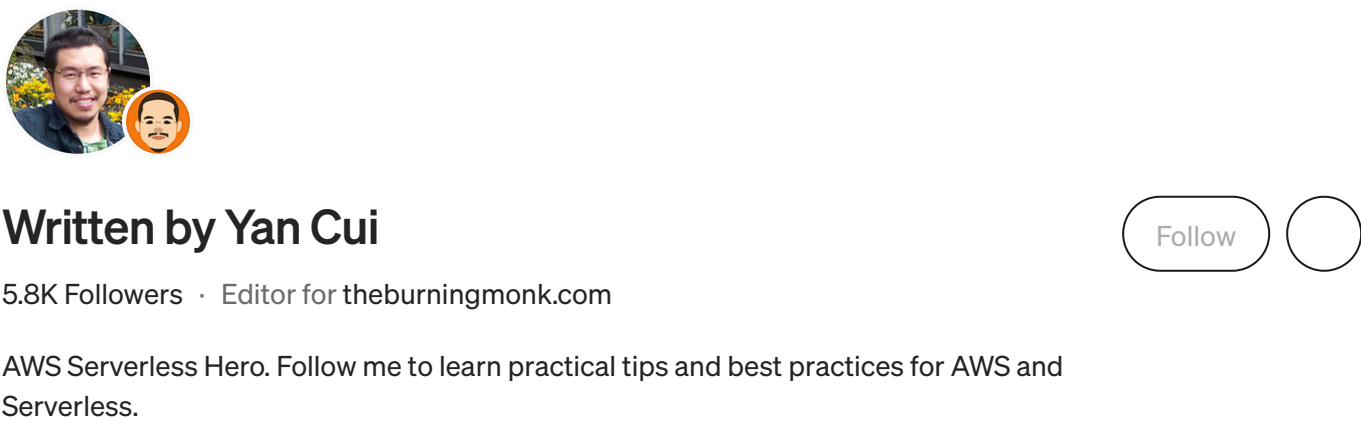
Links

- [1] [AWS Lambda Web Adapter](#)
- [2] [AWS Lambda: function URL is live!](#)
- [3] [Introducing AWS Lambda response streaming](#)
- [4] [Embedded metric format specification](#)
- [5] [API Gateway: Why you should use Service Proxies](#)
- [6] [How to use the Decoupled Invocation pattern with AWS Lambda and serverless](#)
- [7] [Hit the 6MB Lambda payload limit? Here's what you can do](#)

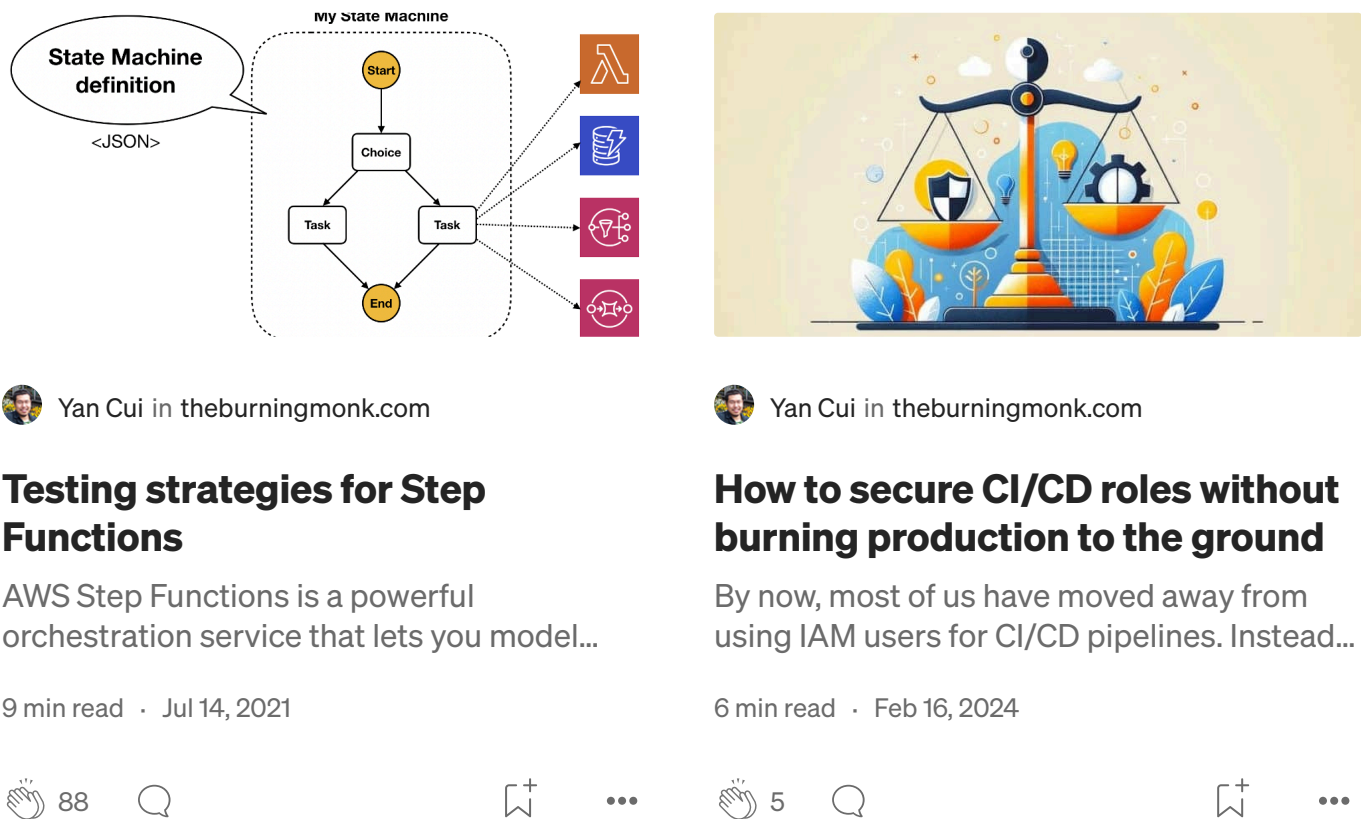
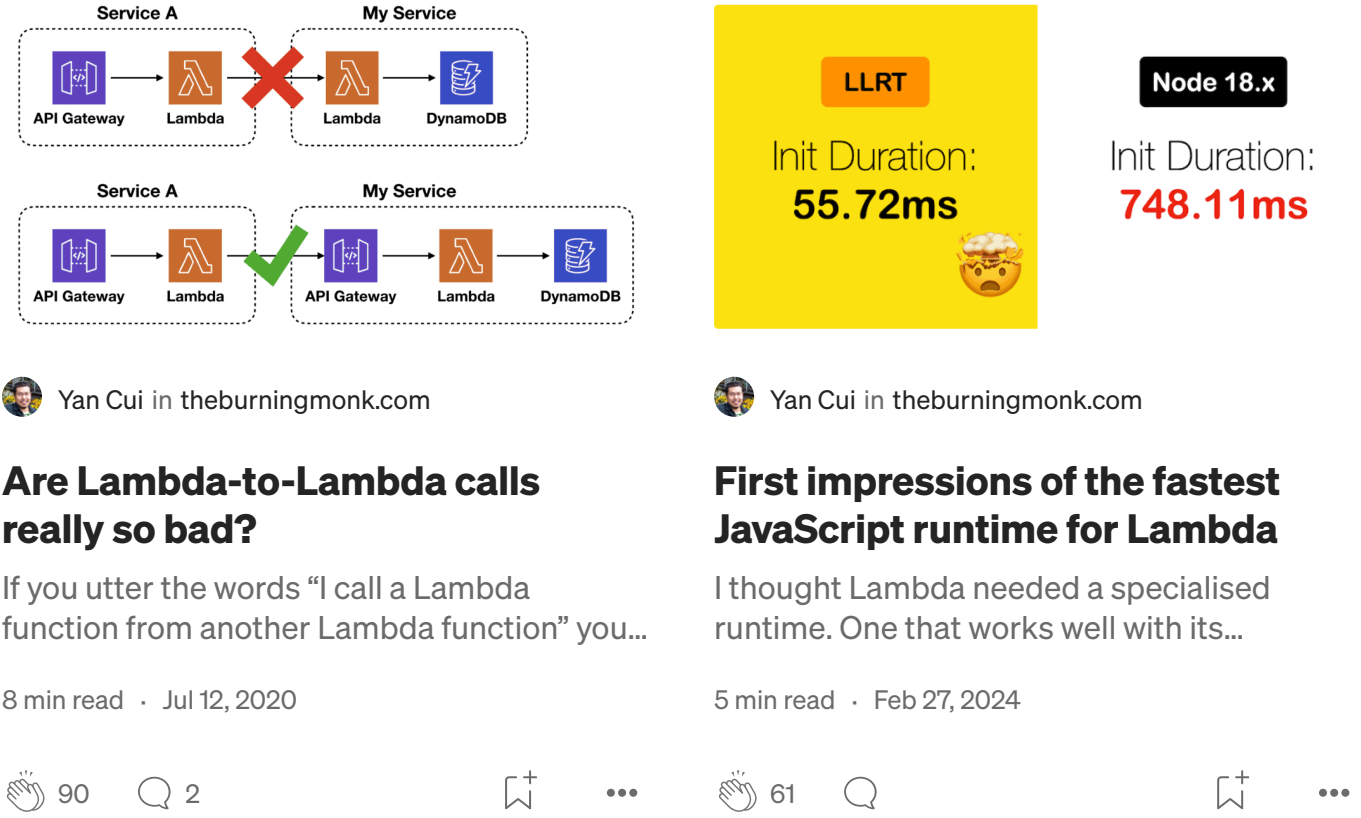


Originally published at <https://theburningmonk.com> on March 3, 2024.

AWS AWS Lambda Api Gateway Cloud Cloud Computing



More from Yan Cui and theburningmonk.com



See all from Yan Cui See all from theburningmonk.com