There Is More to Serverless Than AWS Lambda

Table of Contents

Intro

You've put in the research and decided that AWS Lambda is the right choice to power your website or app. You've chosen wisely as Lambda offers a number of benefits that make it an attractive option for many businesses. Before you can deploy though, there are a couple more decisions that you'll need to make: how you'll connect your front end to your back end and where you'll store all of your data. Since Lambda functions are all stateless, your code can access stateful data via other web services — Amazon S3 and DynamoDB among others. Those stateless functions enable Lambda to rapidly scale as needed by launching copies of the function that match the rate of incoming events.

So, how do you get started?

Step 1: Connect Your Front End To Your Back End

Step 1: Connect Your Front End To Your Back End

If your needs include any of the following:

- Authorization and authentication for your API
- Request validation before the integration request
- Rate throttling for better throughput
- SDK (software development kit) generation
- Running multiple API versions simultaneously
- Flexible security controls with Lambda authorizer
- Restful API endpoints to that respond in the expected language target
- Real-time two-way communication applications with WebSocket APIs

Then API Gateway is for you.

Step 1: Connect Your Front End To Your Back End

API Gateway is built for creating, publishing, monitoring, maintaining and securing REST APIs and web services at any scale. It's the catalyst for routing calls from clients to their appropriate microservices and allows any two applications to communicate without interruption.

Compelling customer service, communicating with partners and suppliers, and connecting internal applications are all among APIGateway's top features and benefits. Its core mission is to facilitate API requests, thus helping enterprises increase their revenue, improve developer productivity, and reduce the risk of downtime or slow performance.

Where API Gateway particularly comes in handy is by allowing users to maintain a single API domain with multiple end-point versions, instead of supporting a cluster of interface systems.

The API Gateway Alternative: Application Load Balancer

If you don't need all the features that API Gateway offers, a viable substitute to connect your front end user experience to your backend business logic is AWS Lambda via application Load Balancer (ALB) routing. ALB is not only a viable alternative; it's a much less expensive one as well. Something to keep in mind is that you're not locked into one choice. Starting out, if you find that ALB is a better fit for you, you can always start out using that and switch to API Gateway as your business needs scale over time. Just because you don't have the above needs today, doesn't mean that will always be the case. In fact, you're almost certain to need additional functionality as you grow.

API Gateway Pricing

Once you've decided API Gateway is for you, you'll pay $3.50 per one million API requests for the first 333 million requests each month. Confused? Don't worry, you're not alone. Pay-as-you-go pricing structures are often very complex and misleading. It's easy to look at that pricing equation and think that $3.50 is cheap. However, if you have an API that averages 450 requests per second, that's pushing 39 million requests every day. In turn, that comes out to $136 per day, or over $4,000 a month, for API Gateway — a fairly daunting amount if you're not expecting it.

This type of pricing led AWS to announce at the 2018 "re:Invent" conference that ALB would also support routing requests to Lambda functions. Notably, ALBs are priced out by the hour and usually cost $15-$20 per month on an hourly rate, depending on the specifications, stream, and nature of the requests.

API Gateway Pricing

Diving deeper into the cost of ALB, we can look at Load Balancer Cost Units (LCUs). These complex set of structures take into account the number of new and active connections each second, as well as the bytes processed by the ALB, and the number of routing rules it needs to evaluate. You'll need to figure out what your application workload is before determining which is right for you but generally, if you just need Lambda routing ALB is about eight times less than API Gateway.

Choosing between API Gateway and ALB ultimately comes down to what specific needs your company has and how much you're willing to pay to address those needs. A more robust solution only works when the log of requests calls for it. In other words, don't bring a power drill to your application if all you need is a small screwdriver. Evaluate your choices and make the right one for you.

Step 2: How Will You Persist Your Data?

Step 2: How Will You Persist Your Data?

While storage used to typically refer to a database, today's world is different. Now, you can use file storage, streams, queues, and cache, in addition to various databases. When choosing a database, you get to pick between structures such as NoSQL, Relational, Graph, Blockchain, and more. Selecting the best solution for your data all comes down to whether or not your storage needs are temporary or permanent. In most cases, you'll need a little bit of both.

One of AWS Lambda's biggest advantages is its ability to interact with the rest of the AWS suite of services. In most cases, you'll spend some time using AWS IAM (Identity and Access Management) getting it to operate properly. Ultimately, this approach is easier than interacting with other locations. Additionally, most services have the ability to trigger AWS Lambda functions as a form of a state change. That enables builders to construct complex data flows and interactions.

But, how do you make up your mind?

File Storage

If you are building something that requires storage and the ability to retrieve file objects, Amazon S3 is worth investigating. One of its core benefits is that customers can access it directly, rather than going through a pre-set gateway. When choosing a storage solution, don't forget about security. It's worth it to spend time figuring out how to allow only authorized objects into your data stream. Our advice: generate a specialized URL and validate frequent access control.

This approach can be used to both read and write objects directly to Amazon S3. It's also quite common to utilize Amazon S3 triggers to invoke a Lambda function that protects the metadata from the AWS DynamoDB objects when storing such objects for queries.

Storage Synchronization with Pub-Sub Messaging

Transferring and replicating data across platforms is key to providing seamless microservice solutions. With pub-sub messaging middleware within an AWS serverless function, you can make use of the AWS SNS (Simple notification service). This keeps microservices interlinked with their assigned SNS topics in their role as publisher, subscriber, or both. This feature can be deployed alongside triggers or streams like AWS DynamoDB, Aurora, Kinesis, and more.

For example, let's look at a scenario where the User Profile Data is shared between two microservices — one for User Profile and the other for User Analytics. User Profile service uses AWS DynamoDB and User Analytics service uses AWS Relatable Database Service (RDS) for their respective internal storage. Since they are interlinked, the User Analytics service will receive any update in the user's name that the User Profile registers. This can be handled by having an SNS topic for User Data Change where User Analytic Microservice internally acts as a subscriber while User Profile Microservice Data change acts as the publisher.

Processing and Data Storage

Another useful tool in the AWS services bundle is Kinesis Streams which can be used to temporarily store large number of events and route them to the appropriate AWS service. The advantages are once again in the operating cost and efficiency since the computational resources are utilized to process these events ephemerally.

Benefits and Drawbacks of Using Serverless Services

If we look at using a serverless solution as a whole, there are a number of benefits it offers. Conversely, it's not free of drawbacks as well.

Benefits

Scale up to millions of requests

- Elastic scale and high throughput
- Independent requests don't interact with each other and demand smaller granules of computation
- Greater focus on business logic

Smaller attack surface

- Cloud vendors are responsible for more of the operating model
- Minimized human error

Cost is driven by the # of requests

- Only pay for what you use
- No payments by the hour

Easy to deploy and package the change

- Eliminates concerns about the runtime
- Compatible with the latest vendor and open source deployment products

Drawbacks

Lambda Cold Starts

- AWS has to create containers for your function and if it hasn't run in a while it can take longer for the first time. Recent Lambda VPC enhancements have provided a way to nearly eliminate this problem so make sure to plan out your architecture ahead of time.

Vendor Lock-in

- Some AWS features might not be available in other public clouds

Optimizations are limited to code

- Can no longer buy a bigger server when up against the wall

Capped execution duration

- Functions are ephemeral and stateless

More lines of code

- Maintenance efforts need to evolve to match new paradigm

A Final Word

At the end of the day, choosing how you'll connect your front end to your back end and where you'll store all of your data all comes down to your specific needs. If you examine them upfront and do your homework as to how each will affect your bottom line and efficiency, you'll be in good shape. Plus, as we've already noted, you aren't locked into anything. If your chosen methods end up not working out for you, you can — and should — switch to something that does. Furthermore, even if your method is working, there's no harm in switching things up from time to time. Experiment with everything that's out there and find what works best for you. Finally, take the time to forecast your costs and for goodness sakes pay attention to your bill. Don't be another sad case of assuming the cloud is cheap only to find out you spent beyond your means.

Digital Ideas **Accelerated**

There Is More to Serverless Than AWS Lambda

Twitter: @MentorMate | www.mentormate.com