Lambda - Serverless Architecture

lean about Lamdba our new best friend and how you can make your architecture cheaper and easy to use.

The evolution of Lambda or the timeline looks something like this On-Prem DataCenter –> IAAS –> PaaS –> containerisation /Docker –> Serverless

Lambda is a compute services where you can upload your code and create a Lamdba function AWS Lambda takes care of provisioning and managing the servers that you use to run the code. You do not have to worry about operating systems, patching, scaling etc.

It is essentially described as an event-driven compute service where AWS Lambda runs your code in response to events. These events could be changes to the data in an Amazon S3 bucket or an Amazon Dynamo DB table.

Lambda events can trigger other Lamdba events or call other AWS services like SQS or SNS.

Why use Lambda?

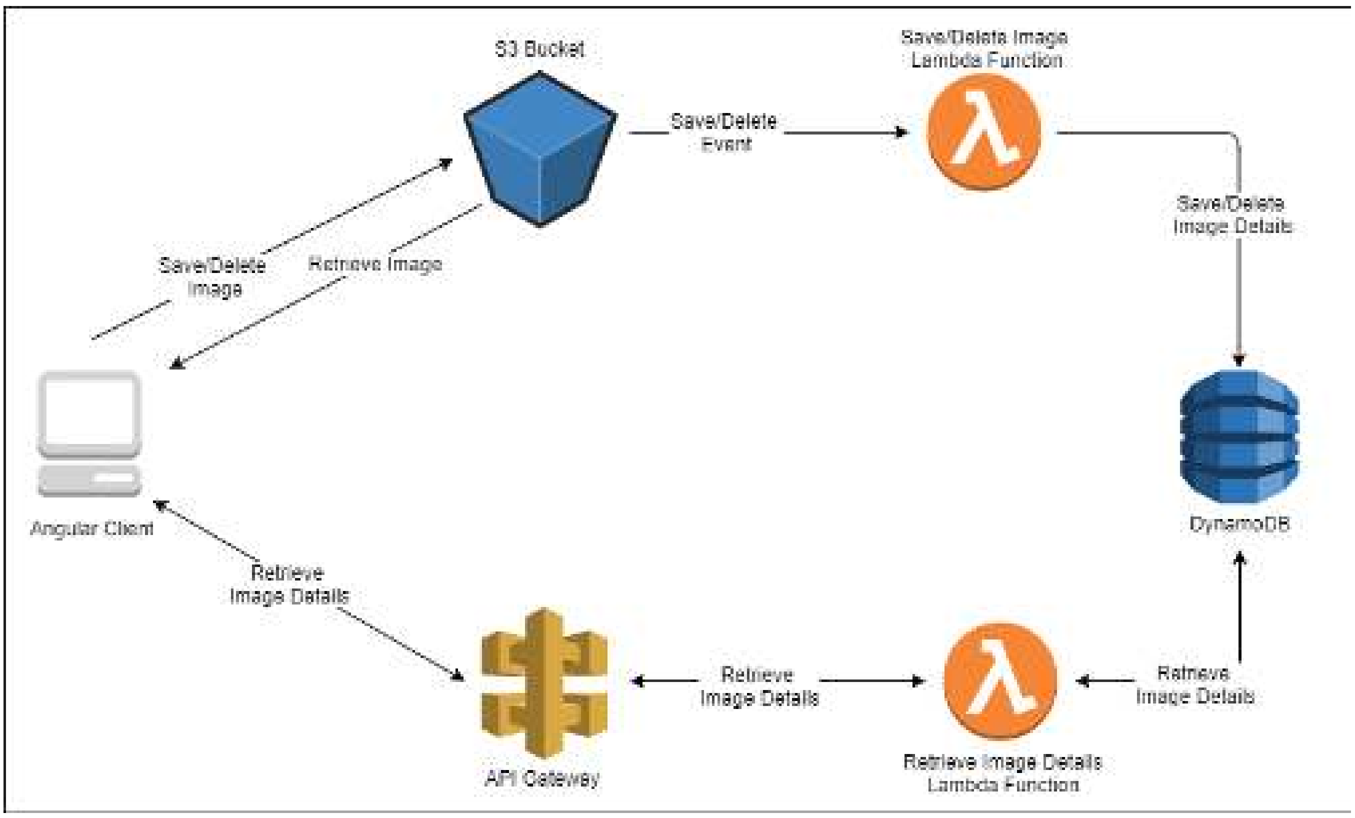
1. The Lambda runtime is fully managed by AWS. Once a function is uploaded and configured, Lambda is responsible for managing the resources required to run the code.
2. Developers are free from the traditional overhead of configuring and maintaining server instances.
3. Lambda will immediately scale to meet spikes in demand.
4. Lambda is cost effective as you only pay for the computational resources used. This is, of course, true for other AWS compute services, but the cost model for Lambda is more granular than EC2 for example, with resources being charged per 100 milliseconds.
5. Lambdas event-driven model means you can integrate nicely with a range of AWS services, but still ensure loose coupling.

Very low cost the first 1 million request are free. 0.20 per 1 million requests

Very low cost the first 1 million request are free. 0.20 per 1 million requests there after!!

Duration the execution time of your code and memory as well but it's almost insignificant in most cases.

Lamdba Architecture Example



Lamdba Architecture Example

Let me describe the various components in the above Architecture:-

1. **Angular Client** allows the user to add and delete images from S3. It also calls an API Gateway endpoint to retrieve details for all images uploaded.
2. **Save/Delete Lambda Function** will handle image upload and delete events from S3. It will be invoked by S3 when a new image is uploaded - the function will use the supplied event data to call back to S3 and retrieve the image. The image details will be saved to DynamoDB. As an image is deleted the function will use the supplied event data to identify the deleted image and remove the associated details from DynamoDB.
3. **Retrieve Image Details Lambda Function** will retrieve image details from DynamoDB and return JSON result.
4. **Dynamo DB** a single Dynamo DB table will be used to persist image

details. All interactions with DynamoDB will happen via the Lambda functions.

5. **API Gateway** an API Gateway endpoint will be used as a bridge to the Retrieve Image Details Lambda function, from the web app.

That should give you a good overview of how Lambda functions work. Also, note that some cloud providers call this as “Function as a Service” *FaaS*.