LAMBDA

- AWS lambda is a compute service that lets you run the code without provisioning or managing servers.
- With AWS lambda, you can run code for virtually any type of application or backend service- all with zero administration.
- AWS lambda manages all the administration it manages:
    - Provisioning and capacity of the compute fleet that offers a balance of memory, CPU, network and other resources.
    - Server and O.S maintenance
    - High availability and automatic scaling
    - Monitoring fleet health
    - Applying security patches
    - Deploying your code
    - Monitoring and logging your lambda functions.
    - AWS lambda runs your code on a high-availability compute infrastructure.
- AWS lambda runs your code on a high availability compute infrastructure.
- AWS lambda executes your code only when needed and scales automatically form a few requests per day to thousands per seconds.
- You pay only for the compute time, you consume no charge when your code is not running.
- All you need to do is supply your code in the form of one or more lambda functions to AWS lambda, in one of the languages that AWS supports (currently Node.js, java, powershell, C#, Ruby, Python and Go) and the service can run the code on your behalf.
- Typically the lifecycle for an AWS lambda based application includes authoring code, deploying code to AWS lambda and then monitoring and troubleshooting.
- This is in exchange for flexibility, which means you cannot log into compute instances or customize the operating system of language runtime.
- If you do want to manage your own compute, you can use EC2 of Elastic Beanstalk.

How Lambda Works:

- First you upload your code to lambda in one or more lambda function.
- AWS lambda will then execute the code in your behalf.
- After the code is invoked, lambda automatically take care of provisioning and managing the required servers.

Difference between AWS Lambda and EC2:

AWS Lambda:

- AWS lambda is a platform-as-a-service.
- It supports only limited languages like Node.js, python, java, C#, Ruby, Go and powershell.
- Write your code and push the code into AWS lambda.
- You cannot log into compute instances, choose customized O.S or language platform.

AWS EC2:

- AWS EC2 is an infrastructure—as-a-service.
- No environment restrictions, you can run any code or language.
- For the first time in EC2, you have to choose the O.S and install all the software required and then push your code in EC2.
- You can select variety of O.S, instance types, network and security patches, RAM and CPU etc.

Important Terms Used in Lambda:

a. **Function:** a function is a resource that you can invoke to run your code in AWS lambda. A function has code that processes events and a runtime that passes request and responses between lambda and the function code.

b. **Runtime:** lambda runtimes allows functions in different languages to run in the same base execution environment. The runtime sits in between the lambda service and your function code relying invocation events, context information and responses between the two.

c. **Event:** it is a JSON formatted document that contains data for a function to process.

d. **Event Source/ Trigger:** an AWS service such as Amazon SNS or a custom service that triggers your function and executes its logic.

e. **Downstream Resource:** an AWS service, such as Dynamo DB tables or S3 buckets that your lambda function call once it is triggered.

f. **Concurrency:** number of request that your function is serving in any given time.

When Lambda Triggers:

- You can use AWS lambda to run your code in response to :
    - Events such as changes to data in an Amazon S3 or an Amazon Dynamo DB table.
    - To run your code in response to HTTP request using Amazon API gateway.
    - With the capabilities you can use lambda to easily build data processing triggers for AWS services like Amazon S3, and Amazon Dynamo DB process streaming data stored in kinesis or create your own backend that operates at AWS scale, performance and security.

Example of S3:

- The user create an object in a bucket.
- Amazon S3 invokes your lambda functions using the permission provided by the execution role.
- Amazon S3 knows which lambda function to invoke based on the event source mapping that is stored in the bucket notification configuration.

AWS Lambda Function Configuration:

- A lambda function consists of code and any associated dependencies.
- In addition a lambda function also has configuration information associated with it.
- Initially you specify the configuration information when you create a lambda function.
- Lambda provides an API for you to update some of the configuration data.

Lambda function configuration information includes the following key elements:

- Compute resources that you need you only specify the amount of memory you want to allocate from your lambda function.
- AWS lambda allocates CPU power proportional to the memory by using the same ratio as a general purpose Amazon EC2 instance type, such as an M3 type.
- You can update the configuration and request additional memory in 64mb increments from 128mb to 3008mb.
- Functions larger than 1536mb are allocated multiple CPU threads.

Maximum Execution Timeout:

- You pay for the AWS resources that are used to run your lambda function.
- To prevent your lambda function from running indefinitely, you specify a timeout.
- When the specified timeout is reached, AWS lambda terminates your lambda function.
- Default is 3sec and maximum is 9000sec (15min).

- **IAM ROLE:** this is the role that AWS lambda assume when it executes the lambda function on your behalf.

AWS Lambda Function- Services it can access:

- Lambda function can access:
  - AWS services and non-AWS services.
  - AWS services running in AWS VPC (e.g: redshift, elastic cache, RDS instance)
  - Non-AWS services running on instances in an AWS VPC.
  - AWS lambda run your function code securely with in a VPC by default.
  - However to enable your lambda function to access resources inside your private VPC you must provide additional VPC specific configuration information that includes VPC subnet ID and Security group IDs

Different way to invoke Lambda Function:

1. Synchronous invoke (push)
2. Asynchronous invoke (event)
3. Poll-based invoke (pull based)

1. Synchronous Invoke:

- Synchronous invoke are the most straight forward way to invoke your lambda function. In this model your functions execute immediately when you perform the lambda invoke API call.
- Invocation flag specifies a value of "request-response".
- You wait for the function to process the event and return a response.

Here is a list of services that invoke lambda function synchronously:

- Elastic Load Balancer
- Amazon Cognito
- Cloud front
- API Gateway
- Amazon Lex
- Kinesis data firehose

2. **Asynchronous Invoke:**

   - For asynchronous invocation, lambda places the event in a queue and returns a success response without additional information.
   - Lambda queues the event for processing and returns a response immediately.
   - You can configure lambda to send an invocation record to another service like SQS, SNS, lambda and eventbridge.

Here is a list of services that invoke lambda function asynchronously:

- Amazon S3
- Amazon SNS
- SES
- Cloud watch logs
- Cloud watch events
- AWS code commit
- AWS config

3. **Poll-Based Invoke:**
   The invocation model is designed to allow you to integrate with AWS stream and queue based service with no code or server management lambda will poll the following service on your behalf, retrieve records and invoke your function. The following are supported service:
   - Amazon Kinesis
   - Amazon SQS
   - Amazon Dynamo DB Streams

WRITTEN BY:

NAGARJUNA HOTA