# Lambda

29 June 2023 23:12

## What is AWS Lambda ?

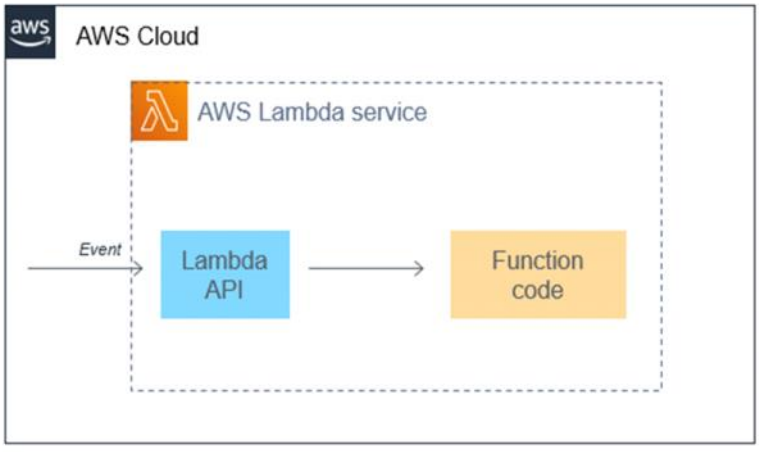
AWS Lambda is a compute service that Runs code without thinking any servers or underlying services. It is a Serverless function that you only responsible for your actual code.

*AWS Lambda is an **event-driven**, s**erverless computing** platform provided by Amazon. It is a computing service that runs code in response to events and automatically manages the computing resources required by that code. It was introduced in November 2014.[1]*
We can trigger Lambda from over 200 AWS services so that means Lambda has incredible natural integrations with AWS resources and also able to integrate SAAS applications.

Run code without provisioning or managing infrastructure. Simply write and upload code as a .zip file or container image. Automatically respond to code execution requests at any scale, from a dozen events per day to hundreds of thousands per second.
Save costs by paying only for the compute time you use by per-millisecond instead of provisioning infrastructure upfront for peak capacity. we call this pay-as-you-go. This is common feature for Serverless services, but for AWS Lambda, AWS charge you as per-millisecond level, its really fair cost when it comes to pay as-you-go model.
Optimize code execution time and performance with the right function memory size. Respond to high demand in double-digit milliseconds with Provisioned Concurrency.

## How does AWS Lambda work?

We said that AWS Lambda fits into the event-driven architectures. So, As you can see the image AWS Lambda triggers by event and executes the function code.

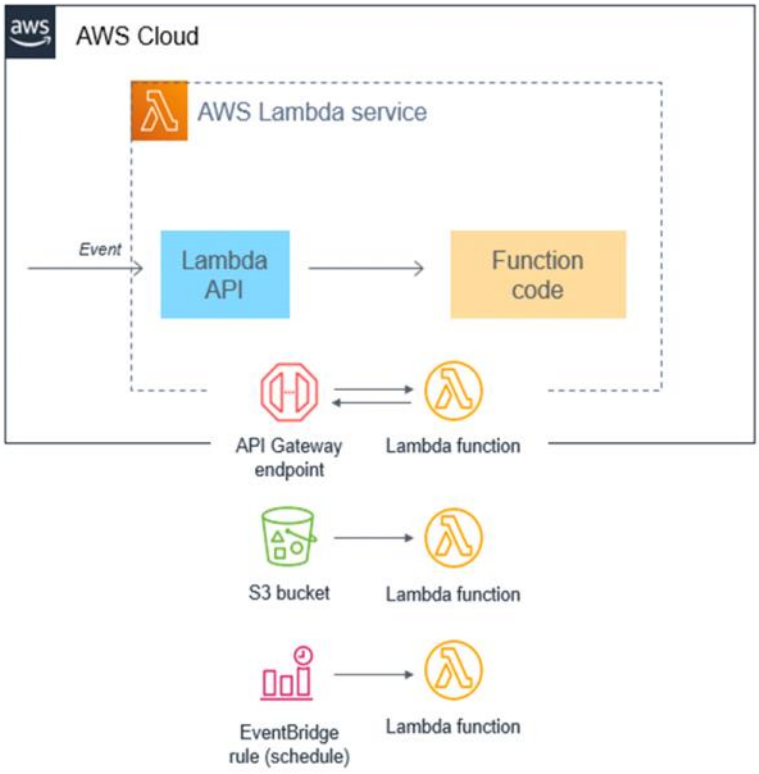



Each Lambda function runs in its own container. You can think every lambda function as a standalone docker containers. When a function is created, Lambda packages it into a new container and then executes that container on a multi-region cloud clusters of servers managed by AWS.
Before the functions start running, each function's container is allocated its necessary RAM and CPU capacity that parameters are configurable in aws lambda.
When the functions finish running, there is a calculation; the allocated RAM and the function execution time is multiplied and calculated charged cost to the customer. So that means customers charged based on the allocated memory and the amount of execution time the function finished.
AWS Lambda's entire infrastructure layer is managed by AWS. Clients don't get a lot of visibility into how the system is running, but they're also don't need to know about underlying machines, network contention, etc. they don't have to worry about things like; AWS handles this itself.
Using AWS Lambda can save you time on operational tasks because the service is fully managed.
When there is no infrastructure to maintain, you can spend more time on application code and your actual business logics; that means it

giving up the flexibility about to your infrastructure.

## AWS Lambda Main Features

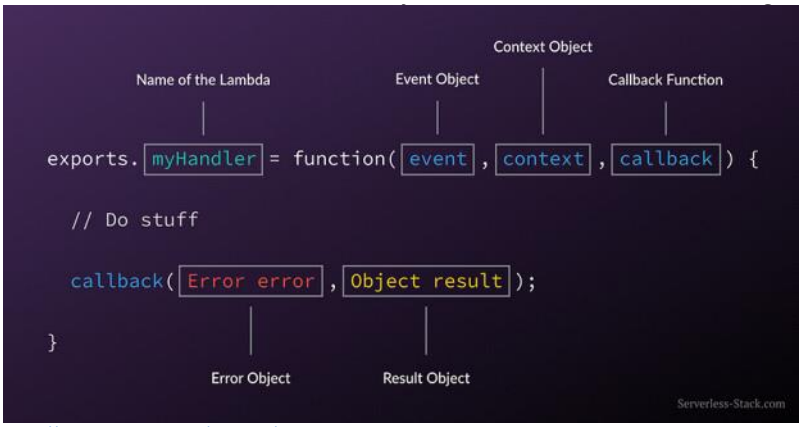
As you remember that we said that Lambda is a compute service that lets you run code without provisioning or managing any servers. Now lets focus on AWS Lambda Main Features.




**Cost Saving with Pay-as-you-go model**
Customers charged based on the allocated memory and the amount of execution time the function finished. You only pay for the compute time and there is no charge when your code is not running.
**Event-driven Architecture with Lambda**
Lambda is an on-demand compute service that runs custom code in response to events. Most AWS services generate events, and many can act as an event source for Lambda.
**Scalability and Availability**
With lambda you can run code for virtually any type of applications or backend services all with zero administration. Upload your code and run your code and scale your code automatically with high availability.
**Supports Multiple Languages and Frameworks**
Lambda has native support for a number of programming languages including Java, Go, PowerShell, Node.js, C#, Python, and Ruby code, and provides a Runtime API that lets you use any additional programming language to write your functions.

## AWS Lambda Function Code

Lambda runs instances of your function to process events. You can invoke your function directly using the Lambda API, or you can configure an AWS service or resource to invoke your function. You can find the image below, which is a Lambda function with Node.js version.




https://serverless-stack.com/chapters/what-is-aws-lambda.html

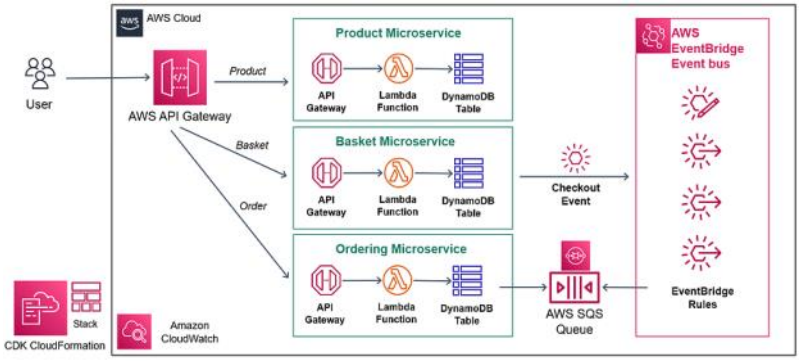
As you can see that we have lambda function name that we export in handler method. And we have defined function which has event, context and callback parameters. With this parameters we develop our actual business function and after that perform some callback operations that can be return a response or not respond for async invocations.

So Lambda function has code to process the events that you pass into the function or that other AWS services send to the function with event json object. The event object contains all the information about the event that triggered this Lambda. For example if lambda invokes from api gateway, then an HTTP request will be the information of event.

The context object contains info about the runtime our Lambda function is executing in. After we do all the work inside our Lambda function, we simply call the callback function with the results or the error and AWS will respond to the HTTP request with it.

## AWS Serverless Microservices for Ecommerce Application Architecture

Here you can find the main overall **Serverless Architecture** for our application. This is the **big picture** of what we are going to develop together for **AWS Serverless Event-driven E-commerce Microservices** application that is Step by Step Implementation together.




Serverless Event-driven E-commerce Microservices Architecture

We will be following the **reference architecture** above which is a **real-world Serverless E-commerce application** and it includes;

- **REST API** and **CRUD** endpoints with using **AWS Lambda, API Gateway**
- **Data persistence** with using **AWS DynamoDB**
- **Decouple microservices** with **events** using **AWS EventBridge**
- **Message Queues** for cross-service communication using **AWS SQS**
- **Cloud stack development** with **IaC** using **AWS CloudFormation CDK**