

EC2 everything is Pay as you go

There is another concept ---> Pay as you use (you have deployed the application, but noone is making a request) . say if application is not getting executed, it will not be charged  
If code is executed only then bill should be generated, till then bill will not be generated  
If code I executed for 10 min, then bill should be generated for only 10 minutes

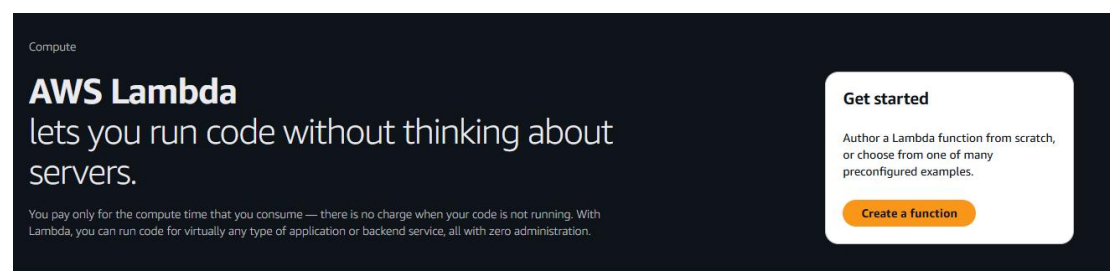
If you want this then we need =====> **Serverless computing comes into picture**

Serverless means how much my application is getting used, only for that much I will be charged. For idle time, I wont be charged at all.

Run your application without thinking about servers. EC2 works like servers

## AWS Lambdas

---



Compute

# AWS Lambda

lets you run code without thinking about servers.

You pay only for the compute time that you consume — there is no charge when your code is not running. With Lambda, you can run code for virtually any type of application or backend service, all with zero administration.

**Get started**  
Author a Lambda function from scratch, or choose from one of many preconfigured examples.  
[Create a function](#)



**How it works** [Run](#) [Next: Lambda responds to events](#)

[.NET](#) [Java](#) [Node.js](#) [Python](#) [Ruby](#) [Custom runtime](#)

You pay only for the compute time that you consume — there is no charge when your code is not running.

AWS Lambdas are used to implement Serverless computing

It is a way to run our application or code without creating / managing /paying for servers

If Application has 100 functionalities ---> we create 100 Lambda functions

For bigger applications with hundreds of functionalities, would you go with AWS Lambda? The answer is No

Depends on what features or functionalities we have, we go with Lambdas

Click Run

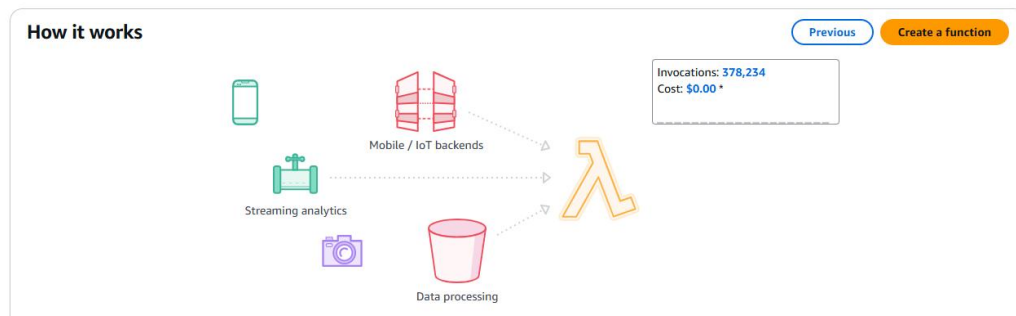


**How it works** [Run](#)

[.NET](#) [Java](#) [Node.js](#) [Python](#) [Ruby](#) [Custom runtime](#)

```
1 package example;
2
3 import com.amazonaws.services.lambda.runtime.Context;
4 import com.amazonaws.services.lambda.runtime.RequestHandler;
5
6 public class Hello implements RequestHandler<Object, String> {
7     public String handleRequest(final Object input, final Context context) {
8         System.out.println(input);
9         return "Hello from Lambda!";
10    }
11 }
12
```

As we get requests, it charges accordingly



## Create function

EC2 Aurora and RDS S3 Billing and Cost Manage... IAM Elastic Beanstalk Lambda

Lambda > Functions > Create function

### Create function [Info](#)

Choose one of the following options to create your function.

☒ **Author from scratch**  
Start with a simple Hello World example.

☐ **Use a blueprint**  
Build a Lambda application from sample code and configuration presets for common use cases.

☐ **Cont...**  
Select

### Basic information

**Function name** [Info](#)  
Enter a name that describes the purpose of your function.

myFunctionName

Function name must be 1 to 64 characters, must be unique to the Region, and can't include spaces. Valid characters are a-z, A-Z, 0-9, hyphens (-), and underscores (\_).

**Runtime** [Info](#)  
Choose the language to use to write your function. Note that the console code editor supports only Node.js, Python, and Ruby.

Node.js 22.x

**Architecture** [Info](#)  
Choose the instruction set architecture you want for your function code.

☒ x86\_64

☐ arm64

**Permissions** [Info](#)  
By default, Lambda will create an execution role with permissions to upload logs to Amazon CloudWatch Logs. You can customize this default role later when adding triggers.

## Create function [Info](#)

Choose one of the following options to create your function.

- ☒ **Author from scratch**  
Start with a simple Hello World example.

- ☐ **Use a blueprint**  
Build a Lambda application from sample code and configuration presets for common use cases.

### Basic information

#### Function name

Enter a name that describes the purpose of your function.

MyFunction

Function name must be 1 to 64 characters, must be unique to the Region, and can't include spaces. Valid characters are a-z, A-Z, 0-9, hyphens (-), and underscores (\_).

#### Runtime [Info](#)

Choose the language to use to write your function. Note that the console code editor supports only Node.js, Python, and Ruby.

Java 21

#### Architecture [Info](#)

Choose the instruction set architecture you want for your function code.

- ☒ x86\_64  
☐ arm64

#### Permissions [Info](#)

By default, Lambda will create an execution role with permissions to upload logs to Amazon CloudWatch Logs. You can customize this default role later when adding triggers.

► **Change default execution role**

### ▼ Additional Configurations

Use additional configurations to set up code signing, function URL, tags, and Amazon VPC access for your function.

☐ **Enable Code signing** [Info](#)

Use code signing configurations to ensure that the code has been signed by an approved source and has not been altered since signing.

☐ **Enable encryption with an AWS KMS customer managed key** [Info](#)

By default, Lambda encrypts the .zip file archive using an AWS owned key.

☒ **Enable function URL** [Info](#)

Use function URLs to assign HTTPS endpoints to your Lambda function.

#### Auth type

Choose the auth type for your function URL. [Learn more](#)

- ☒ **AWS\_IAM**  
Only authenticated IAM users and roles can make requests to your function URL.
- ☐ **NONE**  
Lambda won't perform IAM authentication on requests to your function URL. The URL endpoint will be public unless you implement your own authorization logic in your function.

#### Invoke mode [Info](#)

Choose how your function returns responses. [Learn more](#)

- ☒ **BUFFERED (default)**  
The invocation results are available when the payload is complete. Response payload max size: 6 MB
- ☐ **RESPONSE\_STREAM**  
Stream the invocation results. Streaming responses incurs additional costs. Refer to the documentation for payload size limitations. [Learn more](#)

☐ **Configure cross-origin resource sharing (CORS)**  
Use CORS to allow access to your function URL from any origin. You can also use CORS to control access for specific HTTP headers and methods in requests to your function URL. By default, all origins are allowed. You can edit this after creating the function. [Learn more](#)

☐ **Enable tags** [Info](#)

A tag is a label that you assign to an AWS resource. Each tag consists of a key and an optional value. You can use tags to search and filter your resources, track your AWS costs, and enforce attribute-based access control.

☐ **Enable VPC** [Info](#)

Connect your function to a VPC to access private resources during invocation.

[Cancel](#)

[Create function](#)

Click Enable function URL, AuthType is NONE

## ▼ Additional Configurations

Use additional configurations to set up code signing, function URL, tags, and Amazon VP

☐ **Enable Code signing** [Info](#)

Use code signing configurations to ensure that the code has been signed by an approved source and has no

☐ **Enable encryption with an AWS KMS customer managed key** [Info](#)

By default, Lambda encrypts the .zip file archive using an AWS owned key.

☒ **Enable function URL** [Info](#)

Use function URLs to assign HTTP(S) endpoints to your Lambda function.

### Auth type

Choose the auth type for your function URL. [Learn more](#) [↗](#)

☒ **AWS\_IAM**

Only authenticated IAM users and roles can make requests to your function URL.

☐ **NONE**

Lambda won't perform IAM authentication on requests to your function URL. The URL endpoint will be

### Invoke mode [Info](#)

Choose how your function returns responses. [Learn more](#) [↗](#)

☒ **BUFFERED (default)**

The invocation results are available when the payload is complete. Response payload max size: 6 MB

☐ **RESPONSE\_STREAM**

Stream the invocation results. Streaming responses incurs additional costs. Refer to the documentatio

☐ **Configure cross-origin resource sharing (CORS)**

Use CORS to allow access to your function URL from any origin. You can also use CORS to control acces  
[Learn more](#) [↗](#)

☐ **Enable tags** [Info](#)

A tag is a label that you assign to an AWS resource. Each tag consists of a key and an optional value. You ca

☐ **Enable VPC** [Info](#)

Connect your function to a VPC to access private resources during invocation.

▼ Additional Configurations

Use additional configurations to set up code signing, function URL, tags, and Amazon VPC access for your function.

☐ Enable Code signing [Info](#)

Use code signing configurations to ensure that the code has been signed by an approved source and has not been altered since signing.

☐ Enable encryption with an AWS KMS customer managed key [Info](#)

By default, Lambda encrypts the .zip file archive using an AWS owned key.

☒ Enable function URL [Info](#)

Use function URLs to assign HTTP(S) endpoints to your Lambda function.

Auth type

Choose the auth type for your function URL. [Learn more](#)

☐ AWS\_IAM
 

Only authenticated IAM users and roles can make requests to your function URL.

☒ NONE
 

Lambda won't perform IAM authentication on requests to your function URL. The URL endpoint will be public unless you implement your

Function URL permissions

ⓘ

When you choose auth type **NONE**, Lambda automatically creates the following resource-based policy and attaches policy later. To limit access to authenticated IAM users and roles, choose auth type **AWS\_IAM**.

▼ View policy statement

1 ▼

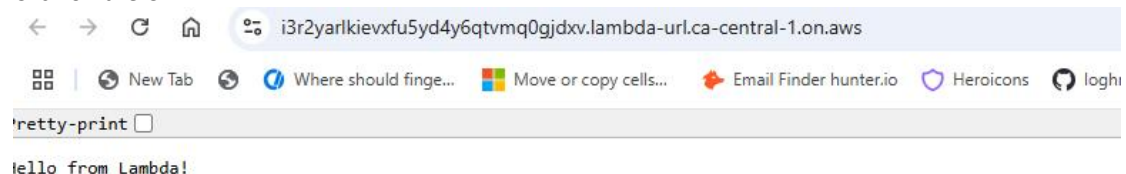
2 "Version": "2012-10-17",

3 ▼ "Statement": [

4 ▼ {

## Create Function

Click on the URL



Code source [Info](#)

ⓘ

The code editor does not support the Java 21 runtime.

But we can upload our own functions  
We can zip application and upload also

AWS Lambda is a way to run code without creating, managing or paying for servers

We pay for it for the time AWS runs and nothing more

AWS will scale our code depending on number of requests we receive. It is promoting event-driven architecture

