

# SuccessLambda 생성

# 단계1: Create function

Lambda > Functions

Lambda

Dashboard

Applications

Functions

▼ Additional resources

Code signing configurations

Event source mappings

Layers

Functions (3)

Last fetched 57 seconds ago

Actions

Create function

Filter by attributes or search by keyword

< 1 >

<input type="checkbox"/>	Function name	Description	Package type	Runtime	Last modified
<input type="checkbox"/>	<a href="#">investment-CRYPTO_UPBIT-etl-lambda</a>	-	Zip	Python 3.11	4 months ago
<input type="checkbox"/>	<a href="#">FirstLambda</a>	-	Zip	Python 3.13	27 minutes ago
<input type="checkbox"/>	<a href="#">investment-slack-alarm-lambda</a>	-	Zip	Python 3.11	4 months ago

☒ **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.

☐ **Container image**  
Select a container image to deploy

## Basic information

### Function name

Enter a name that describes the purpose of your function.

SuccessLambda

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.

Python 3.13



### Architecture [Info](#)

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

☒ x86\_64

☐ arm64

### ▼ Change default execution role

#### Execution role

Choose a role that defines the permissions of your function. To create a custom role, go to the [IAM console](#).

- ☐ Create a new role with basic Lambda permissions
- ☒ Use an existing role
- ☐ Create a new role from AWS policy templates

#### Existing role

Choose an existing role that you've created to be used with this Lambda function. The role must have permission to upload logs to Amazon CloudWatch Logs.

service-role/FirstLambda-role-1l7yhpca

[View the FirstLambda-role-1l7yhpca role](#) on the IAM console.

### ► Additional Configurations

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

Cancel

Create function

## 단계2: configuration > Timeout 수정

[Code](#) | [Test](#) | [Monitor](#) | **[Configuration](#)** | [Aliases](#) | [Versions](#)

**General configuration**  
Triggers  
Permissions  
Destinations  
Function URL

**General configuration** [Info](#)


**Description**  
-

**Timeout**  
0 min 3 sec

**Memory**  
128 MB

**SnapStart** [Info](#)  
None

**Ephemeral storage**  
512 MB

 [Edit](#)

### Timeout

1

min

0

sec

### Execution role

Choose a role that defines the permissions of your function. To create a custom role, go to the [IAM console](#).

- ☒ Use an existing role
- ☐ Create a new role from AWS policy templates

### Existing role

Choose an existing role that you've created to be used with this Lambda function. The role must have permission to upload logs to Amazon CloudWatch Logs.

service-role/FirstLambda-role-1l7yhpcav

[View the FirstLambda-role-1l7yhpcav role](#) on the IAM console.

[Cancel](#) [Save](#)

✓ Successfully updated the function SuccessLambda.



+ Add trigger

Function URL [Info](#)

-

Code

Test

Monitor

**Configuration**

Aliases

Versions

**General configuration**

Triggers

Permissions

Destinations

Function URL

**General configuration** [Info](#)

Edit

Description

-

Memory

128 MB

Ephemeral storage

512 MB

Timeout

1 min 0 sec

SnapStart [Info](#)

None

## 단계3: Layer 추가

Lambda > Functions > SuccessLambda

code properties info

Package size  
299 byte

SHA256 hash  
HAPq9EReJVEC5gLavtc/gyd5vZtd9eiUGF932t0jBxY=

Last modified  
18 seconds ago

► Encryption with AWS KMS customer managed KMS key info

Runtime settings info

Runtime  
Python 3.13

Handler info  
lambda\_function.lambda\_handler

Architecture info  
x86\_64

► Runtime management configuration

EditEdit runtime management configuration

Layers info

EditAdd a layer

Merge order	Name	Layer version	Compatible runtimes	Compatible architectures	Version ARN
1	common_python	1	python3.13	x86_64	arn:aws:lambda:ap-northeast-2:426653742146:layer:common_python:1



## 단계4: 코드 수정

```
import json
from common.aws_s3 import mk_path_json_of_s3, upload_json_to_s3, download_json_from_s3

def lambda_handler(event, context):
    # TODO implement
    path_json = mk_path_json_of_s3("lambda_test")
    path_json = "success/"+path_json
    bucket_name = "lambda-good593"

    upload_json_to_s3(data=event, bucket=bucket_name, path=path_json)
    return {
        'statusCode': 200,
        'body': json.dumps(event)
    }
```

## 단계5: Deploy

☰ [Lambda](#) > [Functions](#) > **SuccessLambda**

✔ Successfully updated the function **SuccessLambda**.

The screenshot displays the AWS Lambda console interface. At the top, a green notification bar states "✔ Successfully updated the function **SuccessLambda**." Below this, the breadcrumb navigation shows "☰ [Lambda](#) > [Functions](#) > **SuccessLambda**".

The main interface is divided into three sections:

- EXPLORER**: On the left, it shows the file structure for the function. Under the "SUCCESSLAMBDA" folder, the file "lambda\_function.py" is listed and selected.
- DEPLOY**: Below the explorer, there are two buttons: "Deploy (Ctrl+Shift+U)" and "Test (Ctrl+Shift+I)". A red arrow points to the "Deploy" button.
- Code Editor**: On the right, the code for "lambda\_function.py" is displayed. The code is as follows:

```
4 def lambda_handler(event, context):
5     # TODO implement
6     path_json = mk_path_json_of_s3("lambda_test")
7     path_json = "success/"+path_json
8     bucket_name = "lambda-good593"
9
10    upload_json_to_s3(data=event, bucket=bucket_name, path=path_json)
11    return {
12        'statusCode': 200,
13        'body': json.dumps(event)
14    }
15
```