**Function-as-a-Service using AWS Lambda**

AWS Lambda is a Function-as-a-Service (FaaS) provided by Amazon Web Services (AWS) that enables developers to run code without provisioning or managing servers. It automatically scales in response to incoming requests and charges only for the actual compute time consumed.

Lambda supports various runtimes, including Python, Node.js, Java, and Go, and can be triggered by AWS services such as S3, DynamoDB, API Gateway, and CloudWatch.

**Getting Started with AWS Lambda**

**Prerequisites**

Before using AWS Lambda, ensure that you have an AWS account.

**Step 1: Creating an AWS Lambda Function**

1. Sign in to AWS Console and go to AWS Lambda Console

A screenshot of a computer

AI-generated content may be incorrect.

2. Create a new function:

- Click Create function.

A screenshot of a box

AI-generated content may be incorrect.

- Choose Author from scratch.

- Enter a function name

- Select a runtime (e.g., Python 3.9)

- Optionally, choose an existing execution role or create a new role.

3. Click Create function.

**Step 2: Writing and Deploying the Function**

1. In the code editor, replace the default code with a function of your liking. For this example, I prepared a function which based on a customer rating and a customer orders’ price, the discount applicable on the order is going to be calculated and returned:  
A screenshot of a computer program

AI-generated content may be incorrect.  
A screenshot of a computer program

AI-generated content may be incorrect.

2. Click Deploy to save the function.

Step 3: Testing the Lambda Function

The easiest way to test the Lambda Function would be by using Postman – or any other URL tester.  
Create a request with the URL of the Lambda Function:

A close-up of a computer screen

AI-generated content may be incorrect.

Result:

A screenshot of a computer

AI-generated content may be incorrect.

In case there is no URL present, it can be generated by the following steps:  
1. A blue and black text

AI-generated content may be incorrect.

2.

A screen shot of a computer

AI-generated content may be incorrect.

1. Generate a Function URL:A screenshot of a computer

   AI-generated content may be incorrect.

**Invoking AWS Lambda with API Gateway using Java Spring**

1. Set the Lambda Function URL in your application.yml file or in your application.properties file in a similar manner:

A black background with white text

AI-generated content may be incorrect.

1. Set the Lambda Function URL in your docker-compose.dev.yml file:

A screenshot of a computer

AI-generated content may be incorrect.

1. Retrieve the URL in the Service you would like to invoke it from:

A black background with colorful text

AI-generated content may be incorrect.

1. Create a new Rest Template instance in the Service you would want to invoke the Lambda function from:

A black screen with white text

AI-generated content may be incorrect.

1. Make an HTTP request to invoke the Lambda Function:

A screen shot of a computer code

AI-generated content may be incorrect.

Additionaly, different data transfer objects can be created for the data sending and the data parsing part, if applicable.

(If you want to copy-paste the function:

import json

def calculate\_discount(total\_price, rating):

"""Calculate discount percentage based on rating (1 to 5)."""

discount\_rates = {

5: 0.20, # 20% discount

4: 0.15, # 15% discount

3: 0.10, # 10% discount

2: 0.05, # 5% discount

1: 0.00 # No discount

}

return total\_price \* discount\_rates.get(rating, 0.00)

def lambda\_handler(event, context):

"""AWS Lambda handler function."""

try:

# Parse input JSON

body = json.loads(event["body"]) if "body" in event else event

prices = body.get("prices", [])

rating = body.get("rating", 1)

# Validate inputs

if not isinstance(prices, list) or not all(isinstance(p, (int, float)) for p in prices):

return {

"statusCode": 400,

"body": json.dumps({"error": "Invalid prices list. Must be a list of numbers."})

}

if not isinstance(rating, int) or rating < 1 or rating > 5:

return {

"statusCode": 400,

"body": json.dumps({"error": "Invalid rating. Must be an integer between 1 and 5."})

}

# Calculate prices

original\_price = sum(prices)

discount = calculate\_discount(original\_price, rating)

final\_price = original\_price - discount

# Return response

response = {

"original\_price": round(original\_price, 2),

"discount": round(discount, 2),

"final\_price": round(final\_price, 2)

}

return {

"statusCode": 200,

"body": json.dumps(response)

}

except Exception as e:

return {

"statusCode": 500,

"body": json.dumps({"error": str(e)})

}

)