Shravani Anil Patil **D15A-38**

Advanced DevOps Lab
Experiment 11

Aim: To understand AWS Lambda, its workflow, various functions and create your first Lambda functions using Python / Java / Nodejs.

Theory:

AWS Lambda

AWS Lambda is a serverless computing service provided by Amazon Web Services (AWS). Users of AWS Lambda create functions, self-contained applications written in one of the supported languages and runtimes, and upload them to AWS Lambda, which executes those functions in an efficient and flexible manner.The Lambda functions can perform any kind of computing task, from serving web pages and processing streams of data to calling APIs and integrating with other AWS services.

The concept of "serverless" computing refers to not needing to maintain your own servers to run these functions. AWS Lambda is a fully managed service that takes care of all the infrastructure for you. And so "serverless" doesn't mean that there are no servers involved: it just means that the servers, the operating systems, the network layer and the rest of the infrastructure have already been taken care of so that you can focus on writing application code.

Features of AWS Lambda

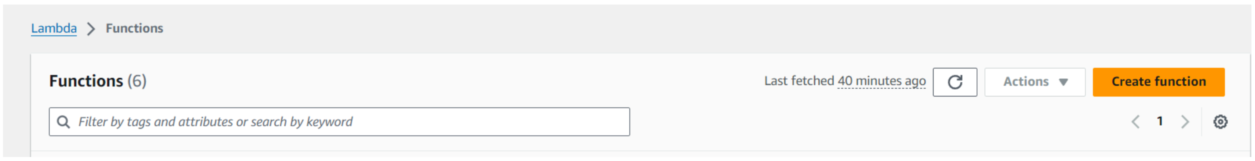
● AWS Lambda easily scales the infrastructure without any additional configuration. It reduces the operational work involved.
● It offers multiple options like AWS S3, CloudWatch, DynamoDB, API Gateway, Kinesis, CodeCommit, and many more to trigger an event.
● You don't need to invest upfront. You pay only for the memory used by the lambda function and minimal cost on the number of requests hence cost-efficient.
● AWS Lambda is secure. It uses AWS IAM to define all the roles and security policies.
● It offers fault tolerance for both services running the code and the function. You do not have to worry about the application down.

Packaging Functions

Lambda functions need to be packaged and sent to AWS. This is usually a process of compressing the function and all its dependencies and uploading it to an S3 bucket. And letting AWS know that you want to use this package when a specific event takes place. To help us with this process we use the Serverless Stack Framework (SST). We'll go over this in detail later on in this guide.

Steps to create an AWS Lambda function

1. Open up the Lambda Console and click on the Create button.



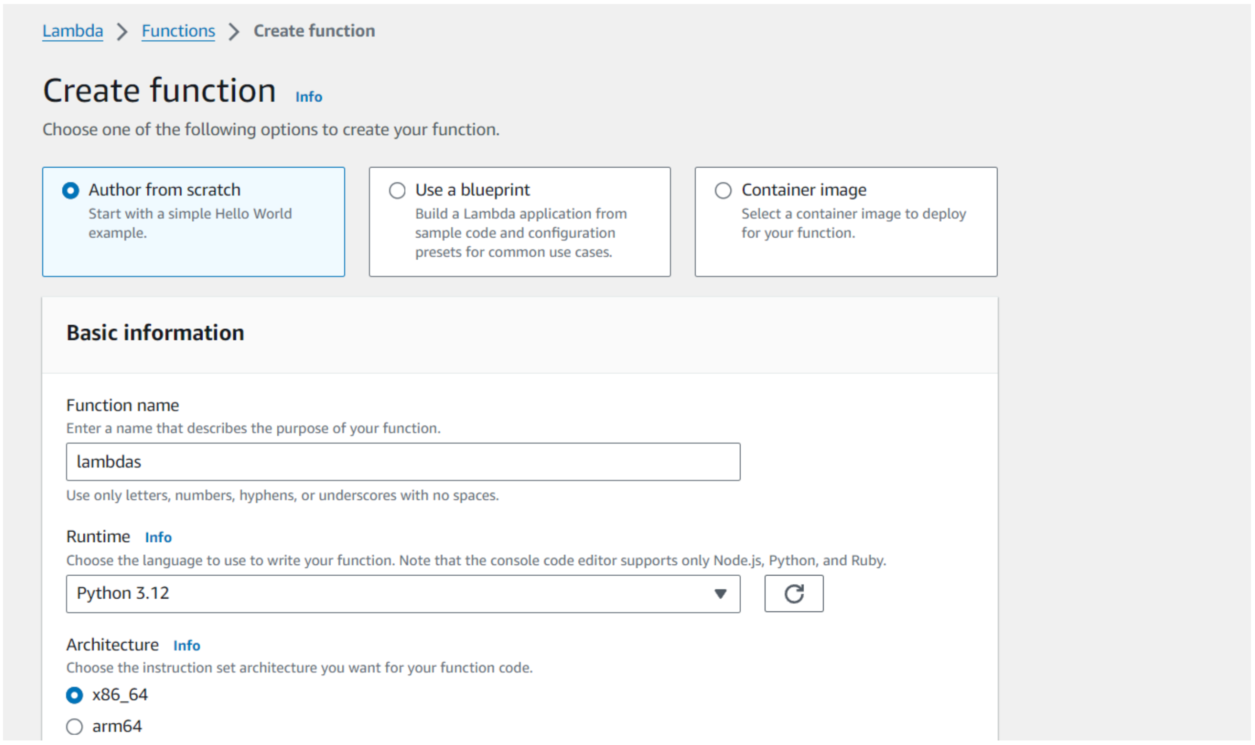
Lambda > Functions

Functions (6)

Last fetched 40 minutes ago

Filter by tags and attributes or search by keyword

2. Choose to create a function from scratch or use a blueprint, i.e templates defined by AWS
for you with all configuration presets required for the most common use cases.
Then,choose a runtime env for your function, under the dropdown, you can see all the options AWS supports, Python, Nodejs, .NET and Java being the most popular ones.



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.

Container image — Select a container image to deploy for your function.

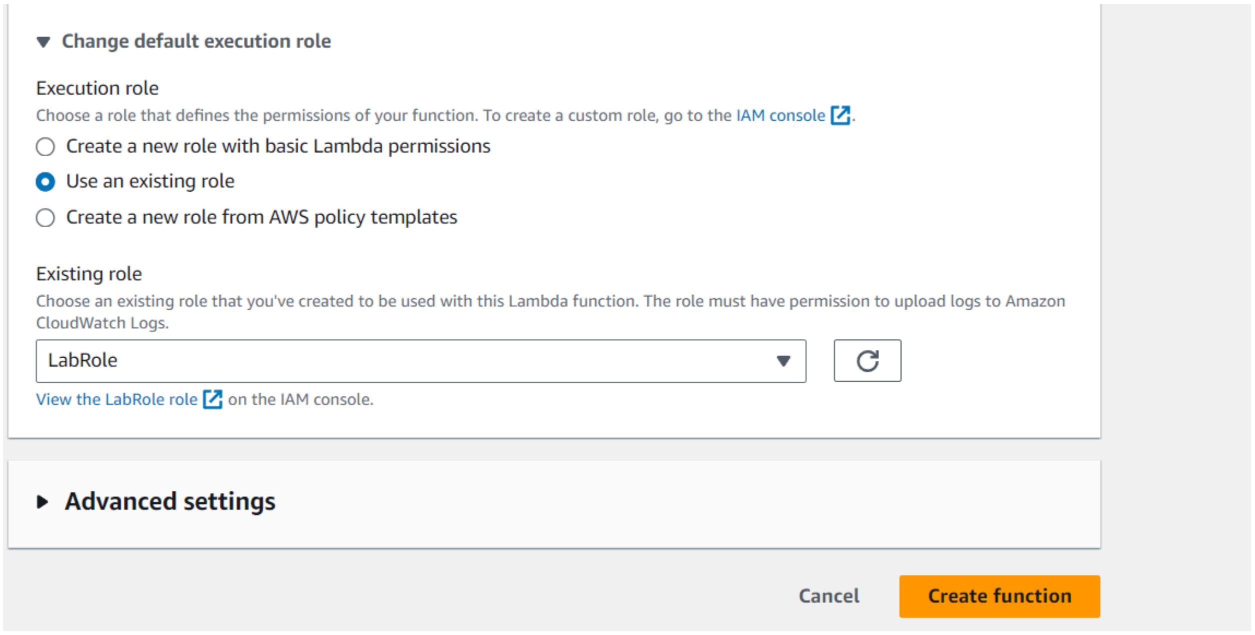
Basic information

Function name
Enter a name that describes the purpose of your function.
lambdas
Use only letters, numbers, hyphens, or underscores with no spaces.

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 3.12

Architecture Info
Choose the instruction set architecture you want for your function code.
[✓ x86_64]
arm64

After that, choose an existing role or create a new role with basic Lambda permissions if you don't have an existing one.



▼ 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

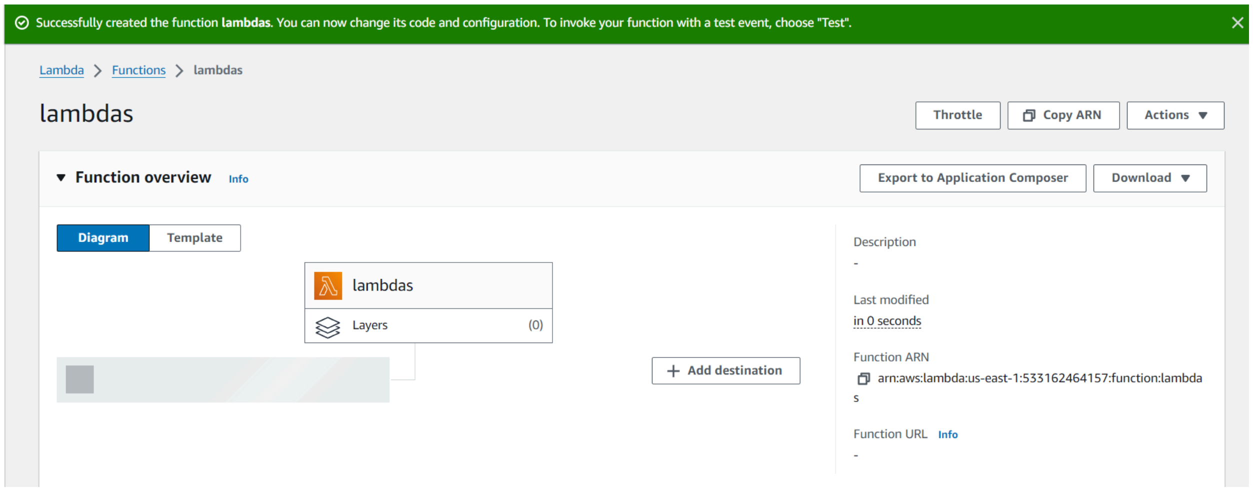
Choose an existing role that you've created to be used with this Lambda function. The role must have permission to upload logs to Amazon CloudWatch Logs.

LabRole

View the LabRole role on the IAM console.

Click on the Create button.

3. This process will take a while to finish and after that, you'll get a message that your function was successfully created.



Successfully created the function **lambdas**. You can now change its code and configuration. To invoke your function with a test event, choose "Test".

Lambda > Functions > lambdas

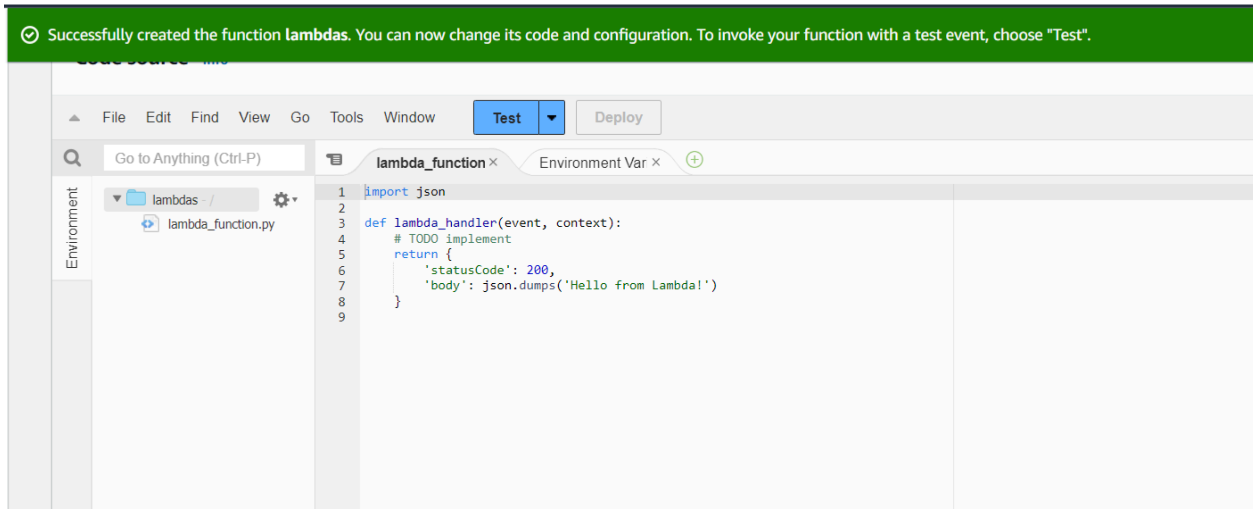
lambdas

Description
-

Last modified
in 0 seconds

Function ARN
arn:aws:lambda:us-east-1:533162464157:function:lambdas

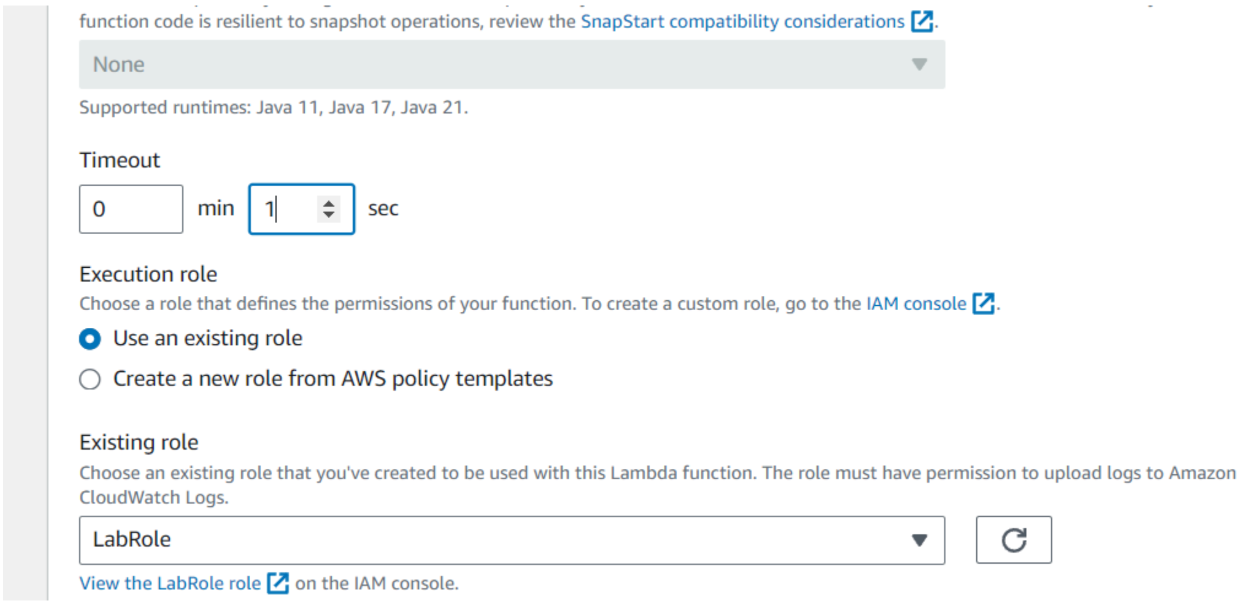
Function URL Info
-


Successfully created the function **lambdas**. You can now change its code and configuration. To invoke your function with a test event, choose "Test".

```
import json

def lambda_handler(event, context):
    # TODO implement
    return {
        'statusCode': 200,
        'body': json.dumps('Hello from Lambda!')
    }
```

4. To change the configuration, open up the Configuration tab and under General Configuration, choose Edit.
Here, you can enter a description and change Memory and Timeout. I've changed the Timeout period to 1 sec since that is sufficient for now.


function code is resilient to snapshot operations, review the SnapStart compatibility considerations.

None

Supported runtimes: Java 11, Java 17, Java 21.

Timeout

0 min 1 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

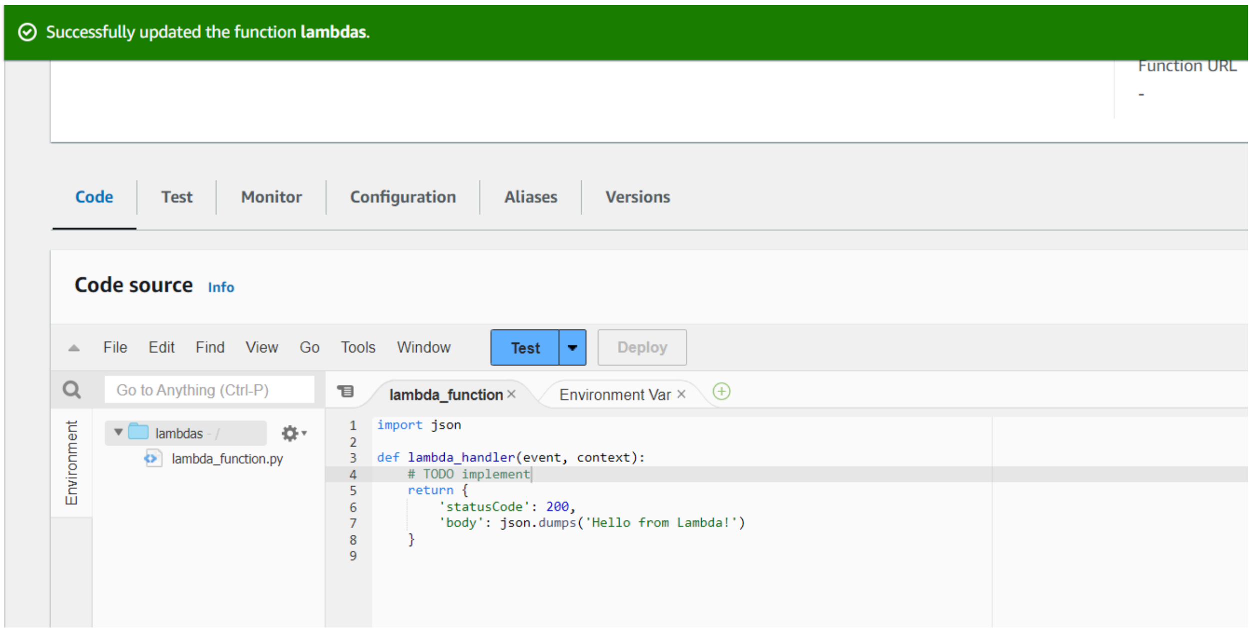
Existing role

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.

LabRole

View the LabRole role on the IAM console.

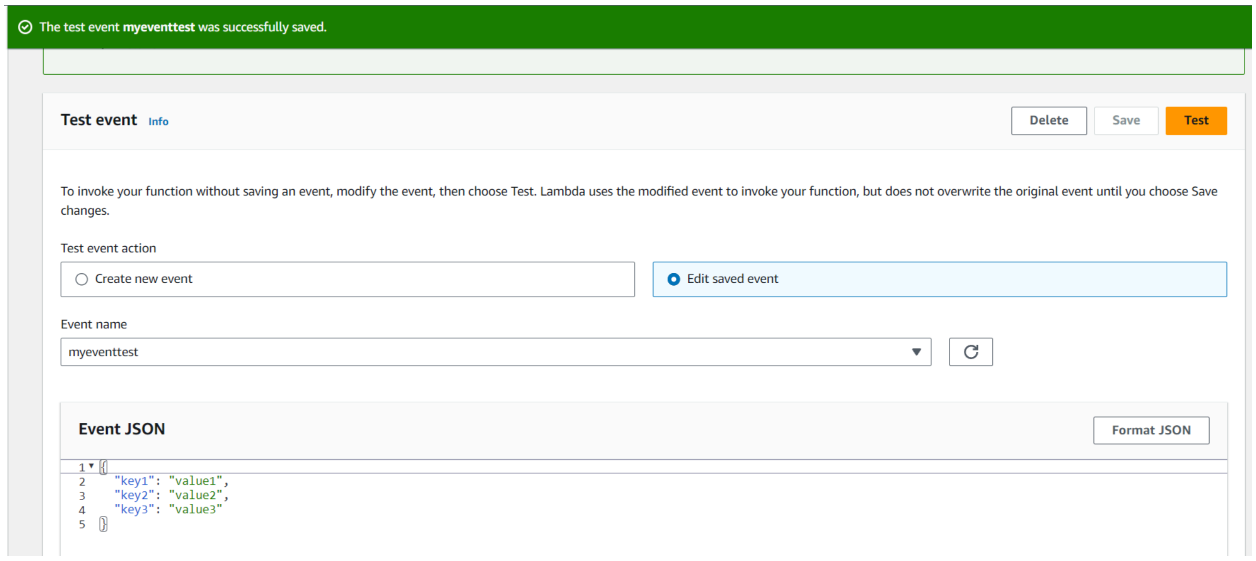
**5. You can make changes to your function inside the code editor. You can also upload a zip file of your function or upload one from an S3 bucket if needed.
Press Ctrl + S to save the file and click Deploy to deploy the changes.**



```python
import json

def lambda_handler(event, context):
    # TODO implement
    return {
        'statusCode': 200,
        'body': json.dumps('Hello from Lambda!')
    }
```

6. Click on Test and you can change the configuration, like so. If you do not have anything in the request body, it is important to specify two curly braces as valid JSON, so make sure they are there.



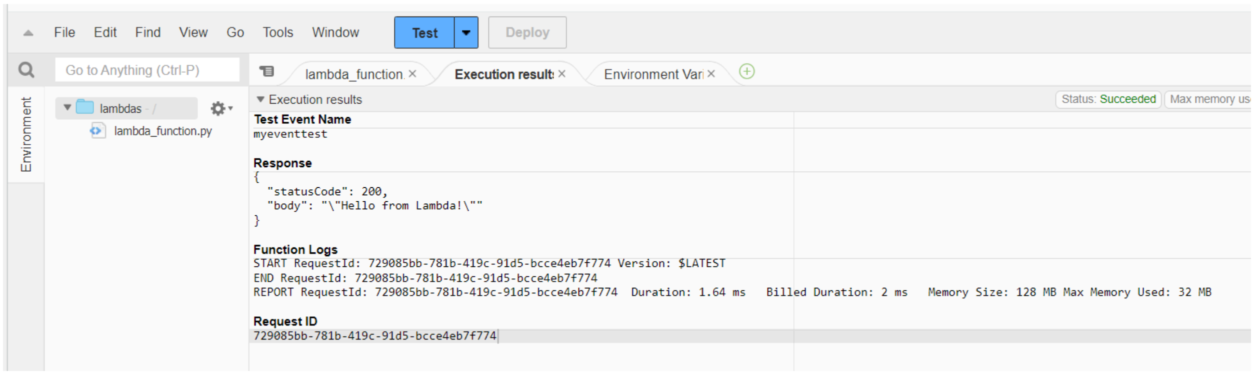
To invoke your function without saving an event, modify the event, then choose Test. Lambda uses the modified event to invoke your function, but does not overwrite the original event until you choose Save changes.

Test event action: Create new event [✓ Edit saved event]

Event name: myeventtest

```json
{
  "key1": "value1",
  "key2": "value2",
  "key3": "value3"
}
```

7. Now click on Test and you should be able to see the results.



Test Event Name

```
myeventtest
```

Response

```
{
  "statusCode": 200,
  "body": "\"Hello from Lambda!\""
}
```

Function Logs

```
START RequestId: 729085bb-781b-419c-91d5-bcce4eb7f774 Version: $LATEST
END RequestId: 729085bb-781b-419c-91d5-bcce4eb7f774
REPORT RequestId: 729085bb-781b-419c-91d5-bcce4eb7f774  Duration: 1.64 ms   Billed Duration: 2 ms   Memory Size: 128 MB Max Memory Used: 32 MB
```

Request ID

```
729085bb-781b-419c-91d5-bcce4eb7f774
```