

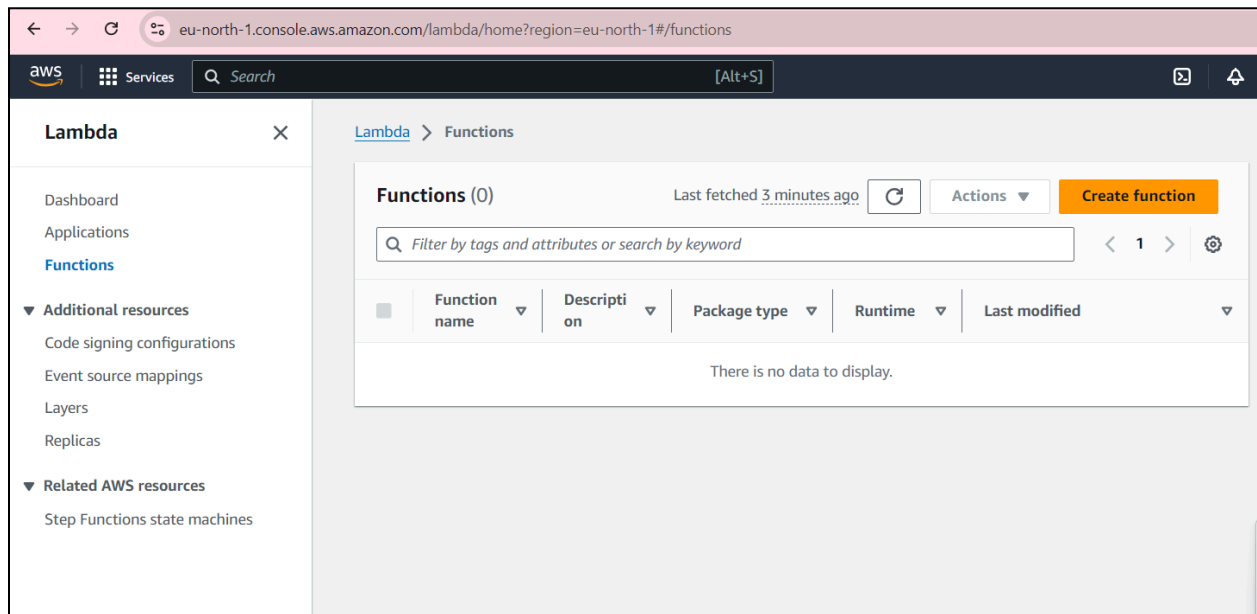
Experiment.11

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

Steps-

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

Be mindful of where you create your functions since Lambda is region-dependent.



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.

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

aws

Services

Search

[Alt+S]

≡

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.

☐ **Browse serverless app repository**
Deploy a sample Lambda application from the AWS Serverless Application Repository.

Basic information

Function name
Enter a name that describes the purpose of your function.

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.

▼

aws

Services

Search

[Alt+S]

≡

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

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

Role creation might take a few minutes. Please do not delete the role or edit the trust or permissions policies in this role.

Lambda will create an execution role named `lambdashreya-role-ni9thou9`, with permission to upload logs to Amazon CloudWatch Logs.

► **Additional Configurations**

Shreya Sawant
D15A 54

The screenshot shows the AWS Lambda console interface. At the top, a green notification bar states: "Successfully created the function **lambdashreya**. You can now change its code and configuration. To invoke your function with a test event, choose 'Test'." Below this, the breadcrumb navigation is "Lambda > Functions > lambdashreya". The main heading is "lambdashreya". To the right of the heading are buttons for "Throttle", "Copy ARN", and "Actions". Below the heading is a "Function overview" section with tabs for "Diagram" and "Template". The "Diagram" tab is active, showing a visual representation of the function with a "Layers" section indicating "(0)" layers. To the right of the diagram are buttons for "Add trigger" and "Add destination". On the far right, a metadata panel displays the following information: "Description" (empty), "Last modified" (1 second ago), "Function ARN" (arn:aws:lambda:eu-north-1:022499029436:function:lambdashreya), and "Function URL" (Info link).

The screenshot shows the "Code source" tab for the "lambdashreya" function. At the top, the same green notification bar is present. Below it, the "Code source" section has an "Upload from" dropdown. A menu bar with "File", "Edit", "Find", "View", "Go", "Tools", and "Window" is visible, along with "Test" and "Deploy" buttons. The "Test" button is highlighted. Below the menu bar, a file explorer on the left shows the "lambdashreya" folder containing "lambda_function.py". The main area displays the Python code for the lambda handler:

```
1 import json
2
3 def lambda_handler(event, context):
4     # TODO implement
5     return {
6         'statusCode': 200,
7         'body': json.dumps('Hello from Lambda!')
8     }
9
```

aws

Services

Search

[Alt+S]

Edit basic settings

Basic settings

Info

Description - optional

Memory

Info

Your function is allocated CPU proportional to the memory configured.

128

MB

Set memory to between 128 MB and 10240 MB

Ephemeral storage

Info

You can configure up to 10 GB of ephemeral storage (/tmp) for your function. [View pricing](#)

512

MB

Set ephemeral storage (/tmp) to between 512 MB and 10240 MB.

SnapStart

Info

Reduce startup time by having Lambda cache a snapshot of your function after the function has initialized. To evaluate whether your function code is resilient to snapshot operations, review the [SnapStart compatibility considerations](#).

None

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

Timeout

0

min

3

sec

Timeout

0

min

3

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

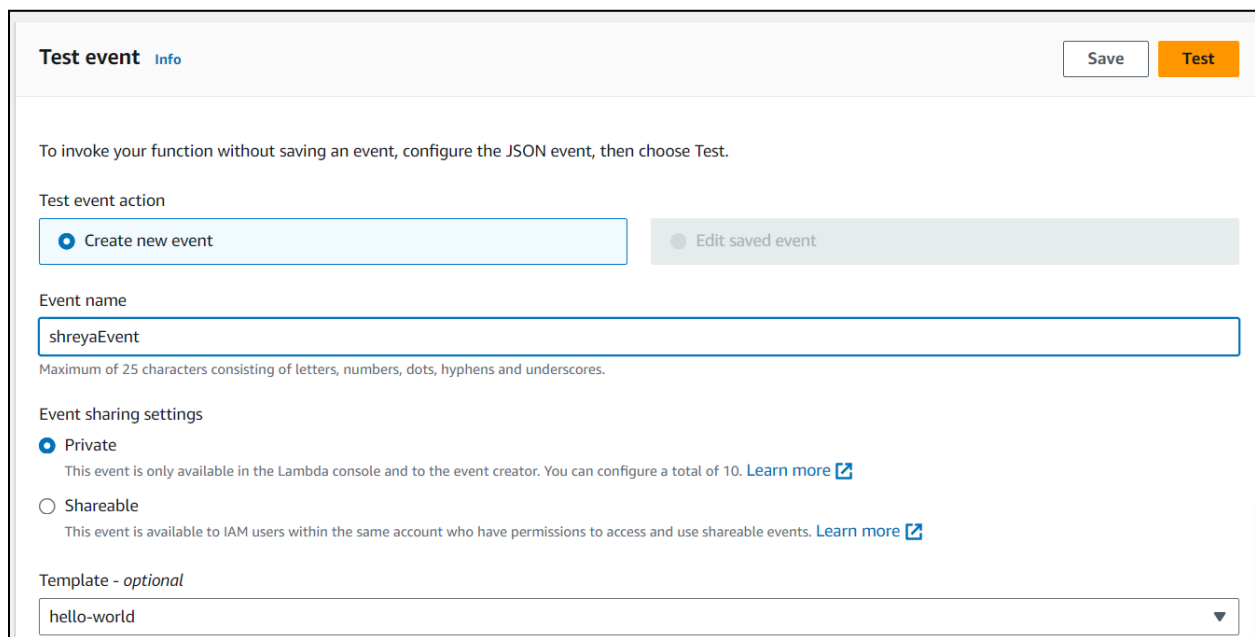
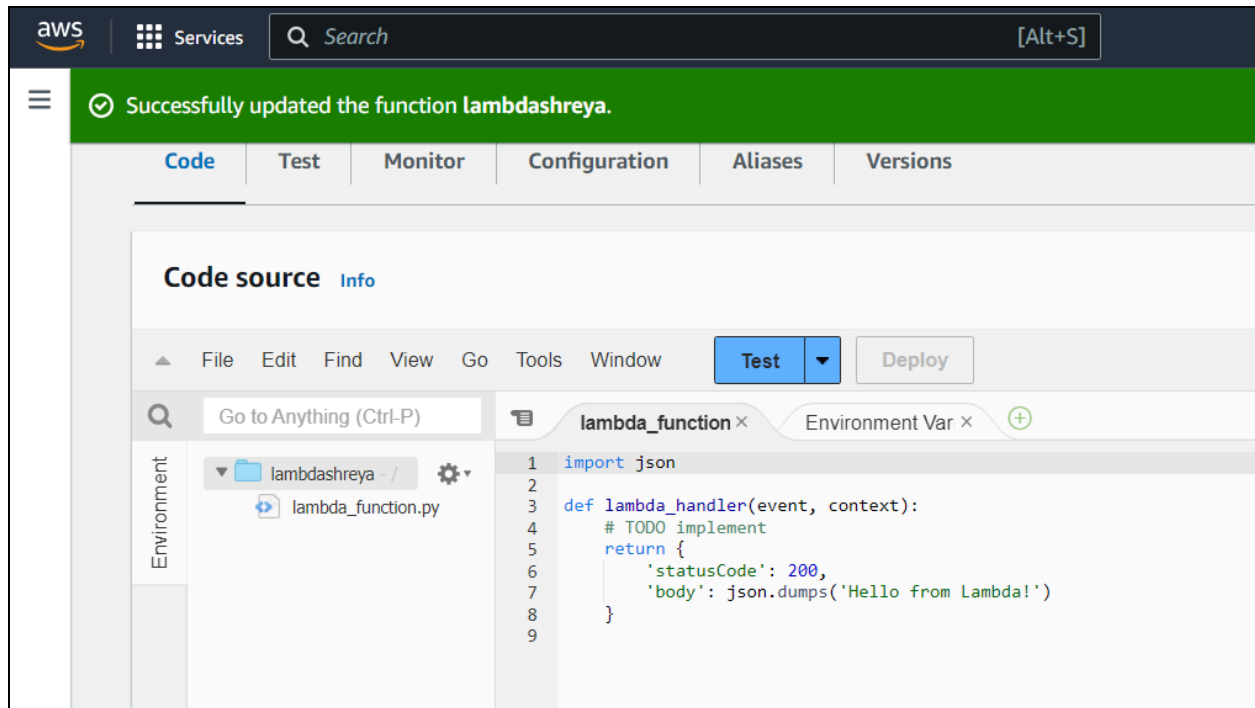
Choose an existing role that you've created to be used with this Lambda function. The role must have permission to upload logs to Amazon CloudWatch Logs.

service-role/lambda-shreya-role-ni9thou9

[View the lambda-shreya-role-ni9thou9 role](#) on the IAM console.

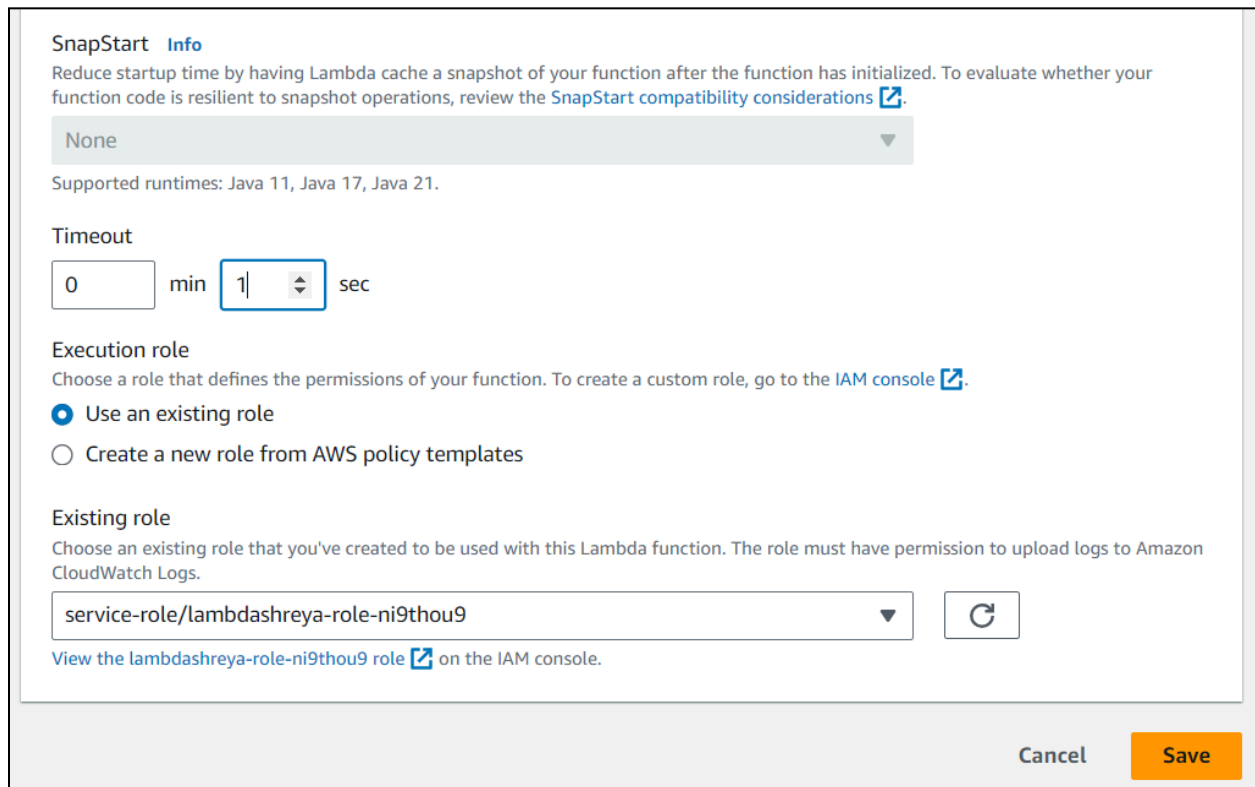
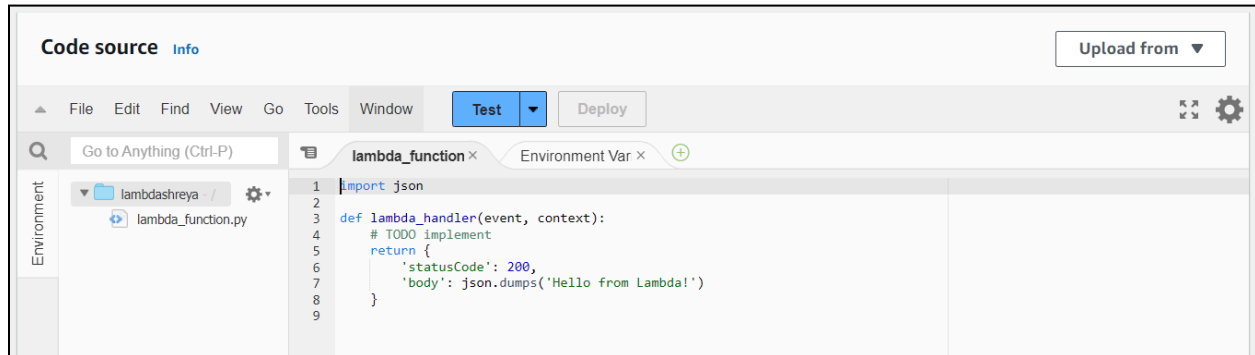
Cancel

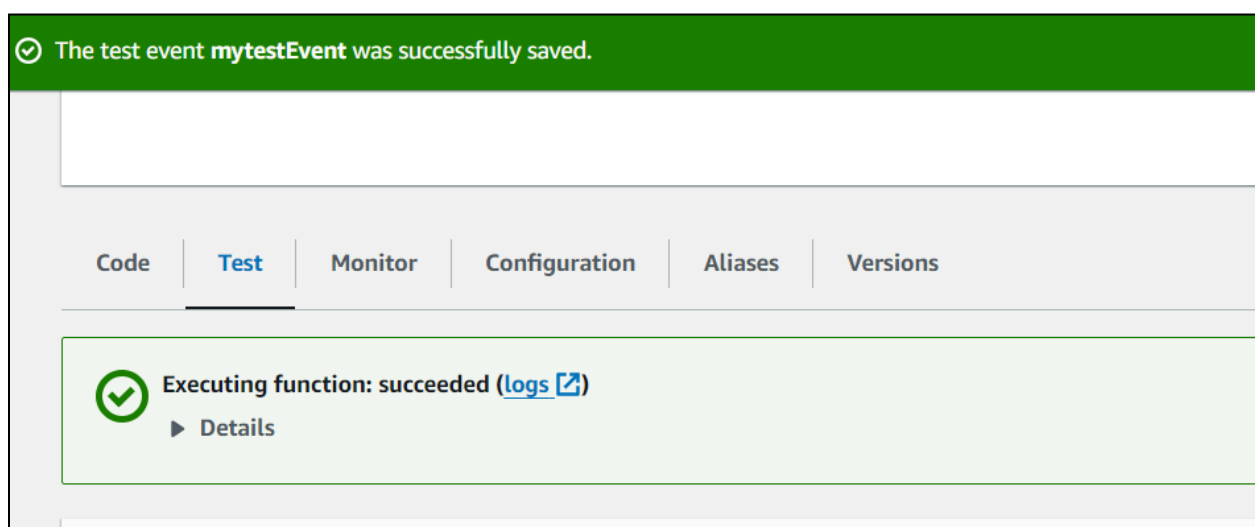
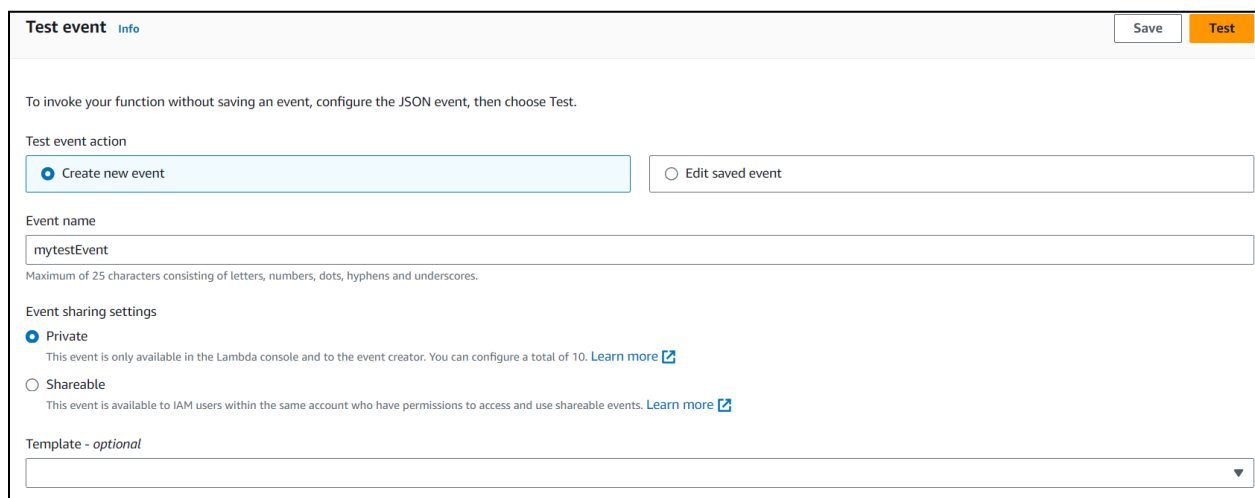
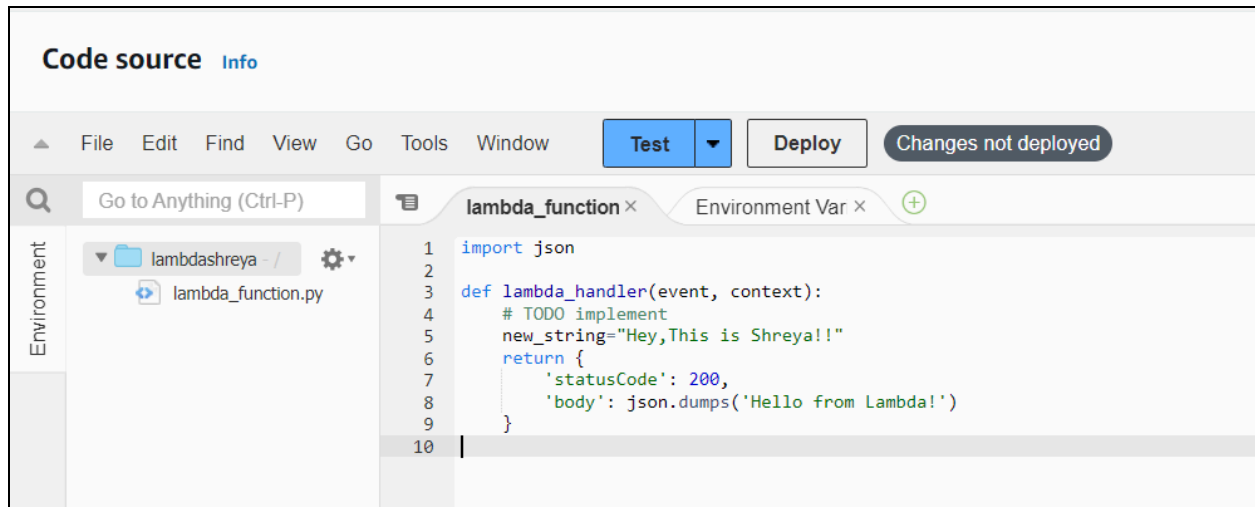
Save



Shreya Sawant
D15A 54

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.





CloudWatch > Log groups > /aws/lambda/lambdashreya

/aws/lambda/lambdashreya

Actions View in Logs Insights Start tailing Search log group

▼ Log group details

Log class Standard	Stored bytes -	KMS key ID -
ARN arn:aws:logs:eu-north-1:022499029436:log-group:/aws/lambda/lambdashreya:*	Metric filters 0	Anomaly detection Configure
Creation time 4 minutes ago	Subscription filters 0	Data protection -
Retention Never expire	Contributor Insights rules -	Sensitive data count -

Code source [Info](#)

Upload from

File Edit Find View Go Tools Window Test Deploy

Environment

Go to Anything (Ctrl-P)

Environment

lambdashreya

lambda_function.py

Execution results

Status: Succeeded Max memory used: 32 MB Time: 2.03 ms

Test Event Name
mytestEvent

Response

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

Function Logs

START RequestId: 50a6c1ec-b60e-4595-821e-6bb783614ed9 Version: \$LATEST
END RequestId: 50a6c1ec-b60e-4595-821e-6bb783614ed9
REPORT RequestId: 50a6c1ec-b60e-4595-821e-6bb783614ed9 Duration: 2.03 ms Billed Duration: 3 ms Memory Size: 128 MB Max Memory Used: 32 MB Init Duration: 86.56 ms

Request ID

50a6c1ec-b60e-4595-821e-6bb783614ed9