

Lesson End Project

Initiating a Lambda Function to Copy Content in S3 Buckets

Project agenda: To launch the Lambda function and perform the trigger operation on Lambda for processing incoming S3 events

Description: As a solution architect, you must replicate the contents of a bucket to other buckets deployed in different regions, thereby making the resources available.

Tools required: AWS Management Console

Prerequisites: None

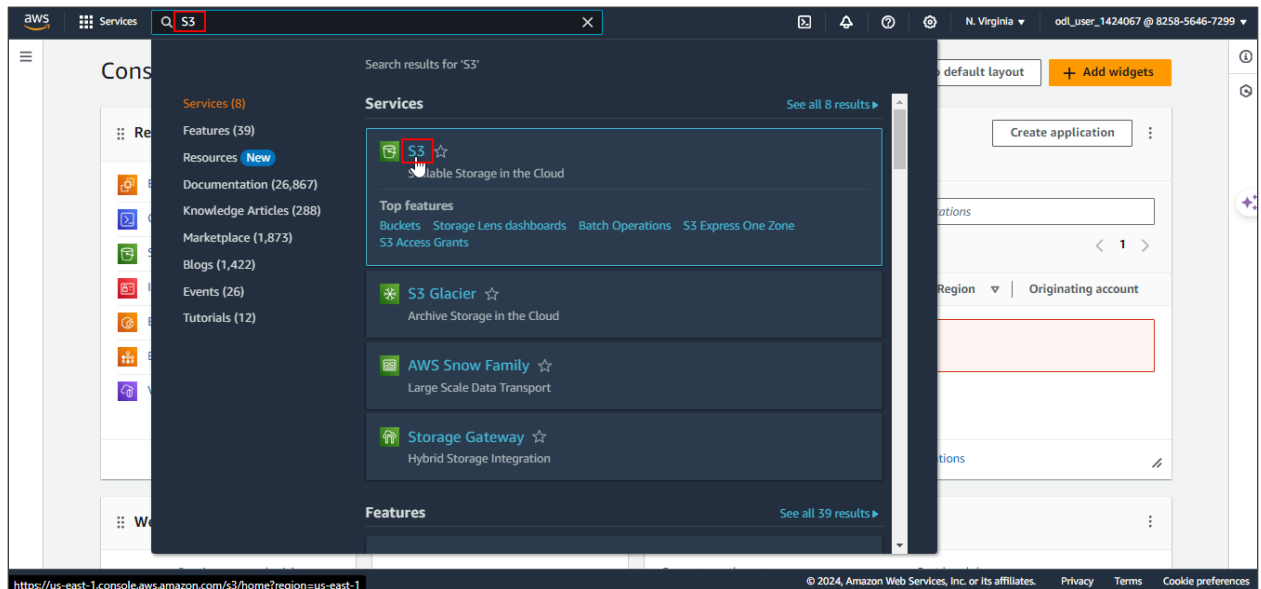
Expected deliverables: A fully functional Lambda function that copies content between specified S3 buckets, with necessary S3 configurations, event triggers, permissions, and documentation, along with test cases, to ensure proper functionality.

Steps to be followed:

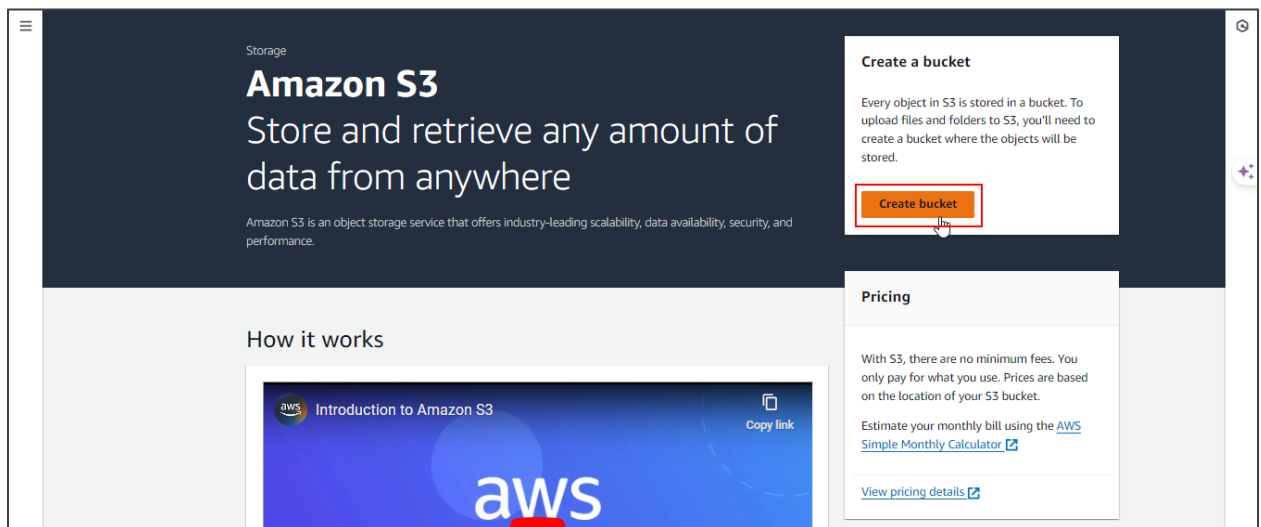
1. Create an S3 bucket
2. Create a Lambda function
3. Test the Lambda function
4. Create a trigger
5. Add a destination

Step 1: Create an S3 Bucket

1.1 Navigate to the AWS Management Console and search for and click on S3



1.2 Click on Create bucket



1.3 Name the bucket as **myawsamplebucketname**

General configuration

AWS Region
US East (N. Virginia) us-east-1

Bucket type [Info](#)

☒ **General purpose**
Recommended for most use cases and access patterns. General purpose buckets are the original S3 bucket type. They allow a mix of storage classes that redundantly store objects across multiple Availability Zones.

☐ **Directory - New**
Recommended for low-latency use cases. These buckets use only the S3 Express One Zone storage class, which provides faster processing of data within a single Availability Zone.

Bucket name [Info](#)
myawsamplebucketname

Bucket name must be unique within the global namespace and follow the bucket naming rules. [See rules for bucket naming](#)

Copy settings from existing bucket - *optional*
Only the bucket settings in the following configuration are copied.

Format: s3://bucket/prefix

Object Ownership [Info](#)
Control ownership of objects written to this bucket from other AWS accounts and the use of access control lists (ACLs). Object ownership determines who can specify access to objects.

1.4 Ensure that **Block all public access** is unselected and **acknowledge** the configuration by checking the box

☐ **Block all public access**
Turning this setting on is the same as turning on all four settings below. Each of the following settings are independent of one another.

- ☐ **Block public access to buckets and objects granted through new access control lists (ACLs)**
S3 will block public access permissions applied to newly added buckets or objects, and prevent the creation of new public access ACLs for existing buckets and objects. This setting doesn't change any existing permissions that allow public access to S3 resources using ACLs.
- ☐ **Block public access to buckets and objects granted through any access control lists (ACLs)**
S3 will ignore all ACLs that grant public access to buckets and objects.
- ☐ **Block public access to buckets and objects granted through new public bucket or access point policies**
S3 will block new bucket and access point policies that grant public access to buckets and objects. This setting doesn't change any existing policies that allow public access to S3 resources.
- ☐ **Block public and cross-account access to buckets and objects through any public bucket or access point policies**
S3 will ignore public and cross-account access for buckets or access points with policies that grant public access to buckets and objects.

Turning off block all public access might result in this bucket and the objects within becoming public
AWS recommends that you turn on block all public access, unless public access is required for specific and verified use cases such as static website hosting.

☒ I acknowledge that the current settings might result in this bucket and the objects within becoming public.

1.5 Click on **Create bucket**

Default encryption [Info](#)
Server-side encryption is automatically applied to new objects stored in this bucket.

Encryption type [Info](#)

- ☒ Server-side encryption with Amazon S3 managed keys (SSE-S3)
- ☐ Server-side encryption with AWS Key Management Service keys (SSE-KMS)
- ☐ Dual-layer server-side encryption with AWS Key Management Service keys (DSSE-KMS)
Secure your objects with two separate layers of encryption. For details on pricing, see [DSSE-KMS pricing](#) on the [Storage](#) tab of the [Amazon S3 pricing page](#).

Bucket Key
Using an S3 Bucket Key for SSE-KMS reduces encryption costs by lowering calls to AWS KMS. S3 Bucket Keys aren't supported for DSSE-KMS. [Learn more](#)

- ☐ Disable
- ☒ Enable

► **Advanced settings**

ⓘ After creating the bucket, you can upload files and folders to the bucket, and configure additional bucket settings.

Cancel **Create bucket**

Successfully created bucket "myawsamplebucketname"
To upload files and folders, or to configure additional bucket settings, choose [View details](#).

[Amazon S3](#) > Buckets

► **Account snapshot - updated every 24 hours** [All AWS Regions](#) [View Storage Lens dashboard](#)
Storage lens provides visibility into storage usage and activity trends. [Learn more](#)

[General purpose buckets](#) | [Directory buckets](#)

General purpose buckets (1) [Info](#) [All AWS Regions](#)

Buckets are containers for data stored in S3.

Find buckets by name

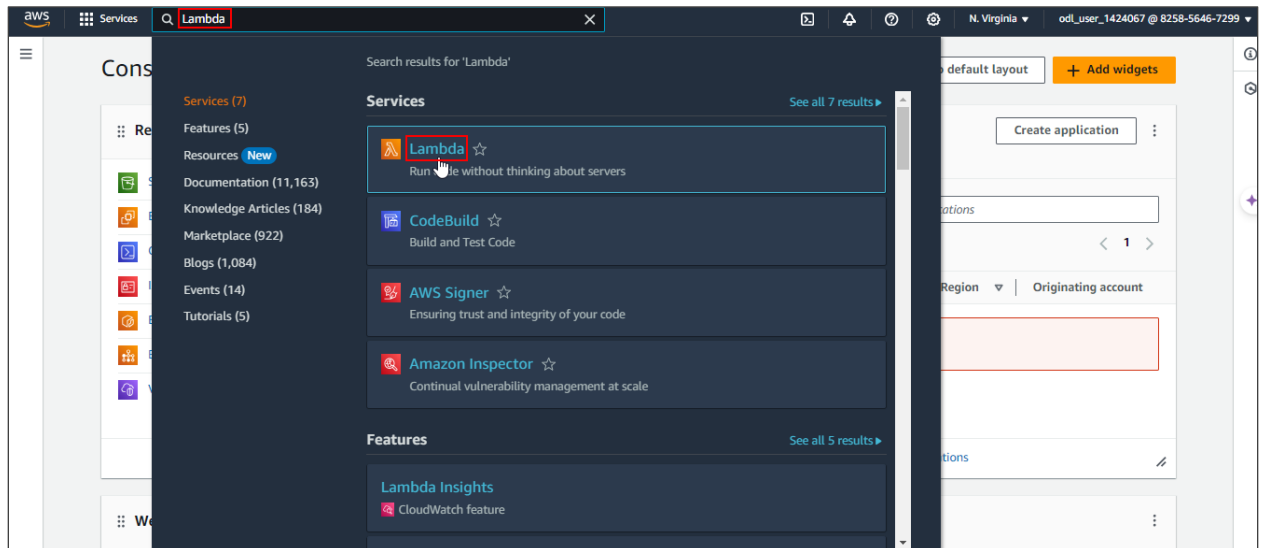
◀ 1 ▶

	Name	AWS Region	IAM Access Analyzer	Creation date
<input type="radio"/>	myawsamplebucketname	US East (N. Virginia) us-east-1	View analyzer for us-east-1	August 14, 2024, 03:04:04 (UTC+05:30)

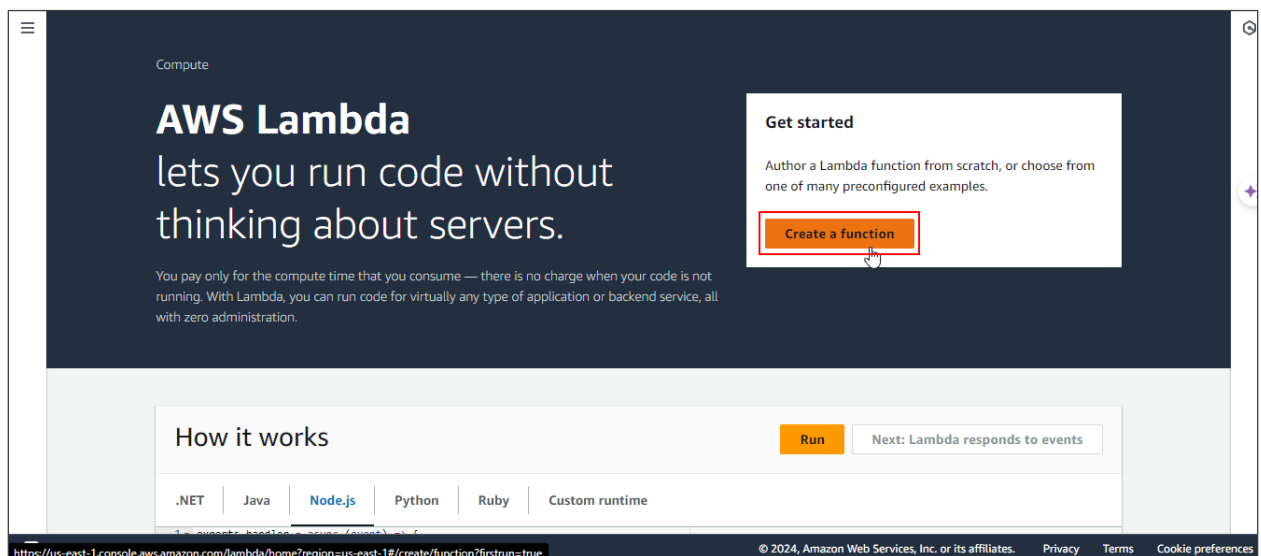
myawsamplebucketname has been created successfully.

Step 2: Create a Lambda function

2.1 Navigate to the AWS console home dashboard, search for and click on **Lambda** as shown



2.2 Click on **Create a function**



2.3 Provide a desired name for the function and choose **Python 3.12** from the **Runtime** option

[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.

 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.
 [Refresh](#)

Architecture [Info](#)

2.4 Click on **Create function**

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.
 [Refresh](#)

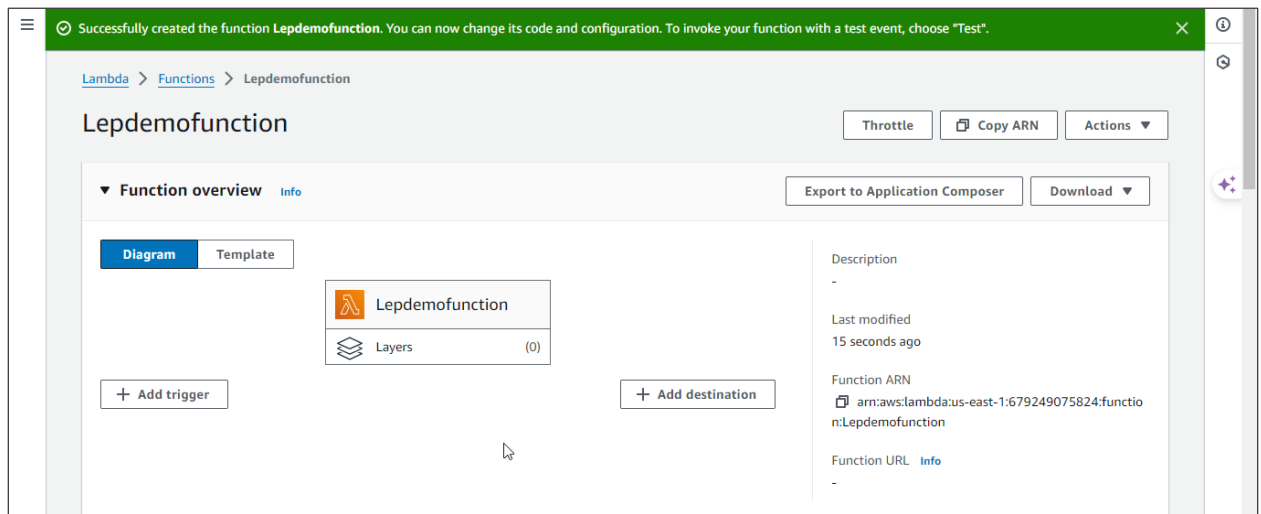
Architecture [Info](#)
 Choose the instruction set architecture you want for your function code.
☒ x86_64
☐ arm64

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](#)

► [Advanced settings](#)

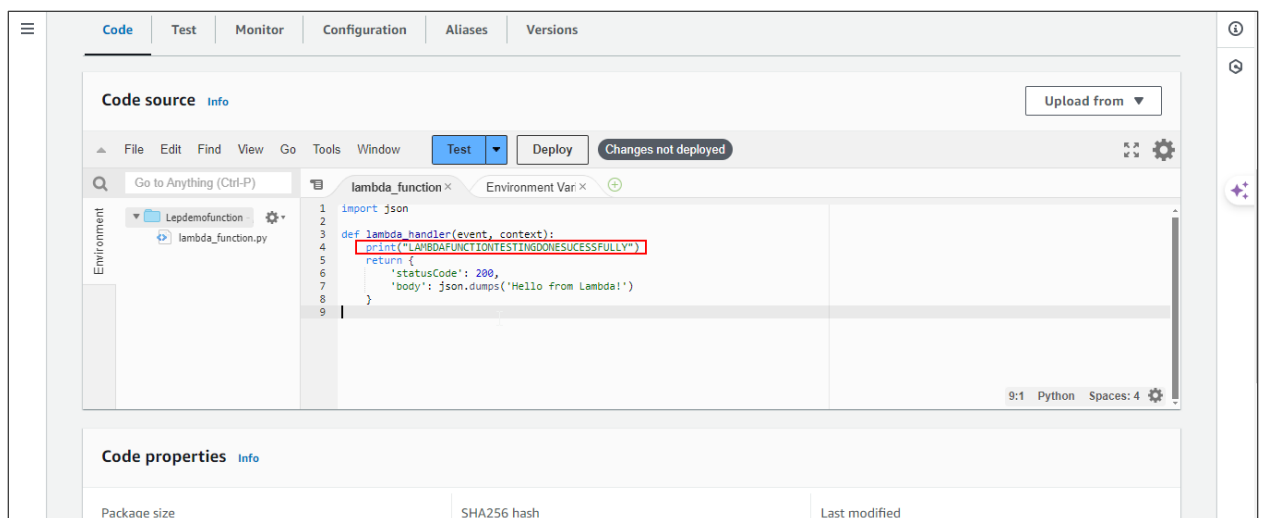
Cancel [Create function](#)



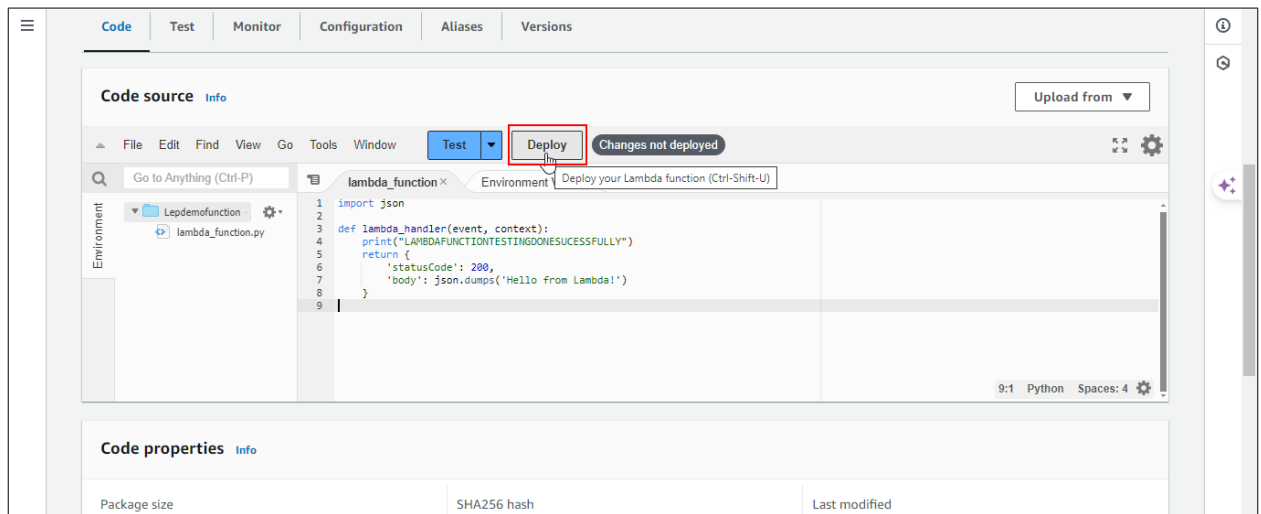
The lambda function has been created successfully.

Step 3: Test the Lambda Function

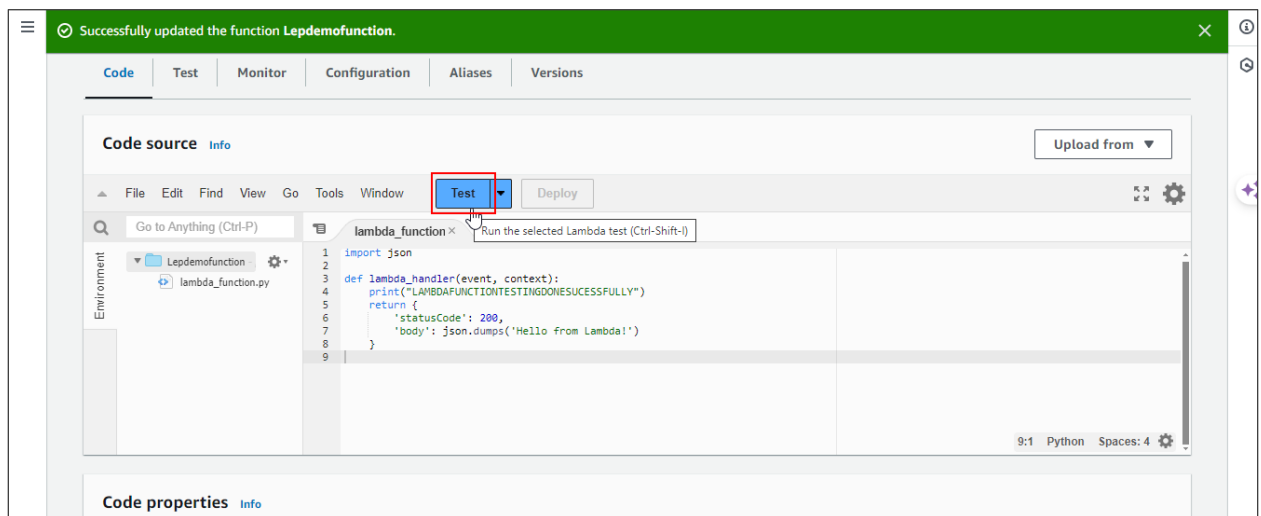
- 3.1 Add the following print statement to the source code as shown:
`print("LAMBDAFUNCTIONTESTINGDONESUCCESSFULLY")`



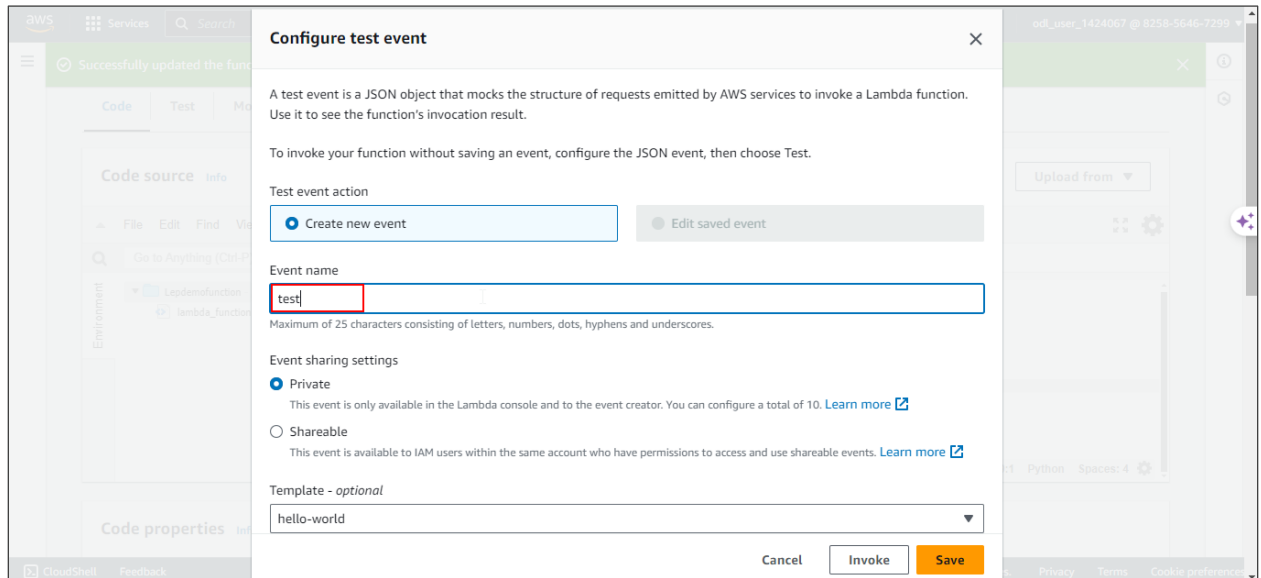
3.2 Click on **Deploy**



3.3 Click on **Test**

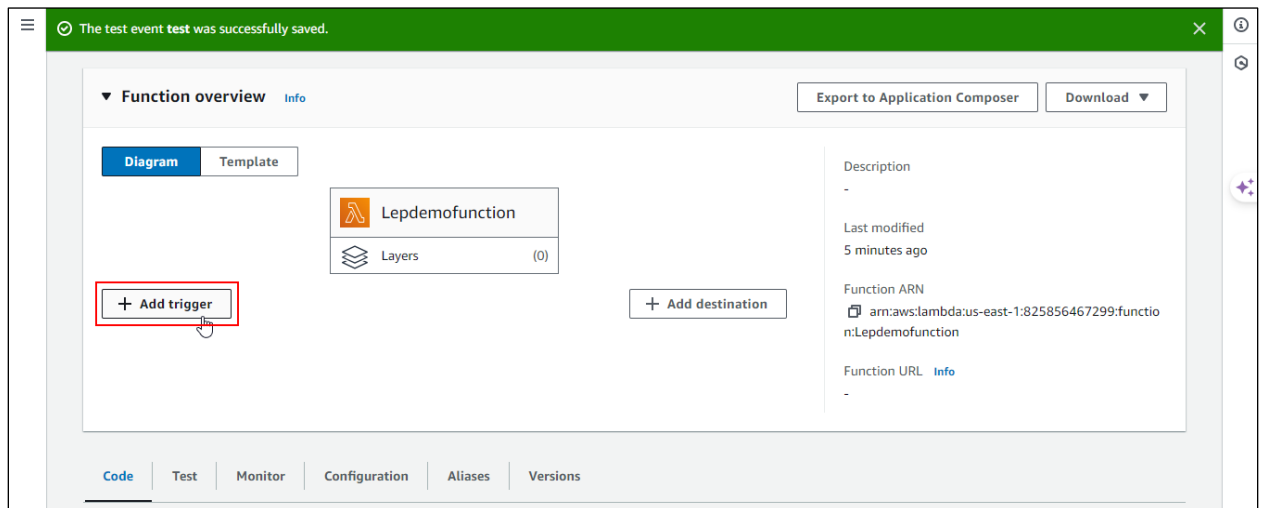


3.4 Add **test** as the **Event name** and click on **Save**

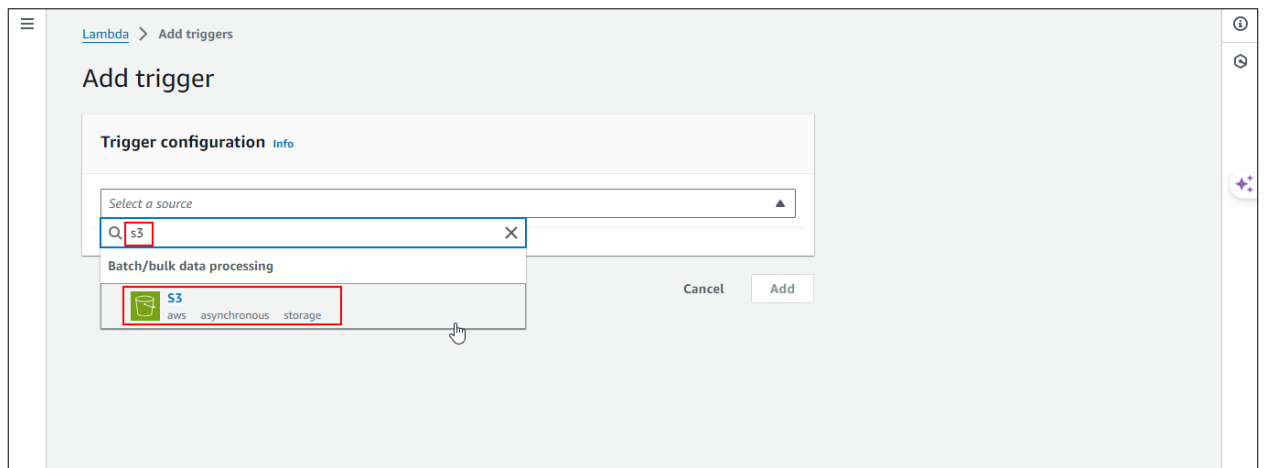


Step 4: Create a Trigger

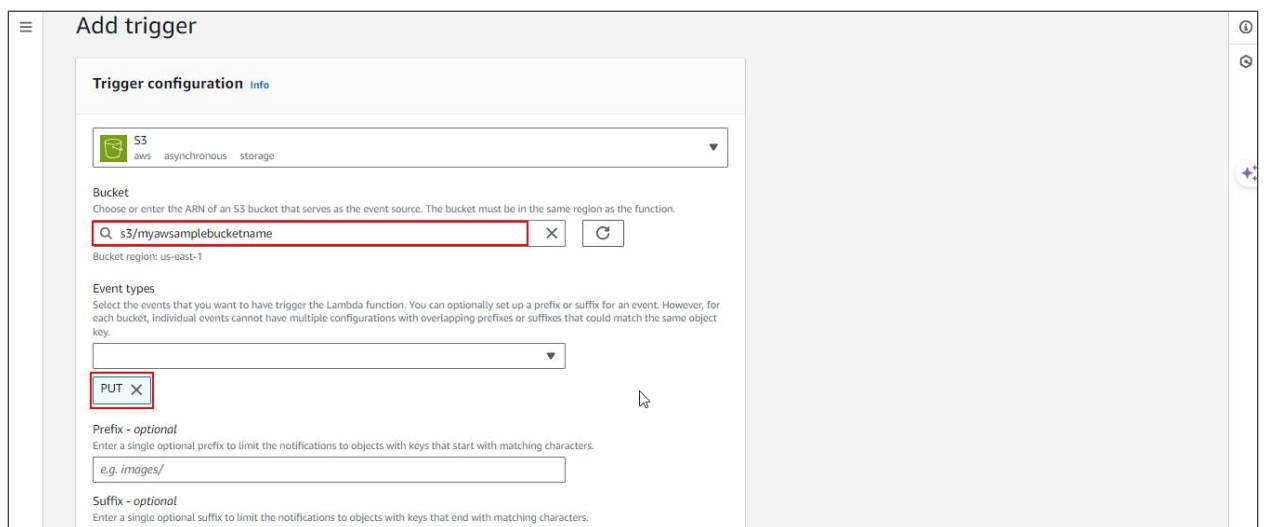
4.1 Click on + **Add trigger** in the **Lambda functions** dashboard



4.2 Search and select the **S3** bucket in the **Trigger configuration**



4.3 Choose the created **S3 bucket** from the **Bucket** dropdown and select **PUT** as the **Event types**



4.4 Check the dialog box and click **Add**

PUT X

Prefix - optional
Enter a single optional prefix to limit the notifications to objects with keys that start with matching characters.
e.g. images/

Suffix - optional
Enter a single optional suffix to limit the notifications to objects with keys that end with matching characters.
e.g. .jpg

Recursive invocation
If your function writes objects to an S3 bucket, ensure that you are using different S3 buckets for input and output. Writing to the same bucket increases the risk of creating a recursive invocation, which can result in increased Lambda usage and increased costs. [Learn more](#)

☒ I acknowledge that using the same S3 bucket for both input and output is not recommended and that this configuration can cause recursive invocations, increased Lambda usage, and increased costs.

Lambda will add the necessary permissions for AWS S3 to invoke your Lambda function from this trigger. [Learn more](#) about the Lambda permissions model.

Cancel **Add**

Lambda > Functions > Lepdemofunction

Lepdemofunction Throttle Copy ARN Actions

✓ The trigger myawsamplebucketname was successfully added to function Lepdemofunction. The function is now receiving events from the trigger.

▼ **Function overview** Info Export to Application Composer Download

Diagram Template

S3 + Add trigger

Lepdemofunction Layers (0)

+ Add destination

Description -

Last modified 12 minutes ago

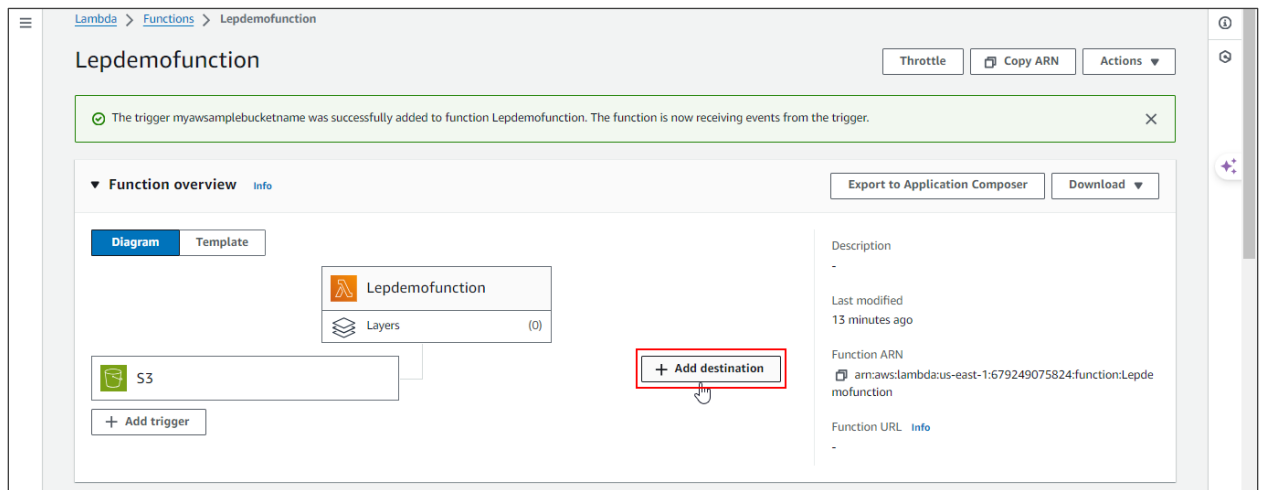
Function ARN arn:aws:lambda:us-east-1:679249075824:function:Lepdemofunction

Function URL [Info](#)

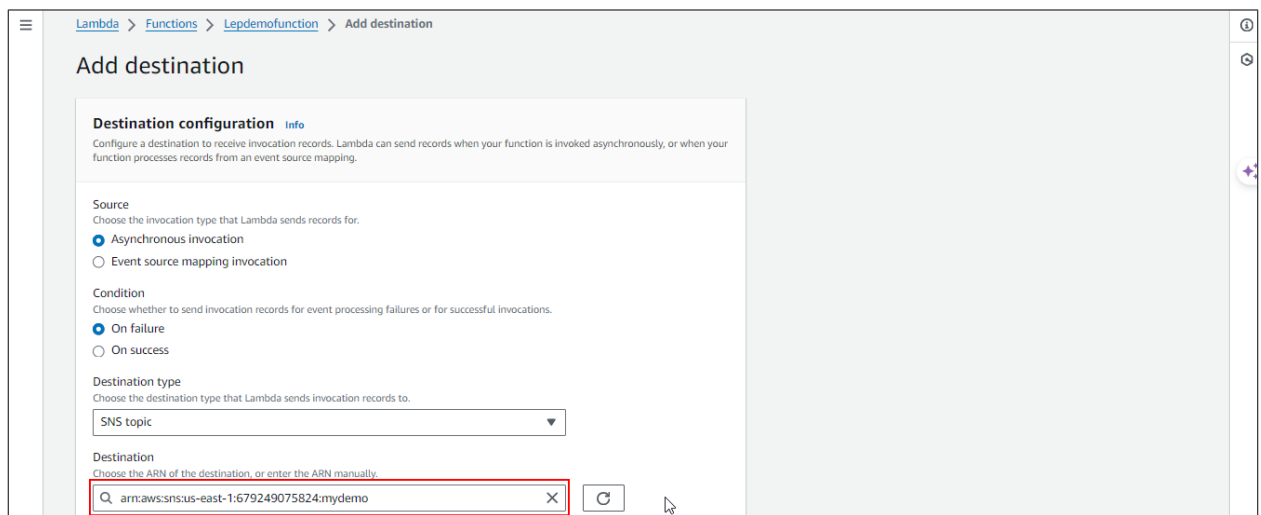
The trigger of the S3 bucket has been successfully added.

Step 5: Add a destination

5.1 Click on **Add destination**



5.2 Add **Destination** in the following format: **arn:aws:sns:region:account-id:topic-name**



Note: Replace **region**, **account-id**, and **topic-name** with the appropriate values for your setup

Note: Ensure that you have chosen or created the topic with the targeted subscribers

5.3 Click on the **Save** button

Source
Choose the invocation type that Lambda sends records for.

☒ Asynchronous invocation
☐ Event source mapping invocation

Condition
Choose whether to send invocation records for event processing failures or for successful invocations.

☒ On failure
☐ On success

Destination type
Choose the destination type that Lambda sends invocation records to.

SNS topic

Destination
Choose the ARN of the destination, or enter the ARN manually.

arn:aws:sns:us-east-1:679249075824:mydemo

Permissions
If your execution role doesn't have the required permissions for the selected destination, then Lambda will attempt to add the permissions to the role.

Cancel **Save**

You will see the following interface:

Your changes have been saved.

Lambda > Functions > Lepdemofunction

Lepdemofunction

Throttle Copy ARN Actions

▼ **Function overview** Info

Export to Application Composer Download

Diagram Template

Lepdemofunction

Layers (0)

S3

+ Add trigger

Amazon SNS

+ Add destination

Description

Last modified 27 minutes ago

Function ARN
arn:aws:lambda:us-east-1:679249075824:function:Lepdemofunction

Function URL Info

Code Test Monitor **Configuration** Aliases Versions

By following these steps, you have successfully created a Lambda function that automates the replication of S3 bucket contents across different AWS regions, ensuring robust data availability.