Name:Krushikesh Shelar 2024-2025 D15C Roll No:51

Aim: To create a Lambda function which will log “An Image has been added” once you add an object to a specific bucket in S3

Theory:

Overview of AWS Lambda

AWS Lambda is a serverless computing service that allows you to run code without managing servers. It automatically scales in response to incoming requests or events, ensuring efficient resource use. You are billed only for the time your code is executed, making it a budget-friendly option for on-demand applications.

How AWS Lambda Works

1. **Function Development**: Begin by writing your code and designating a handler (the entry point). You can use the AWS Management Console, CLI, or upload a package.
2. **Trigger Setup**: Determine the events that will trigger your function, such as uploads to S3 or changes in a DynamoDB table.
3. **Execution**: Upon triggering, Lambda runs your function and handles the necessary logic while automatically scaling to accommodate the volume of events.
4. **Scaling and Concurrency**: Lambda dynamically scales by creating additional instances of your function for simultaneous requests. You can also configure reserved concurrency to manage peak traffic.
5. **Monitoring and Logging**: Integrated with Amazon CloudWatch, Lambda provides logging and monitoring capabilities. You can view logs for each execution, helping you track performance and troubleshoot issues.

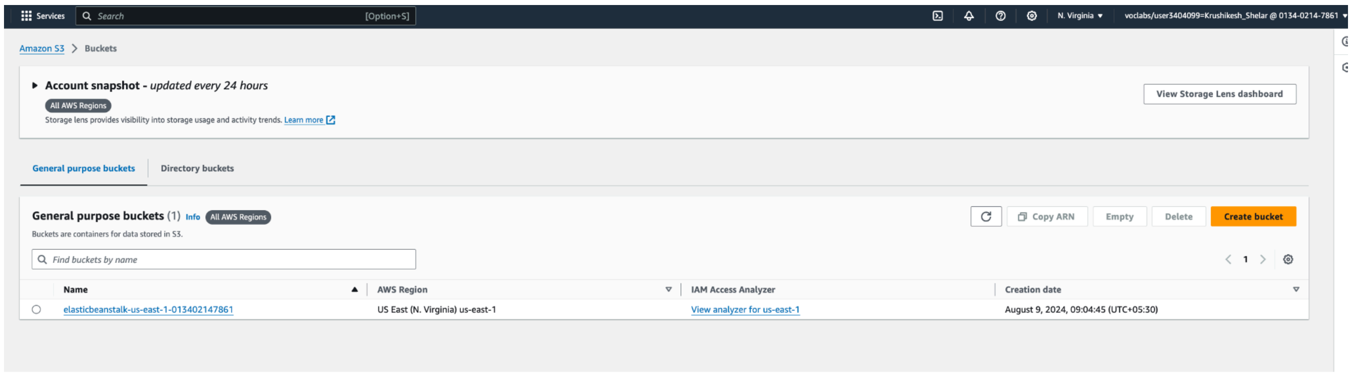
Programming Languages Supported

- **Python**: Excellent for quick development and lightweight tasks due to its extensive standard library.
- **Java**: Suitable for more demanding, compute-intensive applications, though it may have longer cold start times.
- **Node.js**: Well-suited for I/O-heavy applications like APIs, offering fast startup times and efficient resource use.

Steps To create the lambda function:

Name:Krushikesh Shelar **2024-2025** **D15C** **Roll No:51**

Step 1: Login to your AWS Academy account. Now open S3 from services and click on create S3 bucket.



Services | Search [Option+S] | N. Virginia | voclabs/user3404099=Krushikesh_Shelar @ 0134-0214-7861

Amazon S3 > Buckets

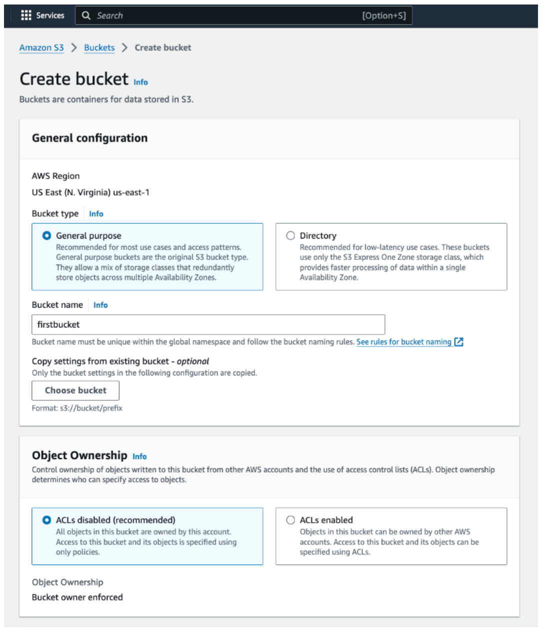
Account snapshot - *updated every 24 hours*
All AWS Regions
Storage lens provides visibility into storage usage and activity trends. Learn more
View Storage Lens dashboard

General purpose buckets | Directory buckets

General purpose buckets (1) Info All AWS Regions
Buckets are containers for data stored in S3.
Copy ARN | Empty | Delete | Create bucket
Find buckets by name

Name	AWS Region	IAM Access Analyzer	Creation date
elasticbeanstalk-us-east-1-013402147861	US East (N. Virginia) us-east-1	View analyzer for us-east-1	August 9, 2024, 09:04:45 (UTC+05:30)

Step 2: Now Give a name to the Bucket, select general purpose project and deselect the Block public access and keep other this to default.



Services | Search [Option+S]

Amazon S3 > Buckets > Create bucket

Create bucket Info
Buckets are containers for data stored in S3.

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 — 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
firstbucket
Bucket name must be unique within the global namespace and follow the bucket naming rules. See rules for bucket naming

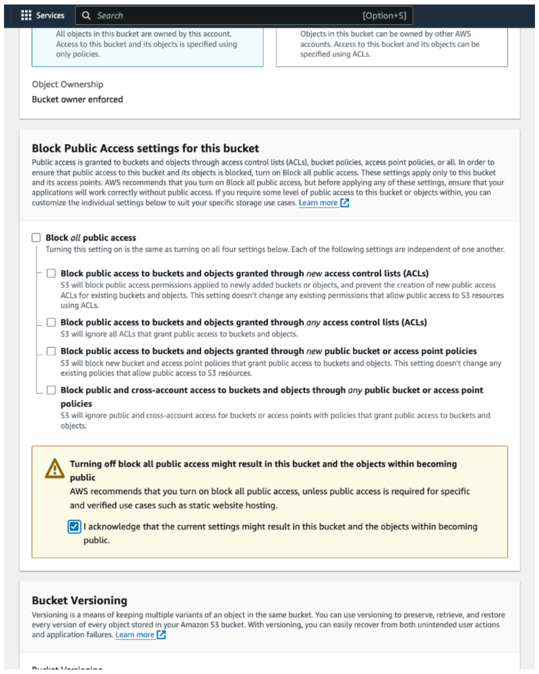
Copy settings from existing bucket - *optional*
Only the bucket settings in the following configuration are copied.
Choose bucket
Format: s3://bucket/prefix

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.

[✓ ACLs disabled (recommended)] All objects in this bucket are owned by this account. Access to this bucket and its objects is specified using only policies.

ACLs enabled — Objects in this bucket can be owned by other AWS accounts. Access to this bucket and its objects can be specified using ACLs.

Object Ownership
Bucket owner enforced



Services | Search [Option+S]

All objects in this bucket are owned by this account. Access to this bucket and its objects is specified using only policies.

Objects in this bucket can be owned by other AWS accounts. Access to this bucket and its objects can be specified using ACLs.

Object Ownership
Bucket owner enforced

Block Public Access settings for this bucket
Public access is granted to buckets and objects through access control lists (ACLs), bucket policies, access point policies, or all. In order to ensure that public access to this bucket and its objects is blocked, turn on Block all public access. These settings apply only to this bucket and its access points. AWS recommends that you turn on Block all public access, but before applying any of these settings, ensure that your applications will work correctly without public access. If you require some level of public access to this bucket or objects within, you can customize the individual settings below to suit your specific storage use cases. Learn more

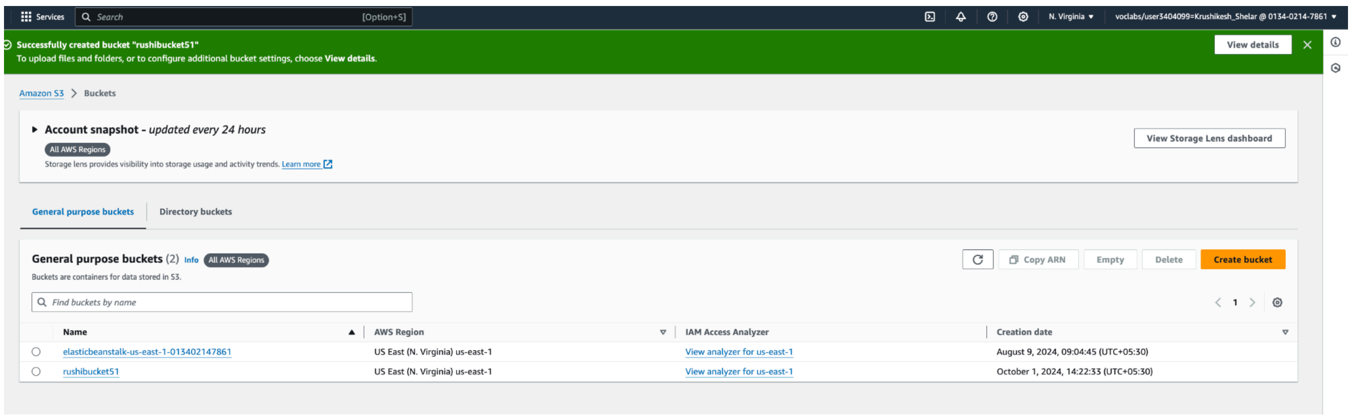
☐ **Block *all* public access**
Turning this setting on is the same as turning on all four settings below. Each of the following settings are independent of one another.

- ☐ **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.

Bucket Versioning
Versioning is a means of keeping multiple variants of an object in the same bucket. You can use versioning to preserve, retrieve, and restore every version of every object stored in your Amazon S3 bucket. With versioning, you can easily recover from both unintended user actions and application failures. Learn more



Services | Search [Option+S] | N. Virginia | voclabs/user3404099=Krushikesh_Shelar @ 0134-0214-7861

Successfully created bucket "rushibucket51"
To upload files and folders, or to configure additional bucket settings, choose **View details**.
View details

Amazon S3 > Buckets

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

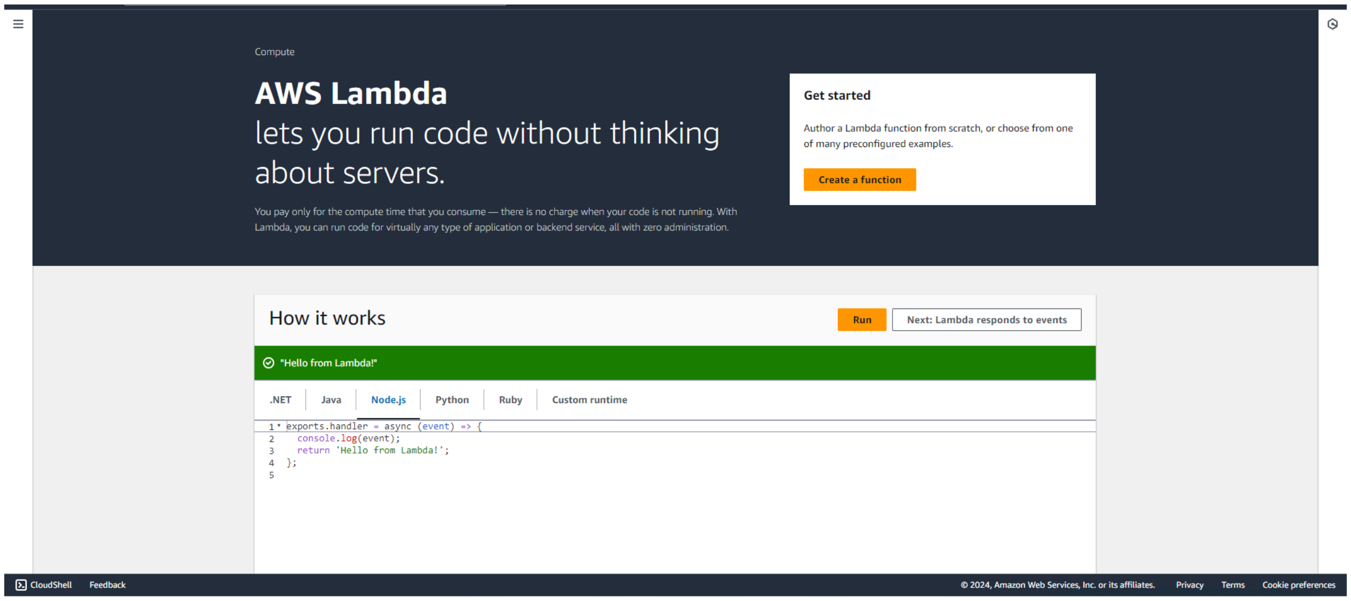
General purpose buckets | Directory buckets

General purpose buckets (2) Info All AWS Regions
Buckets are containers for data stored in S3.
Copy ARN | Empty | Delete | Create bucket
Find buckets by name

Name	AWS Region	IAM Access Analyzer	Creation date
elasticbeanstalk-us-east-1-013402147861	US East (N. Virginia) us-east-1	View analyzer for us-east-1	August 9, 2024, 09:04:45 (UTC+05:30)
rushibucket51	US East (N. Virginia) us-east-1	View analyzer for us-east-1	October 1, 2024, 14:22:33 (UTC+05:30)

Name:Krushikesh Shelar 2024-2025 D15C Roll No:51

Step 3: Open lambda console and click on create function button.


Compute

AWS Lambda
lets you run code without thinking about servers.

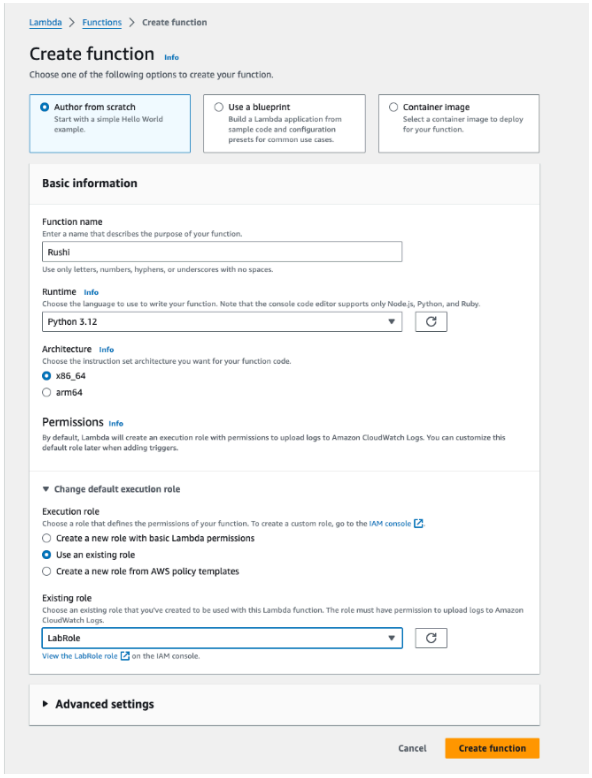
You pay only for the compute time that you consume — there is no charge when your code is not running. With Lambda, you can run code for virtually any type of application or backend service, all with zero administration.

Get started

Author a Lambda function from scratch, or choose from one of many preconfigured examples.

```
exports.handler = async (event) => {
  console.log(event);
  return 'Hello from Lambda!';
};
```

Step 4: Now Give a name to your Lambda function, Select the language to use to write your function. Note that the console code editor supports only Node.js, Python, and Ruby. So will select Python 3.12 , Architecture as x86, and Exceution role to Create a new role with basic Lambda permissions.



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.
Rushi
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

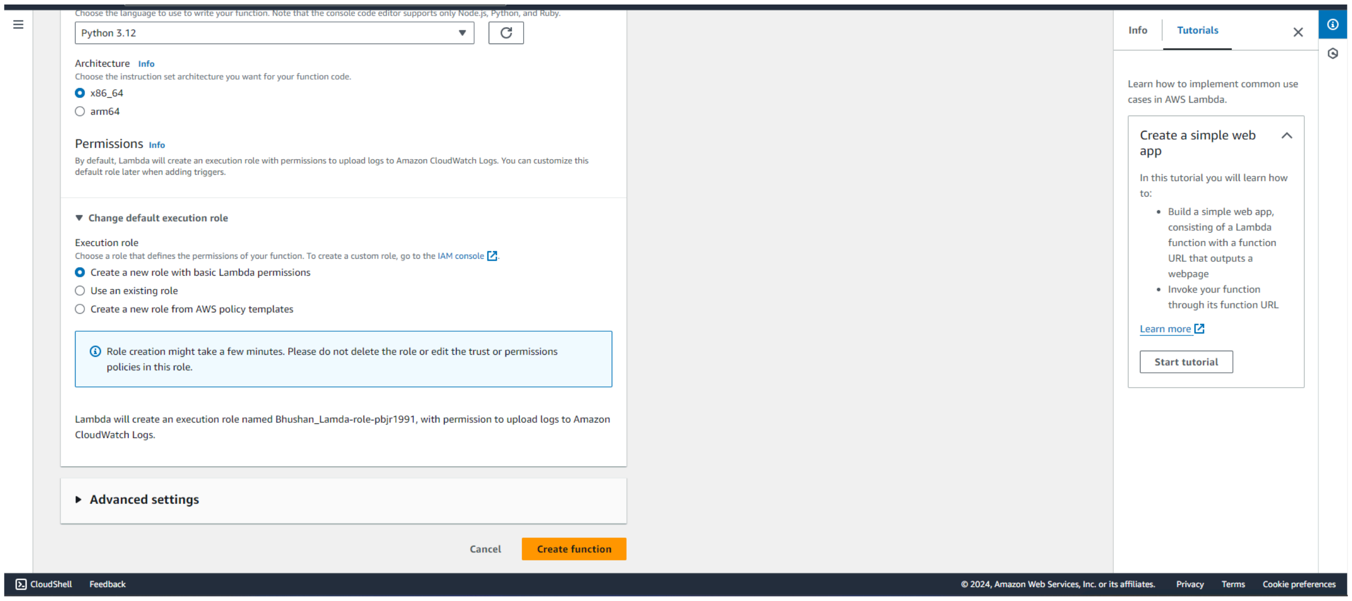
Permissions Info
By default, Lambda will create an execution role with permissions to upload logs to Amazon CloudWatch Logs. You can customize this default role later when adding triggers.

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

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.

Name:Krushikesh Shelar 2024-2025 D15C Roll No:51



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

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 Bhushan_Lamda-role-pbjr1991, with permission to upload logs to Amazon CloudWatch Logs.

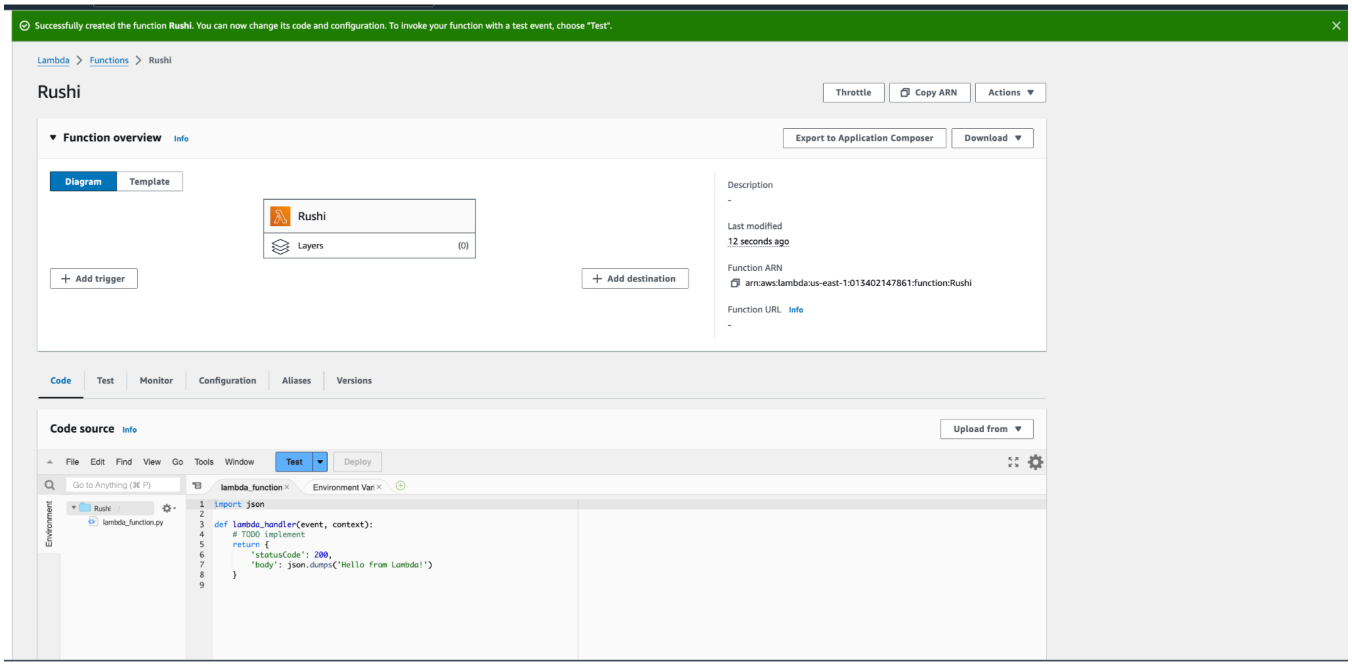
Advanced settings

Learn how to implement common use cases in AWS Lambda.

Create a simple web app

In this tutorial you will learn how to:
- Build a simple web app, consisting of a Lambda function with a function URL that outputs a webpage
- Invoke your function through its function URL

Lambda function code:



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

Lambda > Functions > Rushi

Rushi

Function overview Info

Description
-

Last modified
12 seconds ago

Function ARN
arn:aws:lambda:us-east-1:013402147861:function:Rushi

Function URL Info
-

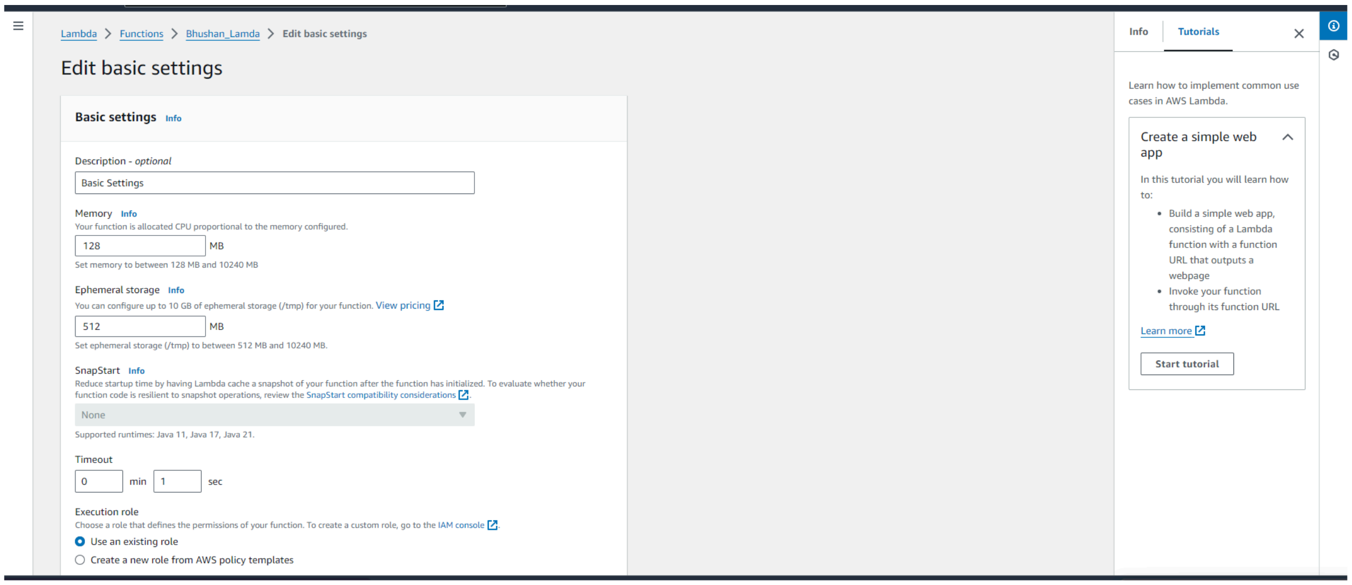
Code source Info

```
import json

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

So See or Edit the basic settings go to configuration then click on edit general setting. 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.

Step 5: Now Click on the Test tab then select Create a new event, give a name to the event and select Event Sharing to private, and select s3 put template.

Name:Krushikesh Shelar **2024-2025** **D15C** **Roll No:51**


Lambda > Functions > Bhushan_Lamda > Edit basic settings

Edit basic settings

Basic settings Info

Description - *optional*

Basic Settings

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 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

Info | Tutorials

Learn how to implement common use cases in AWS Lambda.

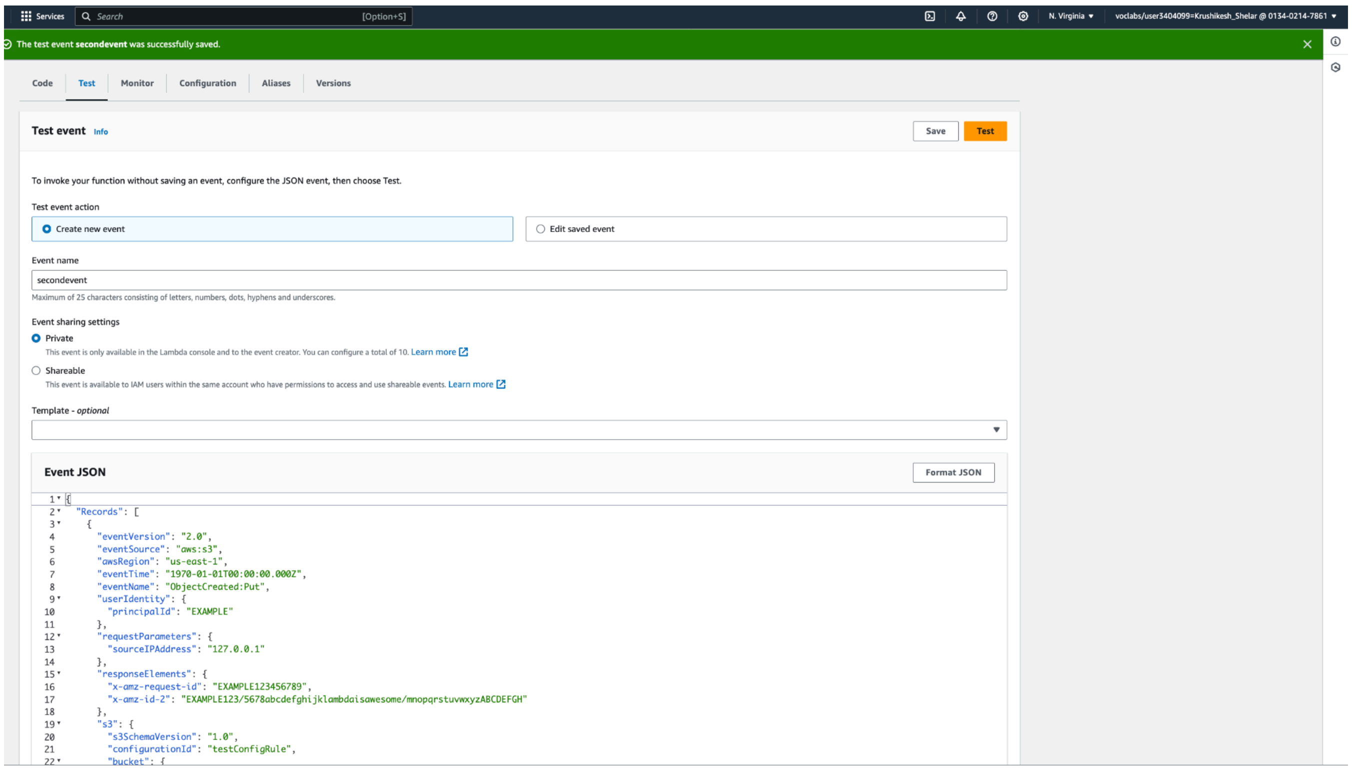
Create a simple web app

In this tutorial you will learn how to:

- Build a simple web app, consisting of a Lambda function with a function URL that outputs a webpage
- Invoke your function through its function URL

Learn more

Start tutorial


Services | Search [Option+S] | N. Virginia | voclabs/user3404099=Krushikesh_Shelar @ 0134-0214-7861

The test event **secondevent** was successfully saved.

Code | Test | Monitor | Configuration | Aliases | Versions

Test event Info   Save Test

To invoke your function without saving an event, configure the JSON event, then choose Test.

Test event action

[✓ Create new event] Edit saved event

Event name

secondevent

Maximum of 25 characters consisting of letters, numbers, dots, hyphens and underscores.

Event sharing settings

[✓ Private]

This event is only available in the Lambda console and to the event creator. You can configure a total of 10. Learn more

Shareable

This event is available to IAM users within the same account who have permissions to access and use shareable events. Learn more

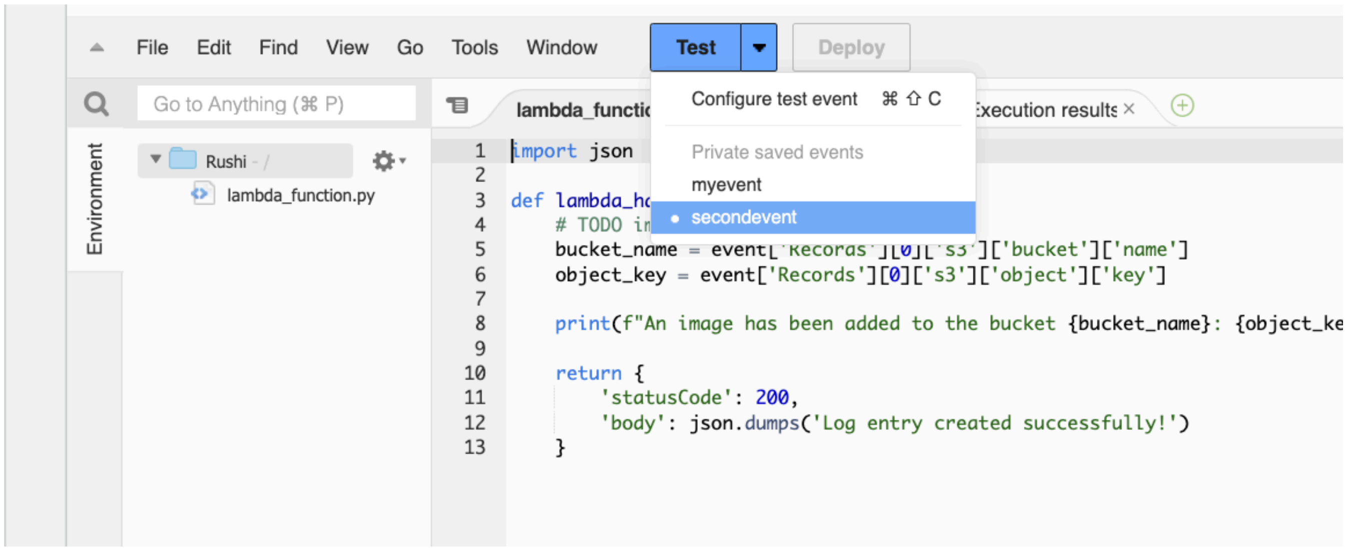
Template - *optional*

Event JSON   Format JSON

```
{
  "Records": [
    {
      "eventVersion": "2.0",
      "eventSource": "aws:s3",
      "awsRegion": "us-east-1",
      "eventTime": "1970-01-01T00:00:00.000Z",
      "eventName": "ObjectCreated:Put",
      "userIdentity": {
        "principalId": "EXAMPLE"
      },
      "requestParameters": {
        "sourceIPAddress": "127.0.0.1"
      },
      "responseElements": {
        "x-amz-request-id": "EXAMPLE123456789",
        "x-amz-id-2": "EXAMPLE123/5678abcdefghijklambdaisawesome/mnopqrstuvwxyzABCDEFGH"
      },
      "s3": {
        "s3SchemaVersion": "1.0",
        "configurationId": "testConfigRule",
        "bucket": {
```

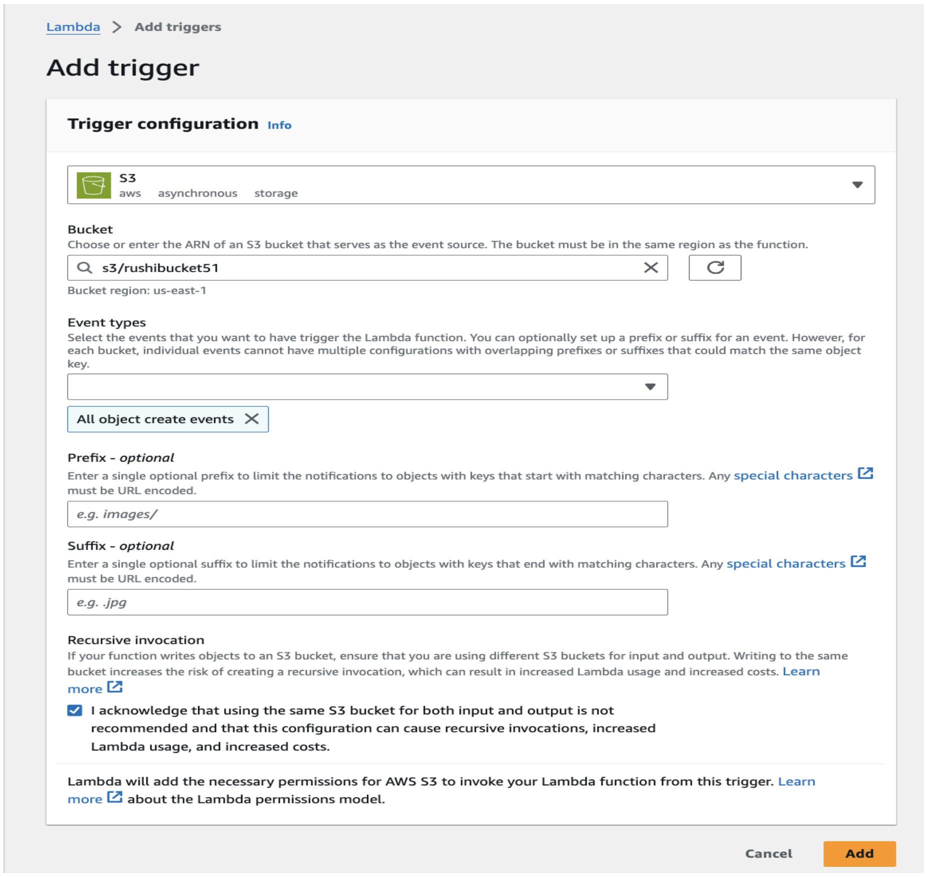
Step 6: Now In Code section select the created event from the dropdown .

Name:Krushikesh Shelar 2024-2025 D15C Roll No:51



```
import json

def lambda_ha
    # TODO im
    bucket_name = event['Records'][0]['s3']['bucket']['name']
    object_key = event['Records'][0]['s3']['object']['key']

    print(f"An image has been added to the bucket {bucket_name}: {object_ke

    return {
        'statusCode': 200,
        'body': json.dumps('Log entry created successfully!')
    }
```

Step 7: Now In the Lambda function click on add tigger.
Now select the source as S3 then select the bucket name from the dropdown, keep other things to default and also you can add prefix to image.



Add trigger

Trigger configuration Info

S3
aws asynchronous storage

Bucket
Choose or enter the ARN of an S3 bucket that serves as the event source. The bucket must be in the same region as the function.
s3/rushibucket51
Bucket region: us-east-1

Event types
Select the events that you want to have trigger the Lambda function. You can optionally set up a prefix or suffix for an event. However, for each bucket, individual events cannot have multiple configurations with overlapping prefixes or suffixes that could match the same object key.
All object create events

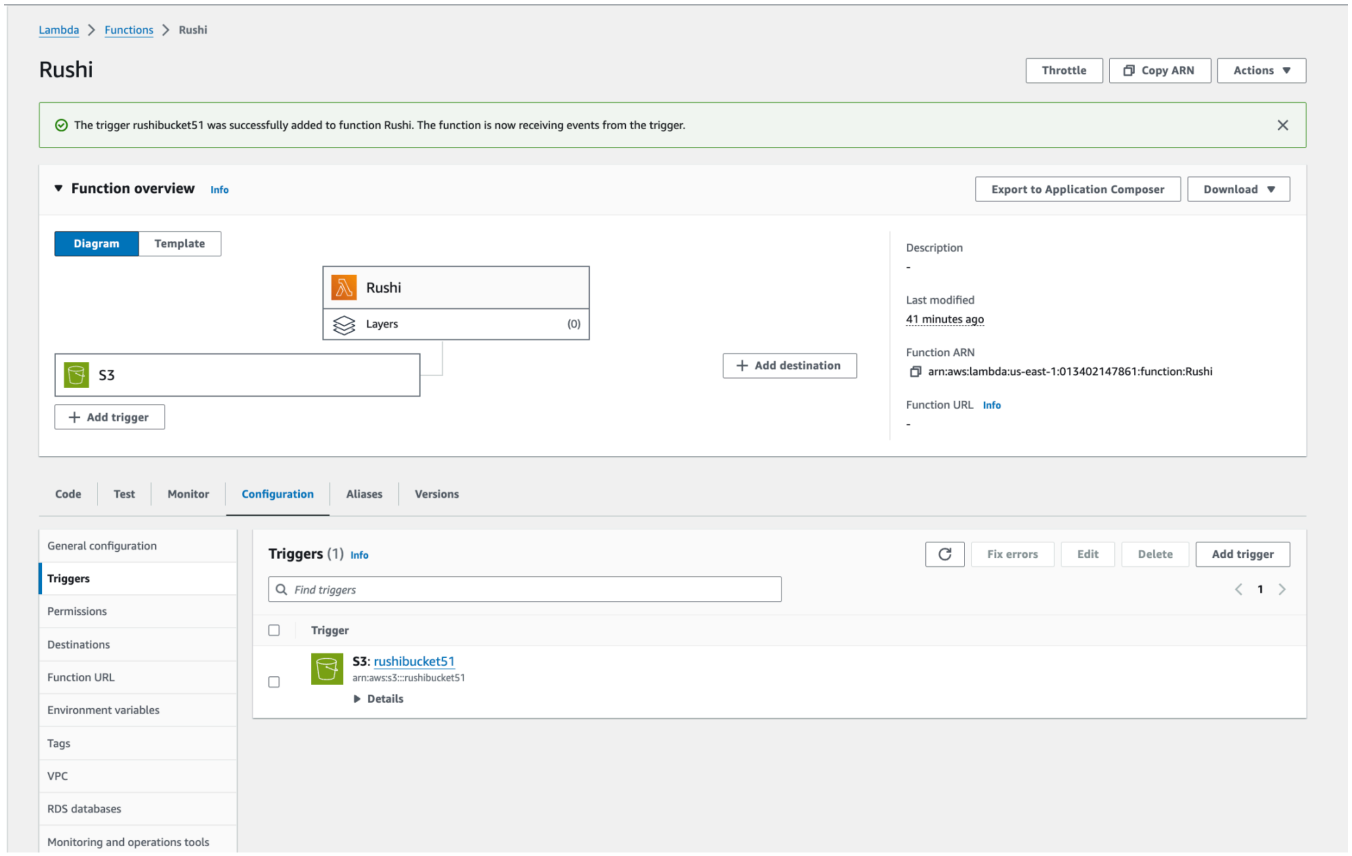
Prefix - *optional*
Enter a single optional prefix to limit the notifications to objects with keys that start with matching characters. Any special characters must be URL encoded.
e.g. images/

Suffix - *optional*
Enter a single optional suffix to limit the notifications to objects with keys that end with matching characters. Any special characters must be URL encoded.
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.



Lambda > Functions > Rushi

Rushi

Throttle | Copy ARN | Actions

The trigger rushibucket51 was successfully added to function Rushi. The function is now receiving events from the trigger.

Function overview Info

Export to Application Composer | Download

Diagram | Template

Rushi
Layers (0)
S3
+ Add trigger
+ Add destination

Description
-

Last modified
41 minutes ago

Function ARN
arn:aws:lambda:us-east-1:013402147861:function:Rushi

Function URL Info
-

Code | Test | Monitor | Configuration | Aliases | Versions

General configuration
Triggers
Permissions
Destinations
Function URL
Environment variables
Tags
VPC
RDS databases
Monitoring and operations tools

Triggers (1) Info

Fix errors | Edit | Delete | Add trigger

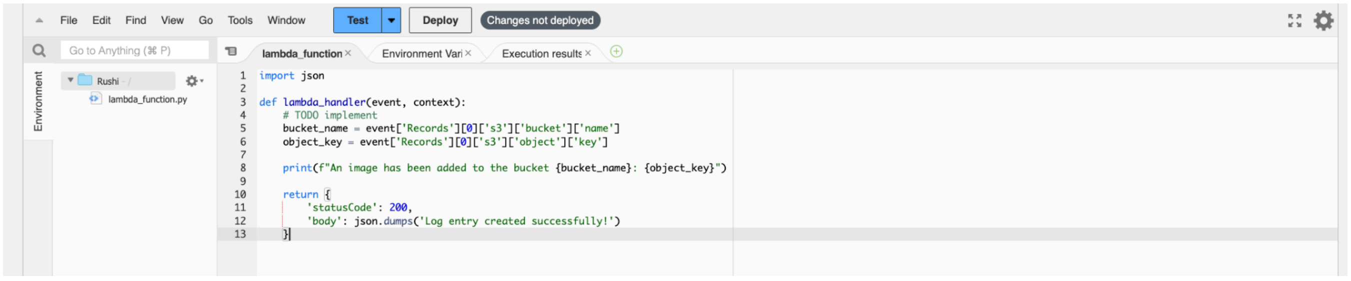
Find triggers

Trigger
S3: rushibucket51 arn:aws:s3:::rushibucket51 Details

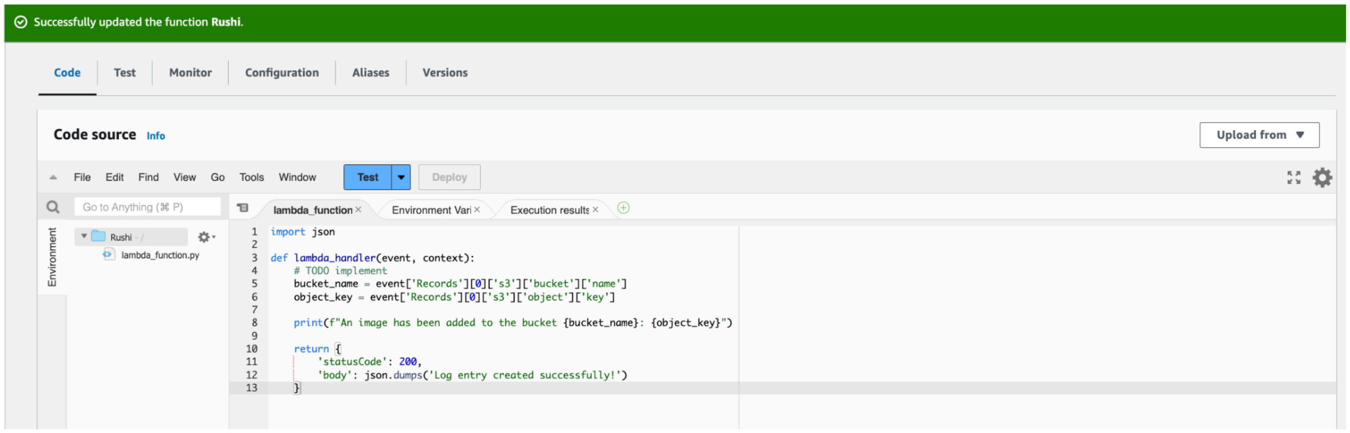
Step 8: Now Write code that logs a message like “An Image has been added” when triggered. Save the file and click on deploy.

```
import json

def lambda_handler(event, context):

    # TODO implement

    bucket_name = event['Records'][0]['s3']['bucket']['name']

    object_key = event['Records'][0]['s3']['object']['key']

    print(f"An image has been added to the bucket {bucket_name}: {object_key}")

    return {

        'statusCode': 200,

        'body': json.dumps('Log entry created successfully!')

    }
```

Name:Krushikesh Shelar **2024-2025** **D15C** **Roll No:51**

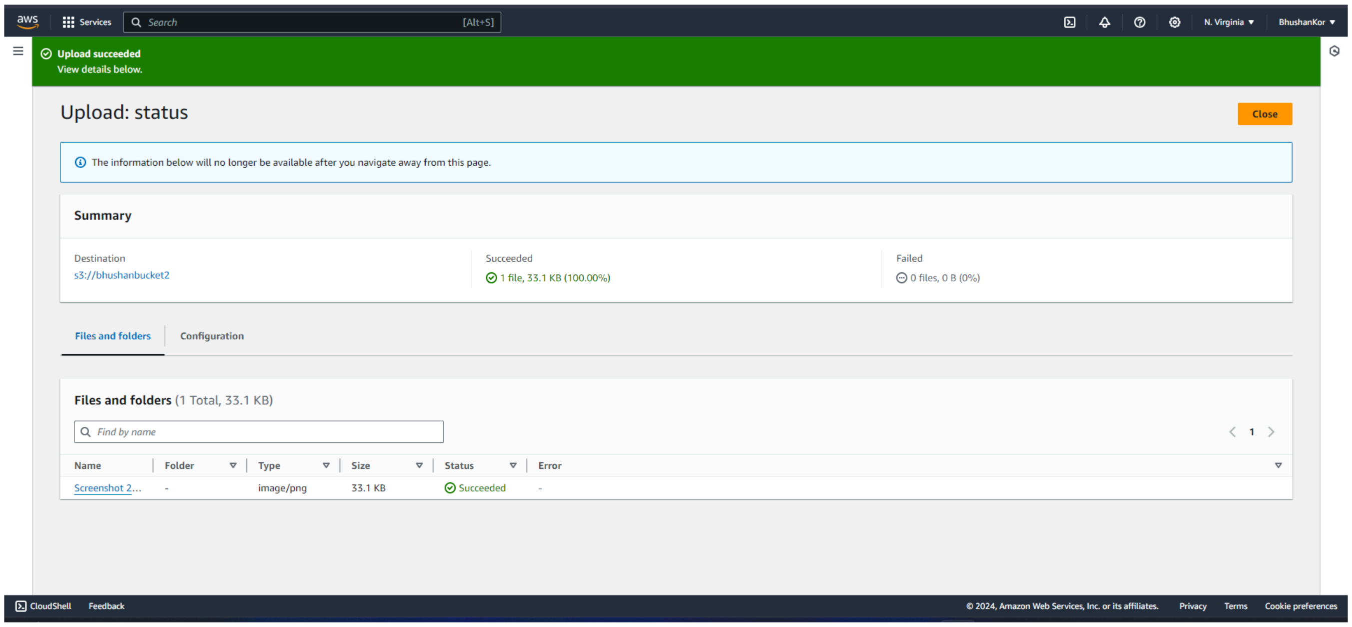


```python
import json

def lambda_handler(event, context):
    # TODO implement
    bucket_name = event['Records'][0]['s3']['bucket']['name']
    object_key = event['Records'][0]['s3']['object']['key']

    print(f"An image has been added to the bucket {bucket_name}: {object_key}")

    return {
        'statusCode': 200,
        'body': json.dumps('Log entry created successfully!')
    }
```


Successfully updated the function **Rushi**.

```python
import json

def lambda_handler(event, context):
    # TODO implement
    bucket_name = event['Records'][0]['s3']['bucket']['name']
    object_key = event['Records'][0]['s3']['object']['key']

    print(f"An image has been added to the bucket {bucket_name}: {object_key}")

    return {
        'statusCode': 200,
        'body': json.dumps('Log entry created successfully!')
    }
```

Step 9: Now upload any image to the bucket.



aws Services Search [Alt+S] N. Virginia BhushanKor
Close
CloudShell Feedback
© 2024, Amazon Web Services, Inc. or its affiliates. Privacy Terms Cookie preferences

Upload succeeded
View details below.

Upload: status

The information below will no longer be available after you navigate away from this page.

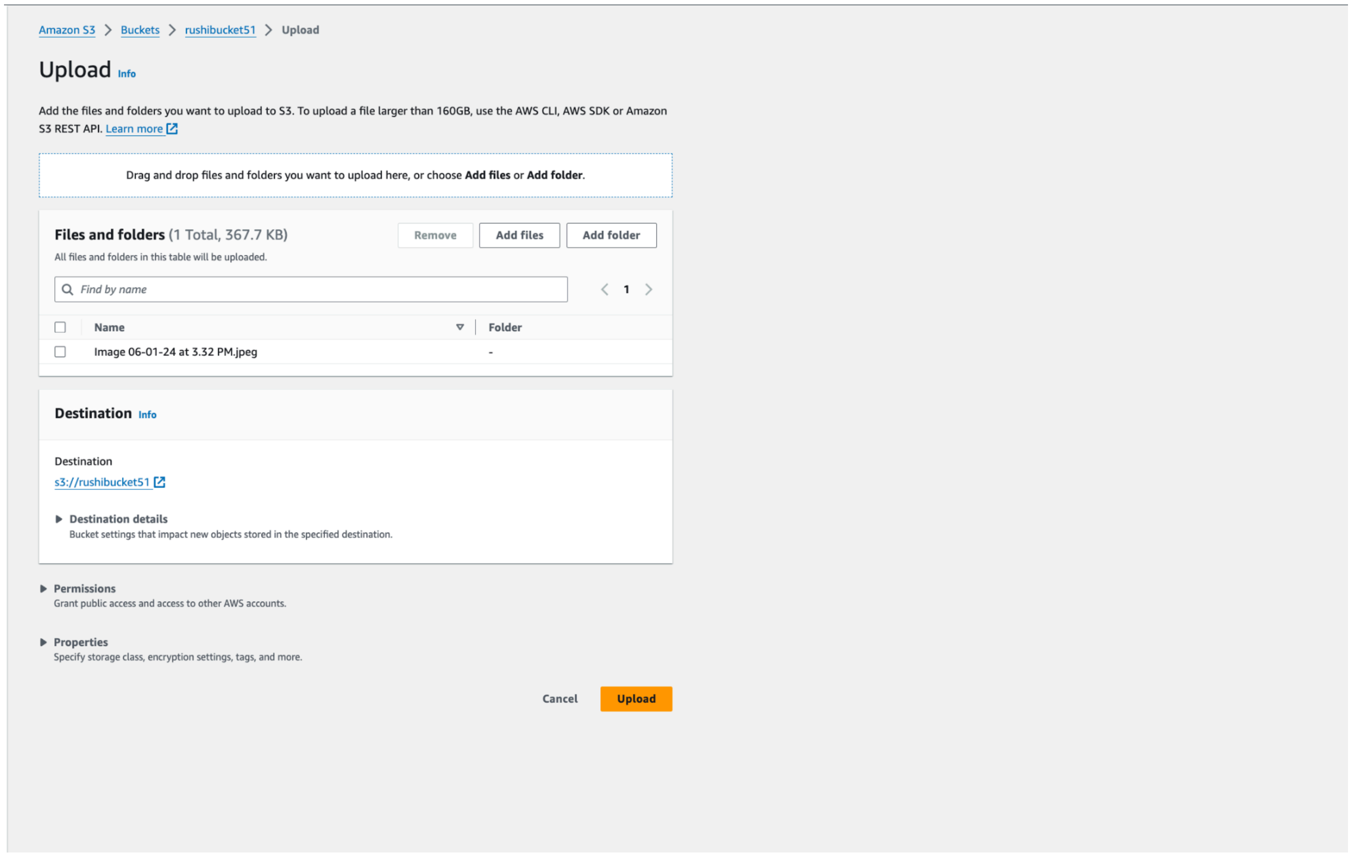
Summary

Destination	Succeeded	Failed
s3://bhushanbucket2	1 file, 33.1 KB (100.00%)	0 files, 0 B (0%)

Files and folders | Configuration

Files and folders (1 Total, 33.1 KB)

Name	Folder	Type	Size	Status	Error
Screenshot 2...	-	image/png	33.1 KB	Succeeded	-



Amazon S3 > Buckets > rushibucket51 > Upload

Upload Info

Add the files and folders you want to upload to S3. To upload a file larger than 160GB, use the AWS CLI, AWS SDK or Amazon S3 REST API. Learn more

Drag and drop files and folders you want to upload here, or choose **Add files** or **Add folder**.

Files and folders (1 Total, 367.7 KB)

All files and folders in this table will be uploaded.

Name	Folder
Image 06-01-24 at 3.32 PM.jpeg	-

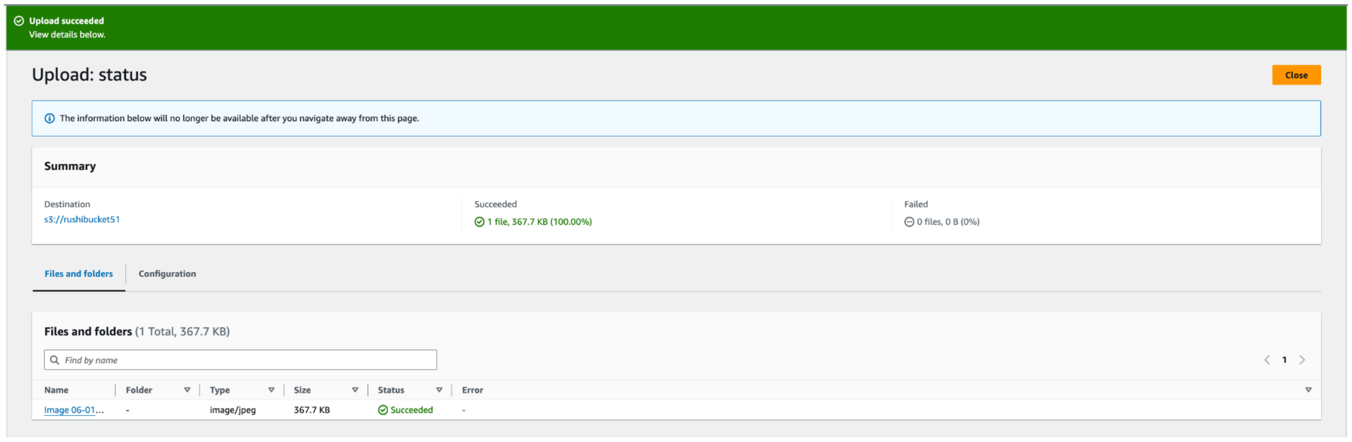
Destination Info

Destination
s3://rushibucket51

Destination details
Bucket settings that impact new objects stored in the specified destination.

Permissions
Grant public access and access to other AWS accounts.

Properties
Specify storage class, encryption settings, tags, and more.



Upload succeeded
View details below.

Upload: status

The information below will no longer be available after you navigate away from this page.

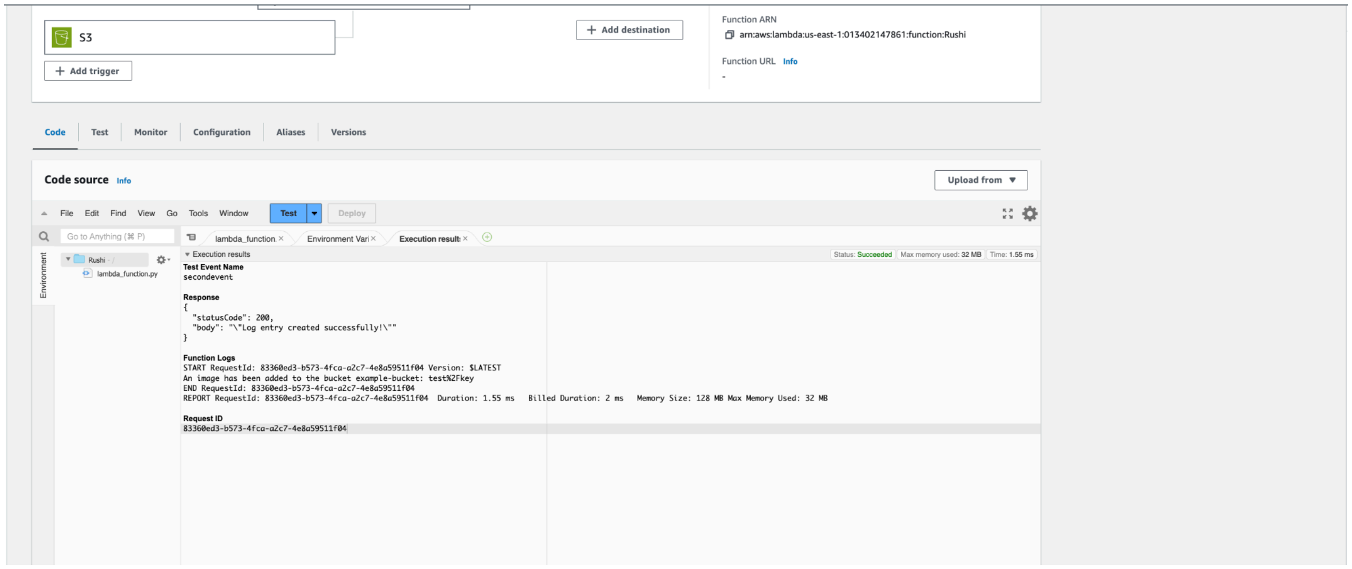
Summary

Destination	Succeeded	Failed
s3://rushibucket51	1 file, 367.7 KB (100.00%)	0 files, 0 B (0%)

Files and folders (1 Total, 367.7 KB)

Name	Folder	Type	Size	Status	Error
Image 06-01...	-	image/jpeg	367.7 KB	Succeeded	-

Step 10: Now to click on test in lambda to check whether it is giving log when image is added to S3.



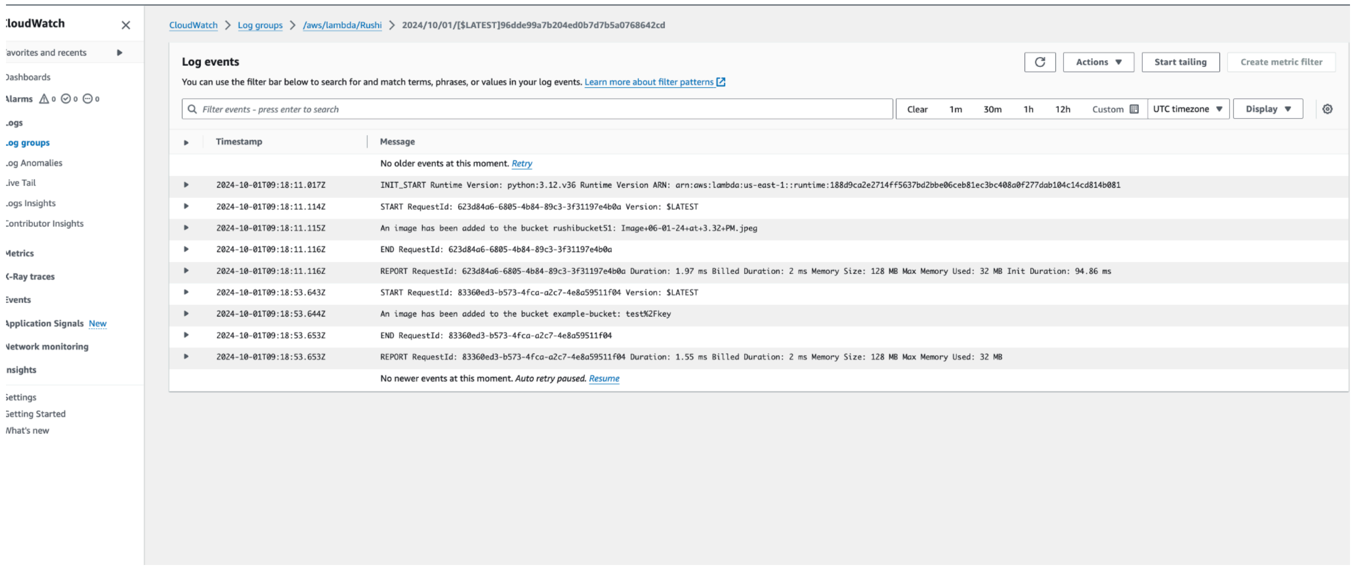
Function ARN: arn:aws:lambda:us-east-1:013402147861:function:Rushi

Function URL: -

Status: Succeeded · Max memory used: 32 MB · Time: 1.55 ms

```
Test Event Name
secondevent

Response
{
  "statusCode": 200,
  "body": "\"Log entry created successfully!\""
}

Function Logs
START RequestId: 83360ed3-b573-4fca-a2c7-4e8a59511f04 Version: $LATEST
An image has been added to the bucket example-bucket: test%2Fkey
END RequestId: 83360ed3-b573-4fca-a2c7-4e8a59511f04
REPORT RequestId: 83360ed3-b573-4fca-a2c7-4e8a59511f04  Duration: 1.55 ms   Billed Duration: 2 ms   Memory Size: 128 MB Max Memory Used: 32 MB

Request ID
83360ed3-b573-4fca-a2c7-4e8a59511f04
```

Step 11: Now Lets see the log on Cloud watch.To see it go to monitor section and then click on view cloudwatch logs.



CloudWatch > Log groups > /aws/lambda/Rushi > 2024/10/01/[$LATEST]96dde99a7b204ed0b7d7b5a0768642cd

Log events

You can use the filter bar below to search for and match terms, phrases, or values in your log events. Learn more about filter patterns

```
Timestamp                 Message
                          No older events at this moment. Retry
2024-10-01T09:18:11.017Z  INIT_START Runtime Version: python:3.12.v36 Runtime Version ARN: arn:aws:lambda:us-east-1::runtime:188d9ca2e2714ff5637bd2bbe06ceb81ec3bc408a0f277dab104c14cd814b081
2024-10-01T09:18:11.114Z  START RequestId: 623d84a6-6805-4b84-89c3-3f31197e4b0a Version: $LATEST
2024-10-01T09:18:11.115Z  An image has been added to the bucket rushibucket51: Image+06-01-24+at+3.32+PM.jpeg
2024-10-01T09:18:11.116Z  END RequestId: 623d84a6-6805-4b84-89c3-3f31197e4b0a
2024-10-01T09:18:11.116Z  REPORT RequestId: 623d84a6-6805-4b84-89c3-3f31197e4b0a Duration: 1.97 ms Billed Duration: 2 ms Memory Size: 128 MB Max Memory Used: 32 MB Init Duration: 94.86 ms
2024-10-01T09:18:53.643Z  START RequestId: 83360ed3-b573-4fca-a2c7-4e8a59511f04 Version: $LATEST
2024-10-01T09:18:53.644Z  An image has been added to the bucket example-bucket: test%2Fkey
2024-10-01T09:18:53.653Z  END RequestId: 83360ed3-b573-4fca-a2c7-4e8a59511f04
2024-10-01T09:18:53.653Z  REPORT RequestId: 83360ed3-b573-4fca-a2c7-4e8a59511f04 Duration: 1.55 ms Billed Duration: 2 ms Memory Size: 128 MB Max Memory Used: 32 MB
                          No newer events at this moment. Auto retry paused. Resume
```

Conclusion: In this experiment, we successfully created an AWS Lambda function that logs a message when an image is uploaded to an S3 bucket. It is important to note that we have to select S3-put template in event other wise code will give an error.The function was successfully triggered by S3 object uploads, validating the functionality of Lambda's event-driven architecture. This experiment demonstrated how Lambda can efficiently respond to S3 events and how to troubleshoot common issues with event structure.