

Objective

The main purpose of this project is to establish an event-driven design in which an Amazon S3 bucket invokes an AWS Lambda function whenever there is an upload of an object (e.g.,.jpg file). The Lambda function processes the object based on the configured logic. The solution automates the workflow without the need to provision server infrastructure, using IAM roles for permission management and CloudWatch Logs for logging.

Advantages

- Serverless and Event-Driven: No server maintenance needed; Lambda executes only when invoked.
- Cost Efficiency: Pay only for compute time used by Lambda.
- Scalability: Scales automatically with the number of S3 object uploads.
- Low Latency Processing: Low-latency processing of S3 object events.
- Highly Available: AWS manages Lambda and S3 with intrinsic fault tolerance.
- Secure: Fine-grained access control with IAM roles and policies.

Project Steps:

1. Create an Amazon S3 Bucket

Service: Amazon S3

Steps:

- Navigate to S3 in AWS Console.
- Click on Create bucket.
- Enter a unique bucket name .
- Select the AWS Region nearest to your deployment resources.
- Leave other options as default .
- Click on Create bucket.

aws

Search

[Alt+S]

United States (N. Virginia)

T%20MANOJ

Storage

Amazon S3

Store and retrieve any amount of data from anywhere

Amazon S3 is an object storage service that offers industry-leading scalability, data availability, security, and performance.

Create a bucket

Every object in S3 is stored in a bucket. To upload files and folders to S3, you'll need to create a bucket where the objects will be stored.

Create bucket

Pricing

With S3, there are no minimum fees. You only pay for what you use. Prices are based on the location of your S3 bucket.

Estimate your monthly bill using the [AWS Simple Monthly Calculator](#)

How it works

aws

Introduction to Amazon S3

Copy link

CloudShell Feedbackamazon.com/console/home?region=us-east-1© 2025, Amazon Web Services, Inc. or its affiliates. Privacy Terms Cookie preferences

aws

Search

[Alt+S]

United States (N. Virginia)

T%20MANOJ

Amazon S3 > Buckets > Create bucket

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

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.

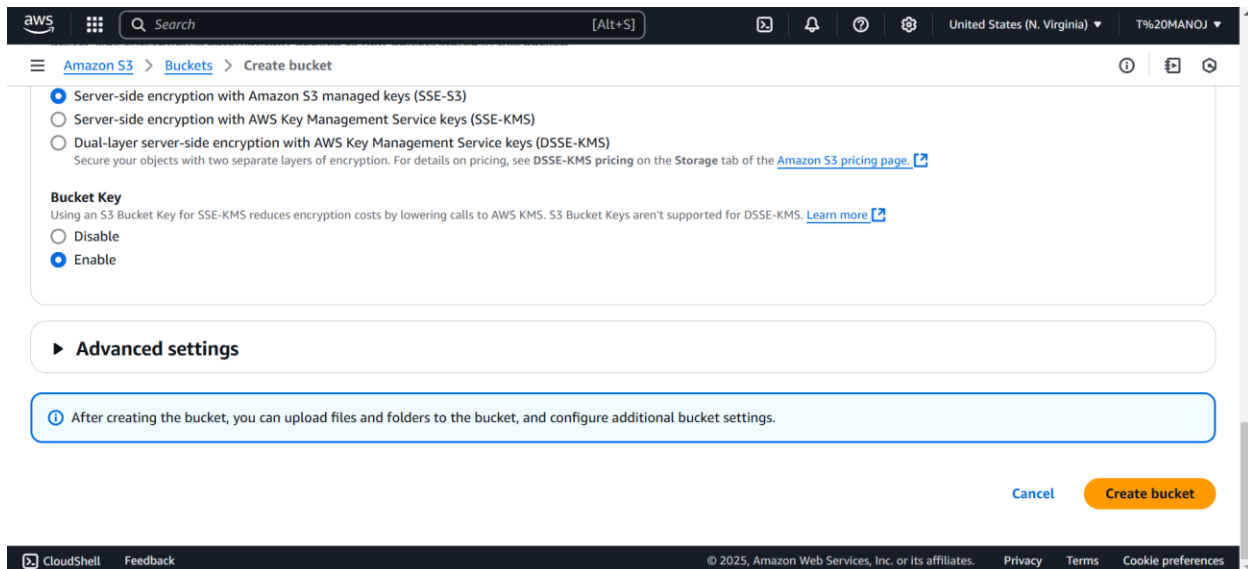
Choose bucket

Format: s3://bucket/prefix

Object Ownership

Info

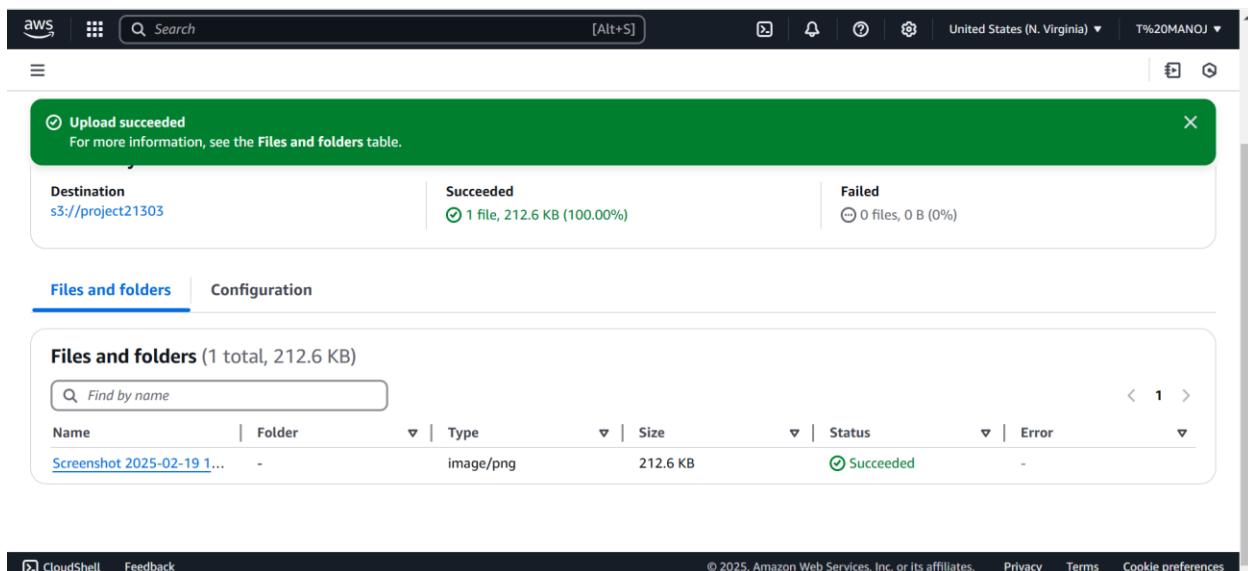
CloudShell Feedback© 2025, Amazon Web Services, Inc. or its affiliates. Privacy Terms Cookie preferences



2. Upload a Test Object (.jpg File)

Select the bucket created.

- Click on Upload → Add files → select any.jpg file from your local machine.
- Click on Upload.



3. Create an IAM Policy

Service: IAM → Policies

Steps:

- Click on Create policy.
- Select the JSON tab.
- Paste the following example JSON policy:

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Effect": "Allow",
      "Action": [
        "logs:PutLogEvents",
        "logs:CreateLogGroup",
        "logs:CreateLogStream"
      ],
      "Resource": "arn:aws:logs:*:*:*"
    },
    {
      "Effect": "Allow",
      "Action": [
        "s3:GetObject"
      ],
      "Resource": "arn:aws:s3:::*/*"
    }
  ]
}
```

- Policy Name: LambdaS3AccessPolicy.

- Description: Policy for Lambda to read from S3 and write to CloudWatch Logs.
- Click Create policy.

The screenshot shows the AWS IAM console 'Create policy' page, specifically the 'Specify permissions' step. The left sidebar shows 'Step 1: Specify permissions' as the active step, with 'Step 2: Review and create' as the next step. The main content area is titled 'Specify permissions' and includes a sub-header 'Add permissions by selecting services, actions, resources, and conditions. Build permission statements using the JSON editor.' Below this is the 'Policy editor' section, which has tabs for 'Visual', 'JSON', and 'Actions'. The 'JSON' tab is selected, showing a JSON snippet for a policy statement. The 'Visual' tab shows an 'Edit statement' section with a 'Select a statement' prompt and an 'Add new statement' button. The bottom of the page shows the AWS footer with 'CloudShell', 'Feedback', and copyright information.

Specify permissions [Info](#)

Add permissions by selecting services, actions, resources, and conditions. Build permission statements using the JSON editor.

Policy editor Visual **JSON** Actions +

```

1 {
2   "Version": "2012-10-17",
3   "Statement": [
4     {
5       "Effect": "Allow",
6       "Action": [
7         "logs:PutLogEvents",
8         "logs:CreateLogGroup",
9         "logs:CreateLogStream"
10      ],
11      "Resource": "arn:aws:logs:*:*:*"
12    },
13    {
14      "Effect": "Allow",
15      "Action": [

```

Edit statement

Select a statement

Select an existing statement in the policy or add a new statement.

[+ Add new statement](#)

CloudShell Feedback © 2025, Amazon Web Services, Inc. or its affiliates. Privacy Terms Cookie preferences

The screenshot shows the AWS IAM console 'Create policy' page, specifically the 'Review and create' step. The left sidebar shows 'Step 2: Review and create' as the active step, with 'Step 1: Specify permissions' as the previous step. The main content area is titled 'Review and create' and includes a sub-header 'Review the permissions, specify details, and tags.' Below this is the 'Policy details' section, which has a 'Policy name' field with the value 'LambdaProject' and a 'Description - optional' field with the value 'Policy for Lambda Project'. The 'Permissions defined in this policy' section is also visible at the bottom. The bottom of the page shows the AWS footer with 'CloudShell', 'Feedback', and copyright information.

Review and create [Info](#)

Review the permissions, specify details, and tags.

Policy details

Policy name
Enter a meaningful name to identify this policy.

LambdaProject

Maximum 128 characters. Use alphanumeric and '+=, @, _' characters.

Description - optional
Add a short explanation for this policy.

Policy for Lambda Project

Maximum 1,000 characters. Use alphanumeric and '+=, @, _' characters.

Permissions defined in this policy [Info](#) [Edit](#)

CloudShell Feedback amazon.com/console/home?region=us-east-1 © 2025, Amazon Web Services, Inc. or its affiliates. Privacy Terms Cookie preferences

4. Create an IAM Role for Lambda

Service: IAM → Roles

Steps:

- Click Create role.
- Use case: Select Lambda.
- Click Next.
- Attach the LambdaS3AccessPolicy created above.
- Role name: LambdaS3TriggerRole.
- Click Create role.

aws [Search] [Alt+S] Global T%20MANOJ

IAM > Roles > Create role

Step 2
○ Add permissions
Step 3
○ Name, review, and create

Trusted entity type

- ☒ **AWS service**
Allow AWS services like EC2, Lambda, or others to perform actions in this account.
- ☐ **AWS account**
Allow entities in other AWS accounts belonging to you or a 3rd party to perform actions in this account.
- ☐ **Web identity**
Allows users federated by the specified external web identity provider to assume this role to perform actions in this account.
- ☐ **SAML 2.0 federation**
Allow users federated with SAML 2.0 from a corporate directory to perform actions in this account.
- ☐ **Custom trust policy**
Create a custom trust policy to enable others to perform actions in this account.

Use case

Allow an AWS service like EC2, Lambda, or others to perform actions in this account.

CloudShell Feedback © 2025, Amazon Web Services, Inc. or its affiliates. Privacy Terms Cookie preferences

aws [Search] [Alt+S] Global T%20MANOJ

IAM > Roles > Create role

☐ Custom trust policy
Create a custom trust policy to enable others to perform actions in this account.

Use case

Allow an AWS service like EC2, Lambda, or others to perform actions in this account.

Service or use case

Lambda

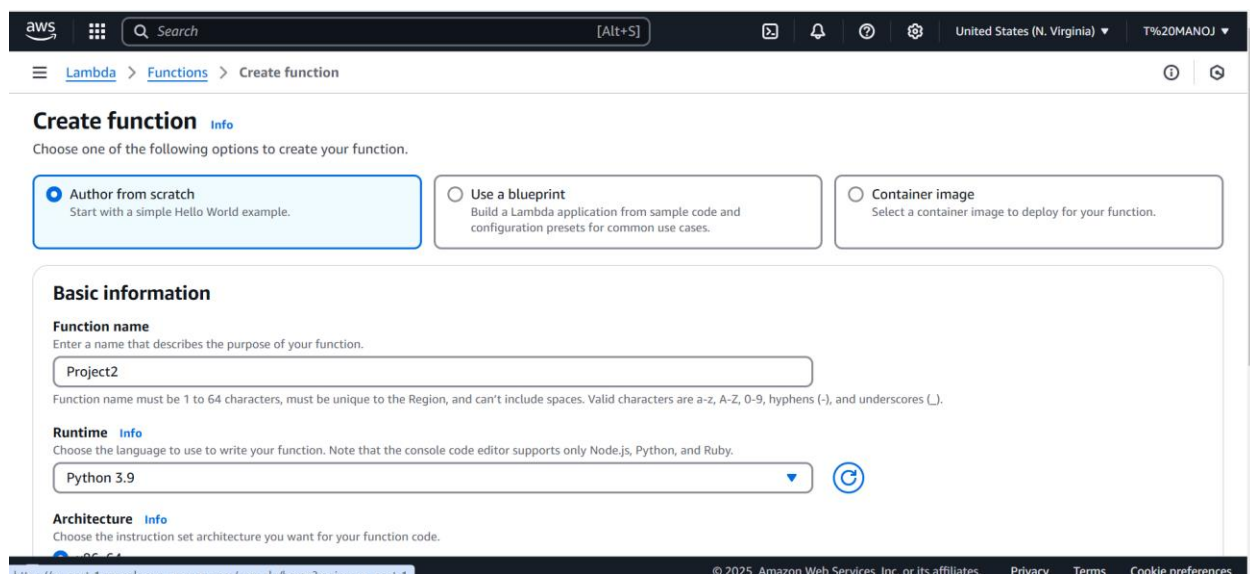
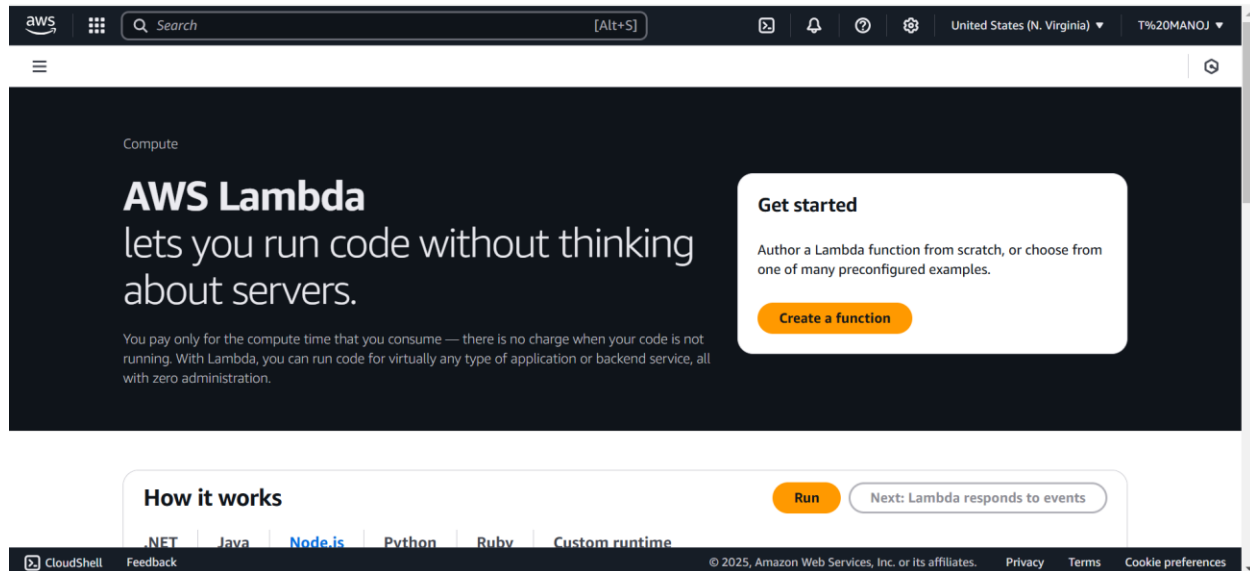
Choose a use case for the specified service.

Use case

- ☒ **Lambda**
Allows Lambda functions to call AWS services on your behalf.

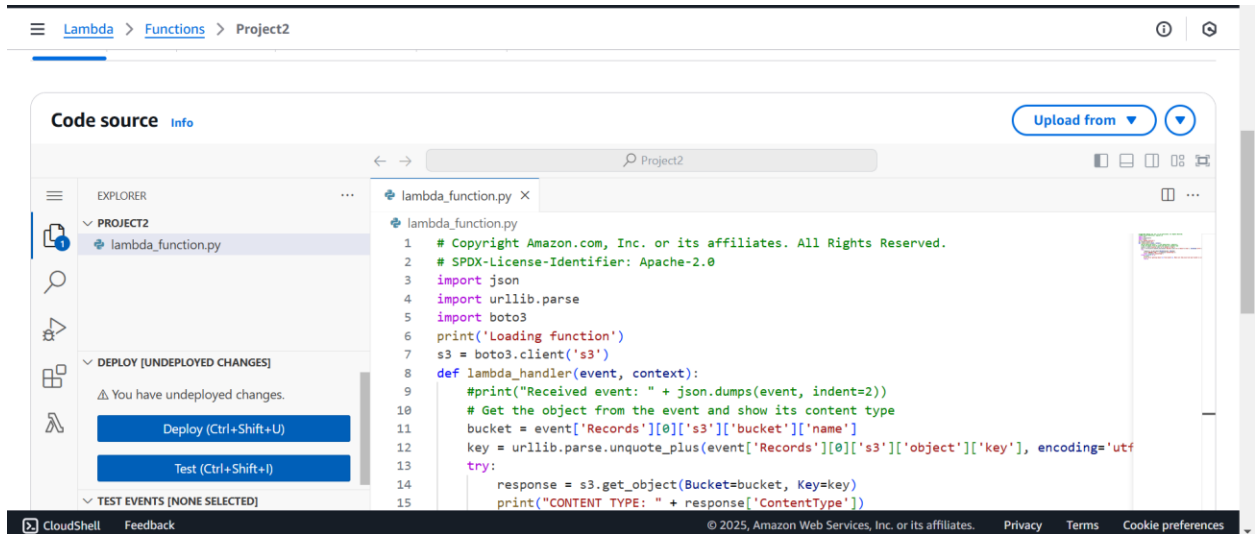
Cancel Next

CloudShell Feedback © 2025, Amazon Web Services, Inc. or its affiliates. Privacy Terms Cookie preferences



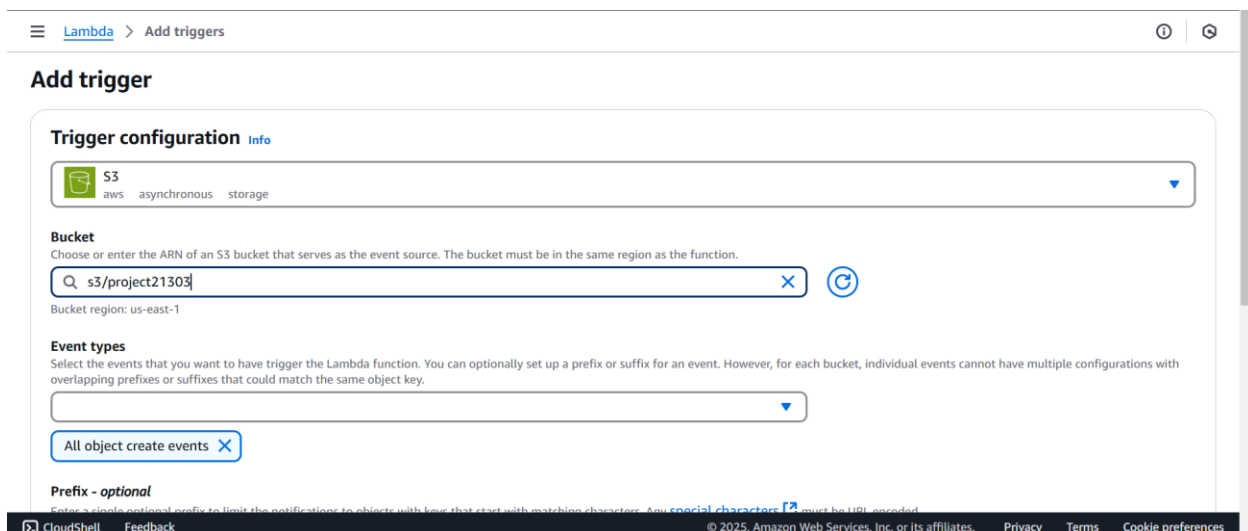
6. Deploy Lambda Code

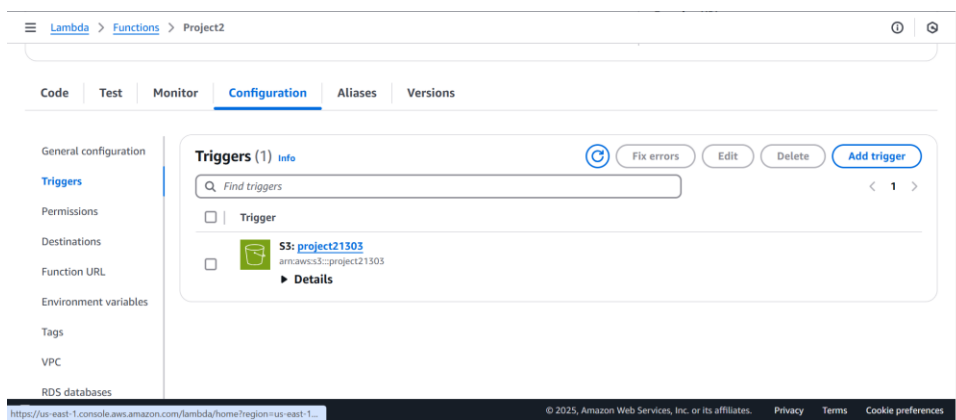
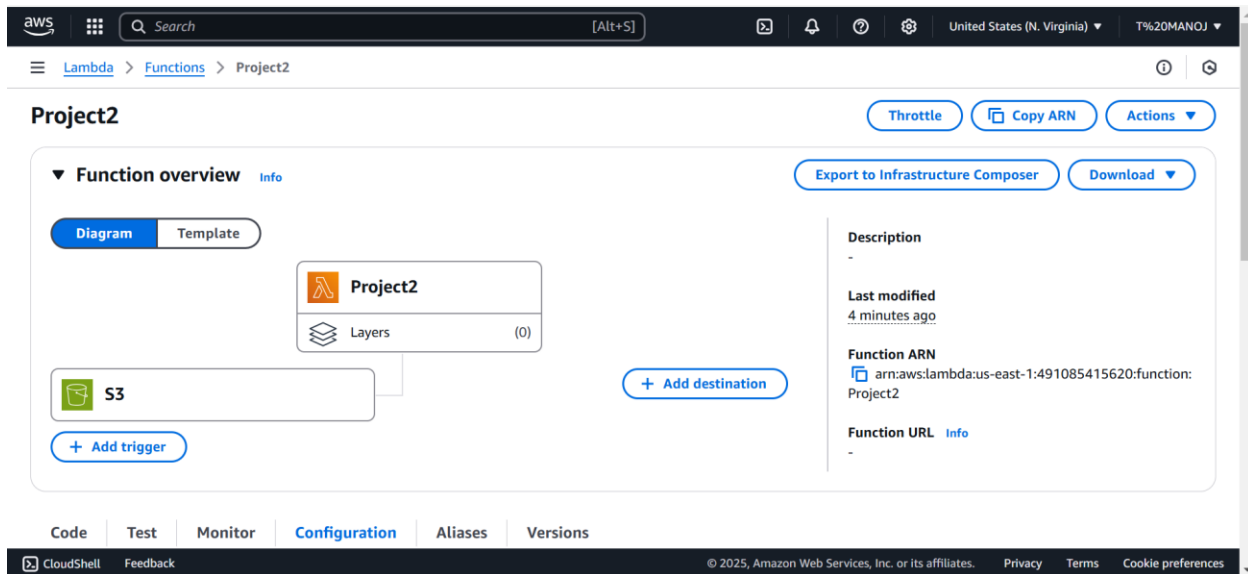
- Sample Lambda Code:
- Steps:
- Paste the code in the Code source section.
- Click Deploy.



7. Set up S3 Trigger

- In the page of the Lambda function, click Configuration → Triggers → Add trigger.
- Trigger configuration:
- Source: S3
- Bucket: Choose the created bucket.
- Event type: PUT (for object uploads).
- Acknowledge the warning regarding recursive invocation.
- Click Add.





8. Test the Lambda Function

- Test with Dummy Event:
- Click Test → Configure test event.
- Use the following JSON and modify the bucket name and key:

```
{
  "Records": [
    {
      "eventVersion": "2.0",
      "eventSource": "aws:s3",
      "awsRegion": "Bucket region",
```

```
"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":
"EXAMPLE123/5678abcdefghijklmbdaisawesome/mnopqrstuvwxyzABCDEFGH"
},
"s3": {
  "s3SchemaVersion": "1.0",
  "configurationId": "testConfigRule",
  "bucket": {
    "name": "S3 bucket name",
    "ownerIdentity": {
      "principalId": "EXAMPLE"
    },
    "arn": "arnawss3:::amzn-s3-demo-bucket"
  },
  "object": {
    "key": "Object_Key",
    "size": 1024,
    "eTag": "0123456789abcdef0123456789abcdef",
    "sequencer": "0A1B2C3D4E5F678901"
  }
}
}
```

```
]
}
```

DEPLOY CODE :

```
# Copyright Amazon.com, Inc. or its affiliates. All Rights Reserved.
```

```
# SPDX-License-Identifier: Apache-2.0
```

```
import json
```

```
import urllib.parse
```

```
import boto3
print('Loading function')
s3 = boto3.client('s3')
def lambda_handler(event, context):
```

```
    #print("Received event: " + json.dumps(event, indent=2)) # Get the object from the
    event and show its content type
```

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

```
    key = urllib.parse.unquote_plus(event['Records'][0]['s3']['object']['key'],
    encoding='utf-8')
```

```
    try:
```

```
        response = s3.get_object(Bucket=bucket, Key=key)
```

```
        print("CONTENT TYPE: " + response['ContentType'])
```

```
        return response['ContentType']
```

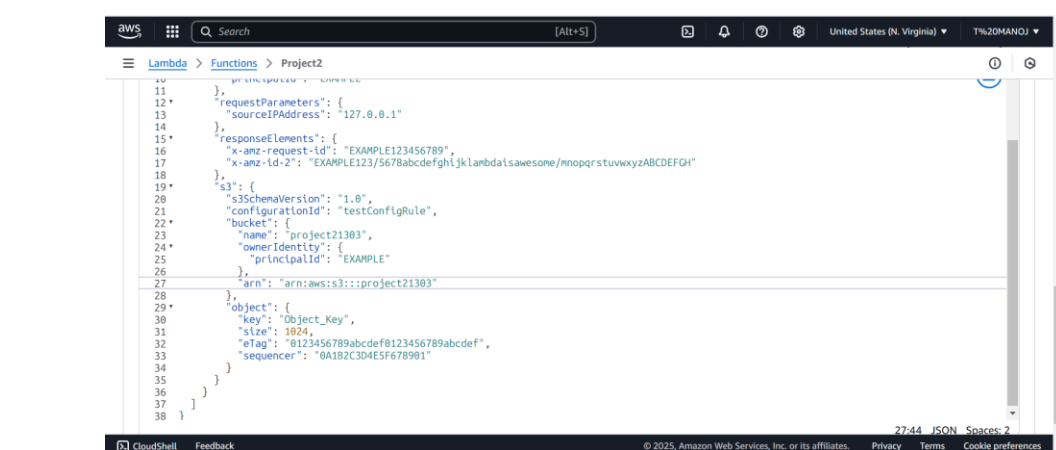
```
    except Exception as e:
```

```
        print(e)
```

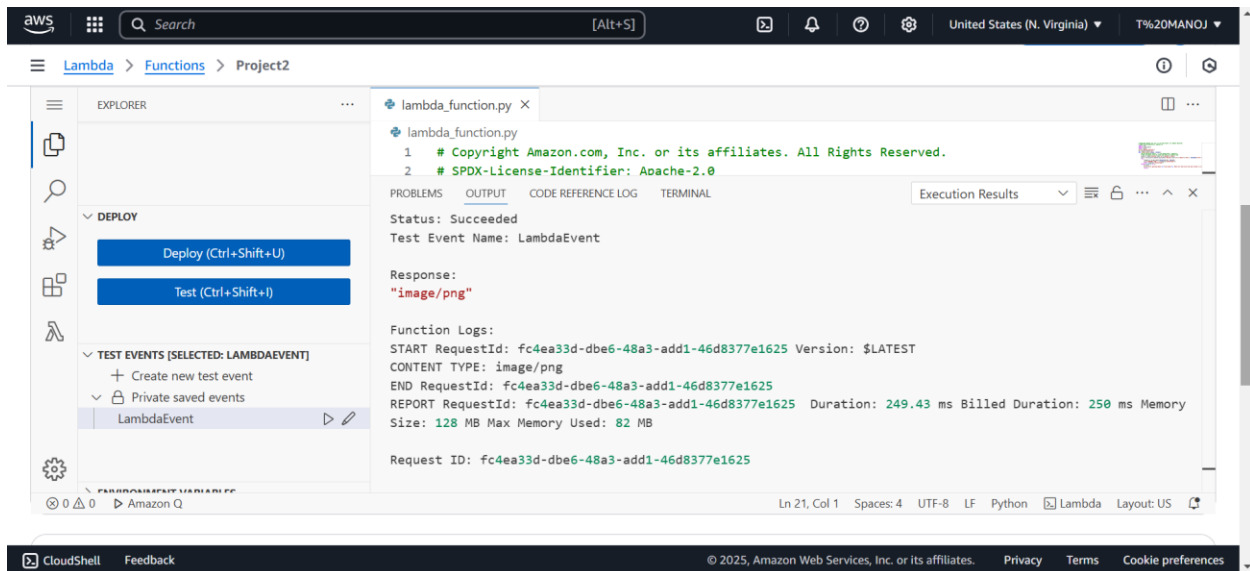
```
        print('Error getting object {} from bucket {}. Make sure they exist and your bucket is in
    the same region as this function.'.format(key, bucket))
```

```
        raise e
```

- Test with Actual Trigger:
- Upload a second.jpg file to the S3 bucket.
- Check the Lambda execution logs in CloudWatch.



- Service: CloudWatch → Logs → Log groups.
- Find the log group for the Lambda function (e.g., /aws/lambda/S3ObjectProcessor).
- Click on the Log stream to see the detailed execution logs.



▶ 2025-02-19T07:51:36.234Z	[ERROR] ClientError: An error occurred (AccessDenied) when calling the GetObject operation: ...
▶ 2025-02-19T07:51:36.257Z	END RequestId: e8d560c2-0354-44cc-83ac-fd7a30e8f705
▶ 2025-02-19T07:51:36.257Z	REPORT RequestId: e8d560c2-0354-44cc-83ac-fd7a30e8f705 Duration: 357.36 ms Billed Duration: ...
▶ 2025-02-19T07:54:11.443Z	START RequestId: fc4ea33d-dbe6-48a3-add1-46d8377e1625 Version: \$LATEST
▶ 2025-02-19T07:54:11.691Z	CONTENT TYPE: image/png
▶ 2025-02-19T07:54:11.693Z	END RequestId: fc4ea33d-dbe6-48a3-add1-46d8377e1625
▶ 2025-02-19T07:54:11.693Z	REPORT RequestId: fc4ea33d-dbe6-48a3-add1-46d8377e1625 Duration: 249.43 ms Billed Duration: ...
No newer events at this moment. <i>Auto retry paused.</i> Resume	
▶ 2025-02-19T07:58:02.248Z	START RequestId: c65c146c-69b8-4781-be42-220b8536ba5d Version: \$LATEST
▶ 2025-02-19T07:58:02.409Z	CONTENT TYPE: application/pdf
▶ 2025-02-19T07:58:02.410Z	END RequestId: c65c146c-69b8-4781-be42-220b8536ba5d
▶ 2025-02-19T07:58:02.410Z	REPORT RequestId: c65c146c-69b8-4781-be42-220b8536ba5d Duration: 162.59 ms Billed Duration: ...
No newer events at this moment. <i>Auto retry paused.</i> Resume	

Use Cases

- Image Processing Pipelines: Resize images automatically upon upload.
- Data Processing: Initiate ETL jobs upon new data upload.
- Document Conversion: Translate uploaded documents (e.g., .docx to .pdf).
- Real-Time Notifications: Provide alerts or notifications upon adding of certain files.
- Audit and Logging: Automatically log upload information for compliance.

Conclusion :

- This project showcases the easy integration of AWS Lambda with Amazon S3 to develop real-time event-driven applications. Utilizing IAM roles for access security, CloudWatch Logs for monitoring, and Python as runtime for Lambda, the solution creates a scalable, cost-saving, and automated file processing workflow. These kinds of architectures are most suitable for current applications needing rapid, consistent, and serverless event processing.