

PROJECT PRESENTATION

PRESENTED BY ANURAG VERMA

INTRODUCTION :-



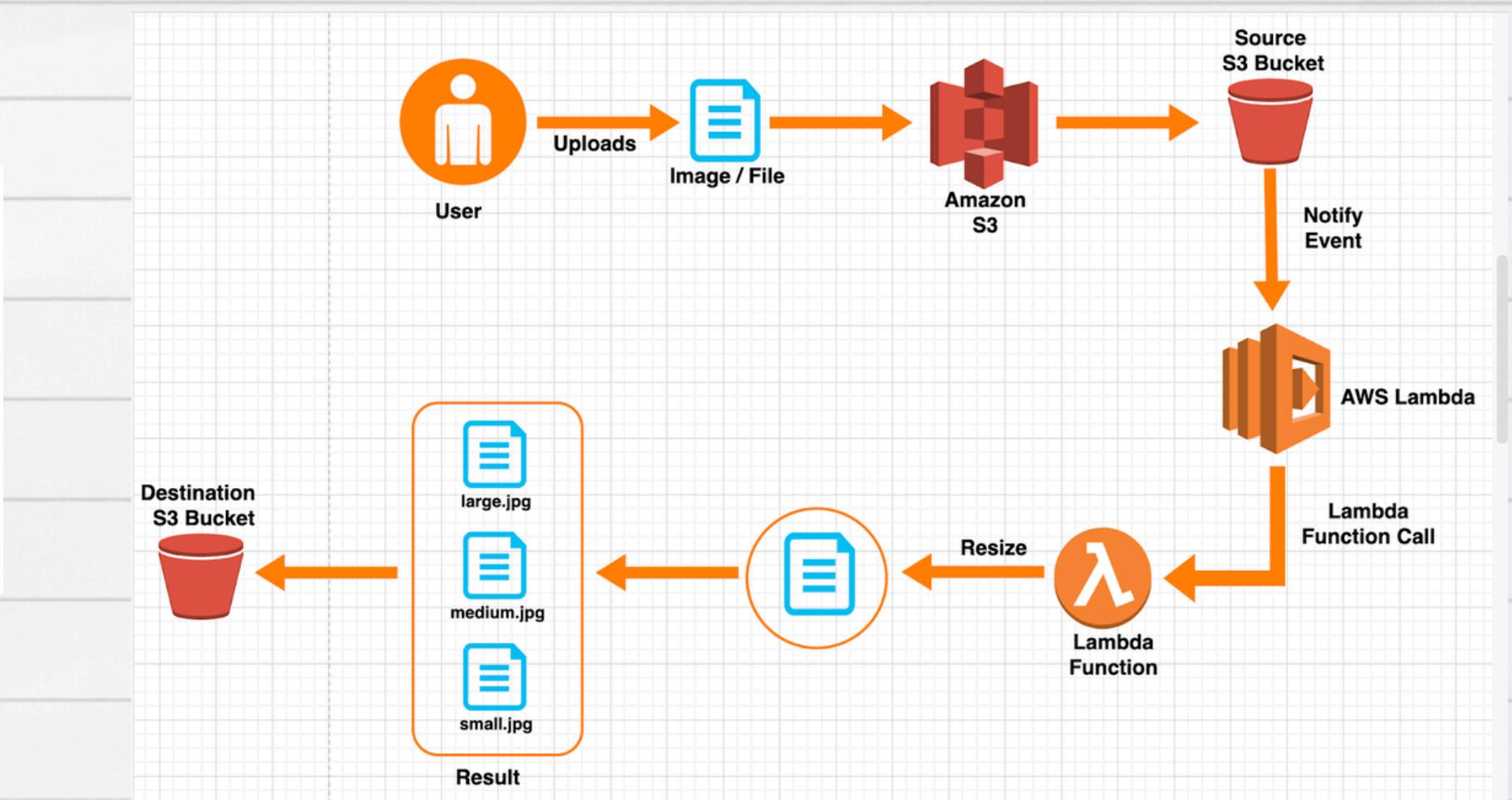
Amazon Web Services (AWS) is a comprehensive cloud computing platform provided by Amazon. It offers a wide range of services, including computing power, storage options, and networking capabilities, on a pay-as-you-go basis. AWS allows businesses to scale and grow efficiently by providing infrastructure as a service (IaaS), platform as a service (PaaS), and software as a service (SaaS). Key services include Amazon EC2 (virtual servers), Amazon S3 (storage), Amazon RDS (database management), and AWS Lambda (serverless computing).

Serverless Image Processing

Create a serverless image processing application that automatically resizes and optimizes images uploaded to an Amazon S3 bucket.

Amazon S3: Used to store original images uploaded by users.

AWS Lambda: Serverless compute service used to execute the image processing tasks.

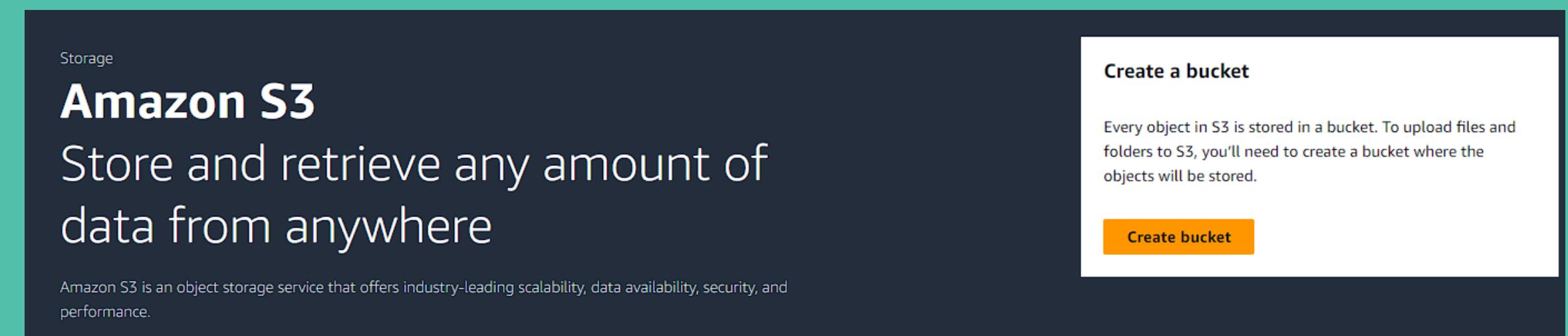


**For Create a serverless image processing application
that automatically resizes and optimizes images
You need to follow the given steps:-**

STEP 1:- Login in your AWS account.

STEP 2:- In AWS console home page,search S3.

STEP 3:- Click on S3.



STEP 4:- Bucket Name :- mysourcebucketanurag

AWS Region
Asia Pacific (Mumbai) ap-south-1

Bucket name [Info](#)

mysourcebucketanurag

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

STEP 5:- Now,click on ACLs enable

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.

STEP 6:- Now, turn off all block public access service

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.

STEP 7:- Now leave all the setting default, and click on create the bucket

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

Cancel

Create bucket

STEP 8:- Once the bucket is created successfull, select your S3 bucket
Click on the Copy ARN button to copy the ARN.
Save the source bucket ARN in a text file for later use
arn:aws:s3:::mysourcebucketanurag

STEP 9:- Create Destination Bucket

STEP 10:- Click on Create bucket button

STEP 11:- Bucket Name: Enter mydestinationbucketanurag

STEP 12:- Once the bucket is created successfully, Select your S3 bucket.

STEP 13:- Click on the Copy ARN button to copy the ARN.
Save the destination bucket ARN in a text file for later use.
arn:aws:s3:::mydestinationbucketanurag

General purpose buckets (2) Info All AWS Regions			
Buckets are containers for data stored in S3.			
<input type="text"/> Find buckets by name < 1 > 			
Name	AWS Region	IAM Access Analyzer	Creation date
mydestinationbucketanurag	Asia Pacific (Mumbai) ap-south-1	View analyzer for ap-south-1	June 20, 2024, 15:54:23 (UTC+05:30)
mysourcebucketanurag	Asia Pacific (Mumbai) ap-south-1	View analyzer for ap-south-1	June 20, 2024, 15:45:46 (UTC+05:30)

STEP 14:- Now we have two S3 buckets (Source and Destination). We will make use of our AWS Lambda function to copy the content from source bucket to destination bucket.

STEP 15:- Now we have to click the mysourcebucketanurag and click the add files option

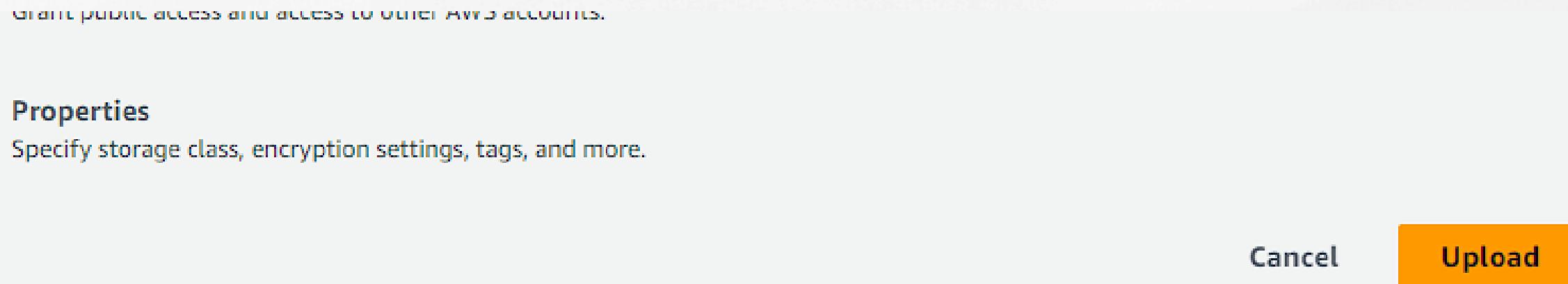
Files and folders (0) [Remove](#) [Add files](#) [Add folder](#)

All files and folders in this table will be uploaded.

[Find by name](#) < 1 >

	Name	Folder
No files or folders		
You have not chosen any files or folders to upload.		

STEP 16:- Now upload the file which you have been selected



STEP 17:- Now ,Create a Lambda Function

Go to the Services menu and click on Lambda under Compute section.

Click on the Create a function button.

Function name : Enter myfunctionanurag

Runtime : Select Node.js 18x

The screenshot shows the 'Create a new function' wizard step 1. It has two main sections:

- Function name:** A text input field containing "myfunctionanurag". Above it, a placeholder text says "Enter a name that describes the purpose of your function."
- Runtime:** A dropdown menu set to "Node.js 18.x". Below the dropdown, a note says "Choose the language to use to write your function. Note that the console code editor supports only Node.js, Python, and Ruby."

now, we have to click on the change default execution role and click on the IAM console

The screenshot shows the 'Permissions' section of the AWS Lambda configuration. It includes a note about Lambda creating a default execution role for log upload to CloudWatch Logs, with a link to customize it later. A 'Change default execution role' button is visible, which is highlighted in the image. Below it, there's a section for selecting an execution role, with a note to choose a role or create a custom one via the IAM console.

STEP 18:- Now , click on the IAM services and create a policy
Go to Services and Select IAM under Security, Identity and Compliance.
Click on Policies in the left navigation bar and click on the Create policy button.

The screenshot shows the 'Policies' page in the AWS IAM service. It displays a search bar, a filter dropdown set to 'All types', and a pagination indicator showing page 1 of 1. At the top right, there are buttons for 'Actions', 'Delete', and 'Create policy'. The main area shows a table header with columns for 'Policy name', 'Type', 'Used as', and 'Description'. A message at the bottom indicates that policies are loading.

Click on the JSON tab, Remove the existing code and copy-paste the below policy statement into the editor:

Policy JSON:

```
{  
    "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:::mysourcebucketanurag/*"  
        },  
        {  
            "Effect": "Allow",  
            "Action": ["s3:PutObject"],  
            "Resource": "arn:aws:s3:::mydestinationbucketanurag/*"  
        }  
    ]  
}
```

Replace the Source and destination ARN name of the bucket (which you have saved before) in the option Resource.

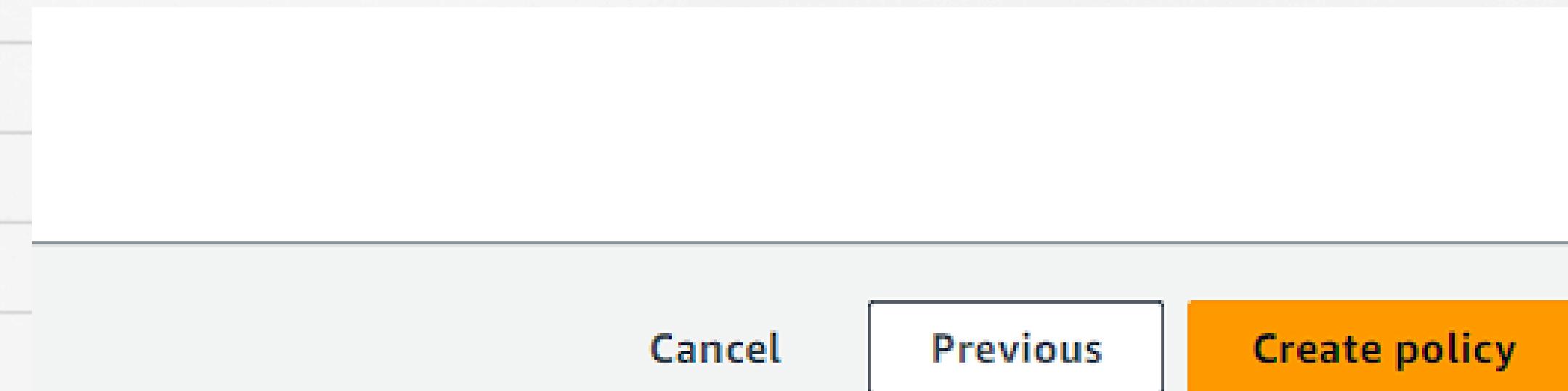
Leave everything as default and click on Next button

STEP 19:- On the Review Policy Page

Click on the Create policy button.

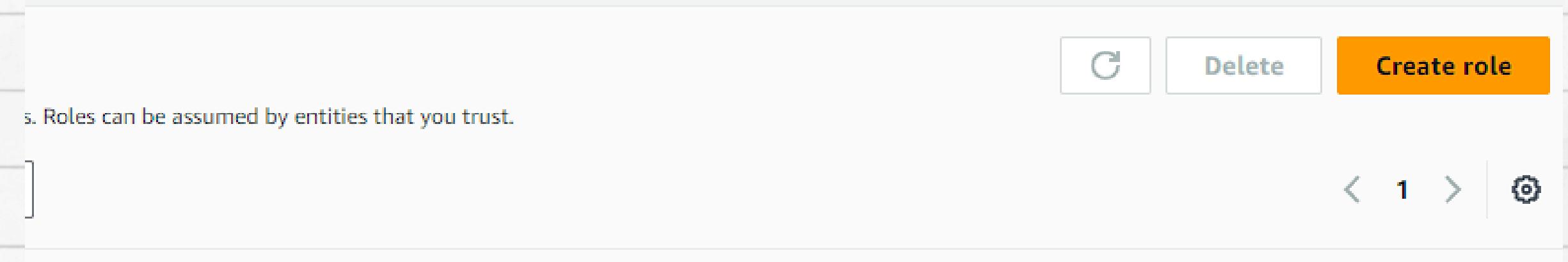
The screenshot shows the 'Create Policy' step of the AWS Lambda function configuration. It has two main input fields. The first field is labeled 'Policy name' with the placeholder 'Enter a meaningful name to identify this policy.' Below it is a text input box containing the value 'resizer'. A note below the input box specifies: 'Maximum 128 characters. Use alphanumeric and '+,.,@,_' characters.' The second field is labeled 'Description - optional' with the placeholder 'Add a short explanation for this policy.' Below it is a large, empty text area for entering a description.

Now, leave all setting as default and click on the create policy



STEP 20:- Create an IAM Role

In the left menu, click on Roles. Click on the Create role button. Select Lambda from AWS Services list.



From Use case: Select Lambda

A screenshot of the 'From Use case' configuration step. It shows a dropdown menu under 'Service or use case' with 'Lambda' selected. Below the dropdown, there is a section titled 'Choose a use case for the specified service.' with a radio button labeled 'Lambda' selected. A tooltip for the Lambda option states: 'Allows Lambda functions to call AWS services on your behalf.' At the bottom right of the form, there are 'Cancel' and 'Next' buttons.

Filter Policies: Now you can see a list of policies. Here you have to select the resizer policy

Permissions policies (1/933) [Info](#)

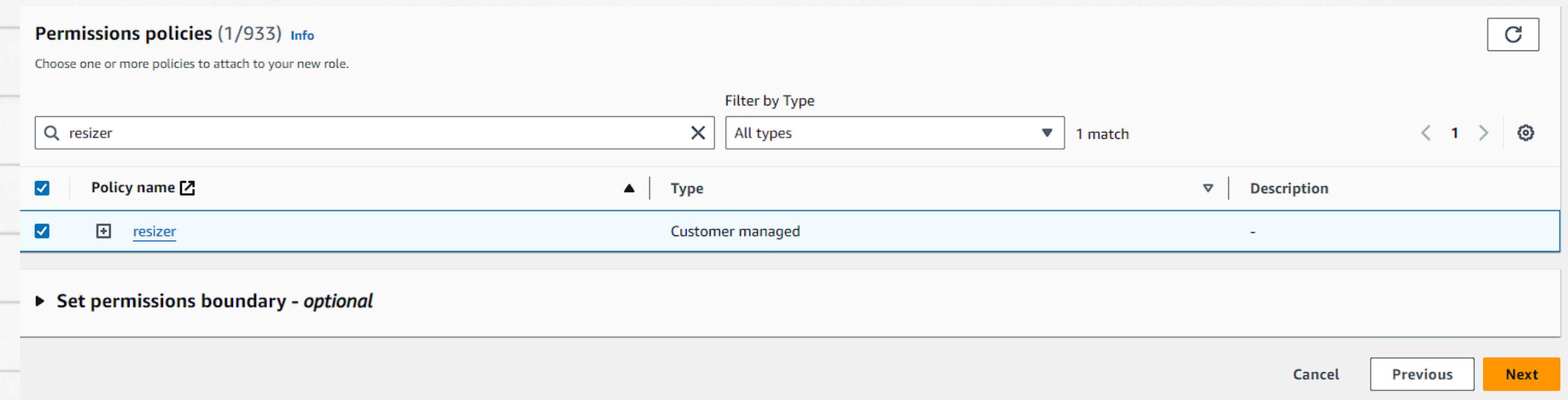
Choose one or more policies to attach to your new role.

Filter by Type X All types ▼ 1 match < 1 > ⚙️

Policy name	Type	Description
<input checked="" type="checkbox"/> resizer	Customer managed	-

▶ Set permissions boundary - optional

Cancel Previous Next



Select your policy and click on the Next button.

STEP 21:- Click on the Create Role button

Role Name: Enter imgresizer

Role details

Role name
Enter a meaningful name to identify this role.

Maximum 64 characters. Use alphanumeric and '+,-,.,@,_' characters.

Description



Now click on the create role

[Cancel](#)

[Previous](#)

[Create role](#)

STEP 22:- Now, after creating the IAM policies and roles you have to get back in the lambda function setting

STEP 23:- Now you have to click change default execution role and select the use an existing role

Now, select the created imgresizer role

▼ 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 CloudWatch Logs.

imgresizer



[View the imgresizer role](#) on the IAM console.

and continue to the create function

Cancel

Create function

STEP 23:- Adding Triggers to Lambda Function



+ Add trigger

Scroll down the list and select S3 from the trigger list.

Bucket: Select your source bucket - mysourcebucketanurag

Leave other fields as default.

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.

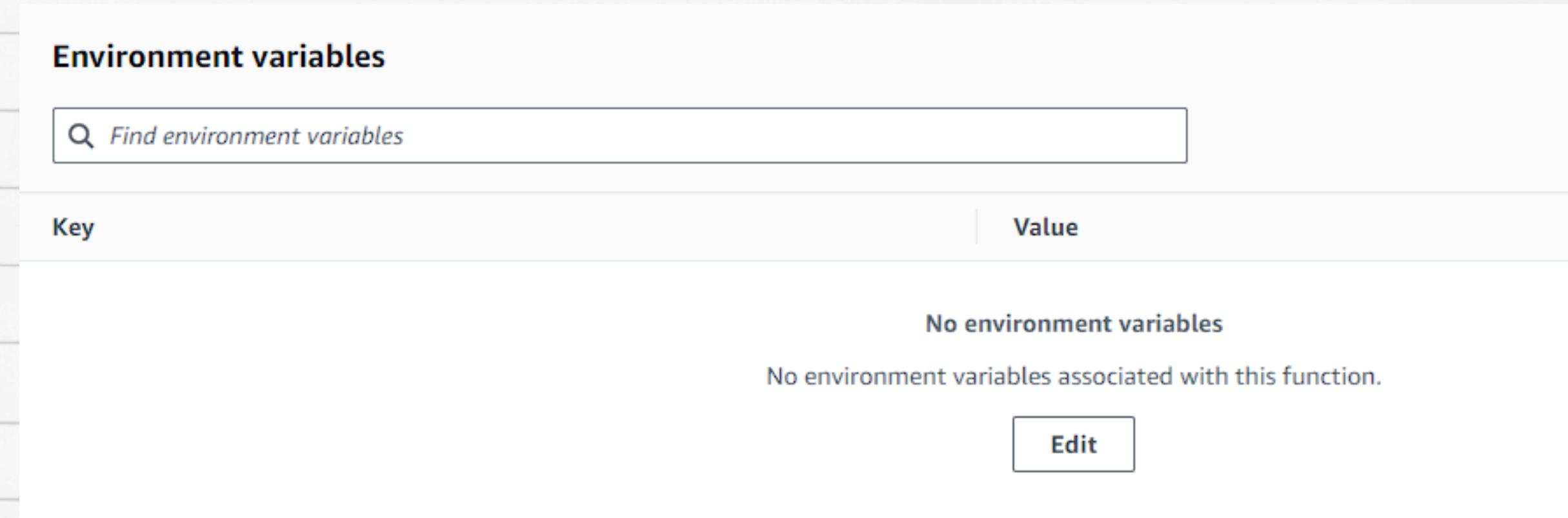
Lambda will add the necessary permissions for AWS S3 to invoke your Lambda function from this trigger. [Learn more ↗](#) about the Lambda permissions model.

Cancel

Add

STEP 24:-

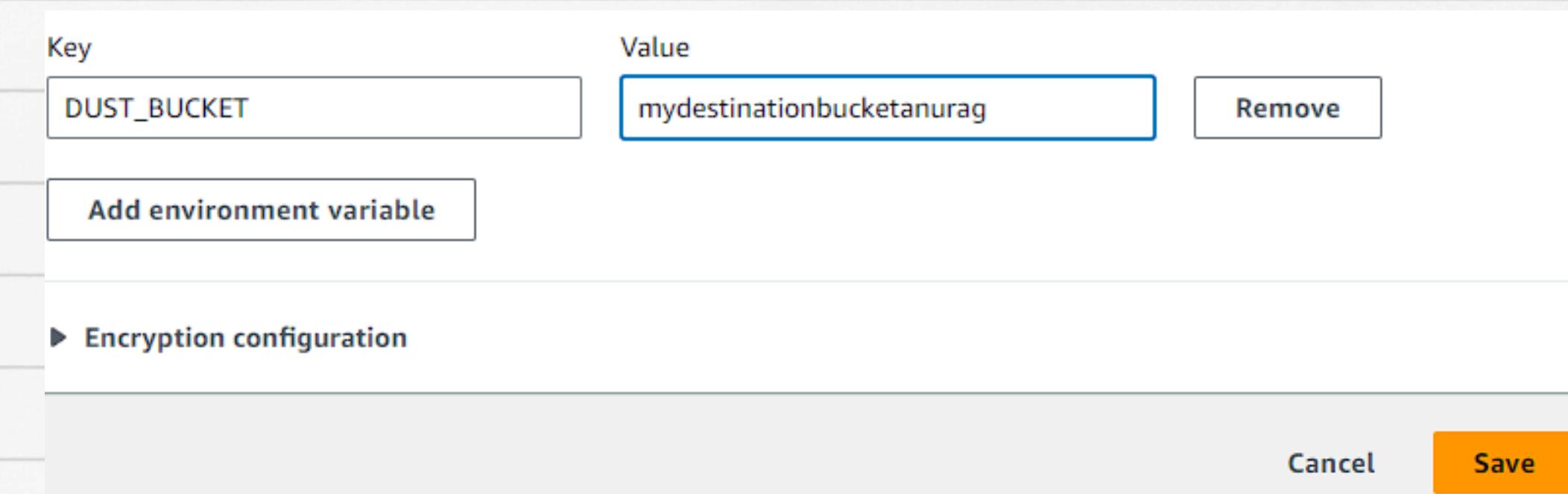
Now, you have to create environment variables



The screenshot shows the 'Environment variables' section of the AWS Lambda function configuration. At the top, there is a search bar labeled 'Find environment variables'. Below it, a table has two columns: 'Key' and 'Value'. A message in the center of the table area says 'No environment variables' and 'No environment variables associated with this function.' There is a single 'Edit' button at the bottom right of the table.

STEP 25:-

click on the edit option and add the key: DEST_BUCKET and value: mydestinationbucketanurag and continue to the save option

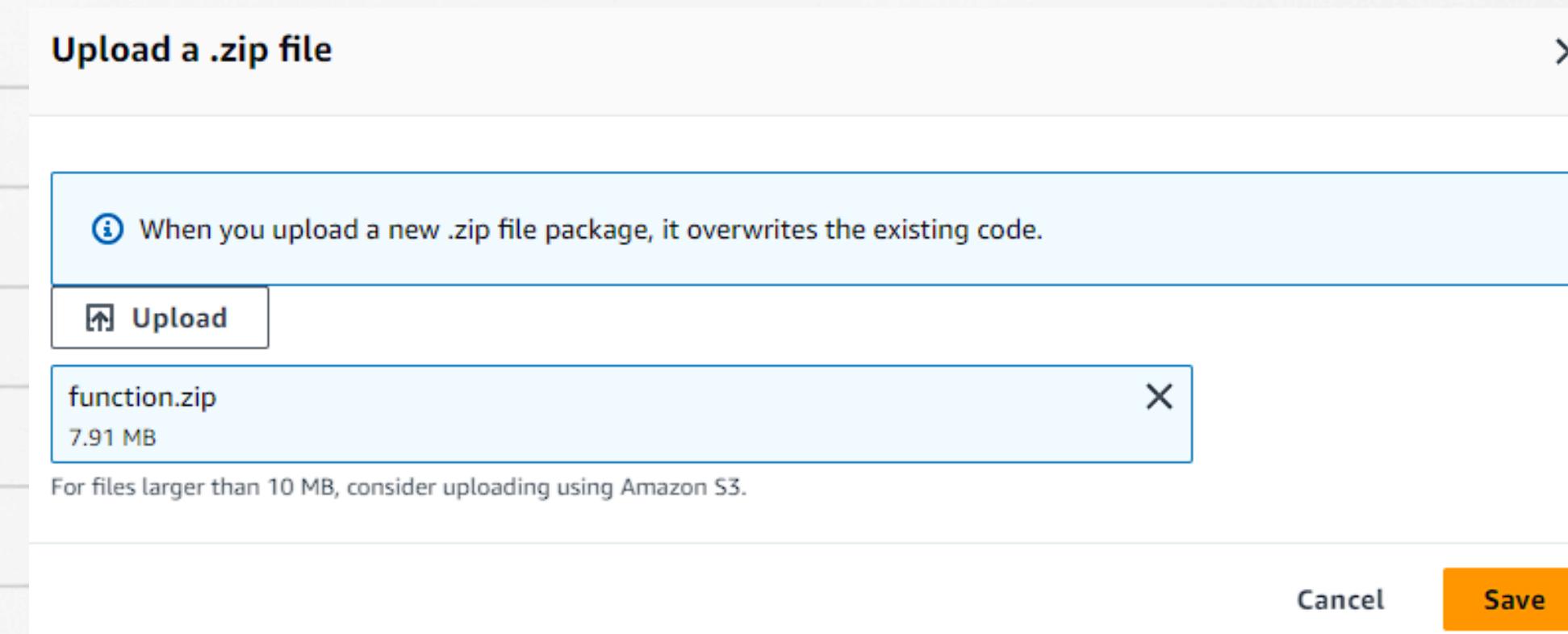


The screenshot shows the 'Edit environment variable' dialog box. It contains a table with two columns: 'Key' and 'Value'. The 'Key' column has a single entry 'DUST_BUCKET' and the 'Value' column has a single entry 'mydestinationbucketanurag'. To the right of the 'Value' column is a 'Remove' button. Below the table is a button labeled 'Add environment variable'. At the bottom of the dialog are 'Cancel' and 'Save' buttons.

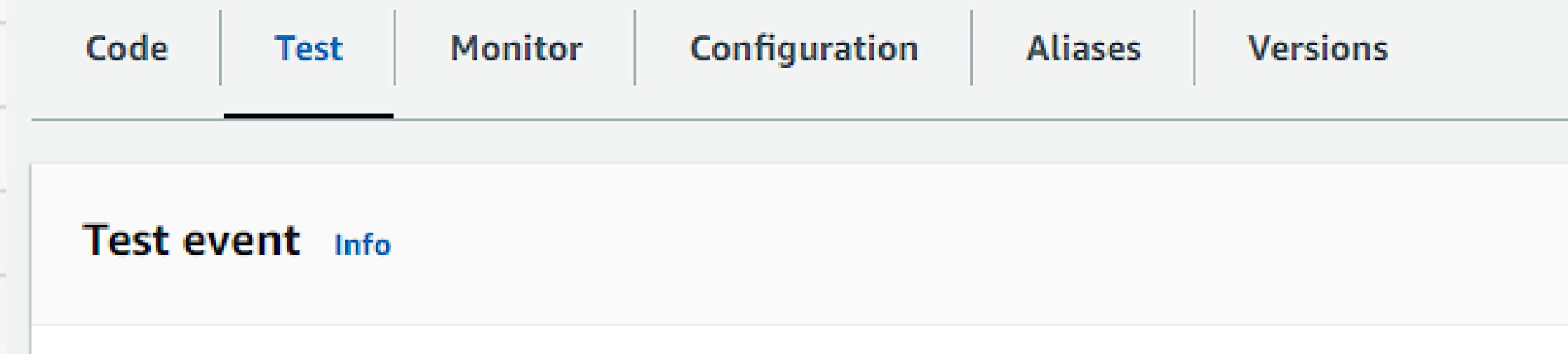
STEP 26:- Now click on the source code and click on the upload from and select .zip file



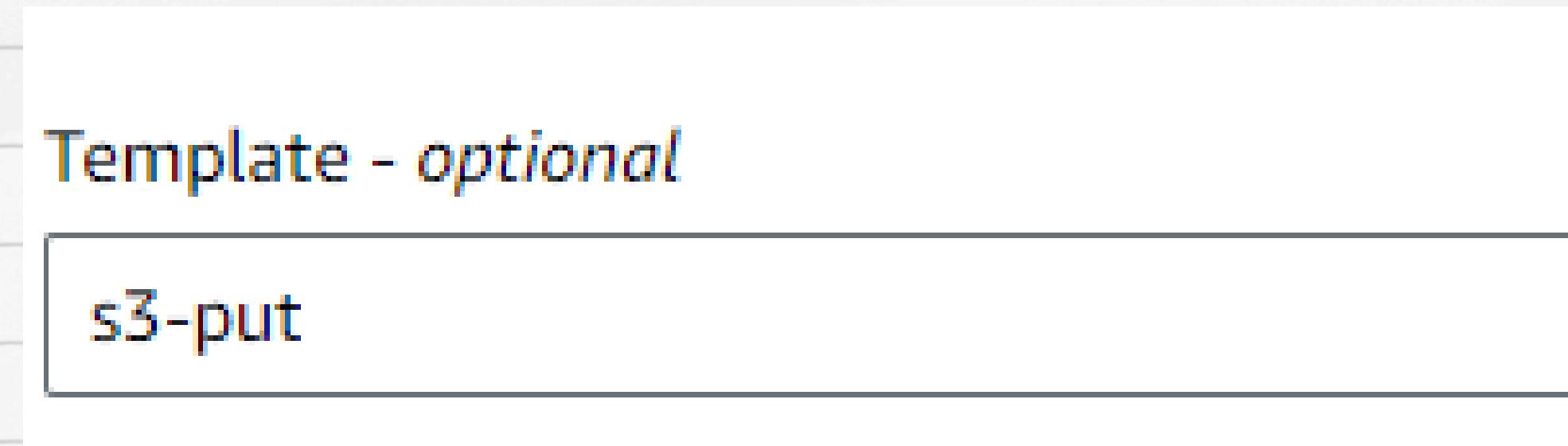
Now upload the given function.zip file



STEP 27:- After that click on the Test



STEP 28:- Now, click on the Template - optional and select the s3-put



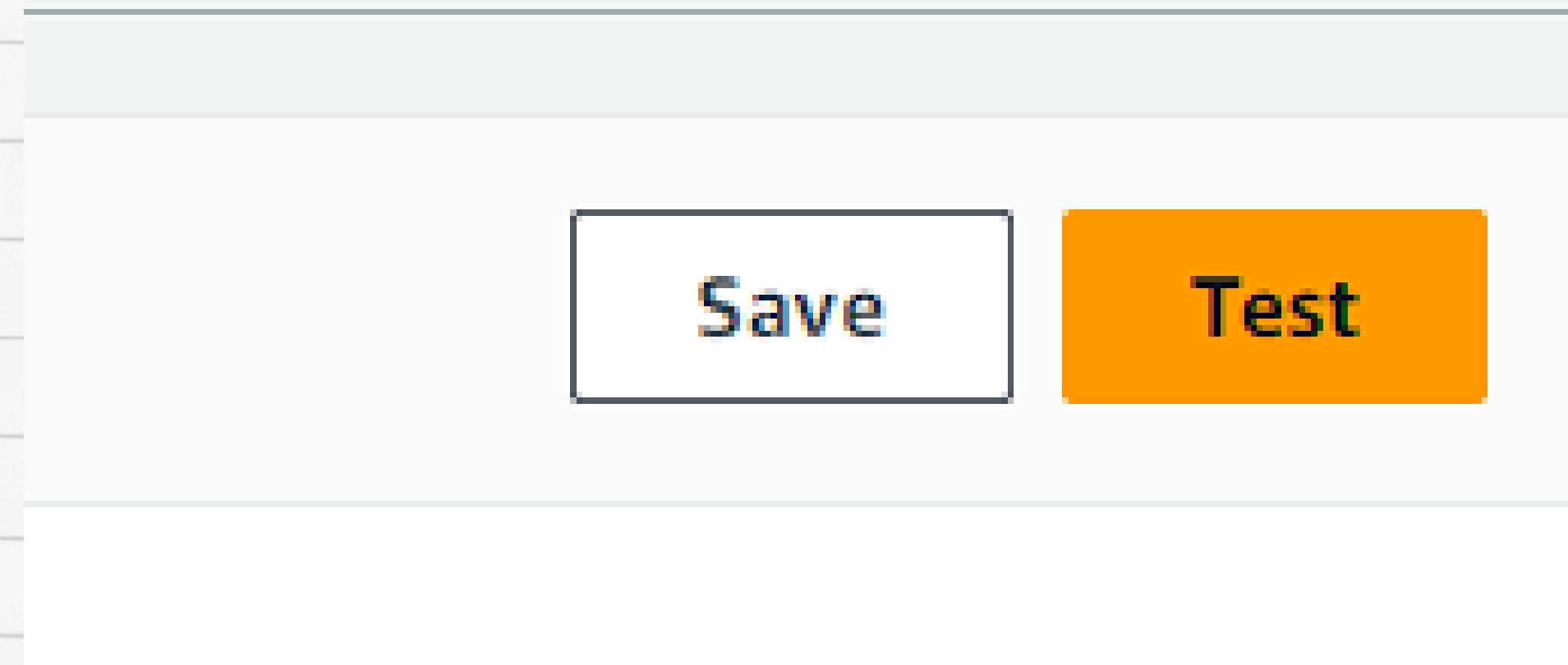
STEP 28:- Now, put the source bucket name : mysourcebucketanurag in the given code

Also put the uploaded file name in the code KEY:shinchan.jpg

```
{  
    "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":  
                    "EXAMPLE123/5678abcdefghijklmbaisawesome/mnopqrstuvwxyzABCDEFGHI"  
            },  
        }  
    ]  
}
```

```
        "s3": {
            "s3SchemaVersion": "1.0",
            "configurationId": "testConfigRule",
            "bucket": {
                "name": "mysourcebucketanurag",
                "ownerIdentity": {
                    "principalId": "EXAMPLE"
                },
                "arn": "arn:aws:s3:::mysourcebucketanurag"
            },
            "object": {
                "key": "shinchan.jpg",
                "size": 1024,
                "eTag": "0123456789abcdef0123456789abcdef",
                "sequencer": "OA1B2C3D4E5F678901"
            }
        }
    }
]
```

STEP 29:- Now, click on the Test



STEP 30:- After the Test completed Now, go to the S3 bucket and click on the
mydestinationbucketanurag
Where you can show the uploaded file in the destination bucket now
select the file and click on the open

mydestinationbucketanurag Info

Objects

Properties

Permissions

Metrics

Management

Access Points

Objects (1) Info

Objects are the fundamental entities stored in Amazon S3. You can use [Amazon S3 inventory](#) to get a list of all objects.

 Find objects by prefix

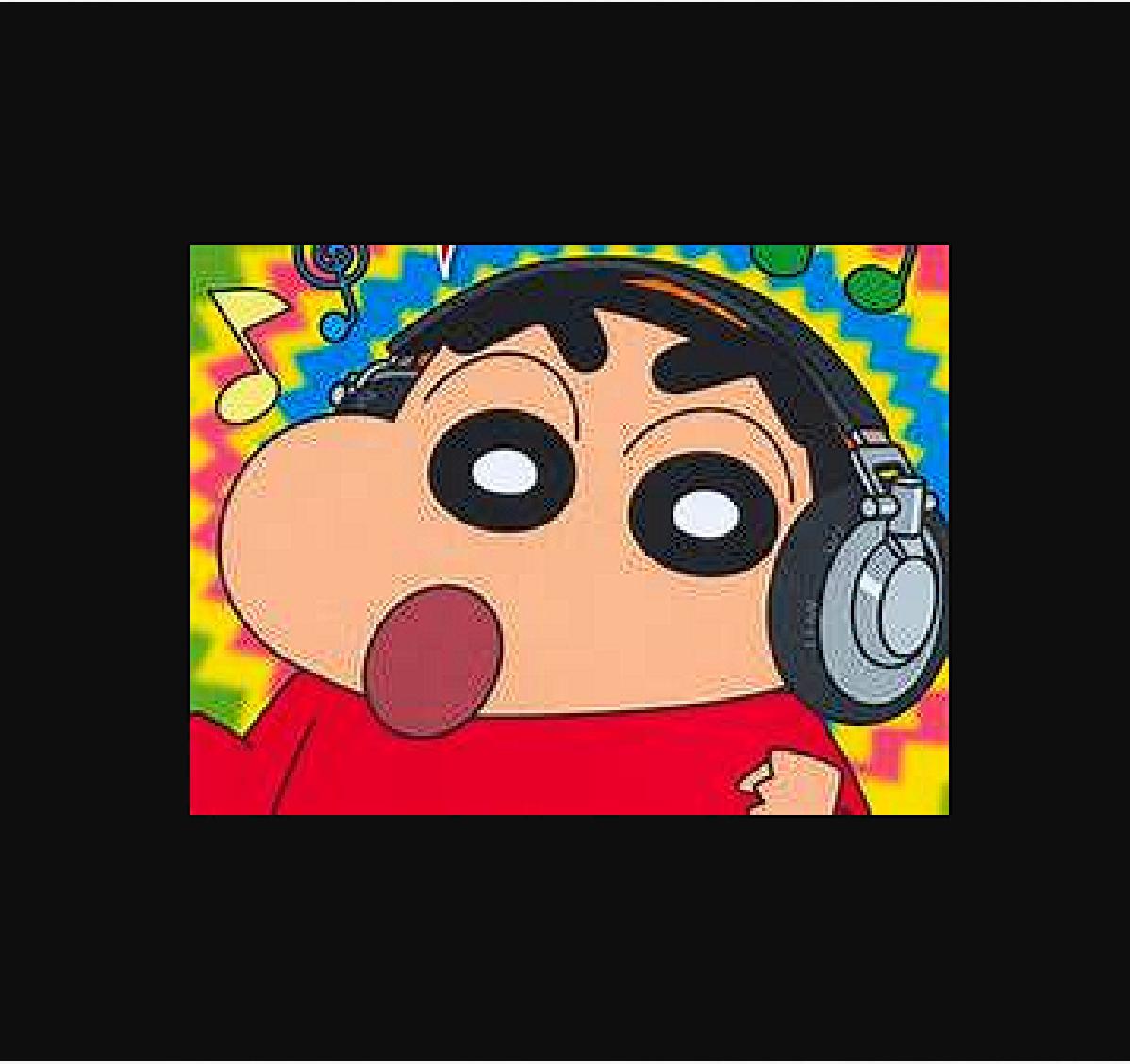
<input type="checkbox"/>	Name	Type
<input type="checkbox"/>	 shinchan.jpg	jpg

Select the file and click on the open

 Download

Open 

Explicitly grant them permissions. [Learn more](#)



THANK YOU VERY MUCH!

Anurag verma
BCA-4A