# PROJECT 1

## PRESENTATION

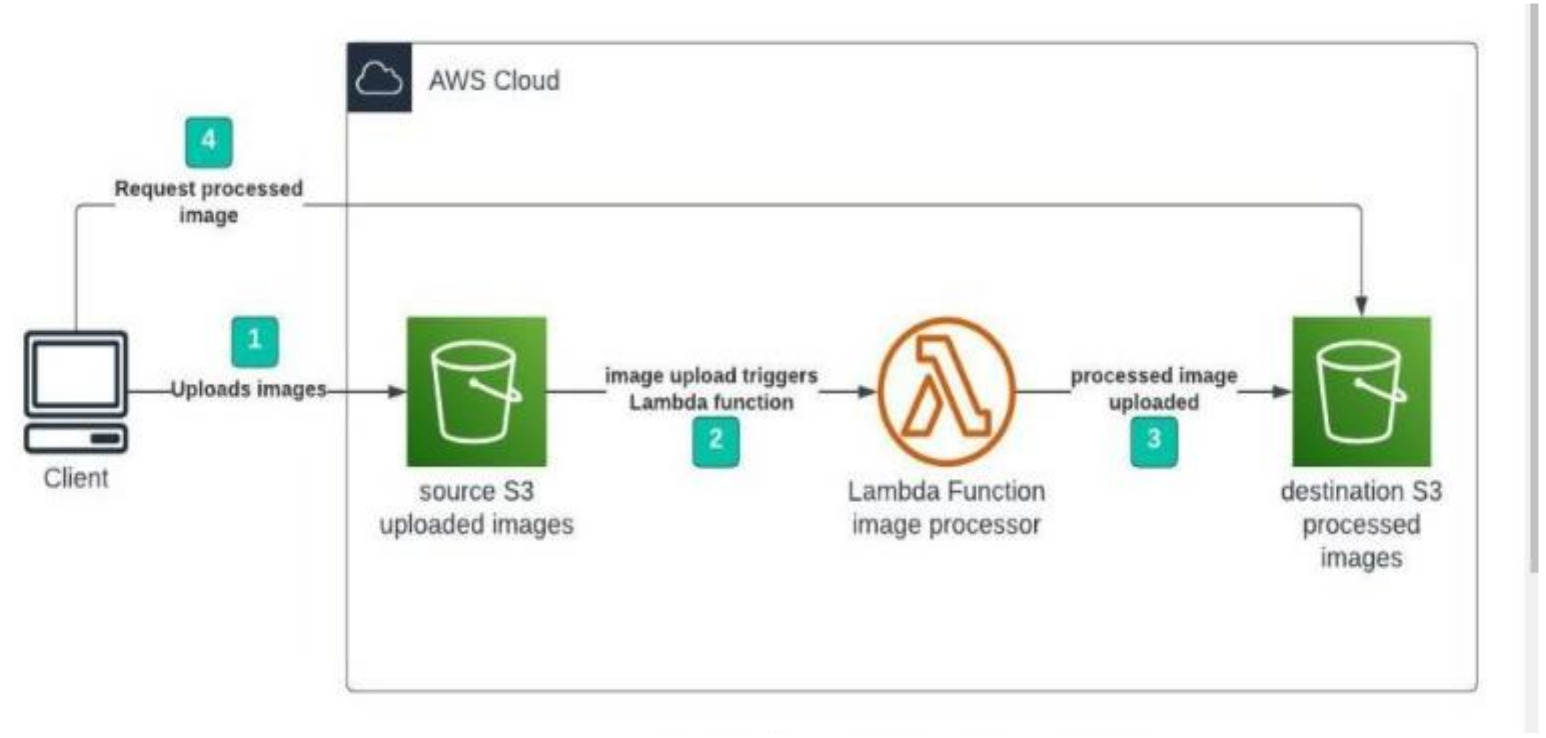
Resize Image with Node.js and
AWS Lambda(and S3)

## ABOUT AWS

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. Key services include:

*EC2 (Elastic Compute Cloud): Scalable virtual servers.
*S3 (Simple Storage Service): Object storage with high durability.
*RDS (Relational Database Service): Managed relational databases.
*Lambda: Serverless computing that runs code in response to events.
*VPC (Virtual Private Cloud): Isolated networks within the AWS cloud.

# Architecture Diagram..


AWS Cloud
4
Request processed image
1
Uploads images
Client
source S3 uploaded images
image upload triggers Lambda function
2
Lambda Function image processor
processed image uploaded
3
destination S3 processed images

## Labs Steps

**<u>Task 1: Sign in to your AWS Management Console</u>.**

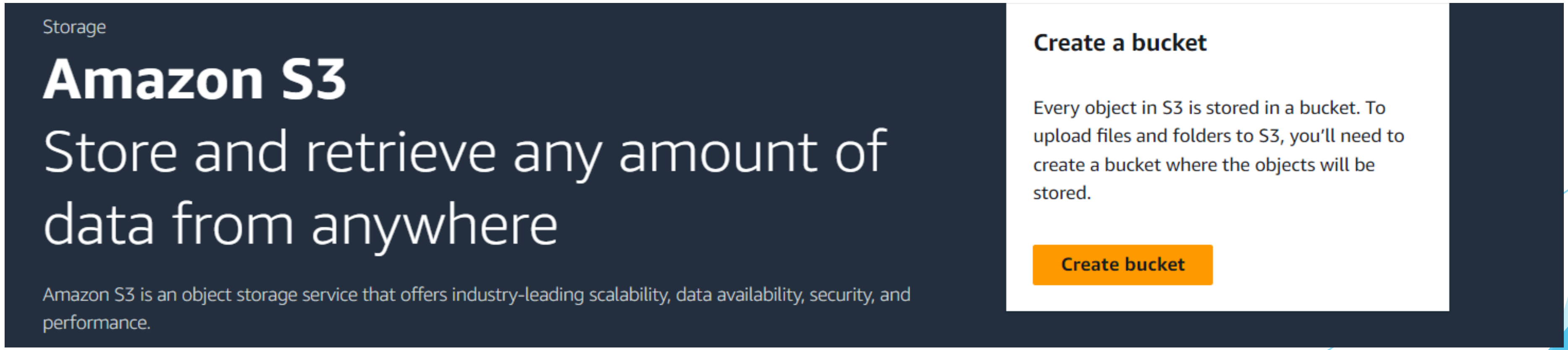
*sign in your root account , make the default AWS region as asia pacific (MUMBAI) ap-south-1.

**<u>Task 2: Create to Amazon S3 Bucket</u>**

In this , we will create two buckets first is the source bucket and destination bucket.

1- Go to the services menu in the top , then search S3 .

2- Click on create bucket.



Storage

Amazon S3

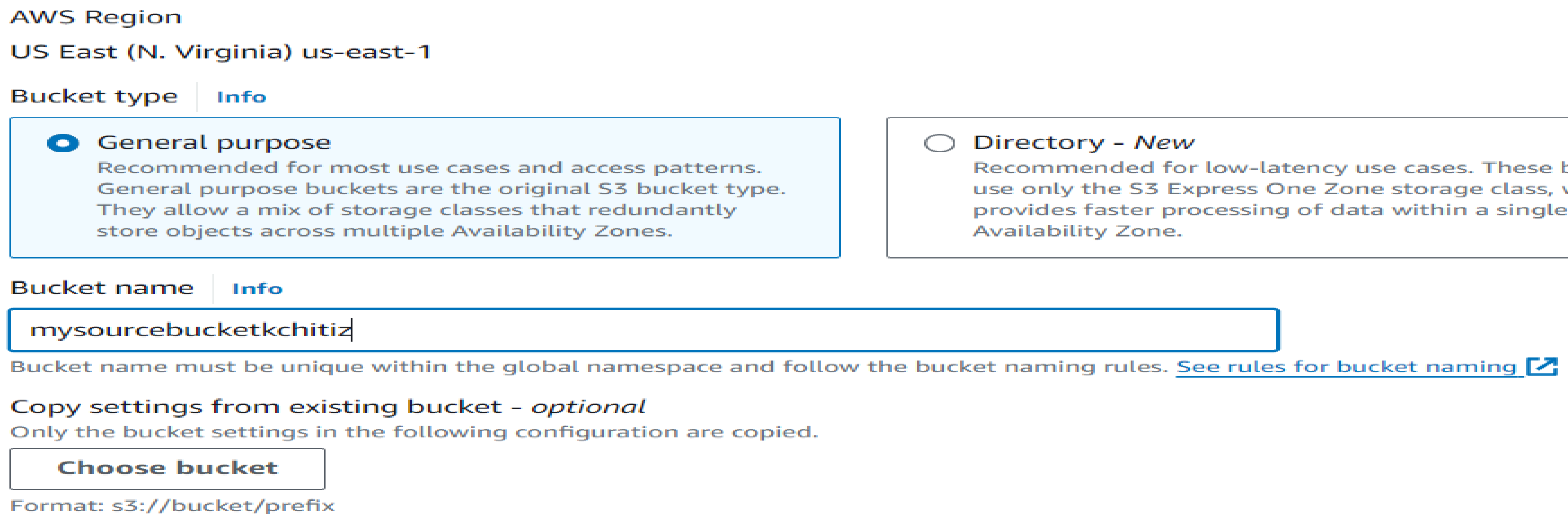
Store and retrieve any amount of data from anywhere

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.

## 3- Firstly we create source bucket , Enter your bucket name .


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

mysourcebucketkchitiz

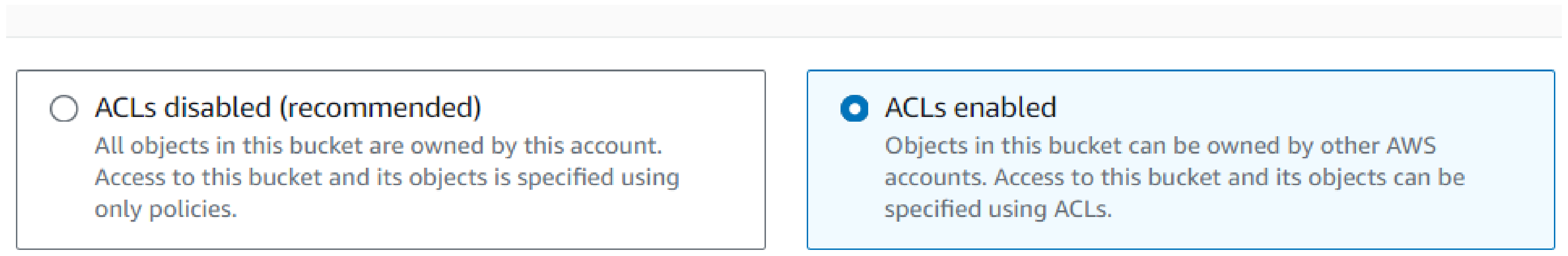
Bucket name must be unique within the global namespace and follow the bucket naming rules. See rules for bucket naming

Copy settings from existing bucket - *optional*

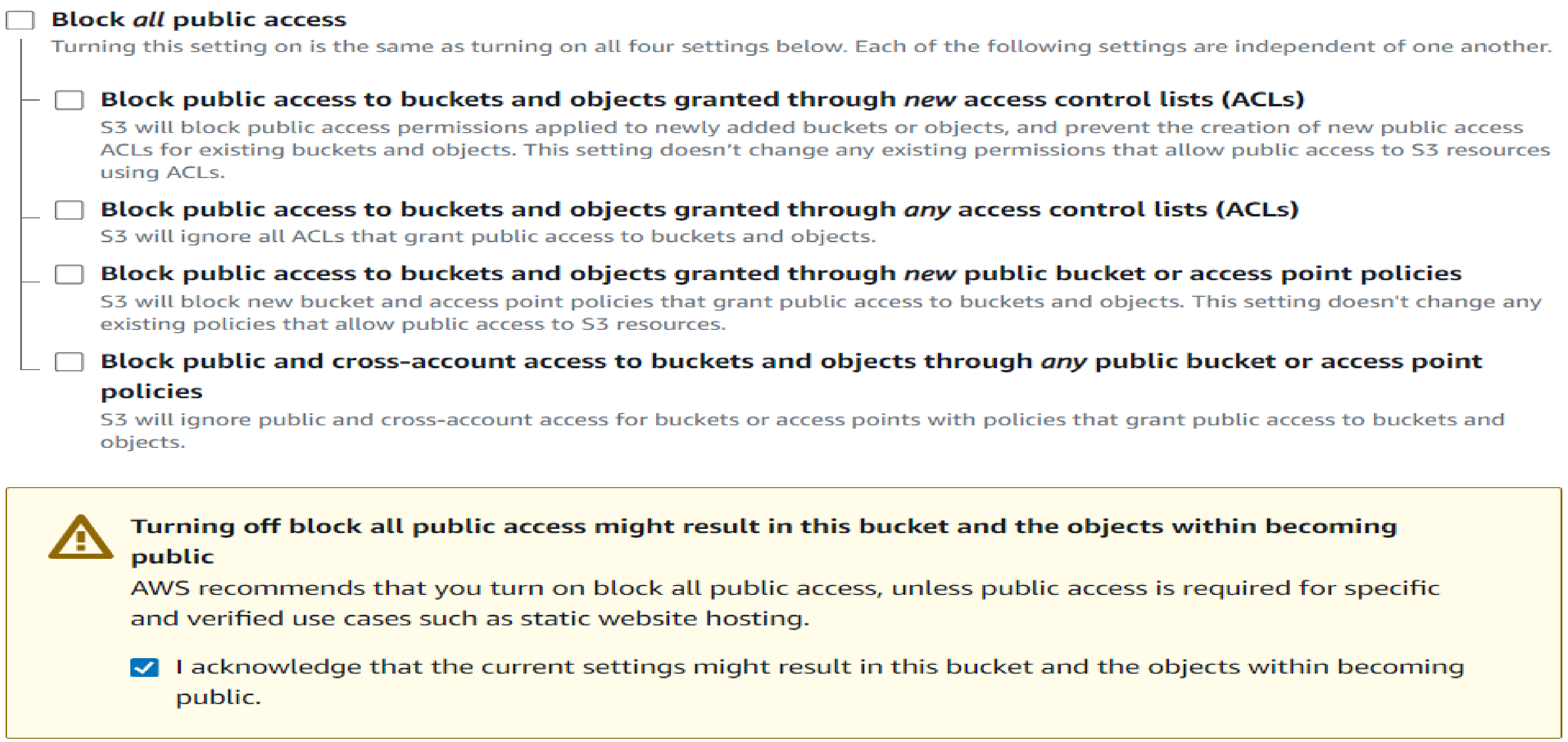
Only the bucket settings in the following configuration are copied.

Format: s3://bucket/prefix

## 4- put ACLs as enable, and block all public access.


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.


☐ 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.

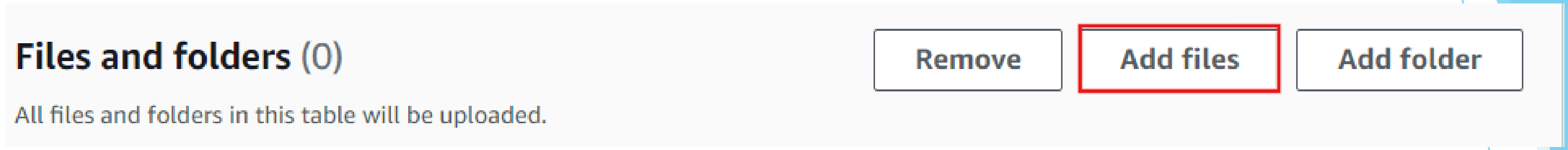
**Turning off block all public access might result in this bucket and the objects within becoming public**
AWS recommends that you turn on block all public access, unless public access is required for specific and verified use cases such as static website hosting.

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

5- Leave other settings as default and click on create button.

6- **Create destination bucket**
* Leave all the settings as same as in source bucket.

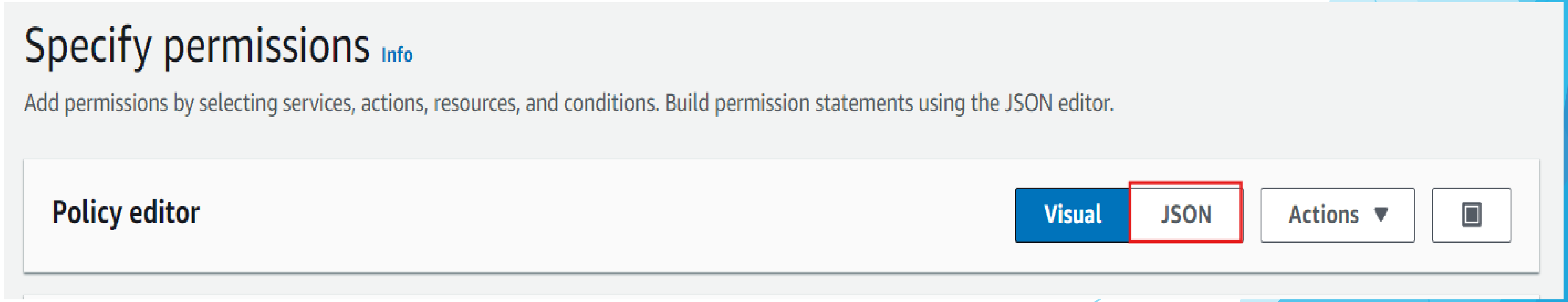
7- Go to the source bucket and upload an image by clicking on add files.


Files and folders (0)

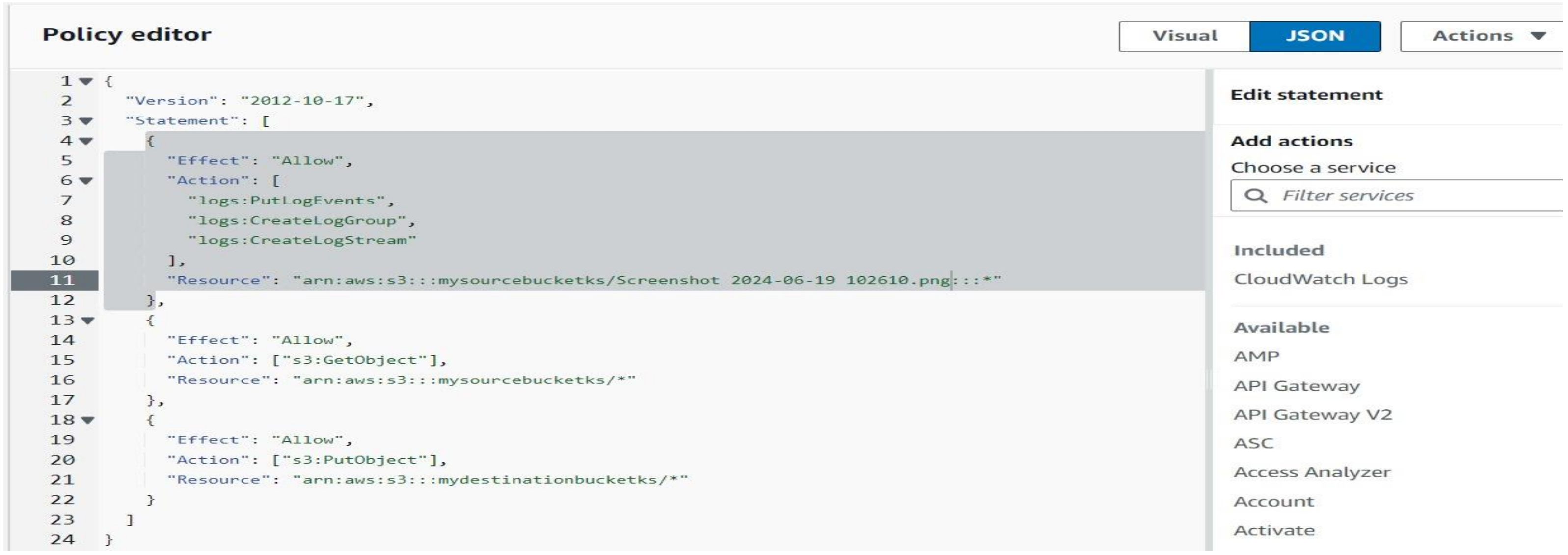
All files and folders in this table will be uploaded.

## Task 3: Create an IAM POLICY.

1- As a pre – requisite for creating lambda function, We firstly create policy and user role.
2- click on POLICIES in the left and click on create policy.
3- Now, click on the **JSON** tab and change the existing code with the given below code .


Specify permissions Info

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

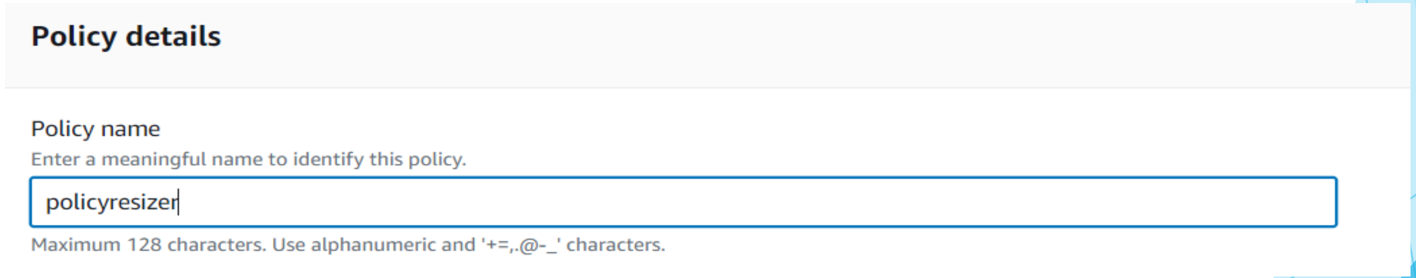


```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Effect": "Allow",
      "Action": [
        "logs:PutLogEvents",
        "logs:CreateLogGroup",
        "logs:CreateLogStream"
      ],
      "Resource": "arn:aws:s3:::mysourcebucketks/Screenshot 2024-06-19 102610.png:::*"
    },
    {
      "Effect": "Allow",
      "Action": ["s3:GetObject"],
      "Resource": "arn:aws:s3:::mysourcebucketks/*"
    },
    {
      "Effect": "Allow",
      "Action": ["s3:PutObject"],
      "Resource": "arn:aws:s3:::mydestinationbucketks/*"
    }
  ]
}
```

4- After implementing this code , replace the **source** and **destination ARN name** of the bucket.

*leave all other setting as default and click on next button.

5- Now, we are at review policy page , Enter your policy name .



**Policy details**

Policy name

Enter a meaningful name to identify this policy.

policyresizer

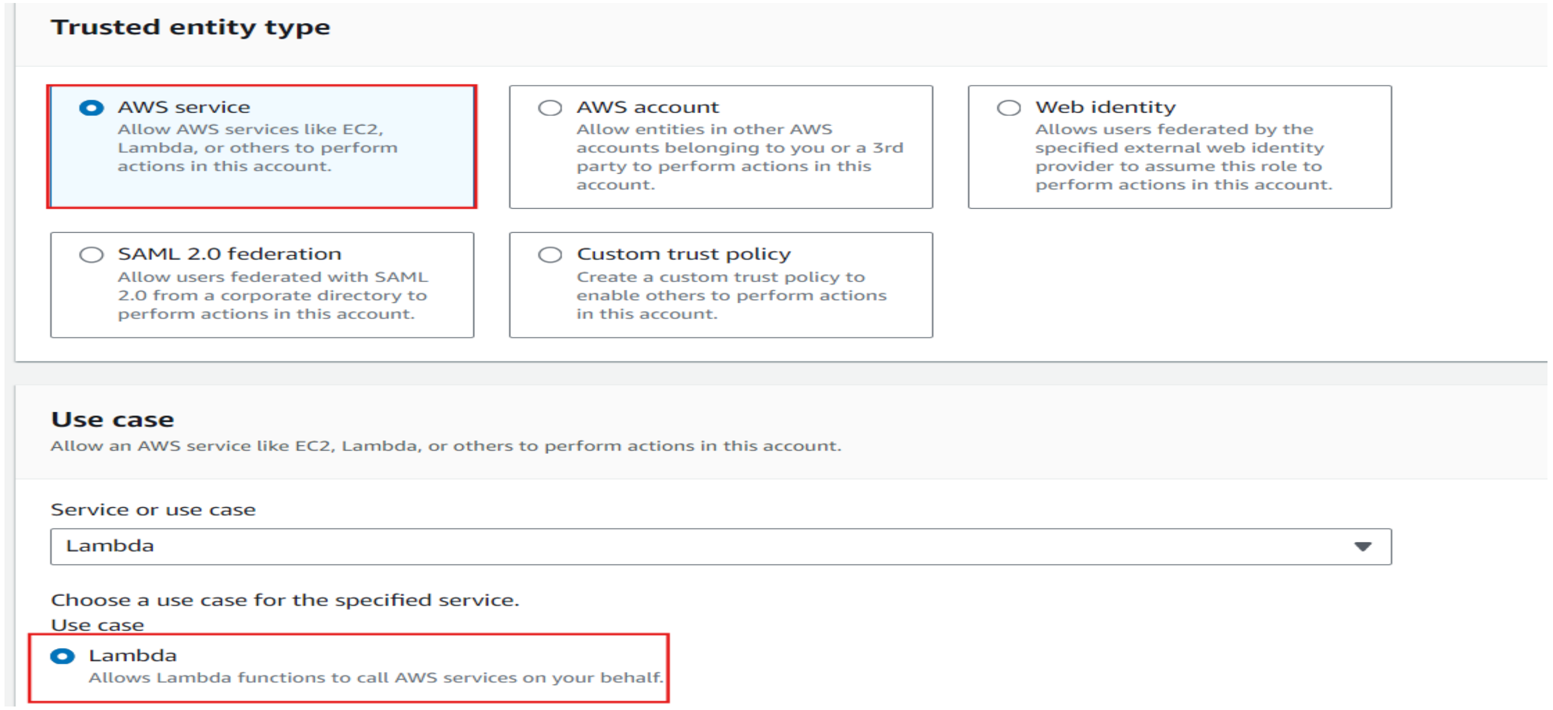
Maximum 128 characters. Use alphanumeric and '+=,.@-_' characters.

* Click on the create policy button

6- An IAM policy is successfully created.

## Task 4: Create an IAM Role.

1- In the left menu, Click on the **Roles** and click on the create role button .
 *Choose the AWS service from the Trusted entity type.
 *Choose the **Lambda** from **Use case.**
 *Click on the Next button



Trusted entity type

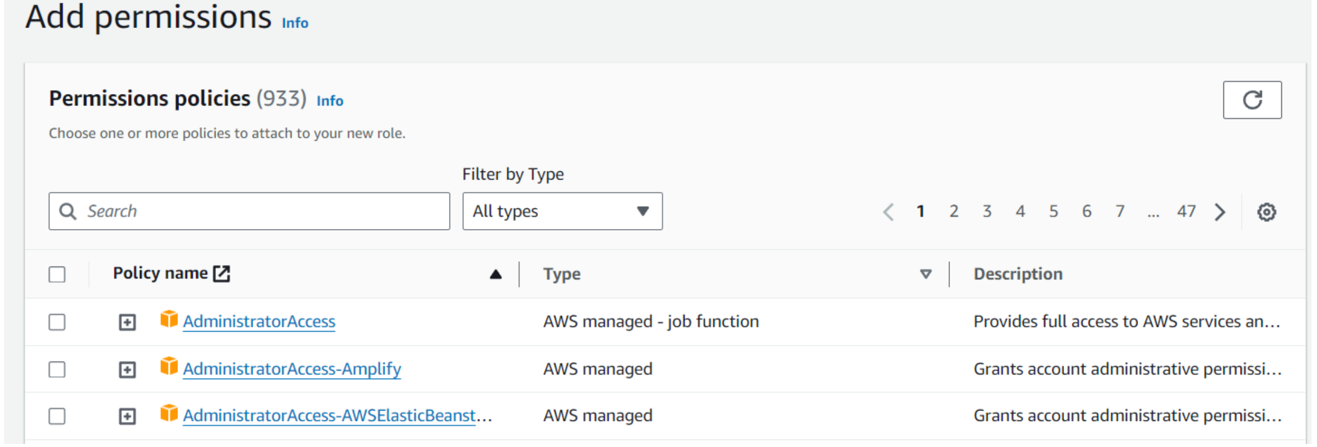
- [✓ AWS service] Allow AWS services like EC2, Lambda, or others to perform actions in this account.
- AWS account: Allow entities in other AWS accounts belonging to you or a 3rd party to perform actions in this account.
- Web identity: Allows users federated by the specified external web identity provider 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.

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.

*Now we select our policy from the list policies , which we created in last task.



Add permissions Info

Permissions policies (933) Info

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

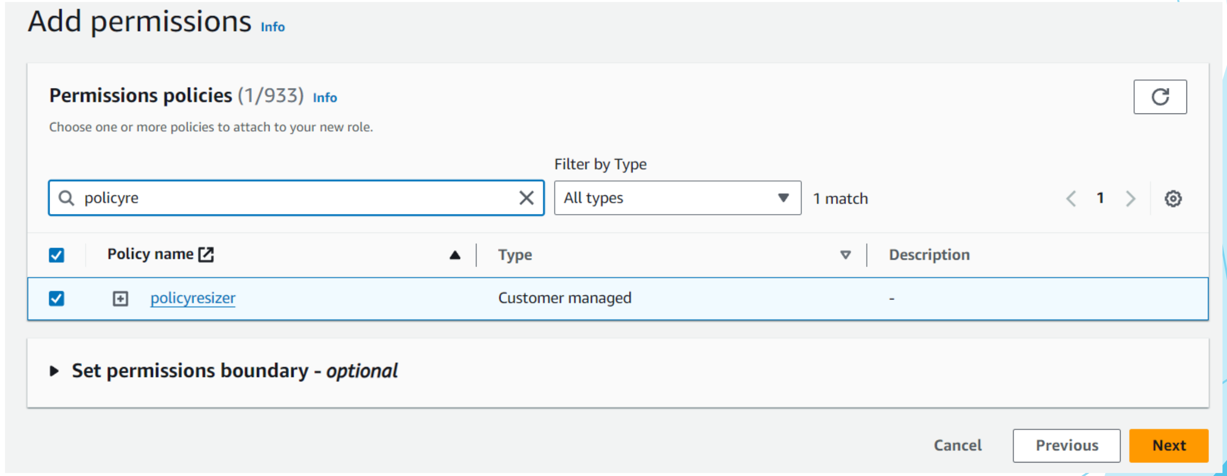
Filter by Type: All types

| Policy name | Type | Description |
|---|---|---|
| AdministratorAccess | AWS managed - job function | Provides full access to AWS services an… |
| AdministratorAccess-Amplify | AWS managed | Grants account administrative permissi… |
| AdministratorAccess-AWSElasticBeanst… | AWS managed | Grants account administrative permissi… |

*Select your policy and click on **next** button .

*Enter **Role** name .

*Click on the **Create Role** button.



Add permissions Info

Permissions policies (1/933) Info

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

Search: policyre

Filter by Type: All types

1 match

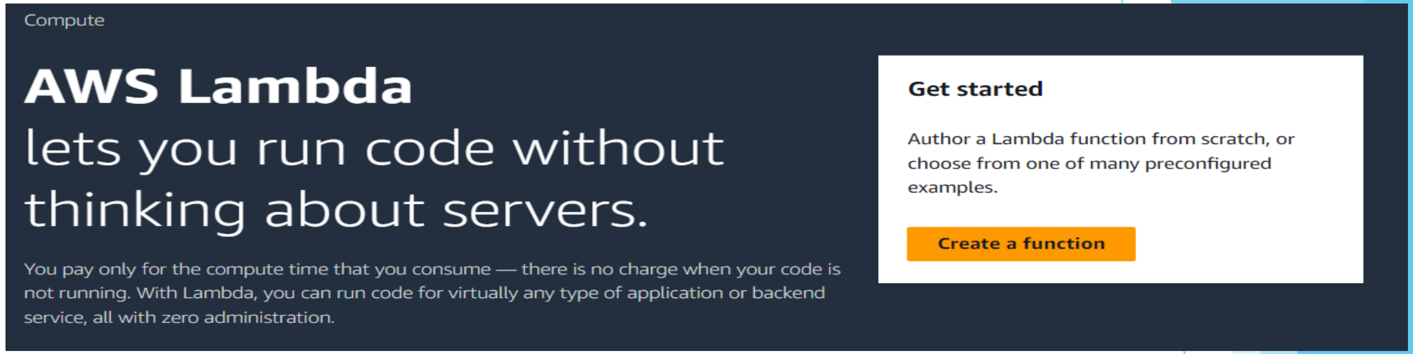
| Policy name | Type | Description |
|---|---|---|
| ✓ policyresizer | Customer managed | - |

Set permissions boundary - *optional*

*We have successfully created an IAM Role .

## TASK 5: Create a Lambda Function.

1- Go to the SERVICES menu and click on the **Lambda** under Compute Section.
2- Click on the create a Function button.



Compute

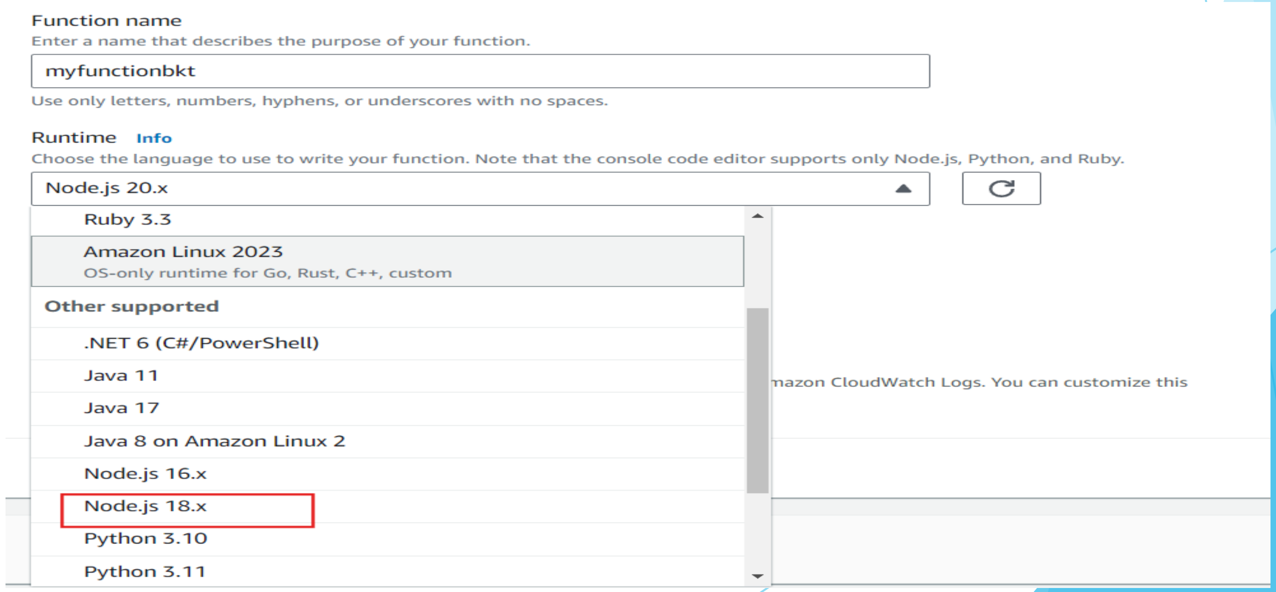
**AWS Lambda**
lets you run code without thinking about servers.

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.

*Choose Author from the scratch.
*Enter the function name.
*Runtime: Select the **node.js 18.x**

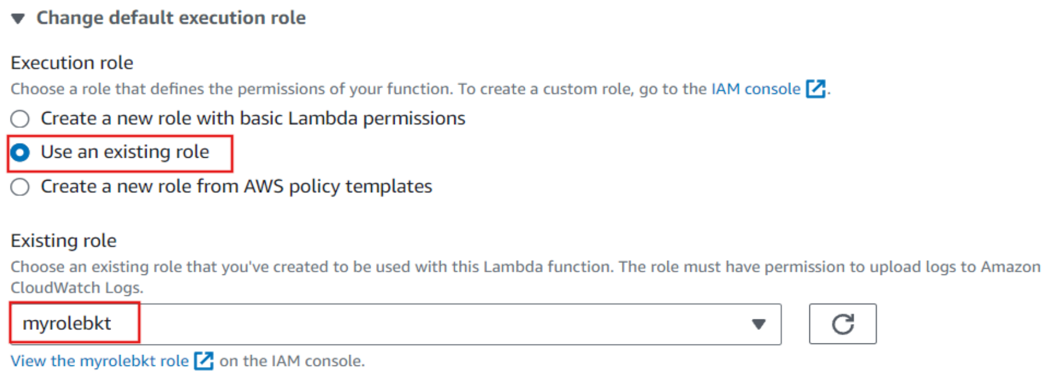


Function name
Enter a name that describes the purpose of your function.
myfunctionbkt
Use only letters, numbers, hyphens, or underscores with no spaces.

Runtime Info
Choose the language to use to write your function. Note that the console code editor supports only Node.js, Python, and Ruby.
Node.js 20.x

3- Click on the **Change Default execution role** and then select the Use an existing role.

*Select the role which we created earlier.



▼ 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.

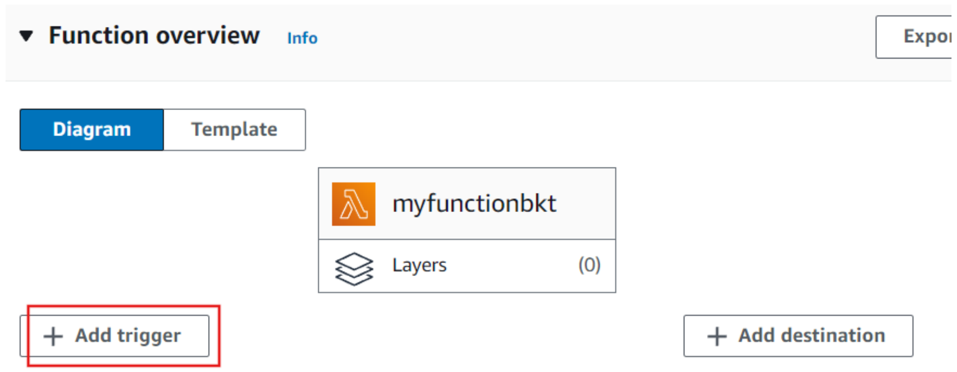
myrolebkt

View the myrolebkt role on the IAM console.

4- click on the **Create Function** button .

*We successfully created function ,now we **Add Trigger** in the function .

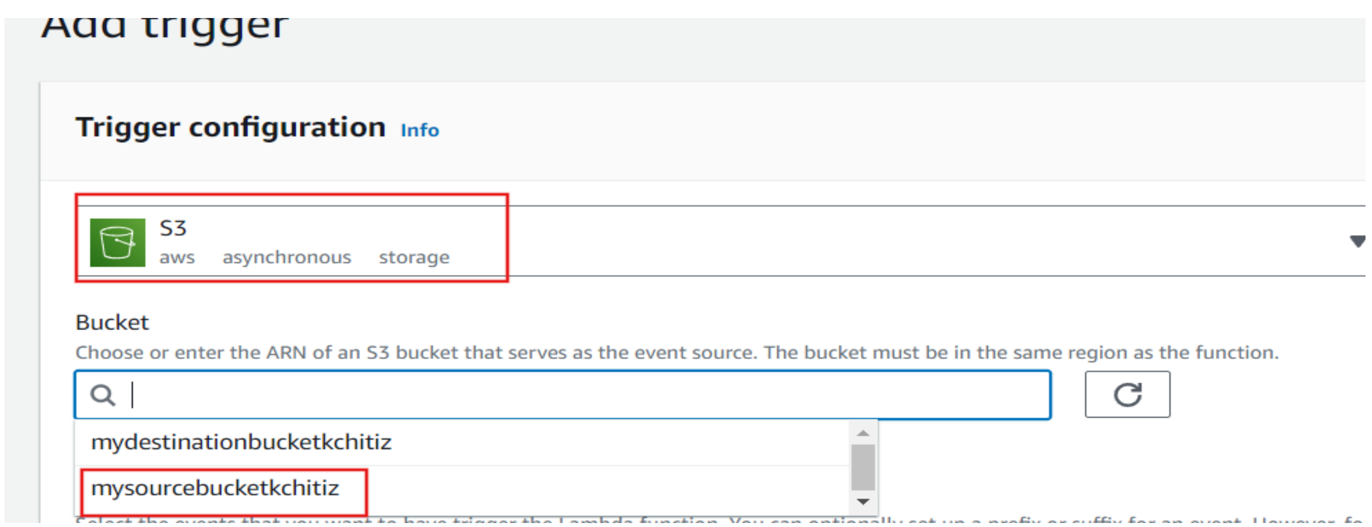
## <u>Task: 6 Adding Triggers to Lambda Function</u>



1 - After clicking on add trigger , we select the source bucket in the bucket column.

2- Tick the acknowledgment box in the **Recursive Innovation**.

*leave other settings as it is, and click on the Add button.



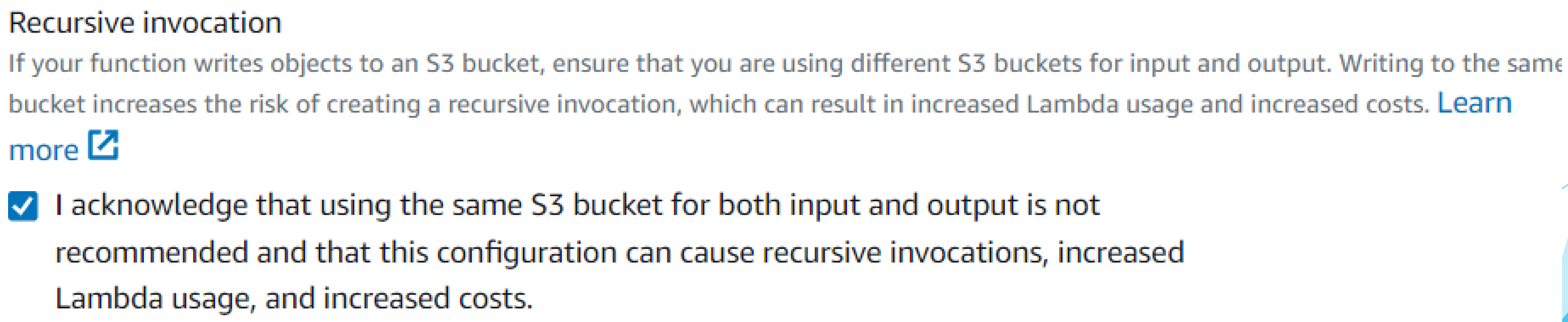
Add trigger

**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.

mydestinationbucketkchitiz
mysourcebucketkchitiz

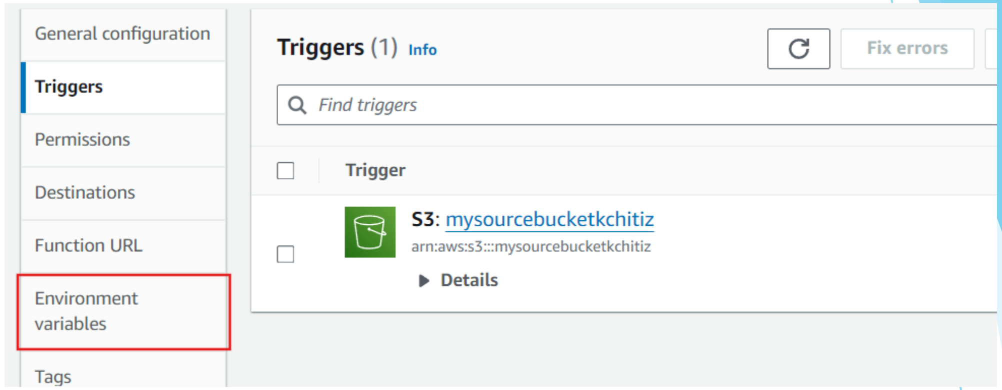


Recursive invocation

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

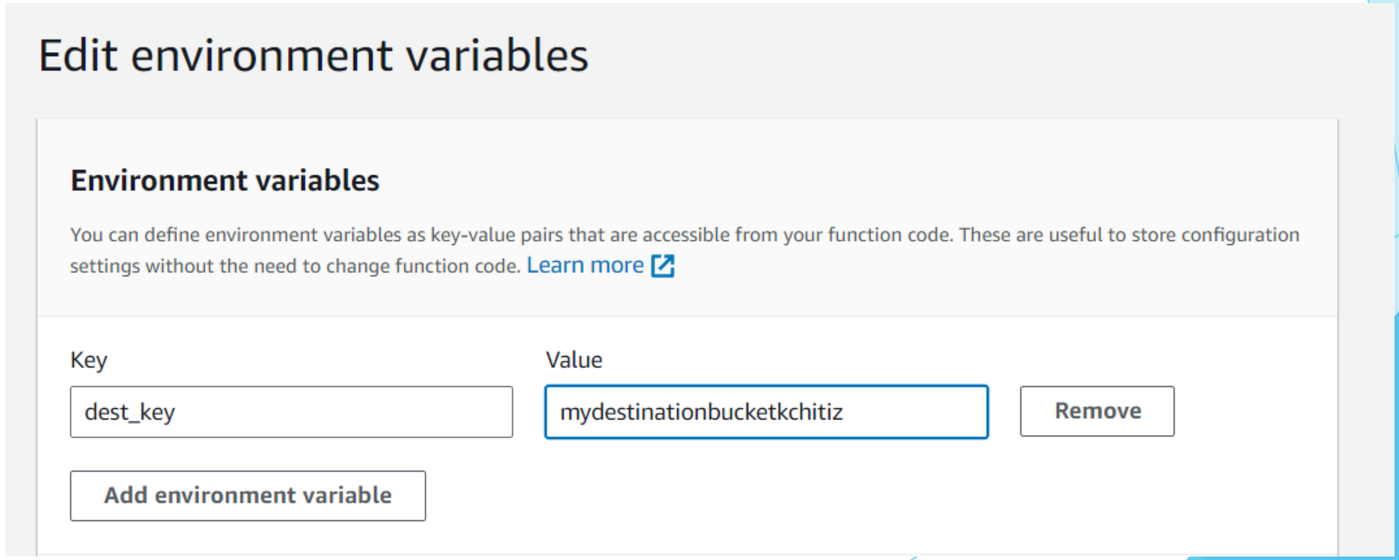
☑ I acknowledge that using the same S3 bucket for both input and output is not recommended and that this configuration can cause recursive invocations, increased Lambda usage, and increased costs.

*After adding the trigger , scroll down the and go to the Configuration
and click on **Environment Variables**.
*Now, edit the Environment Variables
*Enter the key & Value Name ,



**NOTE**- Add the Destination bucket name in the column of value in Environment Variables.

*Click on the Save button


Edit environment variables

Environment variables

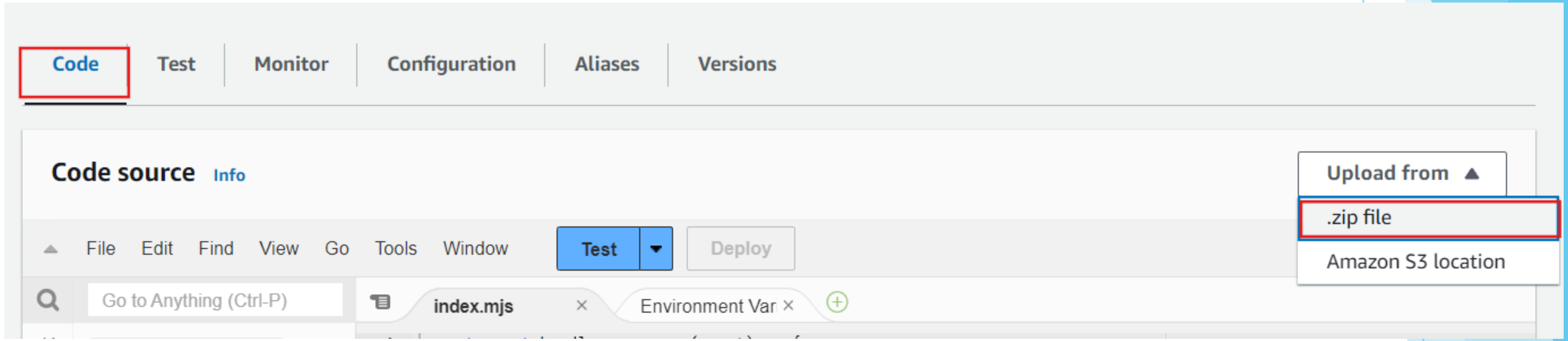
You can define environment variables as key-value pairs that are accessible from your function code. These are useful to store configuration settings without the need to change function code. Learn more

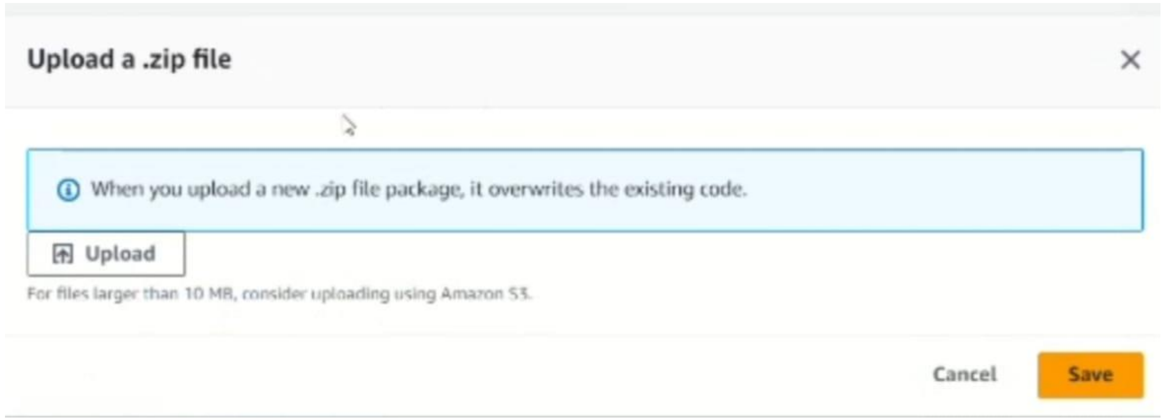
| Key | Value |
|---|---|
| dest_key | mydestinationbucketkchitiz |

*Now, upload the **Zip File**

*For uploading the Zip file go in the column of code and click on **Upload from** and select **.zip file.**

*Select the zip file function from the downloads of the desktop, and Upload the file by clicking on **Save** button.



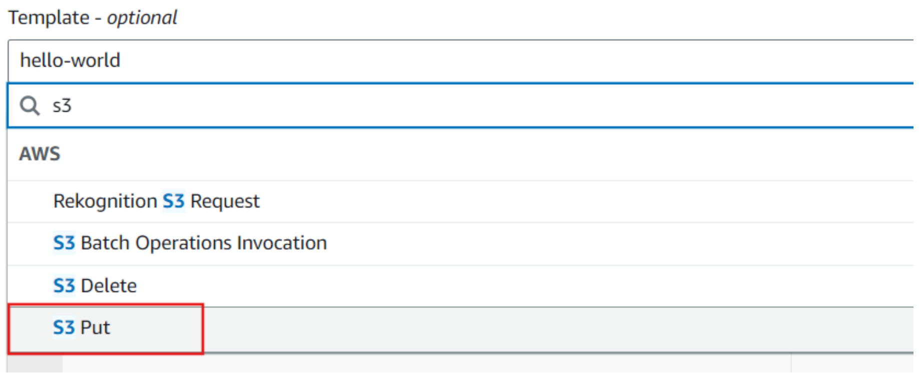


When you upload a new .zip file package, it overwrites the existing code.

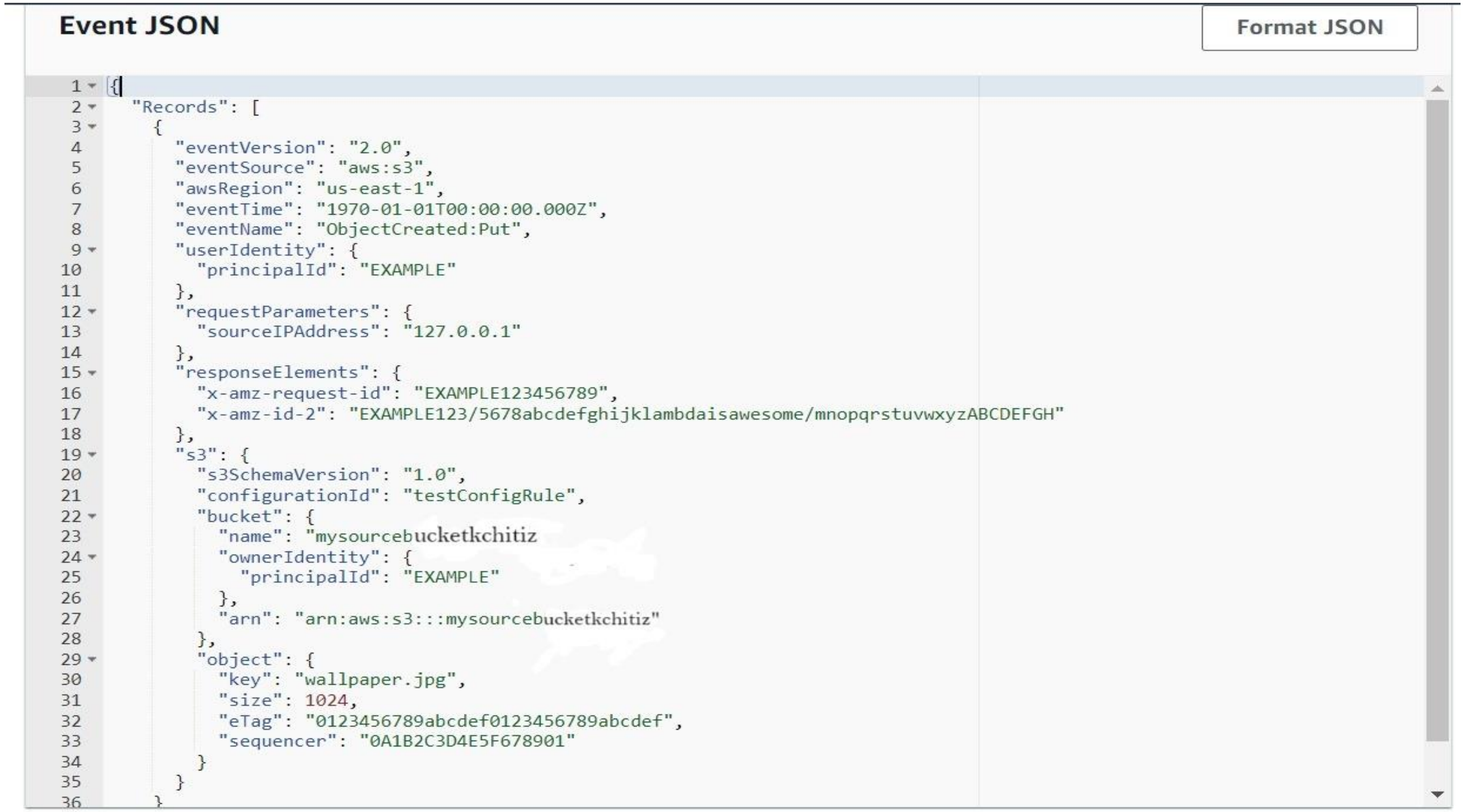
For files larger than 10 MB, consider uploading using Amazon S3.

## Task 7: Test the Lambda Function

*After successfully Upload the zip file , click on the **Test** just next night to the Code column .
*Here we scroll down and select **S3 PUT** in the Template option .


Template - *optional*
hello-world
s3
AWS
Rekognition S3 Request
S3 Batch Operations Invocation
S3 Delete
S3 Put

*Just selecting the S3 put and we seen the Lambda Function code.
*There is some changes in this code , Firstly we copy the source bucket name and paste it in the name of bucket and paste the ARN of the source bucket in place of existing ARN .
*Also copy the Name of the image which we Uploaded on the Source bucket , paste it in the code .


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": {
          "name": "mysourcebucketkchitiz
          "ownerIdentity": {
            "principalId": "EXAMPLE"
          },
          "arn": "arn:aws:s3:::mysourcebucketkchitiz"
        },
        "object": {
          "key": "wallpaper.jpg",
          "size": 1024,
          "eTag": "0123456789abcdef0123456789abcdef",
          "sequencer": "0A1B2C3D4E5F678901"
        }
      }
    }
```

*After doing all the changes scroll up and click on the Test button and Execute the function .

*Now go back to the S3 list and open your Destination bucket.

*We can see the a compress image in our destination bucket , which we upload in our source bucket.

## Thank You

presented by:~Kchitiz Saxena