



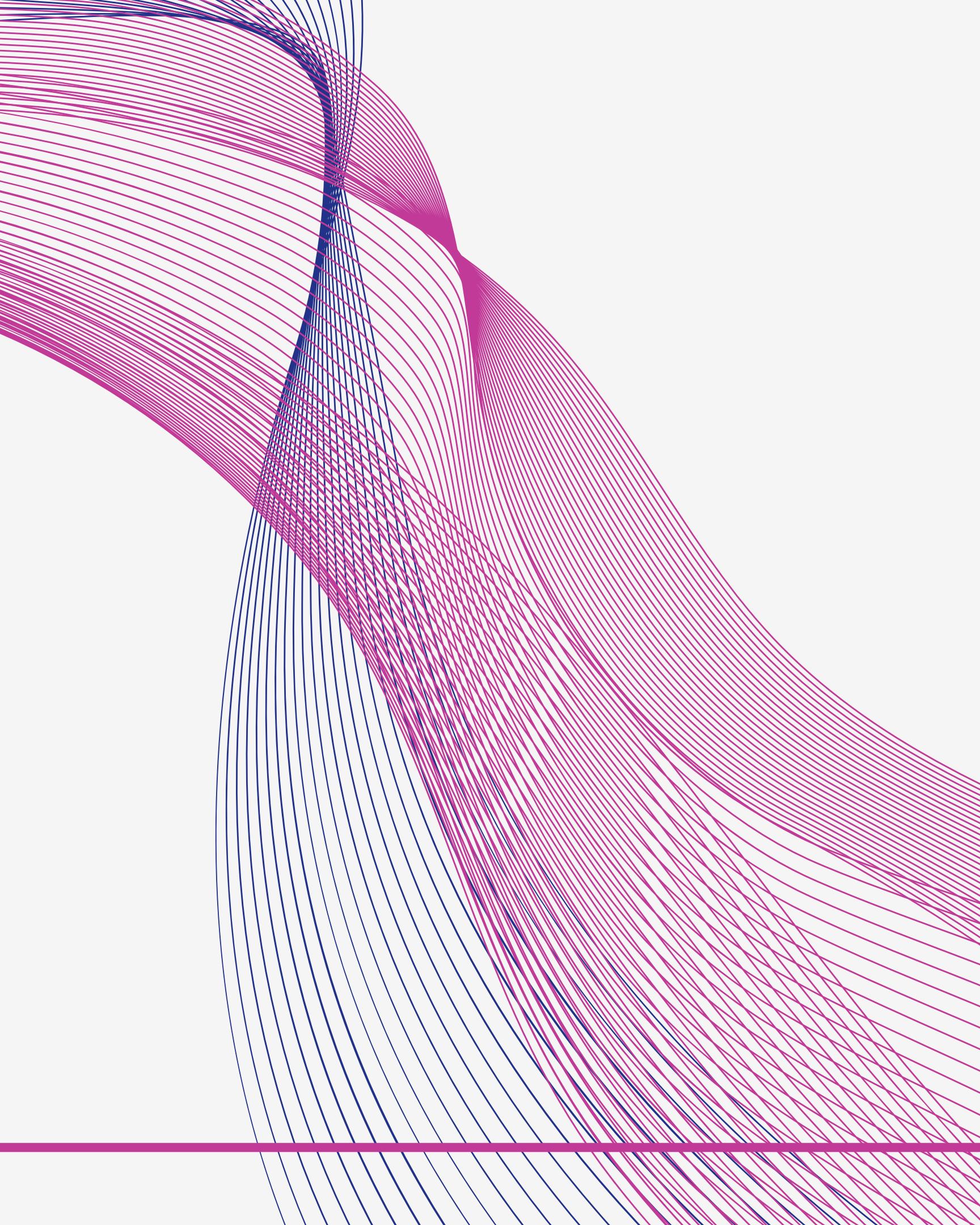
PROJECT -1

# SERVERLESS IMAGE PROCESSING

---

PRESENTED BY AMOD PATHAK ➤





# **INTRODUCTION :-**

Serverless Image Handler on AWS creates a serverless architecture to initiate cost-effective image processing in the AWS Cloud. The architecture combines AWS services with sharp, an open-source image processing software, and is optimized for dynamic image manipulation. You can use this AWS Solution to help you maintain high-quality images on your websites and mobile applications to drive user engagement.

## **GOALS TO ACHIEVE:-**

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

## STEP 1 : Sign in to AWS console

1. Go to AWS console and enter your details and log in to AWS cloud services

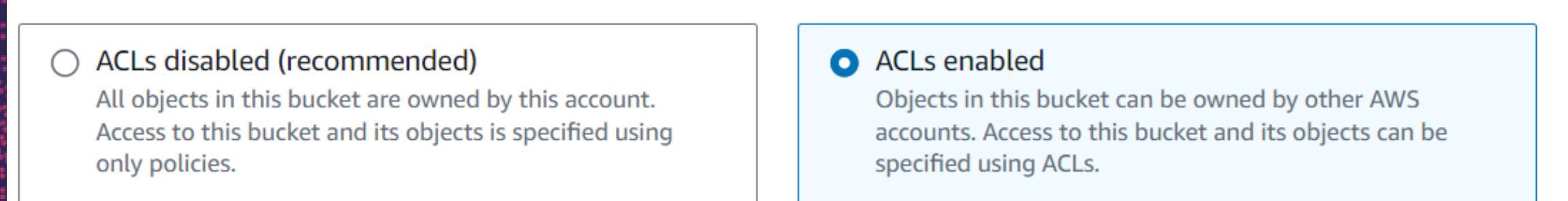
## STEP 2 : CREATE A BUCKET

2. Click on SERVICES and open S3 bucket
3. Now click on create bucket and create a source bucket
4. Enter your bucket name “mysourcebktamod”
5. Now select ACL enabled option

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

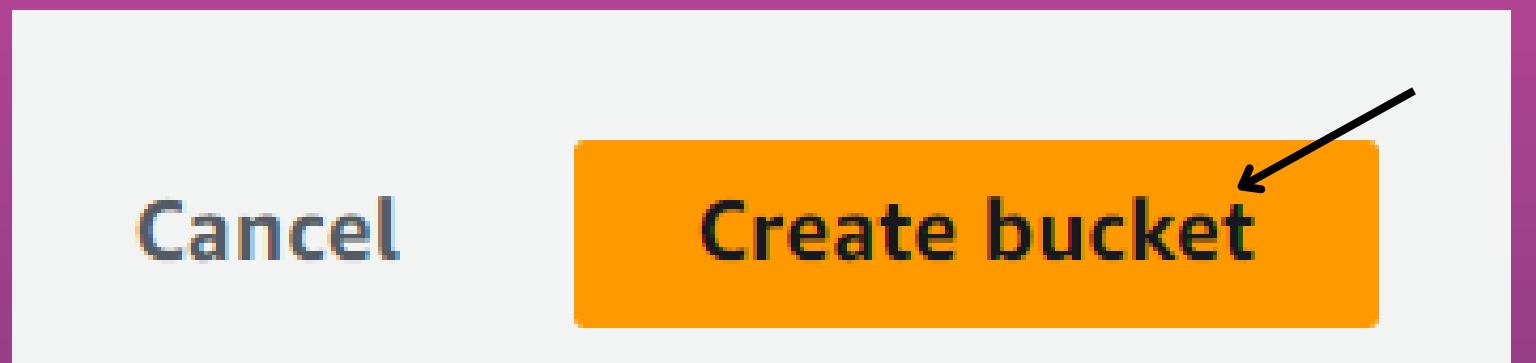
Bucket name [Info](#)  
mysourcebktamod

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



6. Remove “block all public access” and confirm the given request as mentioned

7. Leave all other options as default and create bucket

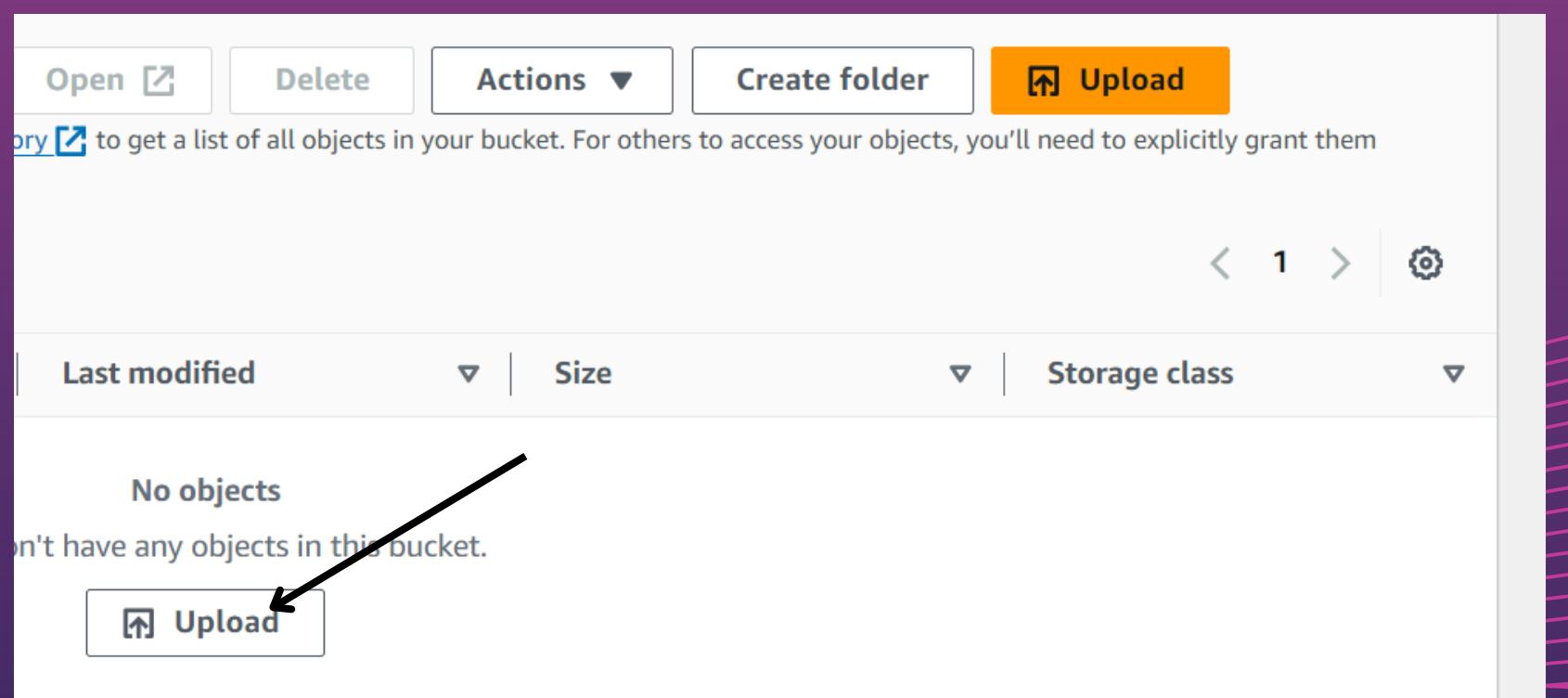


8. Just like source bucket create another bucket named as "mytargetbktamod"

Every bucket name is globally unique so select your unique bucket name

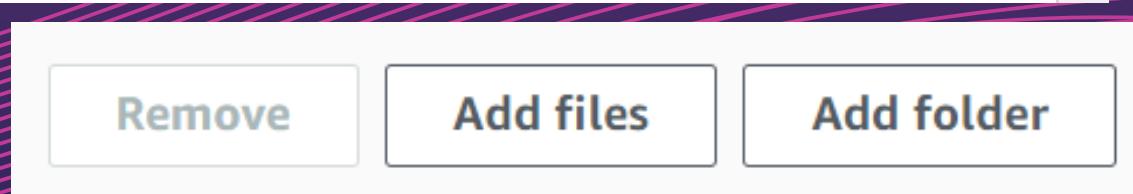
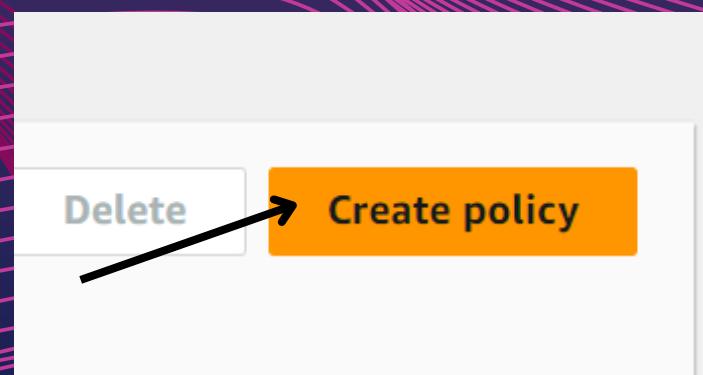
### STEP 3 : UPLOAD YOUR IMAGE

9. Open your source bucket and click on upload and using add files option you can upload your image



### STEP 4: CREATE ROLES AND POLICIES

10. In services menu go to "IAM" and click on policies and create a policy as mentioned



11. Click on Json and write this given code in policy generator and click on NEXT

Policy editor

```
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": ["s3:GetObject"],
16      "Resource": "arn:aws:s3:::mysourcebktkrati/*"
17    },
18    {
19      "Effect": "Allow",
20      "Action": ["s3:PutObject"],
21      "Resource": "arn:aws:s3:::mydestinationbktkrati/*"
22    }
23  ]
24 ]
```

## 12. Now enter your policy name and create policy

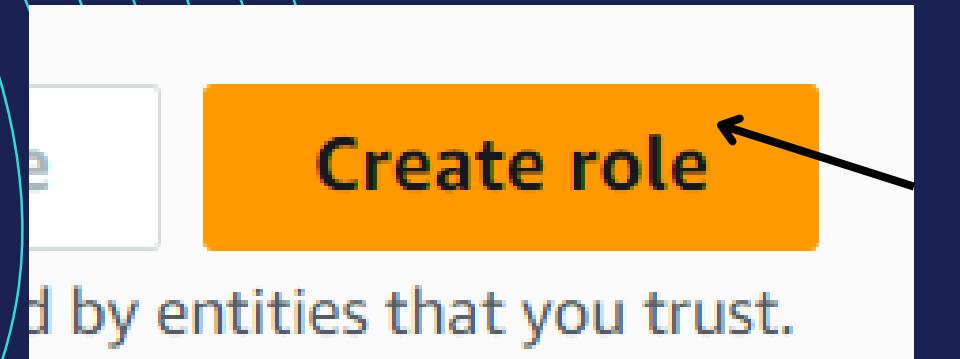
### Policy name

Enter a meaningful name to identify this policy.

mypolicyamod

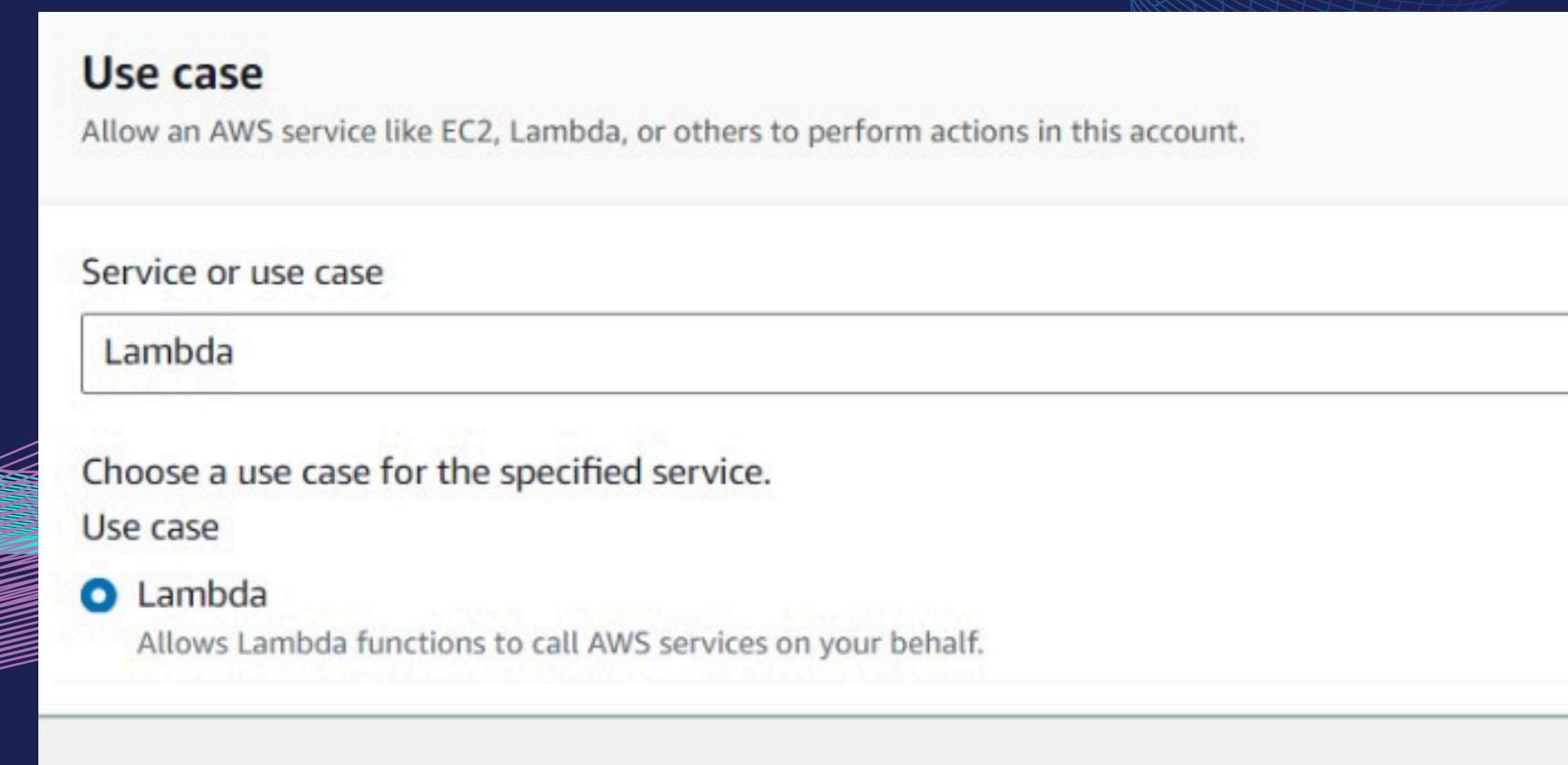
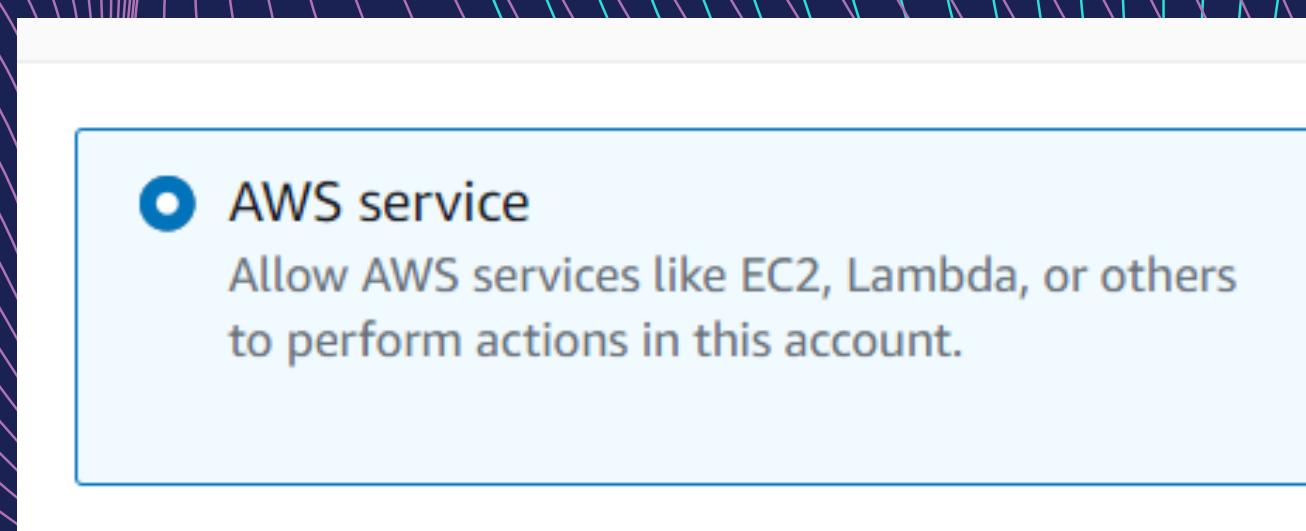
Maximum 128 characters. Use alphanumeric and '+,-,@-\_

## 13. After this go to Roles click on “create roles”



## 14. Now select AWS services

## 15. In use case option select Lambda and click on Next



16. Now go to policy and select your created policy then click on Next

Policy name	Type
<input type="checkbox"/> mypolicy	Customer managed
<input checked="" type="checkbox"/> mypolicyamod	Customer managed
<input type="checkbox"/> mypolicyimage	Customer managed

17. Enter your role name and create your role

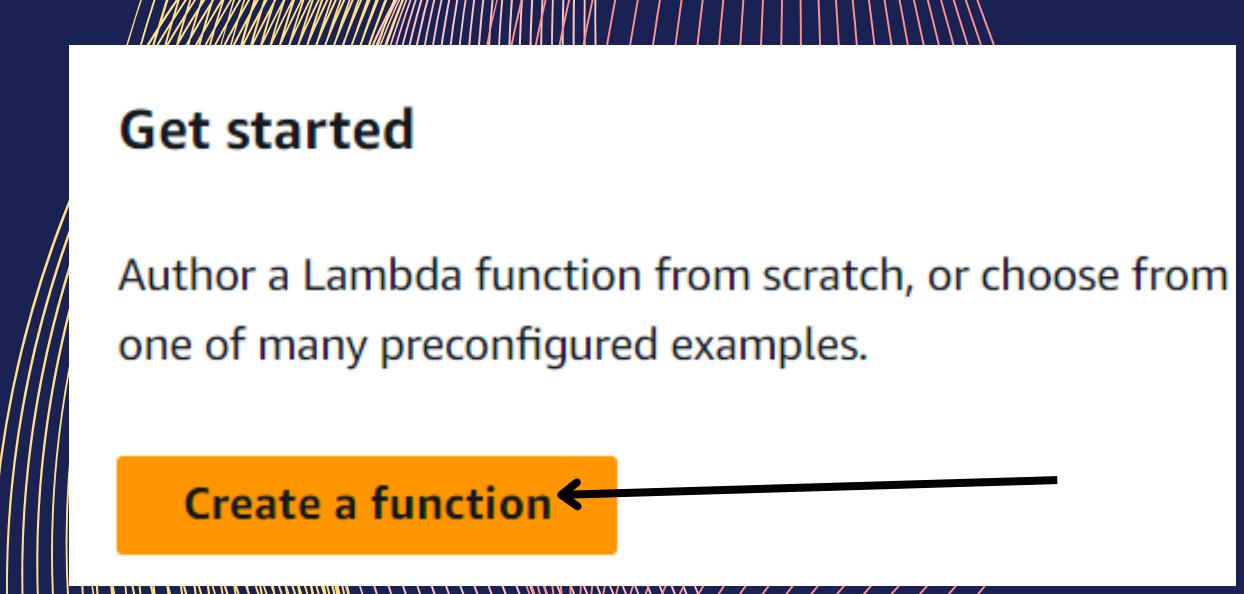
Role name  
Enter a meaningful name to identify this role.

Maximum 64 characters. Use alphanumeric and '+

## STEP 5 : CREATE A LAMBDA FUNCTION

18. Now go to services and open “lambda” and click on “create a function

19. Select author from scratch option and name your function



**Function name**

Enter a name that describes the purpose of your function.

mylambdafunctionAMOD

Use only letters, numbers, hyphens, or underscores with no spaces.

**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.

20. Now in Runtime option select "node.js 18.x"

**Runtime** Info

Choose the language to use to write your function. Note that the console code editor supports only Node.

Node.js 18.x

21. In change default execution role select “use an existing role” and select your created role

22. Now click on create function

## ▼ 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

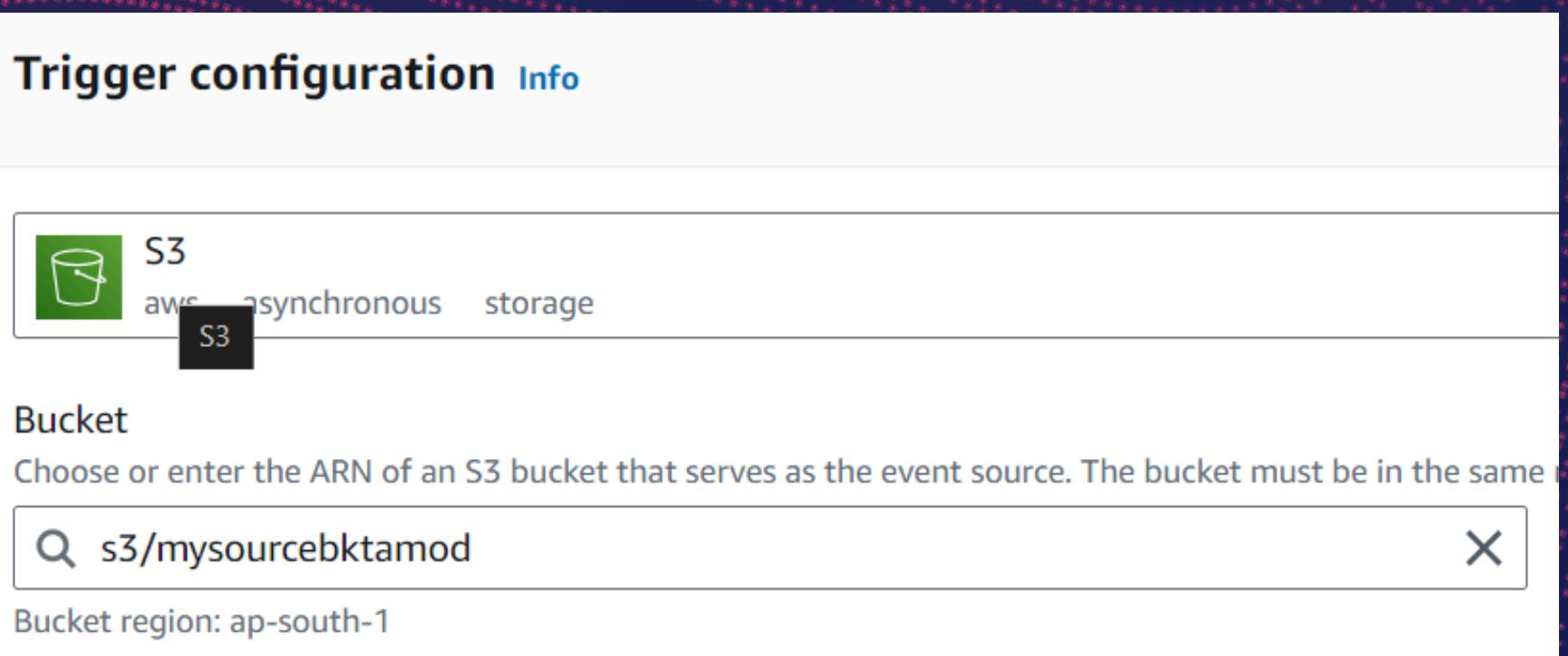
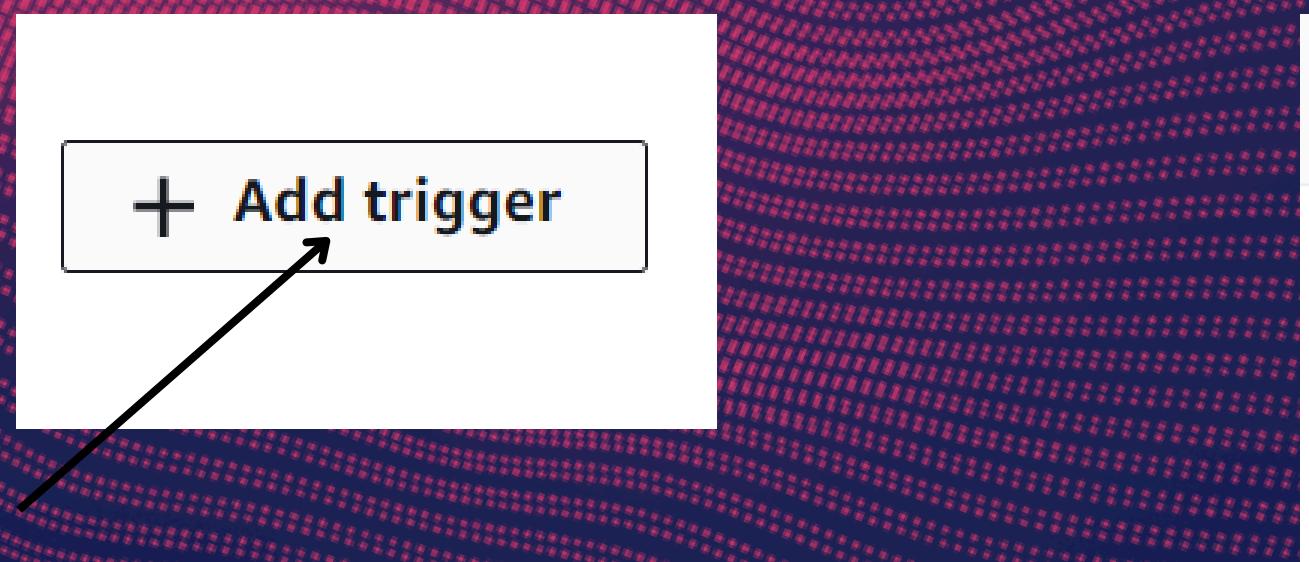
Choose an existing role that you've created to be used with this Lambda function. The role must have permission to access CloudWatch Logs.

myroleamod

## STEP 6 : ADDING TRIGGER AND CREATING ENVIRONMENT VARIABLES

23. Now click on add trigger and select S3

24. Enter your source bucket name and click on add



25. Now go to configuration and select environment variables

**Environment variables**

Find environment variables

Key	Value
No environment variables	
No environment variables associated with this function.	
<a href="#">Edit</a>	

26. Select edit and enter key as "DEST\_BUCKET" and value as ": mytargetbktamod"(use your unique bkt  
27. Click on Save

**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	Remove
DEST_BUCKET	mytargetbktamod	<a href="#">Remove</a>

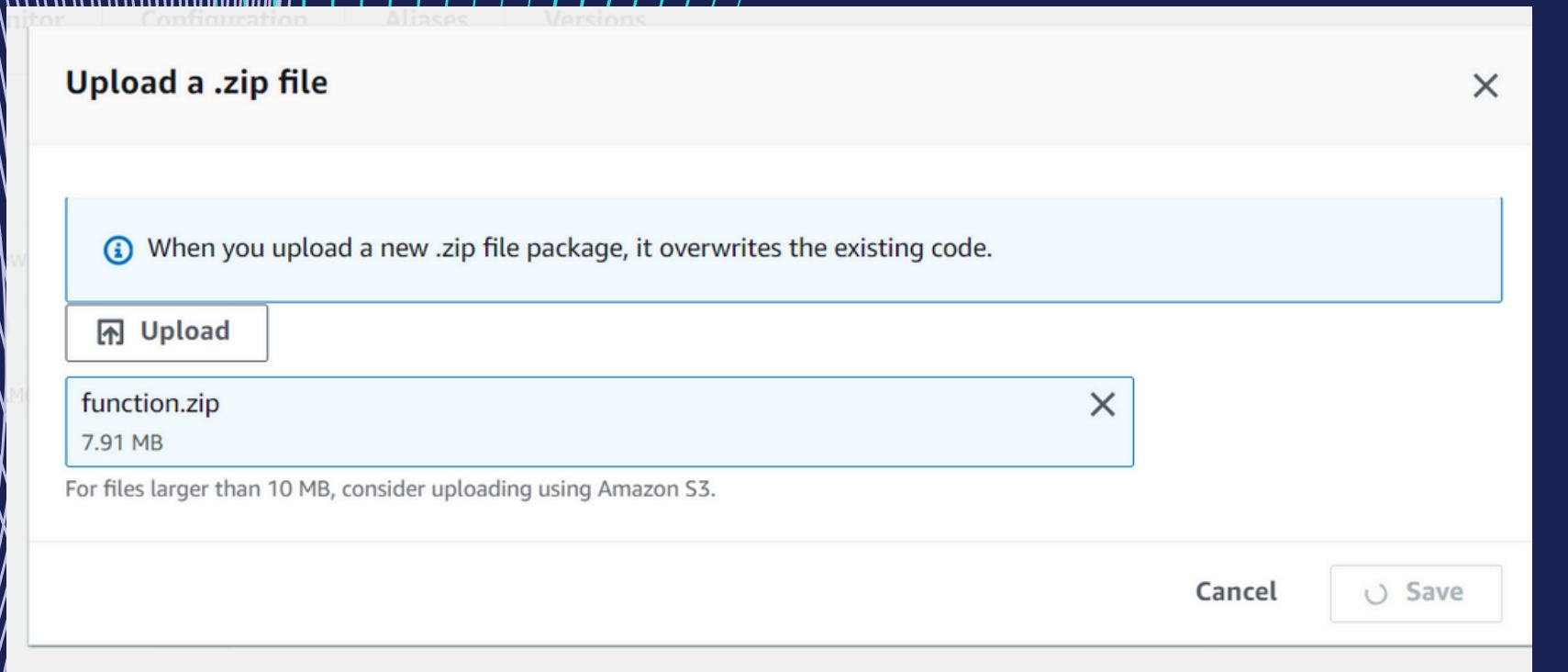
[Add environment variable](#)

[Encryption configuration](#)

[Cancel](#) [Save](#)

## STEP 7 : UPLOADING ZIP FILE AND RUNNING TEST

28. Now go to Code , and under code select upload and upload your ZIP file

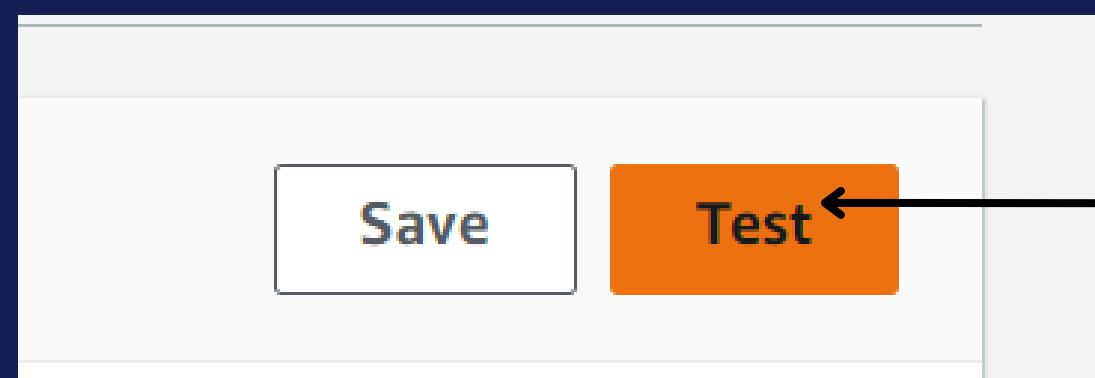


29. Then open “Test” and select s3-put in Template

30. Then enter name of your source bucket and name of your image manually at their respective columns

## Event JSON

```
1 {  
2   "Records": [  
3     {  
4       "eventVersion": "2.0",  
5       "eventSource": "aws:s3",  
6       "awsRegion": "us-east-1",  
7       "eventTime": "1970-01-01T00:00:00.000Z",  
8       "eventName": "ObjectCreated:Put",  
9       "userIdentity": {  
10         "principalId": "EXAMPLE"  
11       },  
12       "requestParameters": {  
13         "sourceIPAddress": "127.0.0.1"  
14       },  
15       "responseElements": {  
16         "x-amz-request-id": "EXAMPLE123456789",  
17         "x-amz-id-2": "EXAMPLE123/5678abcdefghijklmabaisawesome/mnopqrstuvwxyzABCDEFGH"  
18       },  
19       "s3": {  
20         "s3SchemaVersion": "1.0",  
21         "configurationId": "testConfigRule",  
22         "bucket": {  
23           "name": "mysourcebktamod",  
24           "ownerIdentity": {  
25             "principalId": "EXAMPLE"  
26           },  
27           "arn": "arn:aws:s3:::mysourcebktamod"  
28         },  
29         "object": {  
30           "key": "wallpaper.jpg",  
31           "size": 1024,  
32           "eTag": "0123456789abcdef0123456789abcdef",  
33           "sequencer": "0A1B2C3D4E5F678901"  
34         }  
35       }  
36     }  
}
```



31. Click on Test .

32. Now your execution is successfully completed and you can check your resized image in your target bucket



**THANK YOU**

**For Any Queries**

Contact us at :

\*919336453059 

[www.amodaws.com](http://www.amodaws.com) 

[amodpathak51@gmail.com](mailto:amodpathak51@gmail.com) 

E-396 K.D.A colony daheli   
sujanpur kanpur, 208013

SUBMITTED TO Mr .Abhinav Dwivedi