

# PRESENTATION



---

SERVERLESS IMAGE PROCESSING BY USING  
LAMBDA FUNCTION IN S3 BUCKETS

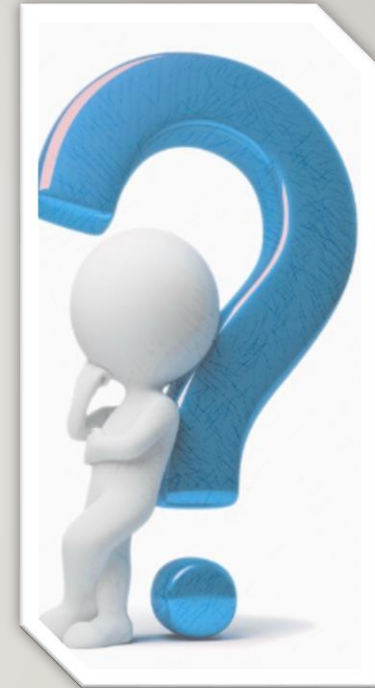
# ACKNOWLEDGEMENT

---

- With the kind guidance of my teacher , Mr ABHINAV DWIVEDI . I would to express my special thanks to my teacher who gave me this golden opportunity to do this wonderful project on serverless processing application .secondly I would also like to thanks to those who helped me in completing my project.

# LIST OF REQUIRED AWS RESOURCES:--

- Create source s3 bucket.
- Create destination s3 bucket.
- Create a bucket to deploy lambda function.
- Policy.
- Role for AWS lambda.



# *ABOUT AWS S3*



AWS (Amazon Web Services) is a comprehensive, evolving cloud computing platform provided by Amazon. It includes a mixture of infrastructure-as-a-service (IaaS), platform-as-a-service (PaaS) and packaged software-as-a-service (SaaS) offerings. AWS offers tools such as compute power, database storage and content delivery services



# AWS LAMBDA

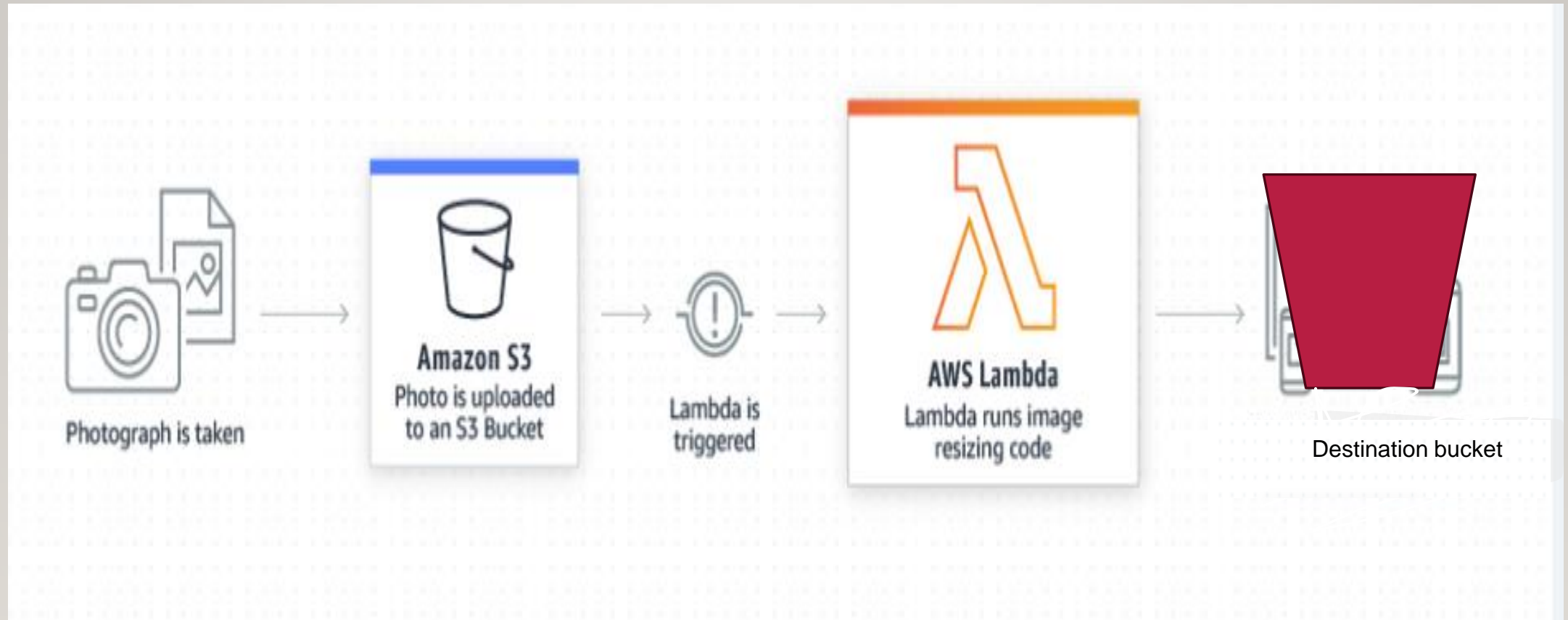


AWS Lambda is a compute service that runs your code in response to events and automatically manages the compute resources, making it the fastest way to turn an idea into a modern, production, serverless applications.

When building serverless application ,AWS lambda is on of the main candidate for running the application code .To complete a serverless stack we need:-

- \*a computing service
- \*a database service
- \*an HTTP gateway service

# Architecture Diagram



**For serverless image processing application that automatically resizes image we have to follow steps given below:-**

**Step 1:** sign in to AWS management console

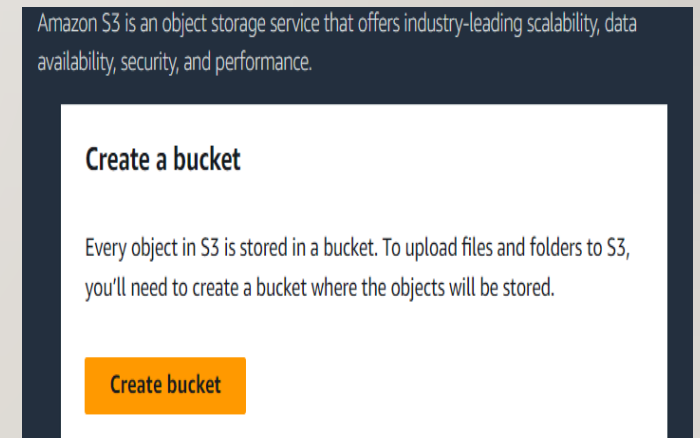
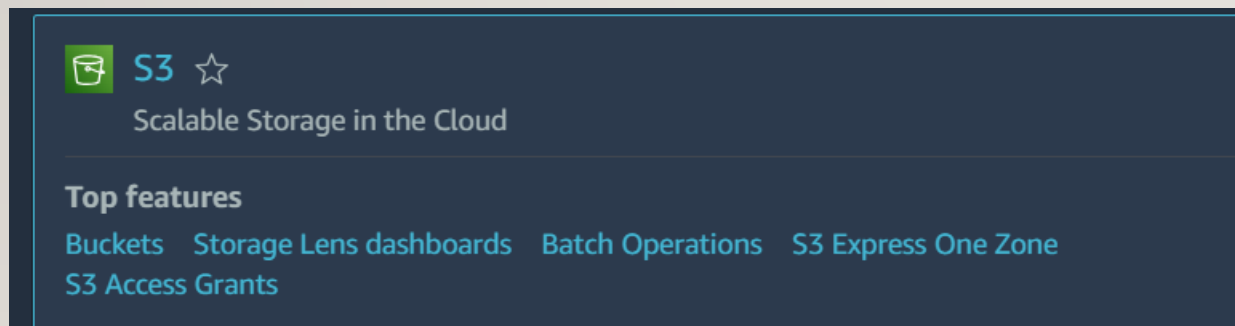
- \*click on open console button

- \*once signed in to aws management console , set region as Mumbai.

**Step 2:**after getting console page in the services menu search s3

- \*here we have to create two buckets source and destination.

- \*Click on create bucket



*\*firstly we create source bucket, enter name of source bucket*

Bucket name

Info

mysourcebucketks

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

## Step 3:-Put ACLs enabled

☐ 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**

**\*Leave other settings as default.**

**\*Then click on create bucket.YOUR SOURCE BUCKET IS CREATED.**



Step 4: Create one more bucket named as destination bucket same as source bucket.  
\*THEN YOUR TWO BUCKETS ARE CREATED.

	Name ▾	AWS Region ▲	IAM Access Analyzer
<input type="radio"/>	<a href="#">mydestinationbucketks</a>	Asia Pacific (Mumbai) ap-south-1	<a href="#">View analyzer for ap-south-1</a>
<input type="radio"/>	<a href="#">mysourcebucketks</a>	Asia Pacific (Mumbai) ap-south-1	<a href="#">View analyzer for ap-south-1</a>

Step 5: Now we upload one image on mysourcebucket.

- \*open your source bucket.
- \*click on upload.
- \*then add file.

Files and folders (1 Total, 937.1 KB)

Remove

Add files

Add folder

All files and folders in this table will be uploaded.

Find by name

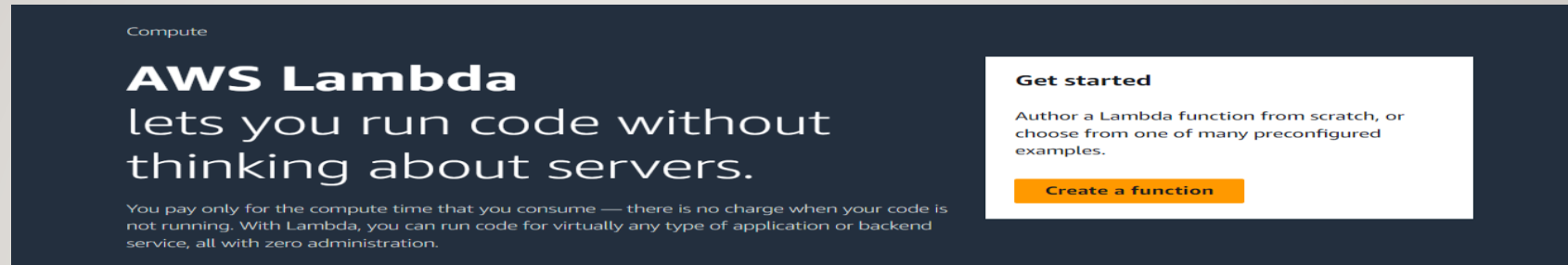
<

1

>

<input type="checkbox"/>	Name ▾	Folder ▾	Type
<input type="checkbox"/>	Screenshot 2024-06-19 102610.png	-	image/png

Step 6: after uploading image , move back to console page and search for LAMBDA function.  
\*Then click on create a function.





Step 7: Give function name and select runtime as (Node.js 18.x).

**Basic information**

**Function name**  
Enter a name that describes the purpose of your function.  
  
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.  





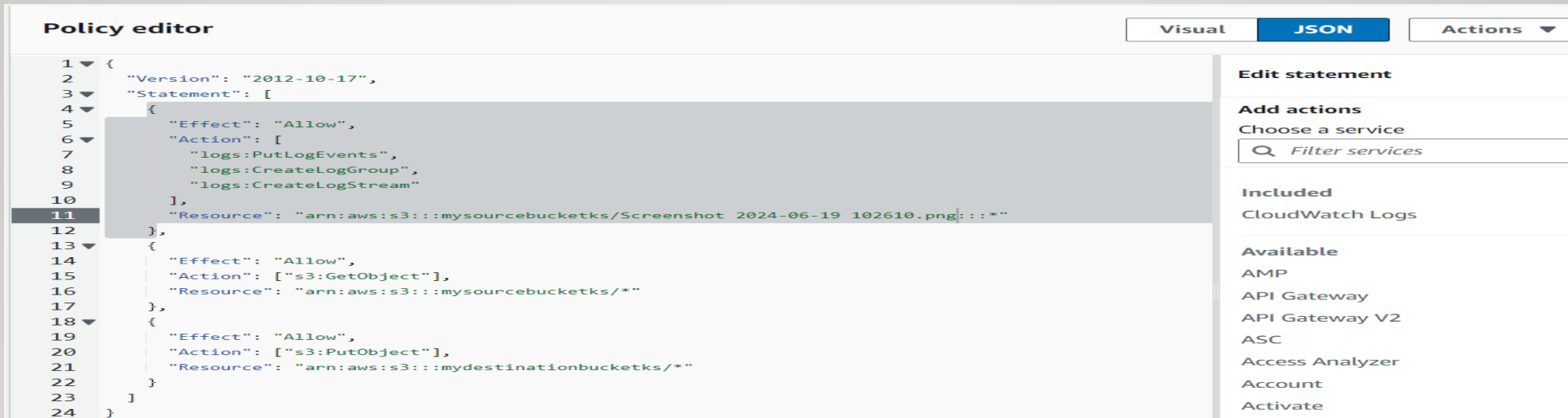
**Architecture** [Info](#)  
Choose the instruction set architecture you want for your function code.  

☒ x86\_64

☐ arm64



Step 8: Search IAM from the services, then click on policies given on left side, select JSON from visual and edit code, change your own ARN of source and destination bucket.

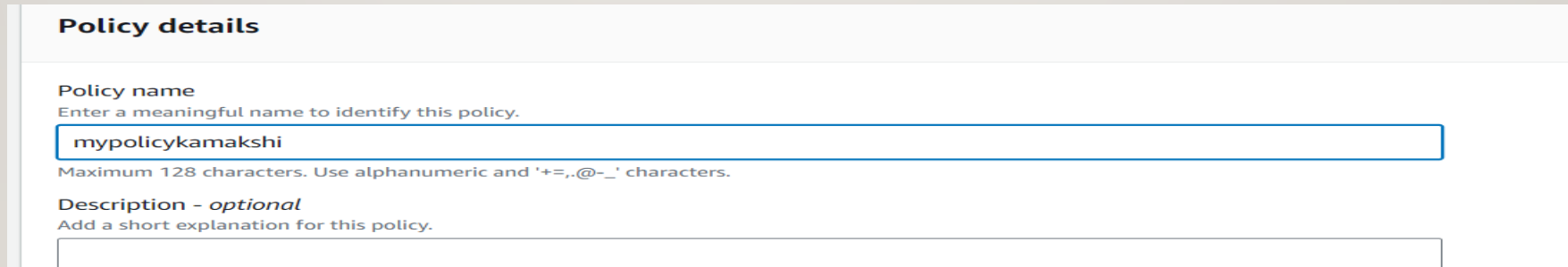


The screenshot shows the AWS IAM Policy Editor interface. At the top, there are three tabs: 'Visual', 'JSON' (which is selected and highlighted in blue), and 'Actions'. The main area displays a JSON policy document with line numbers 1 through 24 on the left. The JSON content is as follows:

```
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:s3::mysourcebucketks/Screenshot_2024-06-19_102610.png::*"
12    },
13    {
14      "Effect": "Allow",
15      "Action": ["s3:GetObject"],
16      "Resource": "arn:aws:s3::mysourcebucketks/*"
17    },
18    {
19      "Effect": "Allow",
20      "Action": ["s3:PutObject"],
21      "Resource": "arn:aws:s3::mydestinationbucketks/*"
22    }
23  ]
24 }
```

On the right side of the editor, there are three sections: 'Edit statement' (with an 'Add actions' button and a 'Choose a service' dropdown), 'Included' (listing 'CloudWatch Logs'), and 'Available' (listing various AWS services like AMP, API Gateway, API Gateway V2, ASC, Access Analyzer, Account, and Activate).

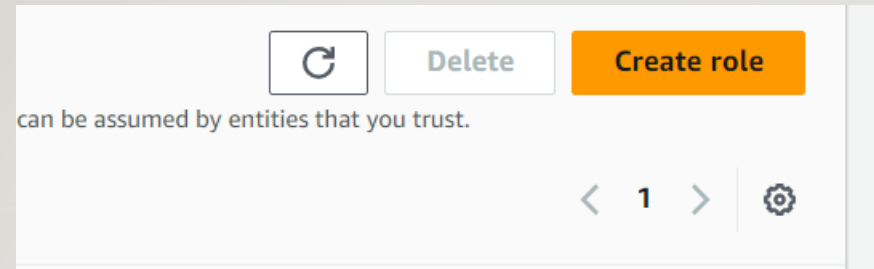
\*click on create policy.



The screenshot shows the 'Policy details' form in the AWS IAM console. It has a title 'Policy details' and a subtitle 'Policy name'. Below the subtitle is a text input field containing 'mypolicykamakshi'. A note below the field states: 'Maximum 128 characters. Use alphanumeric and '+=,.,@-\_' characters.' Below this is a section for 'Description - optional' with a text input field.

Step 9: After creating policy . Go to role on left of the page

\* Click on create role



▼ Access management

User groups

Users

**Roles**

Policies

Identity providers

\*Now choose an AWS services as trusted entity type.

### Select trusted entity Info

**Trusted entity type**

<input checked="" type="radio"/> <b>AWS service</b> Allow AWS services like EC2, Lambda, or others to perform actions in this account.	<input type="radio"/> <b>AWS account</b> Allow entities in other AWS accounts belonging to you or a 3rd party to perform actions in this account.	<input type="radio"/> <b>Web identity</b> Allows users federated by the specified external web identity provider to assume this role to perform actions in this account.
<input type="radio"/> <b>SAML 2.0 federation</b> Allow users federated with SAML 2.0 from a corporate directory to perform actions in this account.	<input type="radio"/> <b>Custom trust policy</b> Create a custom trust policy to enable others to perform actions in this account.	



\*Select Lambda in use case

**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

Click on next.

Filter policies , now you can see a list of policies . Here you have to select your policy .

Search

mypolicy

X

All types

1 match

Policy name

Type

Description

+

[mypolicykamakshi](#)

Customer managed

-

Step 10: Now go back to Lambda and choose the column of use an existing role and select your role which is 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

\*Click on create Lambda function.

\*Then click on add trigger.  
Choose S3 function here..

+ Add trigger


 **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

× ↺

Bucket region: ap-south-1

### 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.

\*Click on next.

Step 11: Then move to Environment Variables on left side , then click on edit.

General configuration

Triggers

Permissions

Destinations

Function URL

**Environment variables**

Tags

VPC

**Environment variables**

Key	Value
No environment variables	
No environment variables associated with this function	
<div>Edit</div>	

\*Give name dest\_bkt in key , and mydestbucketkamakshi in value column.

### 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	
<input type="text" value="dest_bkt"/>	<input type="text" value="mydestbucketkamakshi"/>	<button>Remove</button>

Add environment variable

► Encryption configuration

Cancel

Save

Click on save.

Step 12: Then go to the code interface and upload zip file.

### Upload a .zip file

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

Upload

function.zip

7.91 MB

✕

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

Cancel

Save



\*After uploading zip file go to the test interface and select S3 Put in template column.

Template - <i>optional</i>
s3-put

\*After this we get a code , and in this we have to change the source bucket name and the image name .

\*Test the code.

After testing the code , the compressed image is shown in destination bucket.

**Thank You**



**Presented by**  
**Kamakshi Srivastava**