

TITLE: Image Resize SAAS Lambda Project.

Use Case: when filling forms many times the user needs photos of certain width and height, for that use case this project can be used.

Working: when user uploads image it gets stored in input S3 bucket then AWS lambda function triggers and convert image to appropriate width and size and output stored into another S3 bucket and can be download as required.

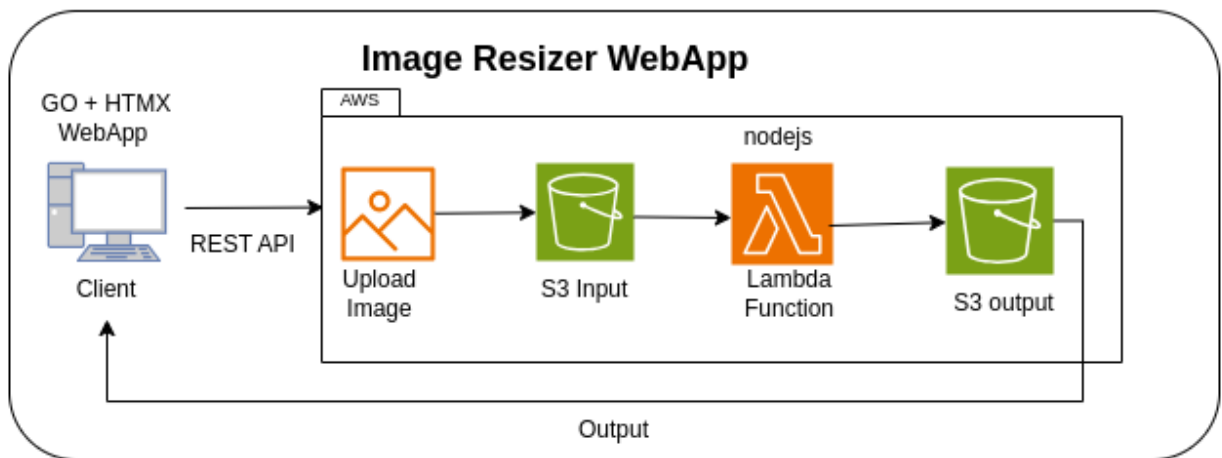
Beneficiaries: Students, Working Professionals etc.

Technology Used:

Client Side: Go lang , Gin framework, HTMX.

Backend : AWS S3 , AWS Lambda, AWS IAM, nodejs, sharp library.

Architecture Diagram:



In the above diagram client send the image to the input S3 Bucket, then the lambda function is evoked and then it processes the image and then saves it to another bucket where the user can take and utilize the resized image.

Steps to perform : 1. Create a 2 s3 bucket with the appropriate names as shown in the images, I have given *image-input-bucket22* and *image-output-bucket22*.

Input Bucket:

The screenshot shows the AWS S3 console interface. At the top, a green banner indicates "Successfully created bucket 'image-input-bucket22'" with a "View details" button. Below this, the breadcrumb "Amazon S3 > Buckets" is visible. The "Account snapshot" section is present, along with a "View Storage Lens dashboard" button. The "General purpose buckets" tab is selected, showing a list of buckets. The table contains one entry: "image-input-bucket22", created on March 17, 2024, at 22:28:49 (UTC+05:30). The table headers are Name, AWS Region, Access, and Creation date.

Name	AWS Region	Access	Creation date
image-input-bucket22			March 17, 2024, 22:28:49 (UTC+05:30)

Output S3 Bucket:

The screenshot shows the AWS S3 console interface. At the top, a green banner indicates "Successfully created bucket 'image-output-bucket22'" with a "View details" button. Below this, the breadcrumb "Amazon S3 > Buckets" is visible. The "Account snapshot" section is present, along with a "View Storage Lens dashboard" button. The "General purpose buckets" tab is selected, showing a list of buckets. The table contains two entries: "image-input-bucket22" and "image-output-bucket22", both created on March 17, 2024, at 22:28:49 and 22:30:09 (UTC+05:30) respectively. The table headers are Name, AWS Region, Access, and Creation date.

Name	AWS Region	Access	Creation date
image-input-bucket22	US West (Oregon) us-west-2	Objects can be public	March 17, 2024, 22:28:49 (UTC+05:30)
image-output-bucket22	US West (Oregon) us-west-2	Objects can be public	March 17, 2024, 22:30:09 (UTC+05:30)

Upload image into input Bucket for Testing

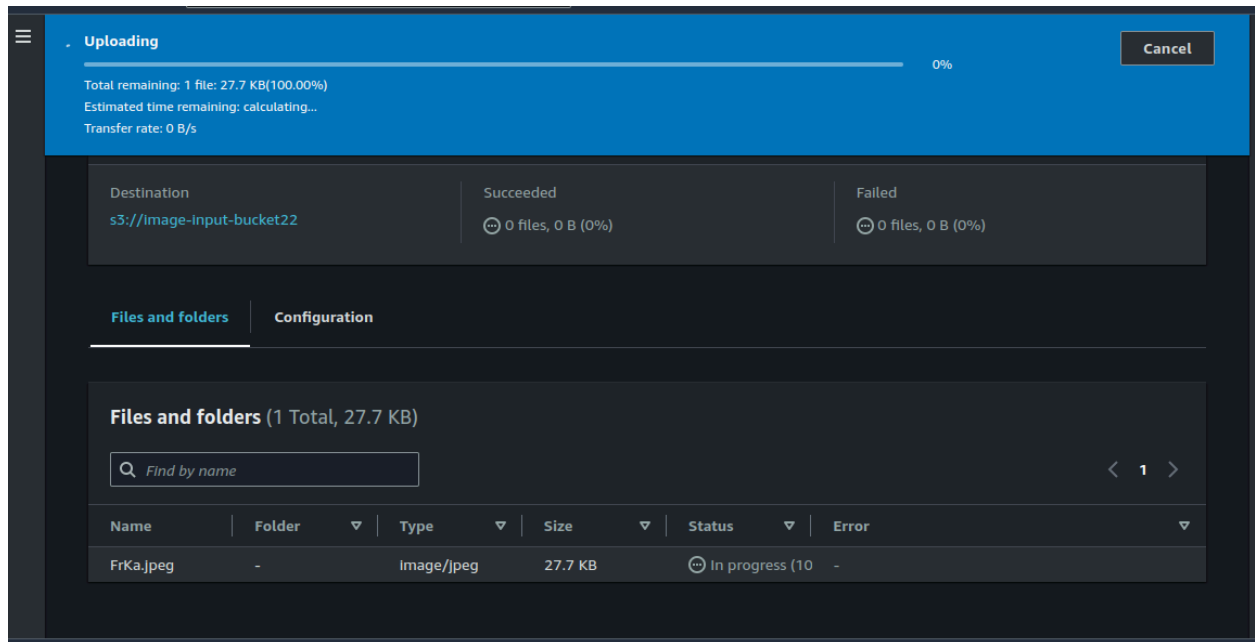
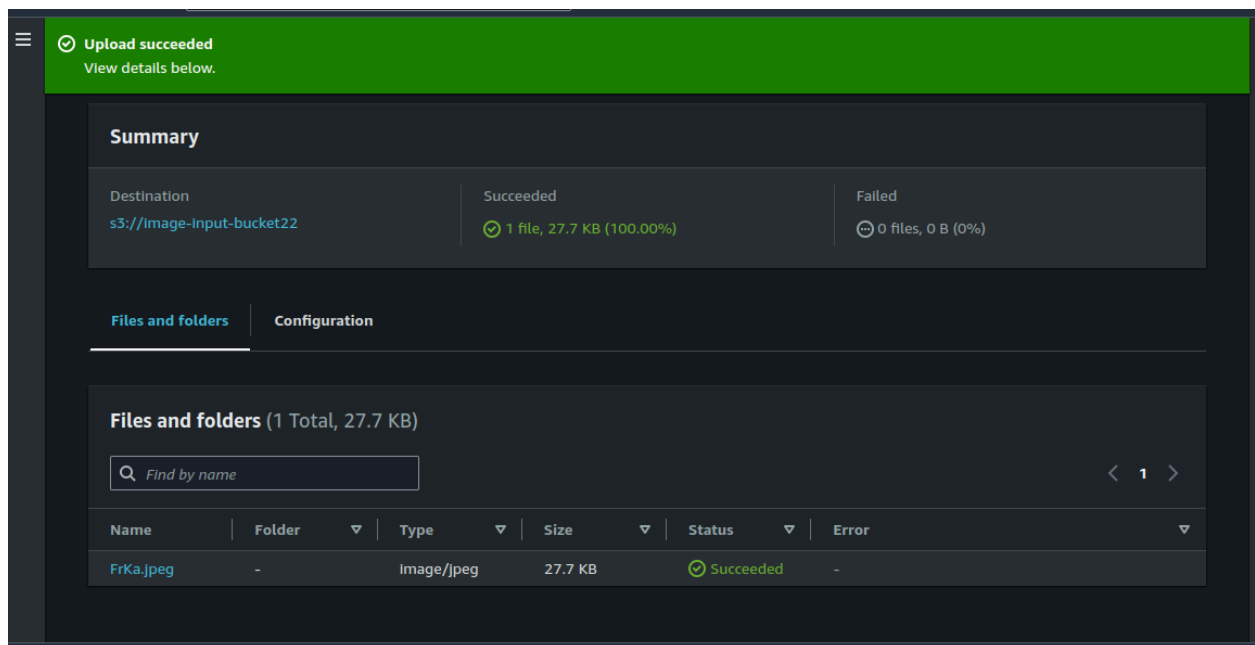
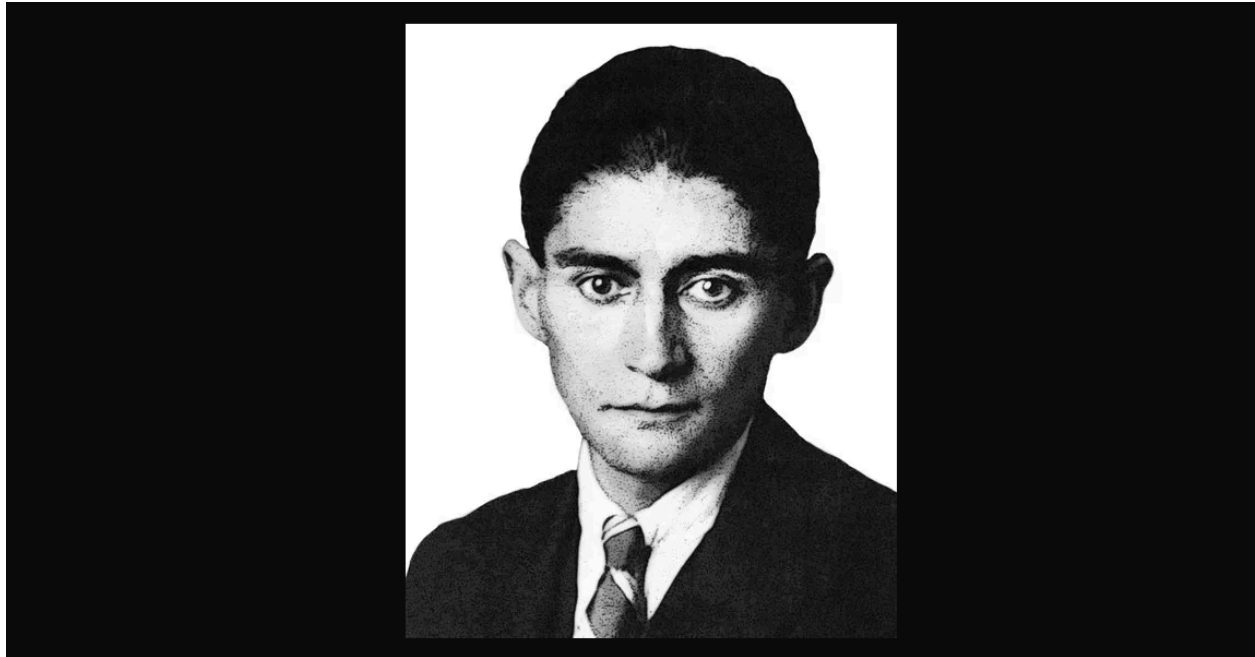


Image upload successfully:



Open original image to see the dimensions of the image:



Create a AWS Lambda Function

The screenshot shows the AWS Lambda console interface. At the top, a green notification bar states: "Successfully created the function image-resize-lambda-project. You can now change its code and configuration. To invoke your function with a test event, choose 'Test'." Below this, the breadcrumb navigation shows "Lambda > Functions > image-resize-lambda-project". The main heading is "image-resize-lambda-project" with buttons for "Throttle", "Copy ARN", and "Actions". The "Function overview" section is active, showing a "Diagram" tab with a visual representation of the function and its layers. To the right, a "Template" tab is also visible. Further right, a "Description" panel lists details: "Last modified 8 seconds ago", "Function ARN arn:aws:lambda:us-west-2:21995232202:function:image-resize-lambda-project", and "Function URL Info". On the far right, a "Tutorials" sidebar offers a "Create a simple web app" tutorial, which includes steps like "Build a simple web app, consisting of a Lambda function with a function URL that outputs a webpage" and "Invoke your function through its function URL". A "Start tutorial" button is provided at the bottom of the sidebar.

Note while creating Lambda Function it requires permission to access S3 bucket, you can use IAM to create a new S3 Policy.

Sample S3 policy:

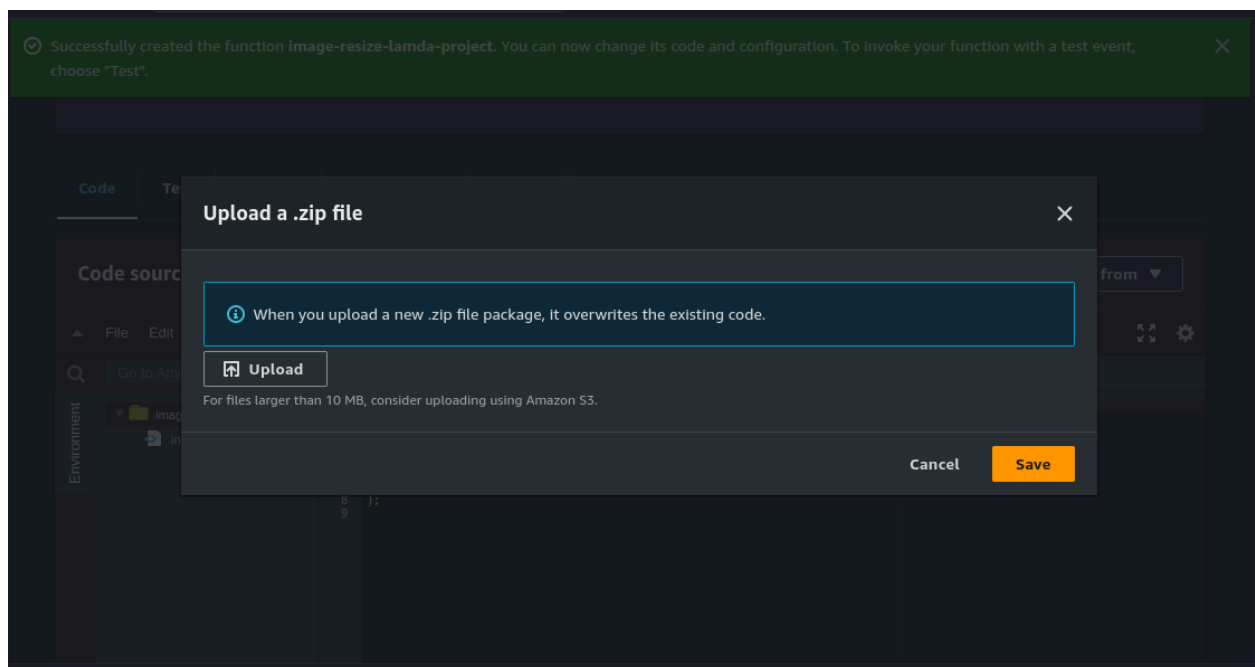
```
{  
  "Version": "1.0.0",
```

```

"Statement": [
  {
    "Effect": "Allow",
    "Action": [
      "logs:PutLogEvents",
      "logs:CreateLogGroup",
      "logs:CreateLogStream"
    ],
    "Resource": "arn:aws:logs:*:*:*"
  },
  {
    "Effect": "Allow",
    "Action": ["s3:GetObject"],
    "Resource": "arn:aws:s3:::image-input-bucket22/*"
  },
  {
    "Effect": "Allow",
    "Action": ["s3:PutObject"],
    "Resource": "arn:aws:s3:::image-output-bucket22/*"
  }
]
}

```

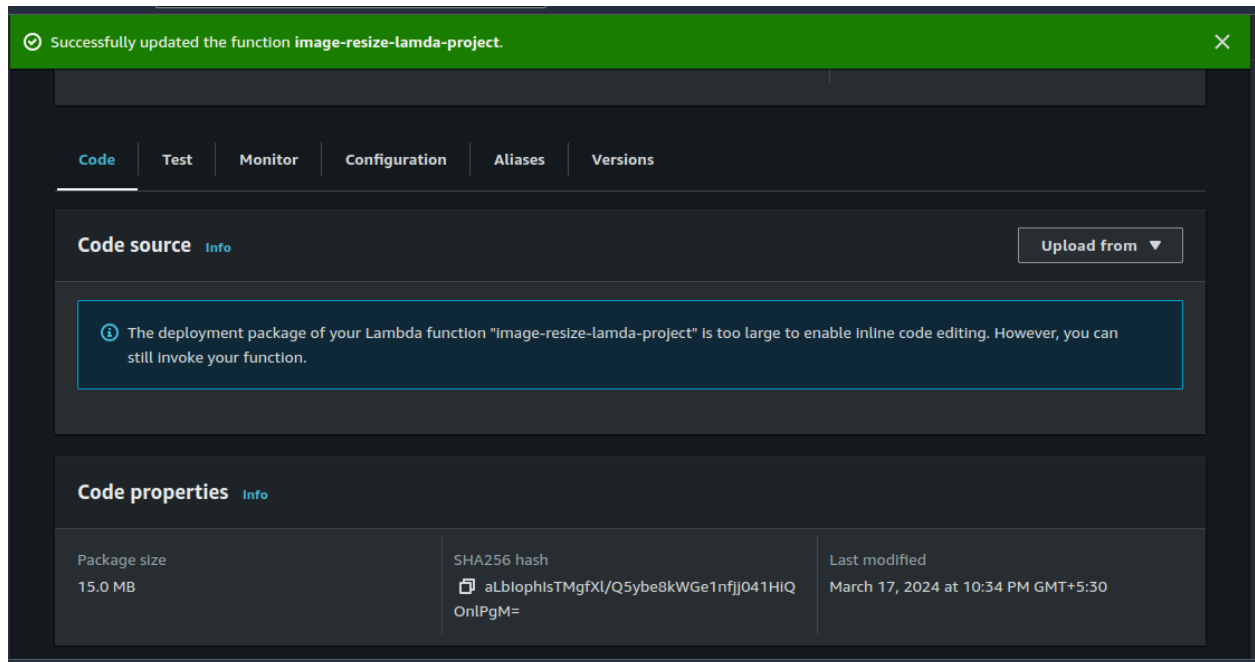
Upload a zip package to AWS Lambda code section.



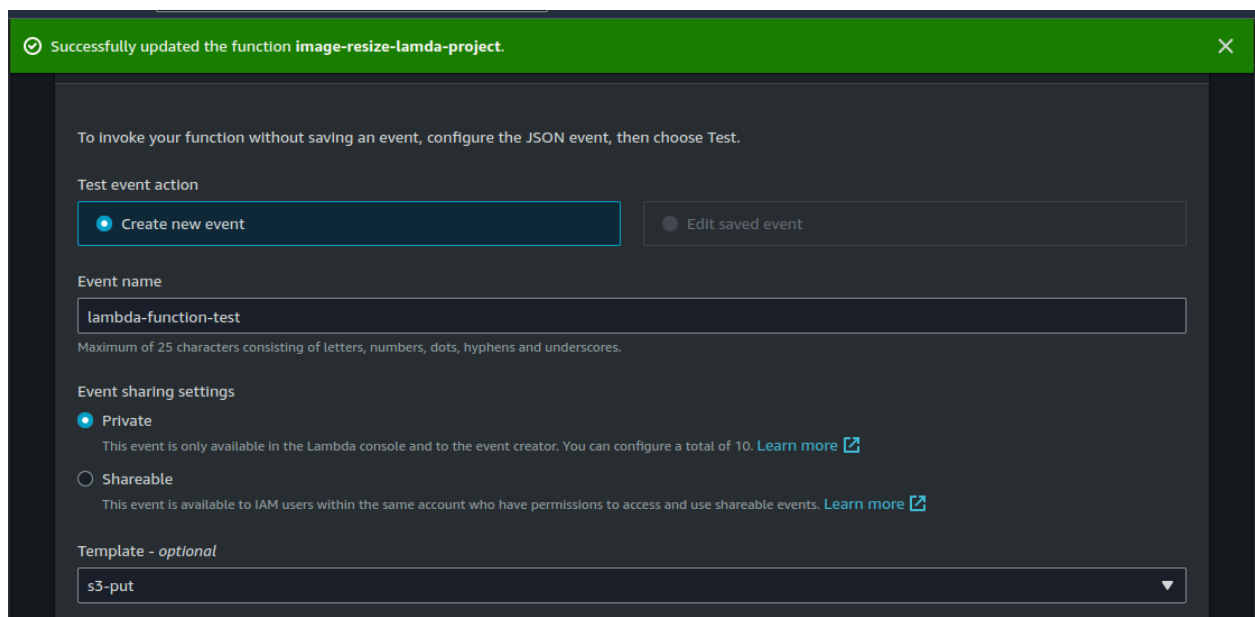
Code done uploading now setup environment variables for the project.

To compile package for production: `npm install --arch=x64 --platform=linux --target=16x sharp`

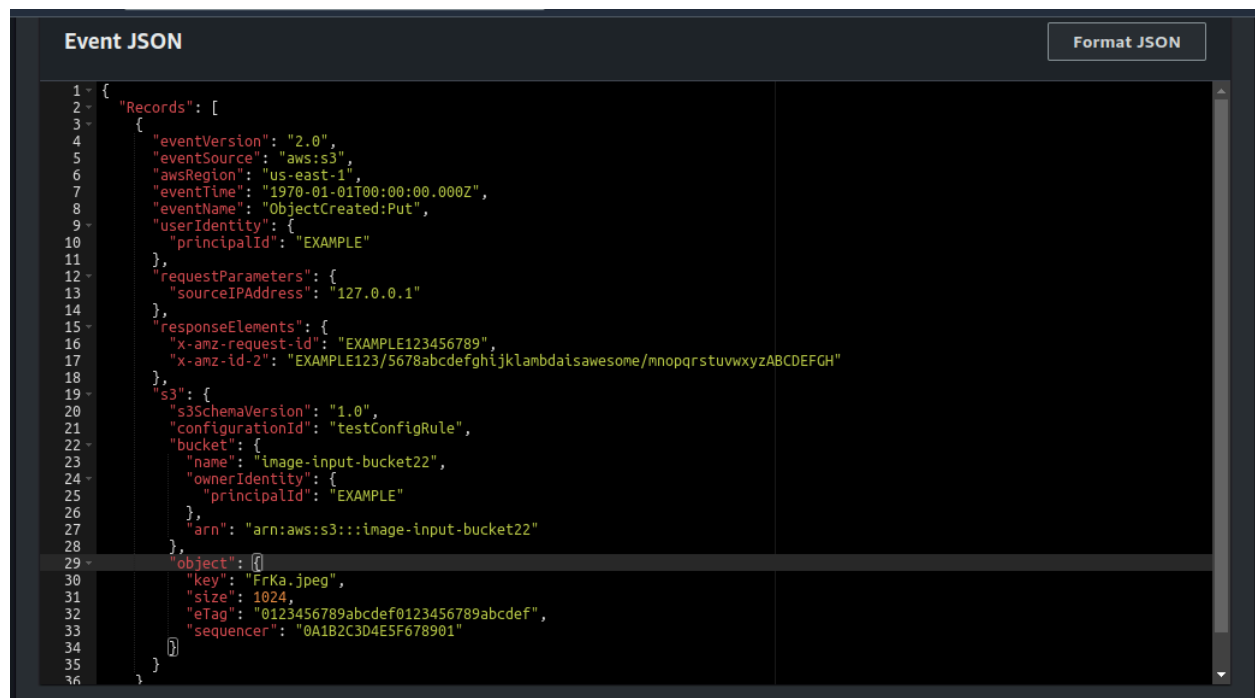
To create a package for upload : `npm run package`



Generate a Test case to Test the newly created Lambda Function.



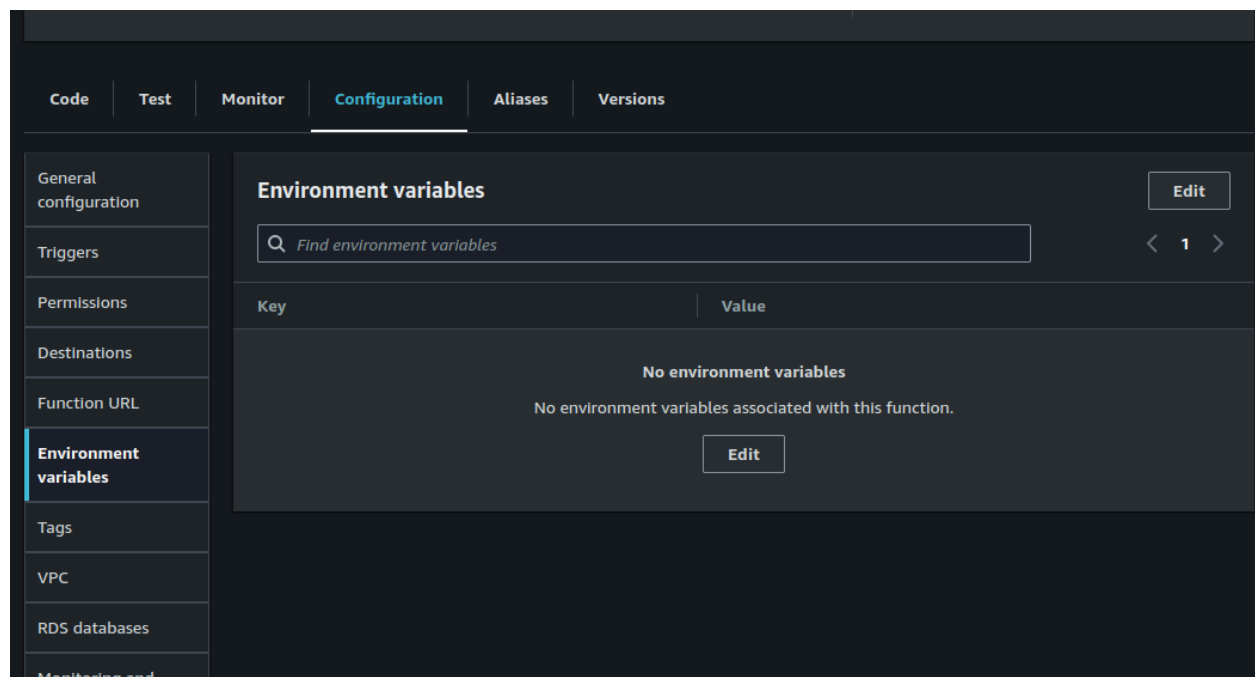
Create an Event JSON to check the Function: create a S3 put Event Template. And change the input AWS S3 key,ARN and input parameters.



The screenshot shows the 'Event JSON' editor in the AWS Lambda console. The JSON is a template for an S3 Put event. It includes fields for eventVersion, eventSource, awsRegion, eventTime, eventName, userIdentity, requestParameters, responseElements, s3, and object. The s3 section contains s3SchemaVersion, configurationId, bucket, and arn. The object section contains key, size, eTag, and sequencer. A 'Format JSON' button is visible in the top right corner.

```
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": "EXAMPLE123/5678abcdefghijklmbdaisawsome/mnopqrstuvwxyzABCDEFGH"
18      },
19      "s3": {
20        "s3SchemaVersion": "1.0",
21        "configurationId": "testConfigRule",
22        "bucket": {
23          "name": "image-input-bucket22",
24          "ownerIdentity": {
25            "principalId": "EXAMPLE"
26          },
27          "arn": "arn:aws:s3:::image-input-bucket22"
28        },
29        "object": {
30          "key": "FrKa.jpeg",
31          "size": 1024,
32          "eTag": "0123456789abcdef0123456789abcdef",
33          "sequencer": "0A1B2C3D4E5F678901"
34        }
35      }
36    ]
37  }
```

Create Env variables to store the image to desired s3 bucket.
Edit > enter desired parameter and values



Key: DEST_BUCKET , **Value:**image-output-bucket22

Lambda > Functions > image-resize-lambda-project > Edit environment variables

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_BUCKET	image-output-bucket22	Remove

[Add environment variable](#)

► Encryption configuration

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

Run the rest: you will get Http status code 200 for your Test API

Code | **Test** | Monitor | Configuration | Aliases | Versions

✓ Executing function: succeeded ([logs](#))

▼ Details

The area below shows the last 4 KB of the execution log.

```
{
  "statusCode": 200,
  "body": "Successfully resized image-input-bucket22/FrKa.jpeg and uploaded to image-output-bucket22/FrKa.jpeg"
}
```

Summary


Code SHA-256	Execution time
aLbIophisTMgfXl/Q5ybe8kWGe1nfjJ041HIQOnIPgM=	13 seconds ago (March 17, 2024 at 10:43 PM GMT+5:30)
Request ID	Function version
3a8d7bf7-4eff-495d-b33e-affaf4772137	\$LATEST
Init duration	Duration
604.06 ms	1337.37 ms
Billed duration	Resources configured
1338 ms	128 MB


Check Your Output S3 bucket you can see your resized image in your bucket, now add the trigger to invoke your Lambda.

[Lambda](#) > [Add trigger](#)


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.
 
Bucket region: us-west-2

Event types
Select the events that you want to have trigger the Lambda function. You can optionally set up a prefix or suffix for an event. However, for each bucket, individual events cannot have multiple configurations with overlapping prefixes or suffixes that could match the same object key.

All object create events 



Prefix - optional

Now whenever you drop image to your input S3 bucket ,it will automatically convert and resize the image and you can see it in output S3 Bucket.

[Lambda](#) > [Functions](#) > [image-resize-lambda-project](#)



image-resize-lambda-project


[Throttle](#) [Copy ARN](#) [Actions](#)

 The trigger Image-Input-bucket22 was successfully added to function Image-resize-lambda-project. The function is now receiving events from the trigger. 


Function overview [Info](#) [Export to Application Composer](#) [Download](#)

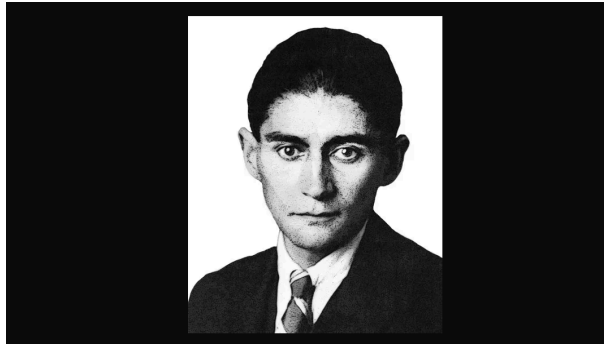
Diagram **Template**

 **image-resize-lambda-project**
 Layers (0)

 **S3**
[+ Add trigger](#)

[+ Add destination](#)

Description
-
Last modified
4 minutes ago
Function ARN
 arn:aws:lambda:us-west-2:221995232202:function:image-resize-lambda-project
Function URL [Info](#)



A. Before Image resize



B. After Image Resize

GoLang + HTMX Client

Image Resize With Lambda

Choose Image

Upload