



AWS Lambda



SERVERLESS IMAGE PROCESSING PROJECT

Content

- | Login to AWS Console
- | Create s3 Bucket
- | Create Lambda Function
- | Create Policy and Role
- | For Executing Function correctly test JSON code
- | Check The Result

Create Bucket

Source Bucket and Destination Bucket

Source Bucket for uploading the images and files, and the destination bucket for their results

The screenshot shows the AWS Create Bucket wizard. On the left, the 'General configuration' tab is active, displaying the AWS Region as 'Asia Pacific (Mumbai) ap-south-1' and the Bucket name as 'my-first-project-source-2'. It also includes sections for 'Copy settings from existing bucket - optional' and 'Object Ownership'. The 'Object Ownership' section shows two options: 'ACLs disabled (recommended)' and 'ACLs enabled' (which is selected). On the right, the 'Block all public access' section is shown, containing four checkboxes. Below it is a warning message: '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.' A checkbox at the bottom of this message is checked, stating 'I acknowledge that the current settings might result in this bucket and the objects within becoming public.'

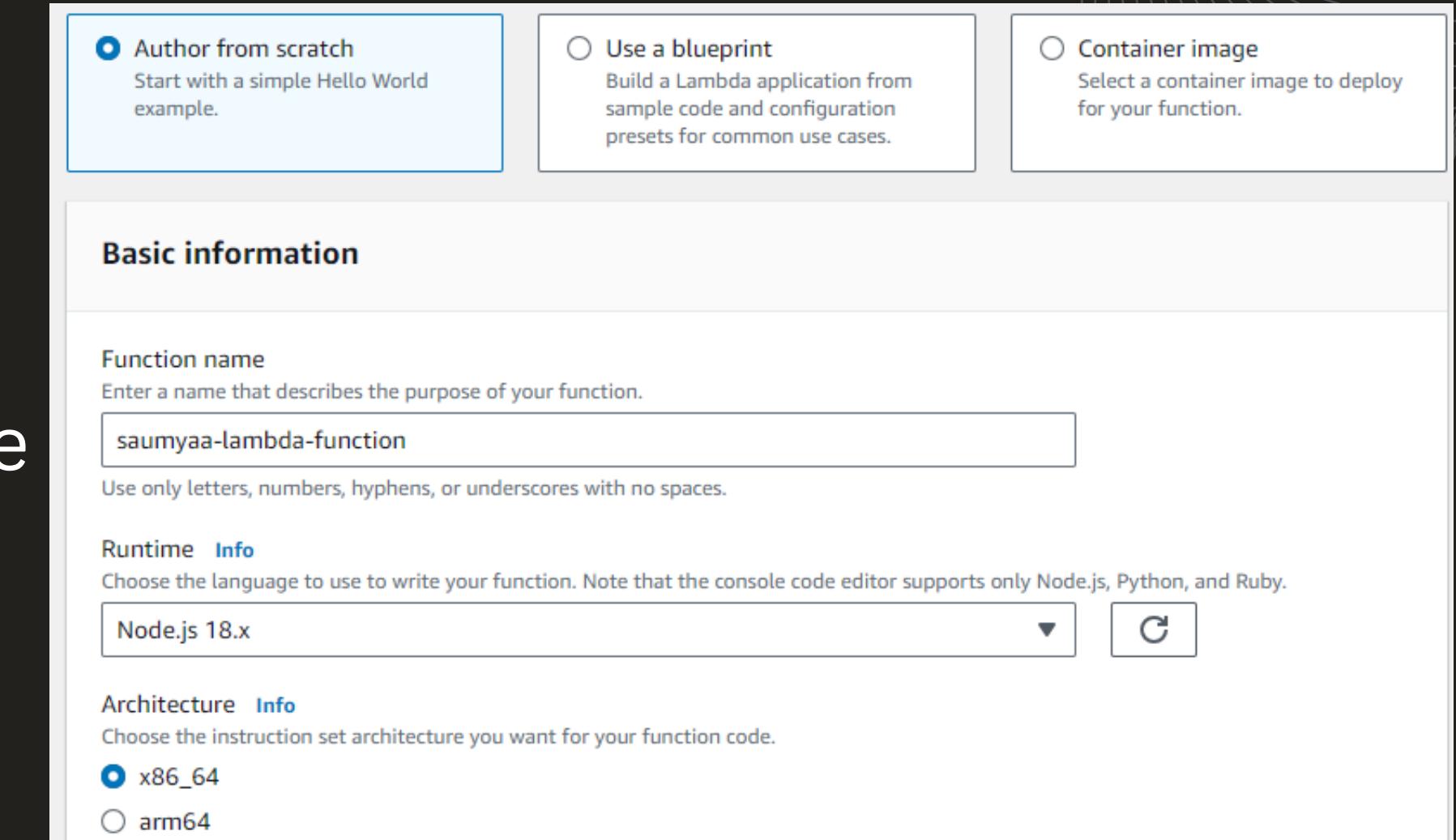
[my-first-project-source-2](#)
[my-second-bucket-project-2](#)

Both Bucket Name

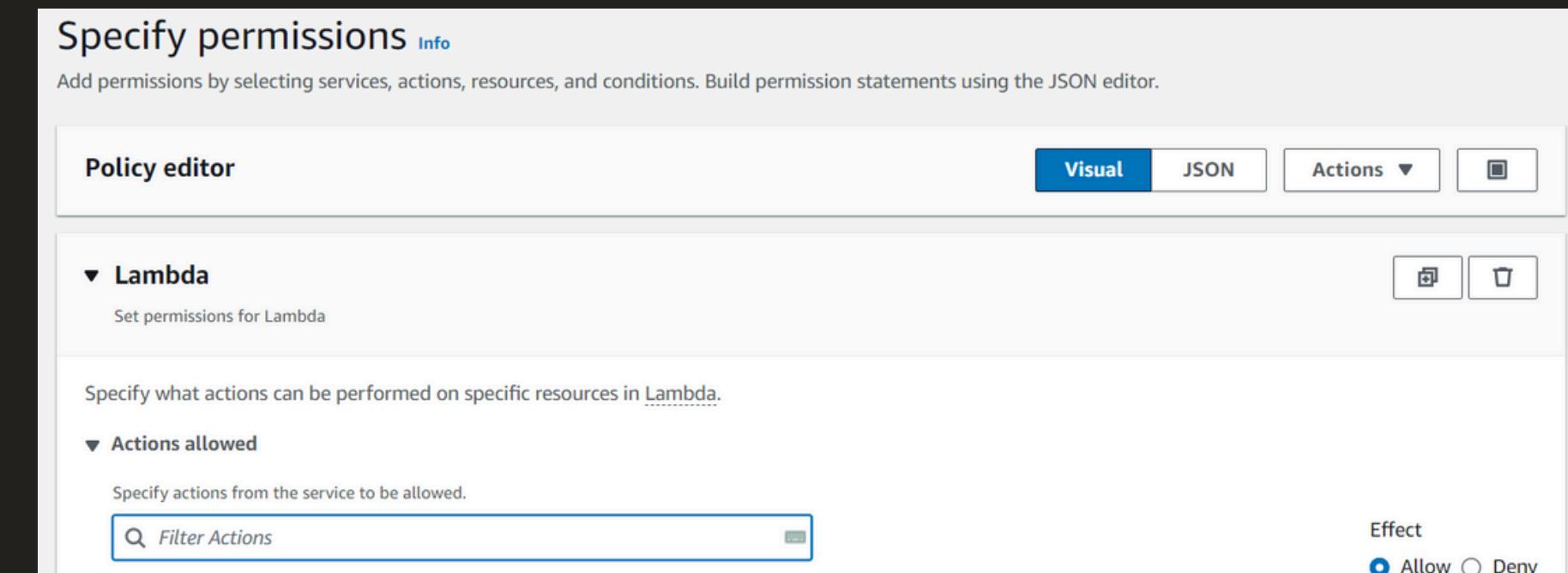
Create Lambda Function

- Create a Function
- Select Node.js 18.x
- Click on Change default Execution Role
- Choose Use an Existing Role
- Go to the IAM console
- Go to the policy tab
- Create Policy

Create function



Create policy



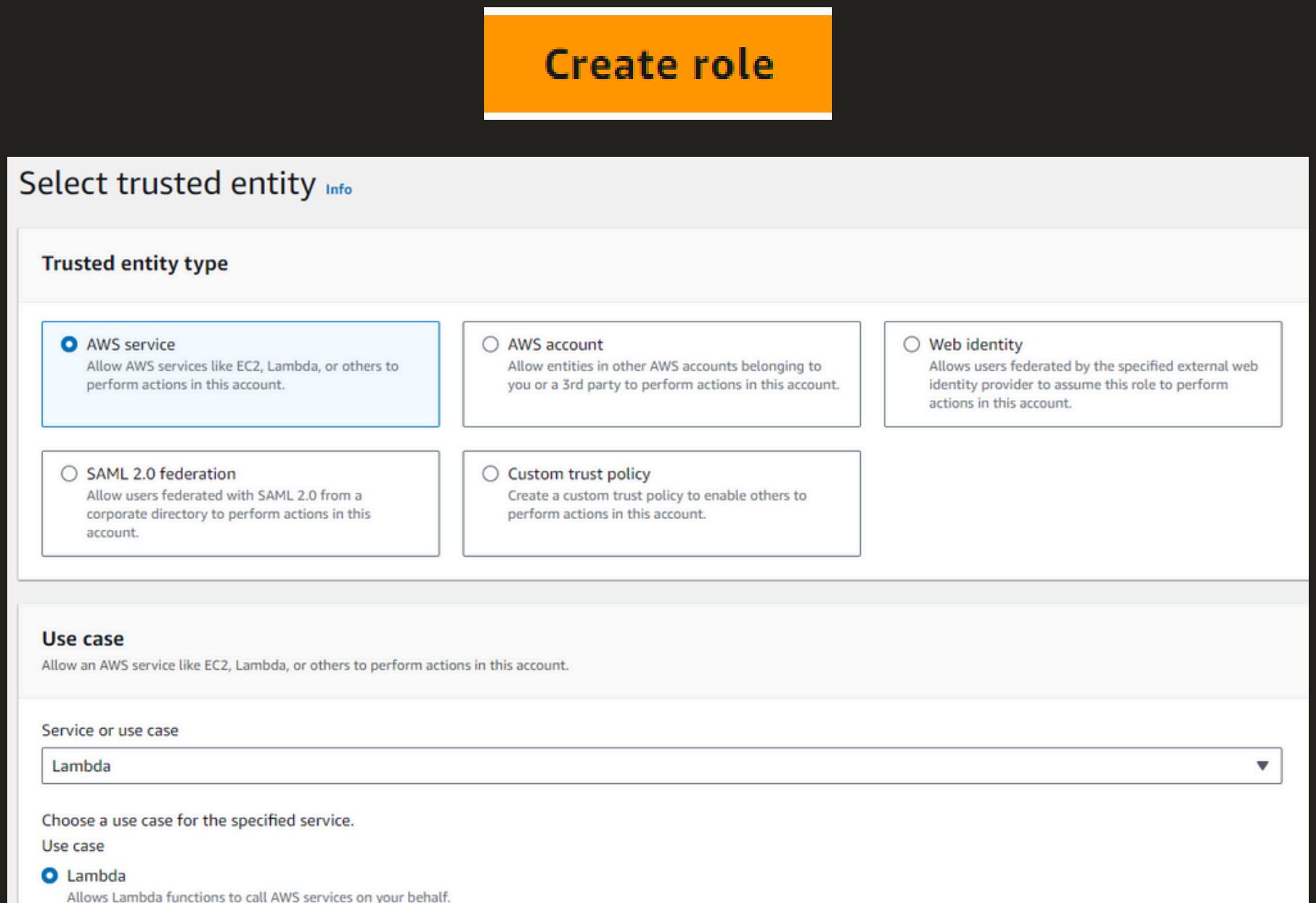
Click on the JSON tab

Write the code to the Policy Editor

Visual

JSON

- Now From the service dropdown, go to Role tab



Policy editor

```
1 Version: "2012-10-17",
2 Statement: [
3 {
4   Effect: "Allow",
5   Action: [
6     "logs:PutLogEvents",
7     "logs>CreateLogGroup",
8     "logs>CreateLogStream"
9   ],
10  Resource: "arn:aws:logs:*:*:*"
11 },
12 {
13   Effect: "Allow",
14   Action: ["s3:GetObject"],
15   Resource: "arn:aws:s3:::my-source-bucket-project/*"
16 },
17 {
18   Effect: "Allow",
19   Action: ["s3:PutObject"],
20   Resource: "arn:aws:s3:::my-destination-bucket-project/*"
21 },
22 ]
23 ]
```

After Writing the code click on Next button

Next

I. Now, add Role Details

Role details

Role name
Enter a meaningful name to identify this role.
Maximum 64 characters. Use alphanumeric and '+,.@-_' characters.

Description
Add a short explanation for this role.
Allows Lambda functions to call AWS services on your behalf.
Maximum 1000 characters. Use letters (A-Z and a-z), numbers (0-9), tabs, new lines, or any of the following characters: _+=,. @-/[\[]!#\$%^&*()";'"<>'`



2. Again come back to the lambda function, and where you choose the option Choose an Existing Role

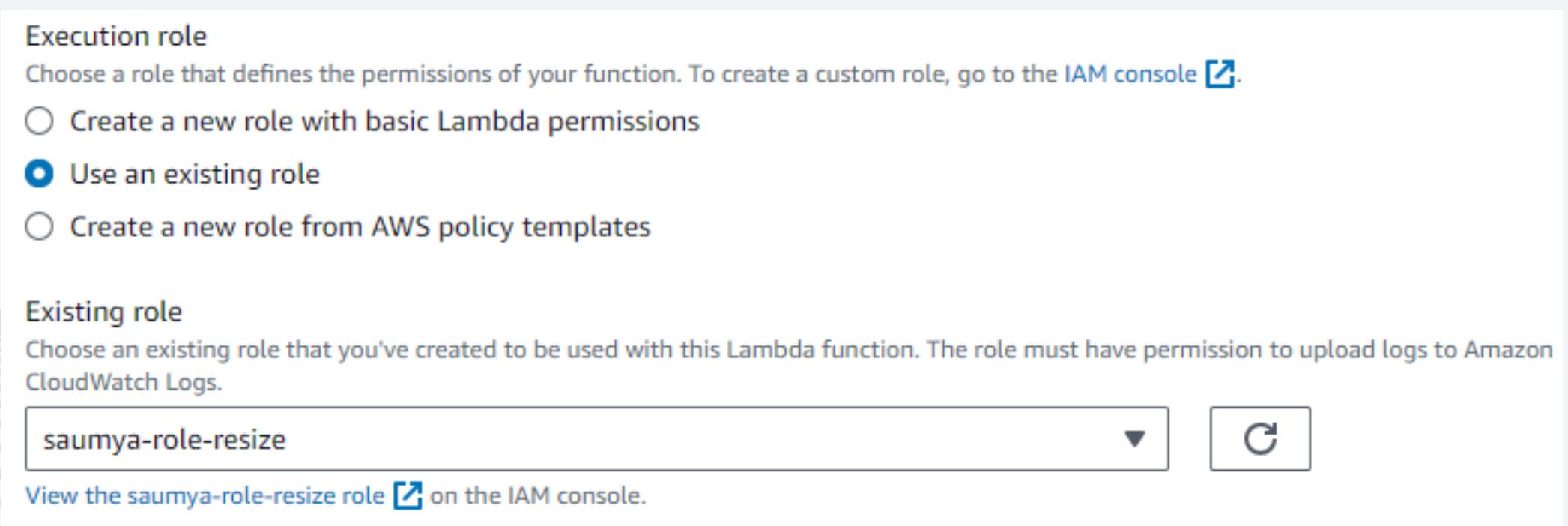
Execution role
Choose a role that defines the permissions of your function. To create a custom role, go to the [IAM console](#).

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.

saumya-role-resize

[View the saumya-role-resize role](#) on the IAM console.



3. In lambda add trigger function

saumya-lambda-function

▼ Function overview [Info](#)

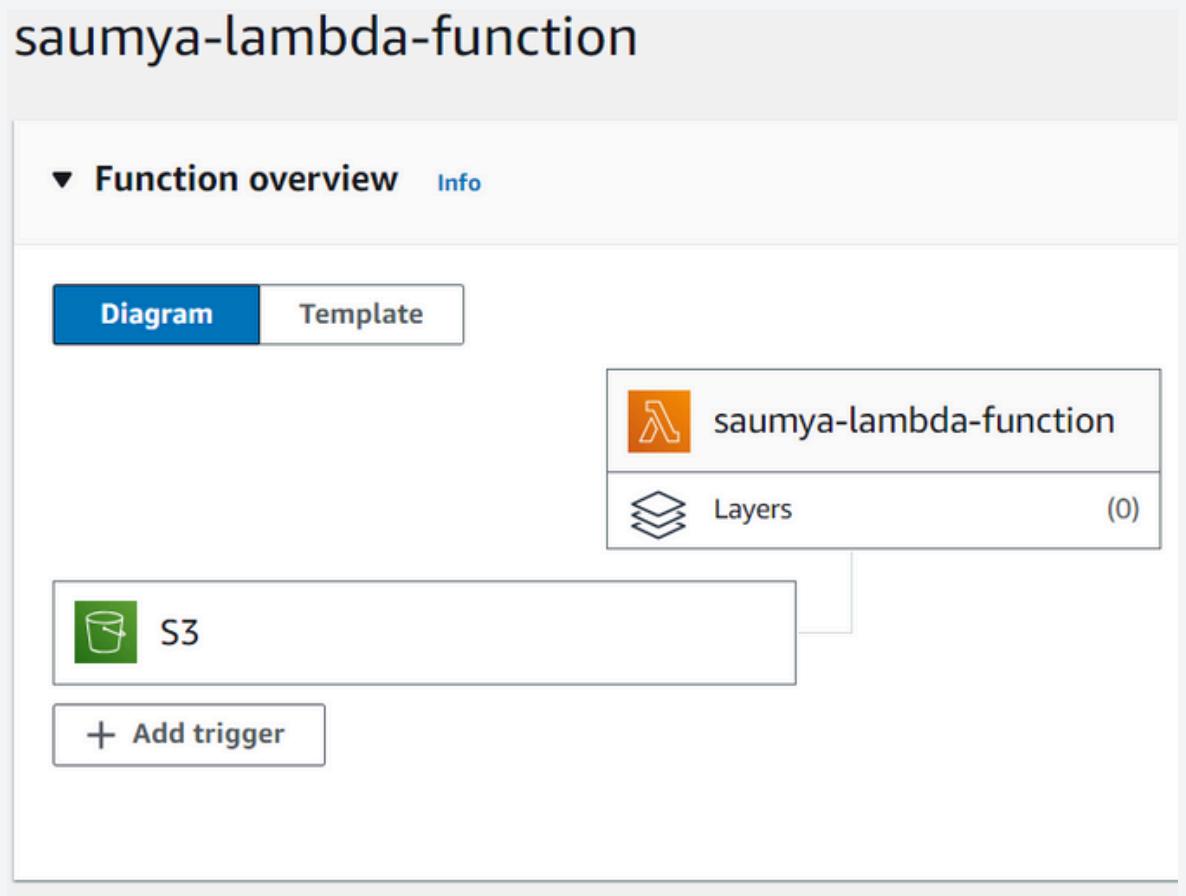
[Diagram](#) [Template](#)

 saumya-lambda-function

 Layers (0)

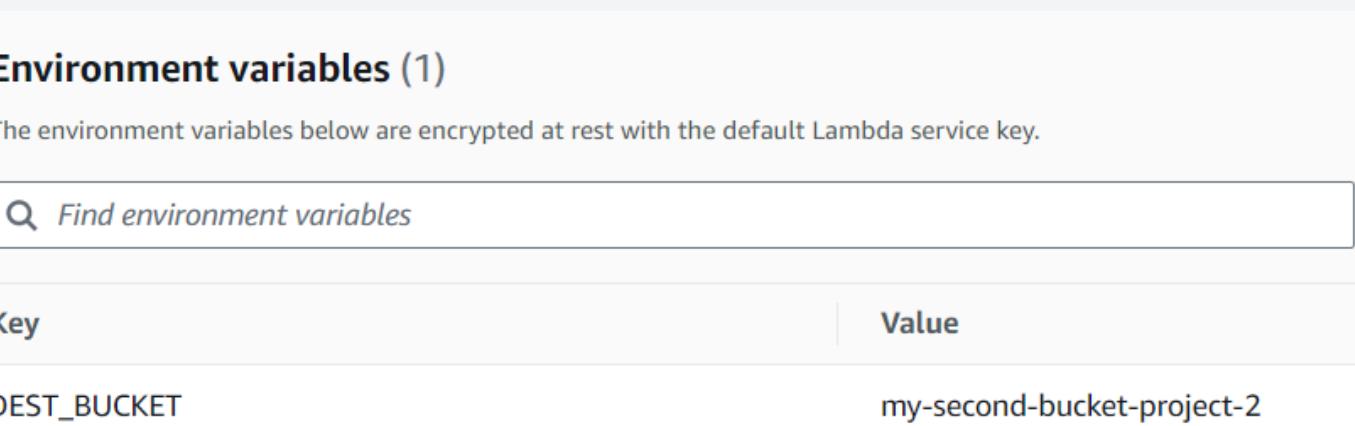
 S3

[+ Add trigger](#)

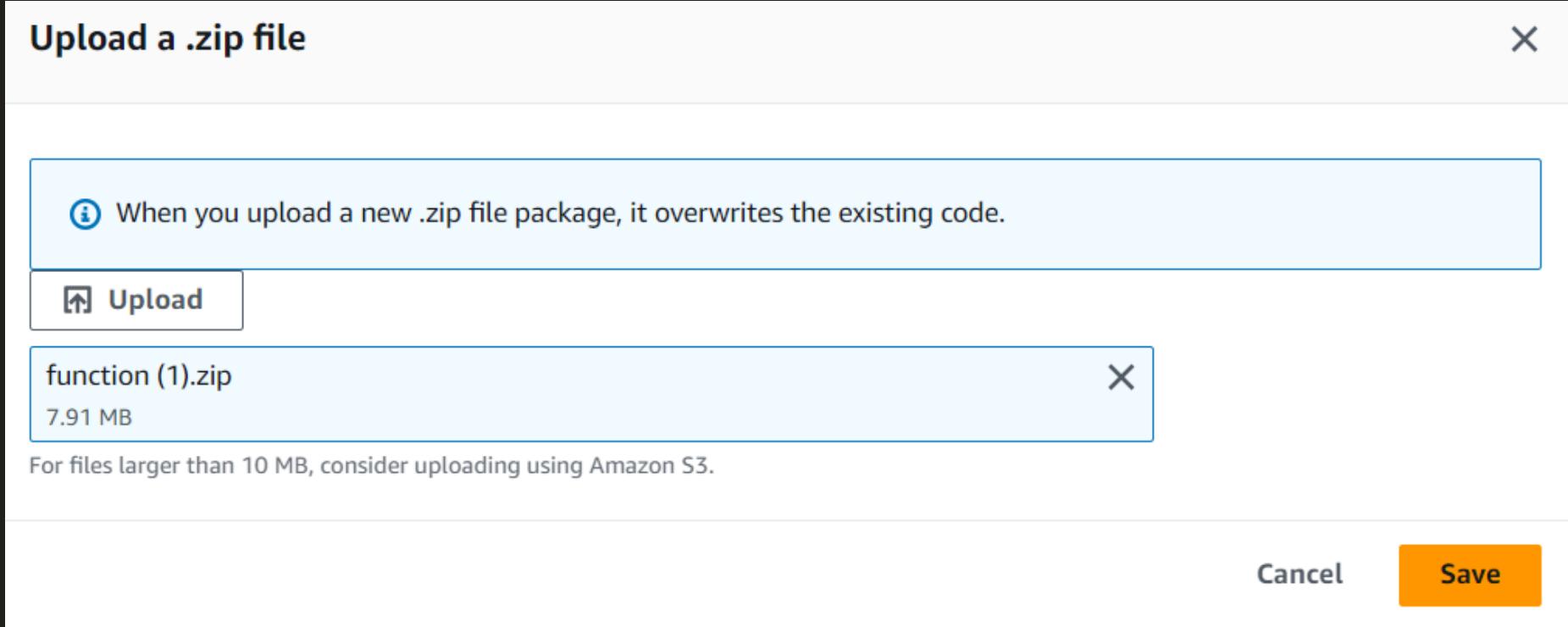


4 . Go to the configuration tab choose environmental variables set key and value where value is your destination bucket name

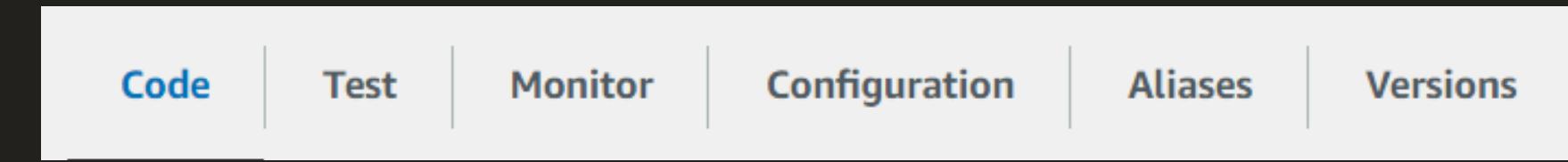
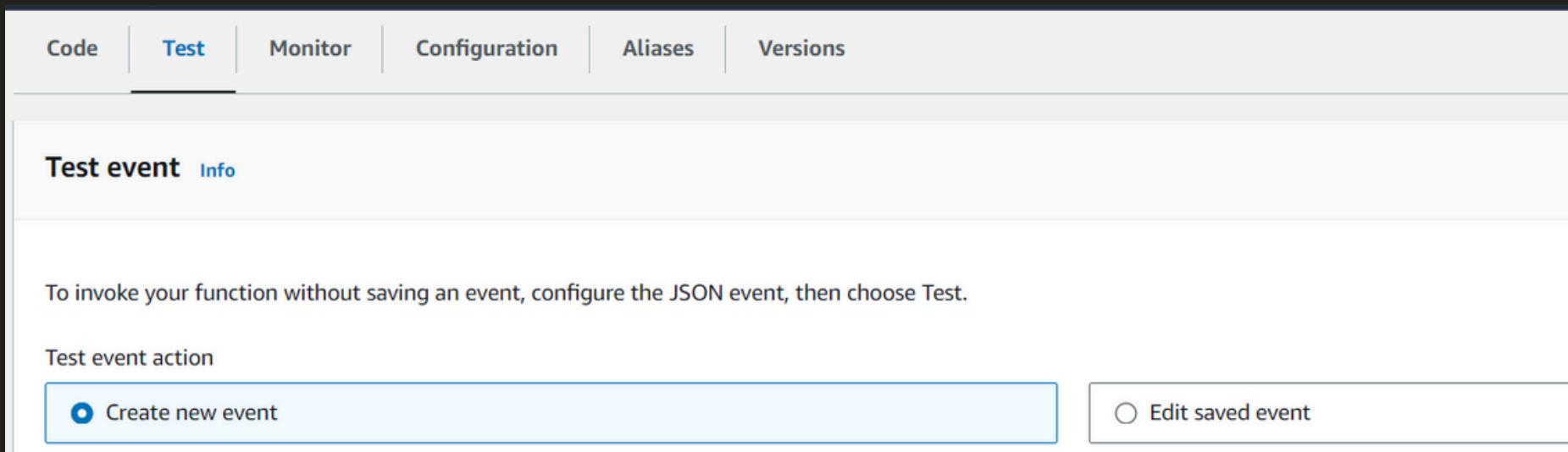
Environment variables (1)	
Key	Value
DEST_BUCKET	my-second-bucket-project-2



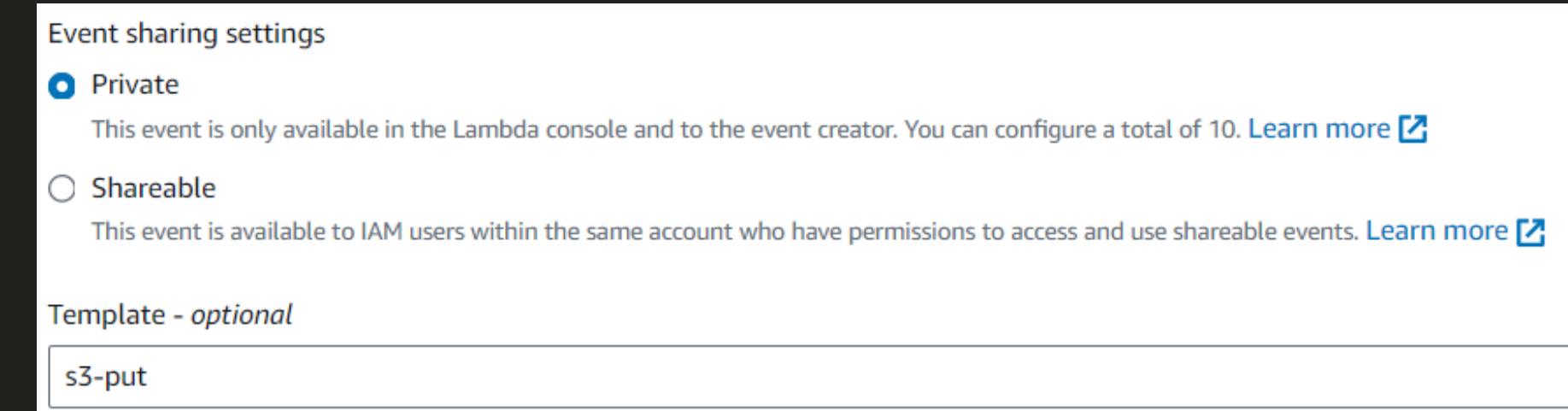
5. Go to code tab and upload a zip file of your automatic resize function code



6. After uploading a file, go to test tab to execute function code successfully



7. Give s3-put to the Template

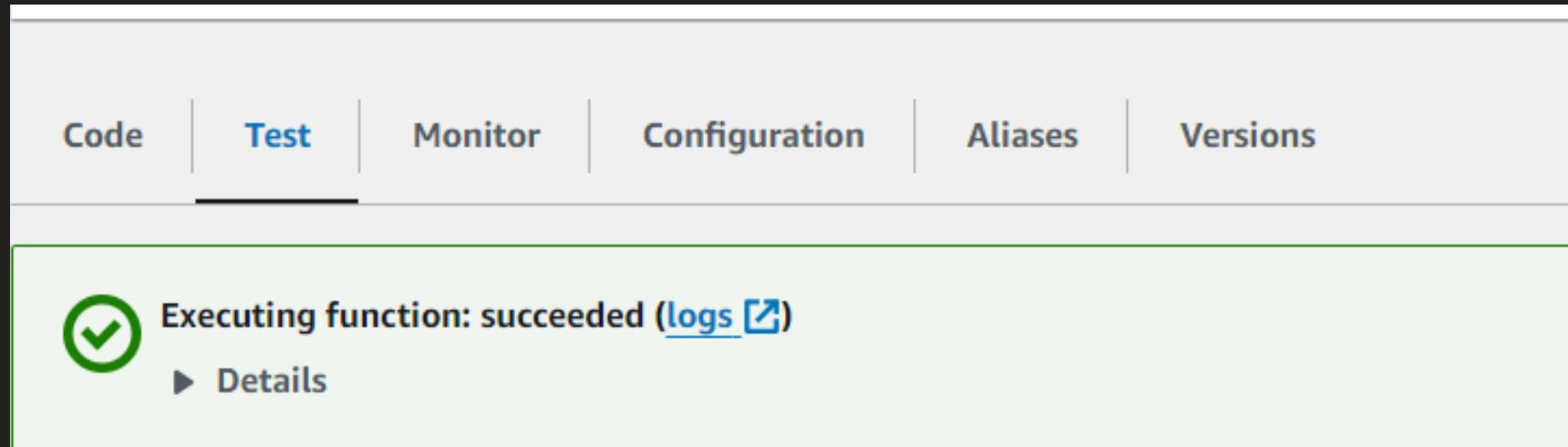


8. now make changes in the code, give the name of your source bucket to the example tag and in the key tag write the name of your uploaded image or file

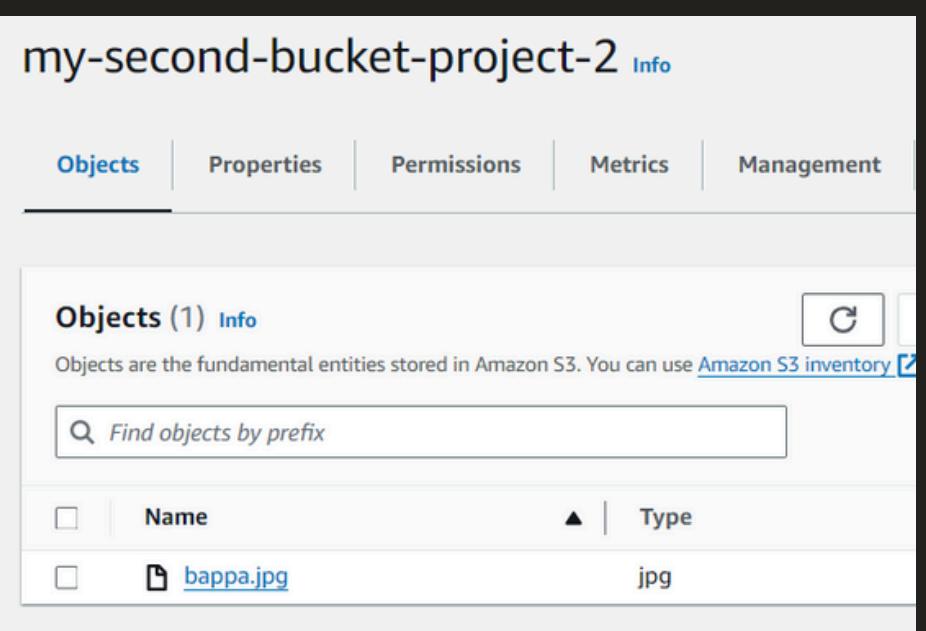
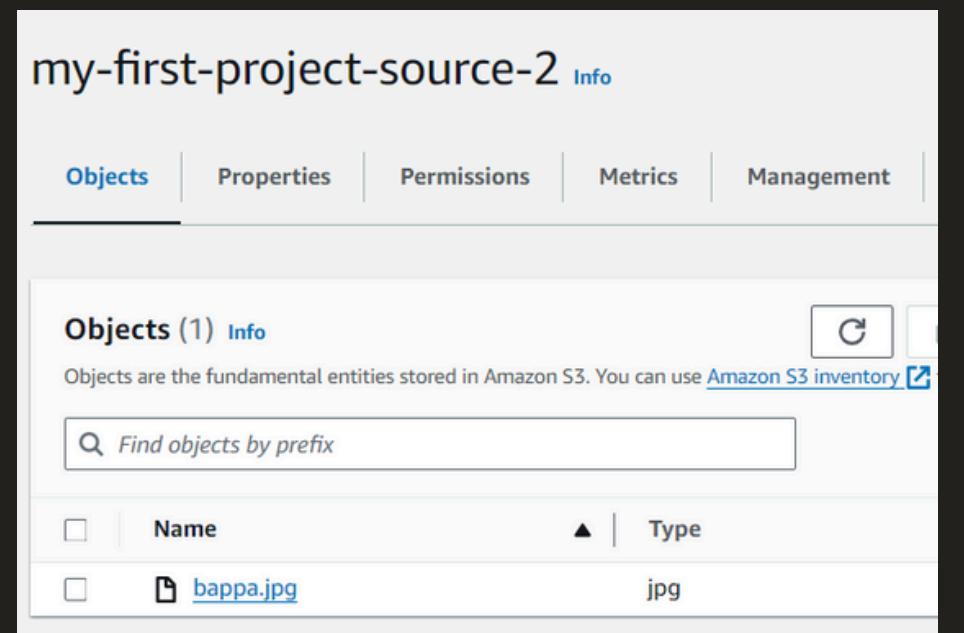
Event JSON

```
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": "EXAMPLE123/5678abcdefghijklmnaaaaaaaaaaaaaaaaqrstuvwxyzABCDEGFGH"
18        },
19        "s3": {
20          "s3SchemaVersion": "1.0",
21          "configurationId": "testConfigRule",
22          "bucket": {
23            "name": "my-first-project-source-2",
24            "ownerIdentity": {
25              "principalId": "EXAMPLE"
26            },
27            "arn": "arn:aws:s3:::my-first-project-source-2"
28          },
29          "object": {
30            "key": "bappa.jpg",
31            "size": 1024,
32            "eTag": "0123456789abcdef0123456789abcdef",
33            "sequencer": "0A1B2C3D4E5F678901"
34          }
35        }
36      }
37    ]
38  }
```

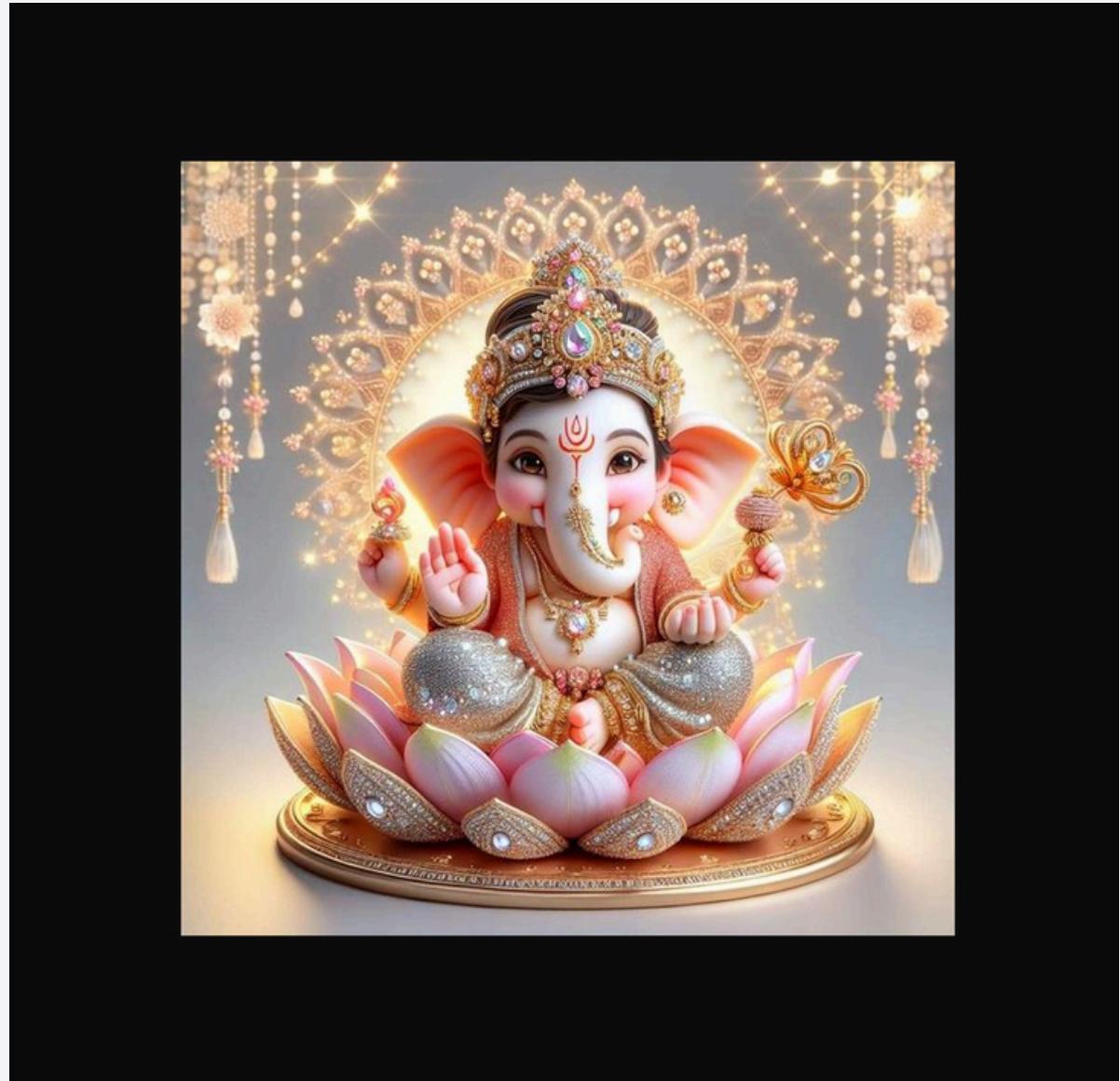
9. Test the code, It must be Executed Successfully



10. Come back to your source bucket , see your uploaded image will be appeared on the Destination Bucket



The Result



Thank You