**CLOUD COMPUTING WITH**

**AWS SERVICES**

**PROJECT**

****

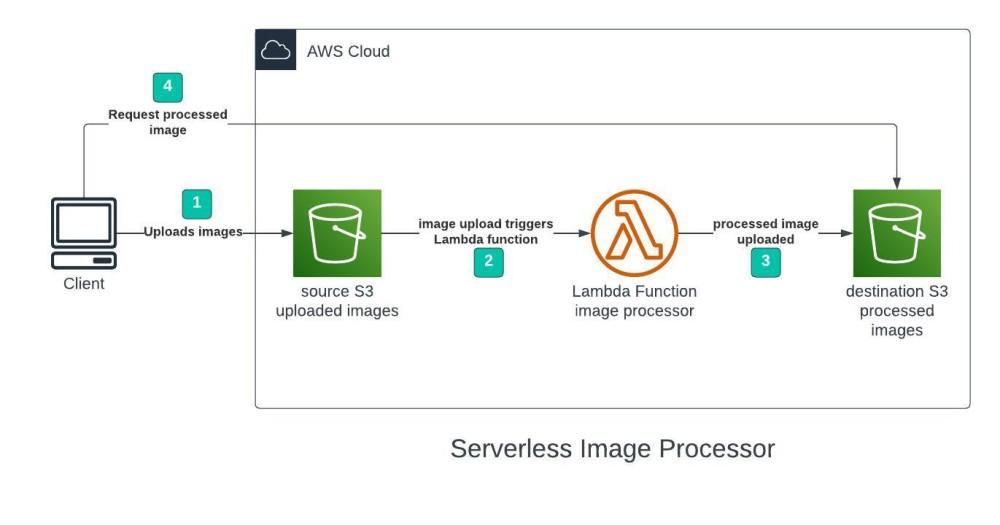
**Submitted by:**

**Himanshi Gupta**

**Project 1**

**Serverless Image Processing**

**Serverless Image Processing Flow**

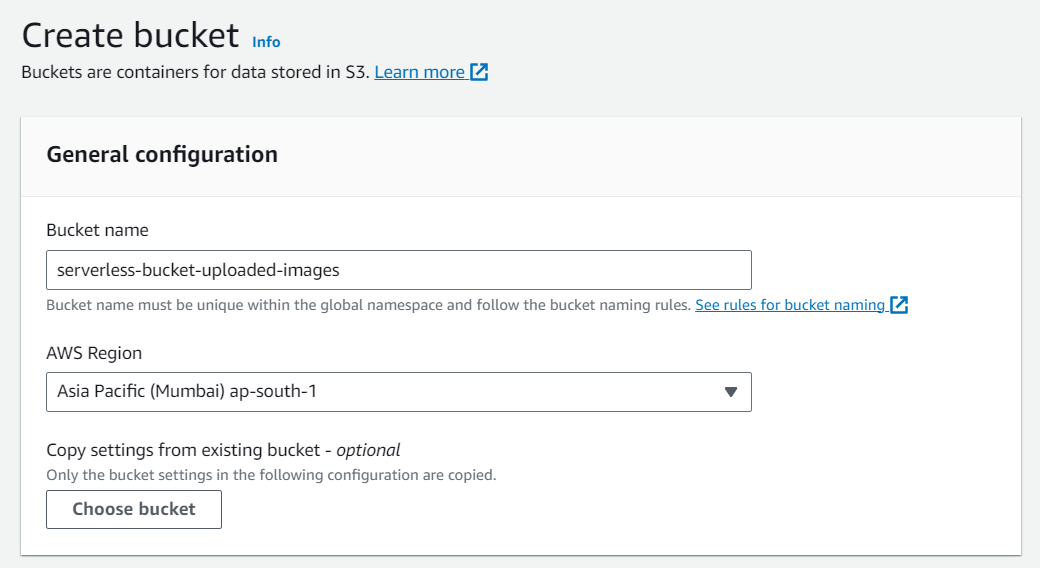
1. The user uploads a file to the source S3 bucket (which is used for storing uploaded images).
2. When the image is uploaded to a source S3 bucket, it triggers an event that invokes the Lambda function. The lambda function processes the image.
3. The processed image is stored in the destination S3 bucket.
4. The processed image is requested by the user.

**Step 1 – Creating S3 buckets**

We will use two S3 buckets:

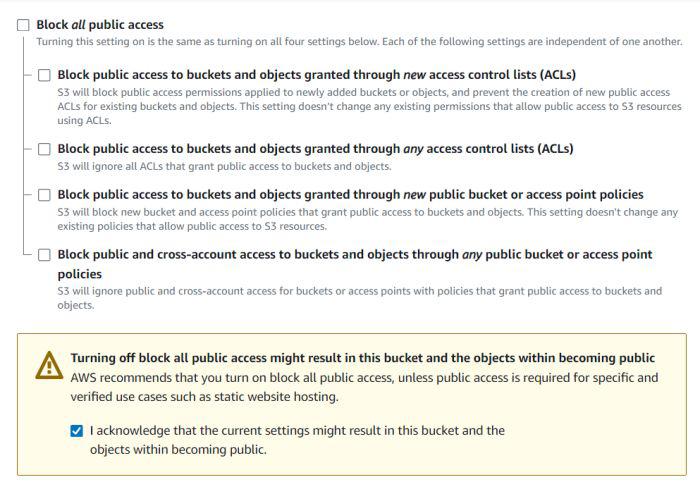
1. **source Bucket:** For storing uploaded images.
2. **destination Bucket:** For storing processed images.

Go to the S3 console and click Create Bucket. Enter bucket name as ‘serverless-bucket-uploaded-images’. Choose any AWS region as ‘ap-south-1’.



**Step 2 – Configuring the S3 bucket policy**

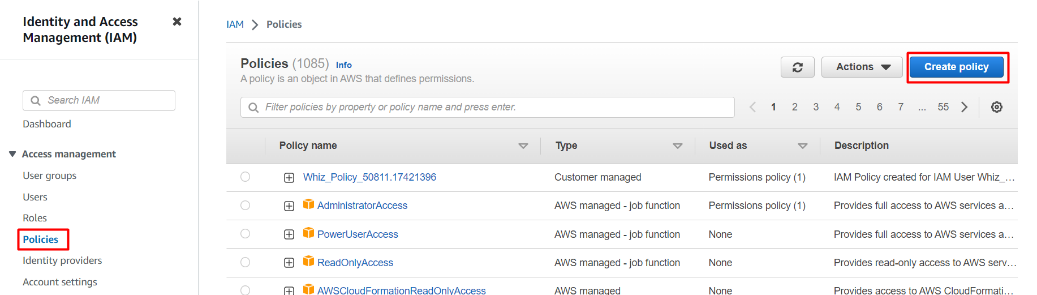
In the ‘Block Public Access settings for this bucket’ section disable “block all public access”. You will get a warning that the bucket and its objects might become public. Agree to the warning**. (Note: we are making this bucket public only for this project, it is not recommended to make an S3 bucket public if not needed)**.



Leave all other settings as default and create a bucket. Similarly, create another bucket named ‘serverless-bucket-processed-images’ with the same region.

## STEP 3: Create an IAM Policy

1. As a pre-requisite for creating the Lambda function, we need to create a user role with a custom policy.
2. Go to **Services** and Select **IAM** under **Security, Identity, and Compliance.**
3. Click on **Policies**in the left navigation bar and click on the **Create Policy**button.



    4. Click on the **JSON** tab, Remove the existing code, and copy-paste the below policy statement into the editor:

* Policy JSON:

**{**

**"Version":"2012-10-17",**

**"Statement":[**

**{**

**"Effect":"Allow",**

**"Action":[**

**"s3:GetObject"**

**],**

**"Resource":[**

**"arn:aws:s3:::mysourcebucket12345/\*"**

**]**

**},**

**{**

**"Effect":"Allow",**

**"Action":[**

**"s3:PutObject"**

**],**

**"Resource":[**

**"arn:aws:s3:::mydestinationbucket12345/\*"**

**]**

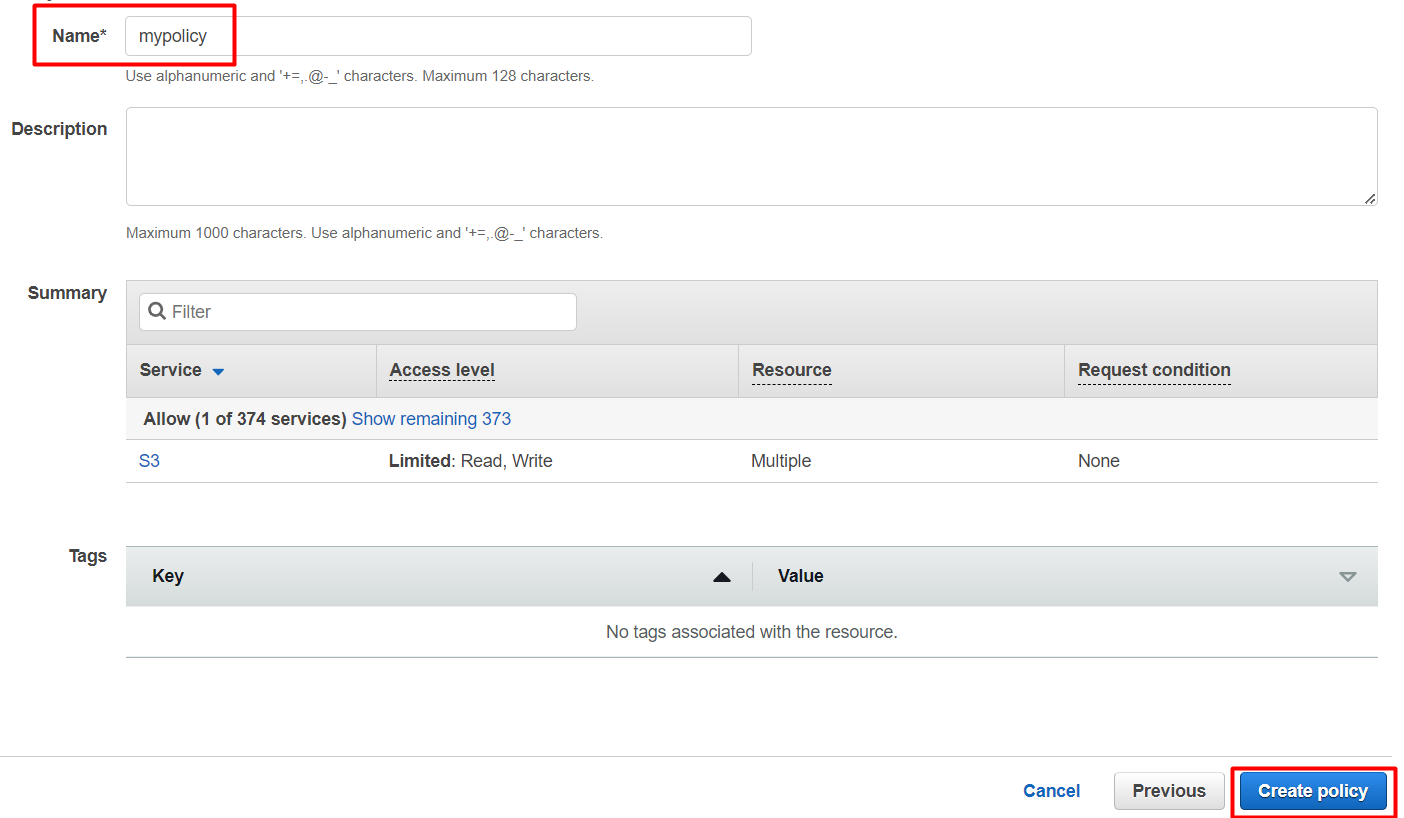
**}**

**]**

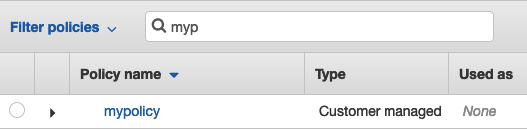
**}**

the **Source**and **destination ARN name**of the bucket (which you have saved before) in the option **Resource**. Make sure to add **/\***at the end of the **ARN name.**

* Leave everything as default and click on the **Next**button.
* On the Review Policy Page:
  + Policy Name: Enter **mypolicy**
  + Click on the **Create policy** button.



1. An IAM Policy with the name **mypolicy** is created.



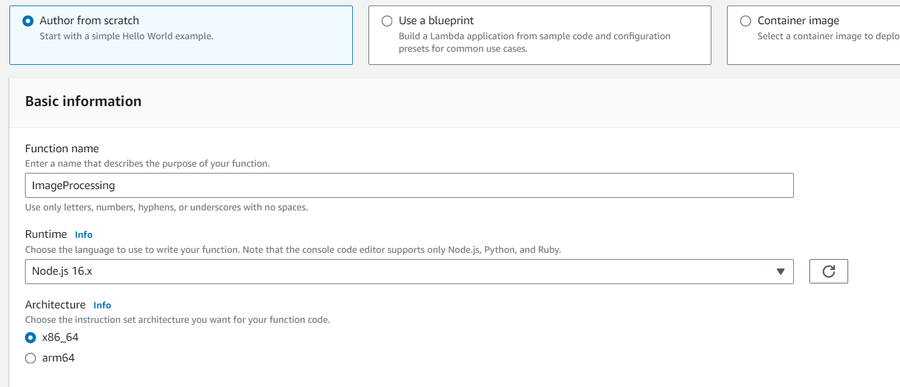
## STEP 4: Create an IAM Role

1. In the left menu, click on **Roles.**Click on the **Create role**button.
2. Select **Lambda** from AWS Services list.

* From **Trusted Entity Type**: Select **AWS Service**
* From **Use case**: Select **Lambda**
* Click on the **Next** button.

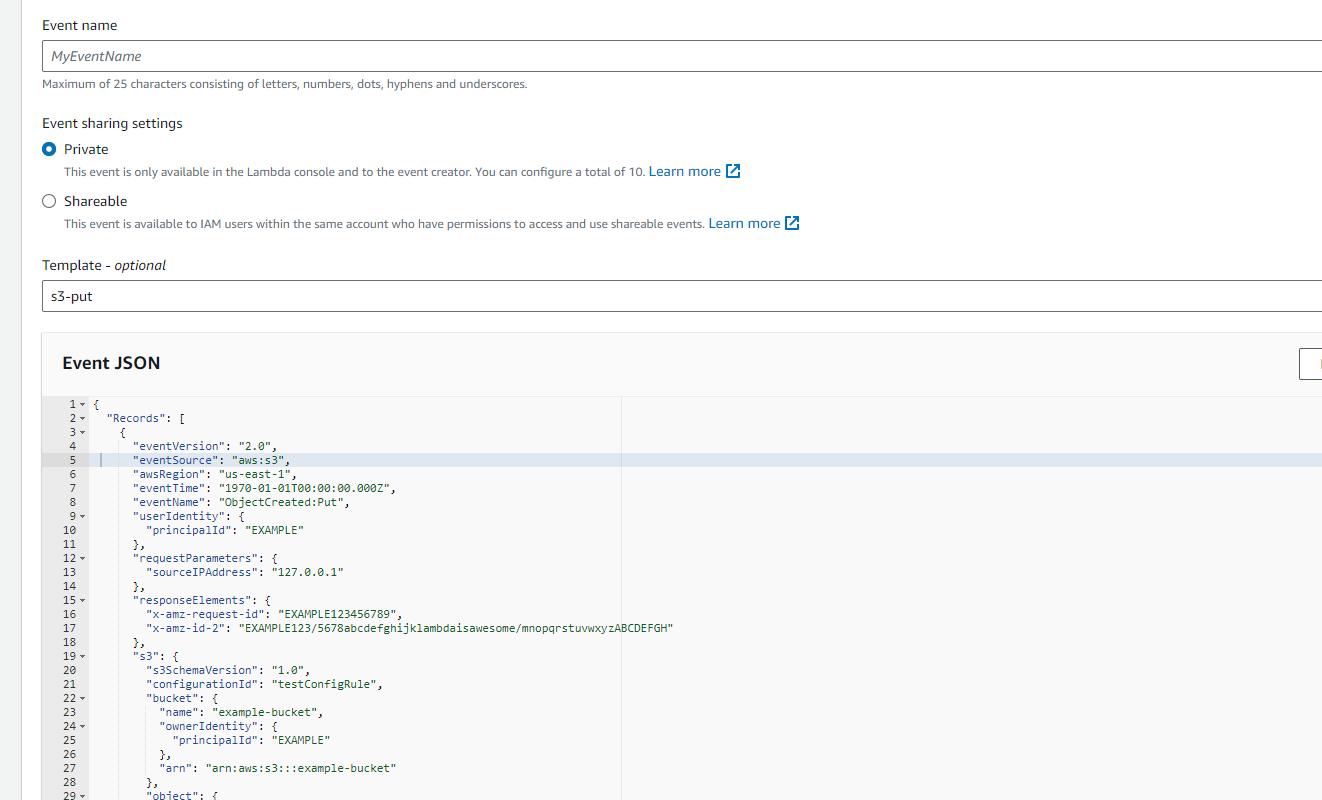
**Step 5 – Creating Lambda function**

Go to AWS Lambda console. Navigate to the Functions section. Click Create Function and name it “ImageProcessing”. Select runtime as “NodeJS 16.x” and architecture as “x86\_64”. Leave all other settings as default. Create the function.



In the code editor on the Lambda function page upload the zip file Then click on the configuration and go to the environment variables.

* Use key: **DUST\_BUCKET** and fill in the name of your destination bucket in the value block.
* Now click on the test tab and select the S3-put.



* Change the example bucket to the source bucket and also change "test%2Fkey", to your uploaded image name. know it’s ready to see so click on the test button

**STEP 6: Testing the application**

Upload an image file to the source S3 bucket (“serverless-bucket-uploaded-images”). Wait for a few seconds and check the destination bucket (“serverless-bucket-processed-images”). There you will see two images

(thumbnail and cover photo).

Congratulations, you just built a serverless Image-processing application.