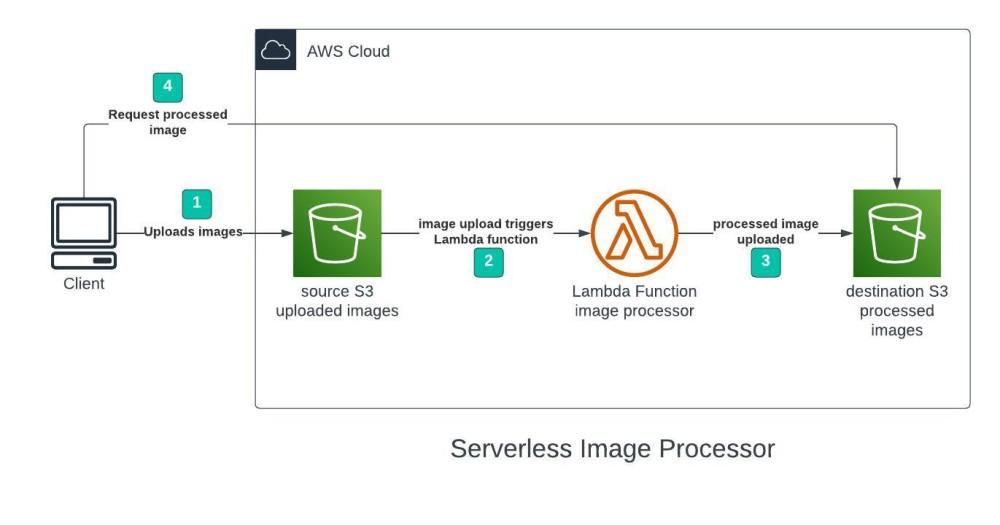
**Project 1**

**Serverless Image Processing**

Create a serverless image processing application that automatically resizes and optimizes images

uploaded to an Amazon S3 bucket.



Tanuja Tiwari

Table of Contents

* **Introduction**
* **Objective**
* **Abstract of the Project**
* **Services used**
* **Implementation**
* **Step 1 - Creating S3 buckets**
* **Step 2 - Configuring S3 bucket policy**
* **Step 3 – Creating policy in IAM**
* **Step 4 -upload zip file in Lambda function**
* **Step 5 - Edit environment variables**
* **Step 6 – Test Lambda Function**
* **Introduction**

**This document outlines a mini-project for implement ng an AWS serverless image processing system. The project utilizes AWS services such as S3, Lambda, and IAM. The aim is to automatically resize and optimize images uploaded to an S3 bucket using a Lambda function on integrated with the Sharp image processing library.**

* **Objective**

**The objective of creating a serverless image processing application that automatically resizes and optimizes images uploaded to an Amazon S3 bucket is to streamline and automate the handling of image assets in a scalable and cost-effective manner**

* **Abstract of this project**

**This project focuses on the development of a serverless image processing application designed to automatically handle image uploads to an Amazon S3 bucket. Leveraging AWS services such as Lambda, S3, and CloudWatch, the system detects new image uploads, then dynamically resizes and optimizes them into various predefined dimensions and formats suitable for responsive applications. The processed images are stored back in S3, enabling seamless delivery through a content delivery network if needed. By adopting a serverless architecture, the application ensures high scalability, reduced operational costs, and minimal maintenance, making it an efficient solution for modern web and mobile platforms that rely heavily on fast and optimized image delivery.**

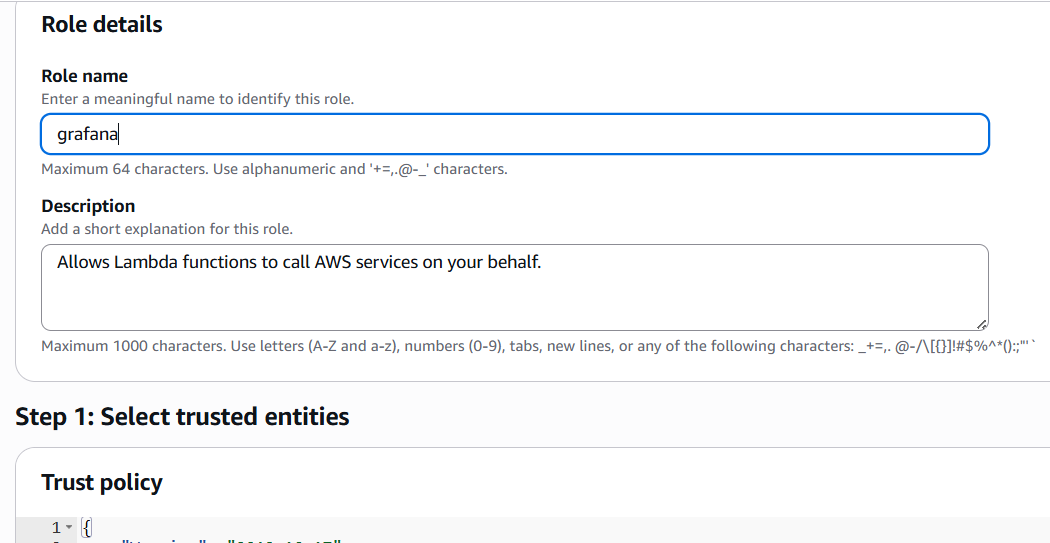
* **Services used**
* **S3**
* **Lambda**
* **IAM**
* **AWS cloudshell**
* **Implementation**

**Step 1 - Creating S3 buckets:**

**We will use two S3 buckets:**

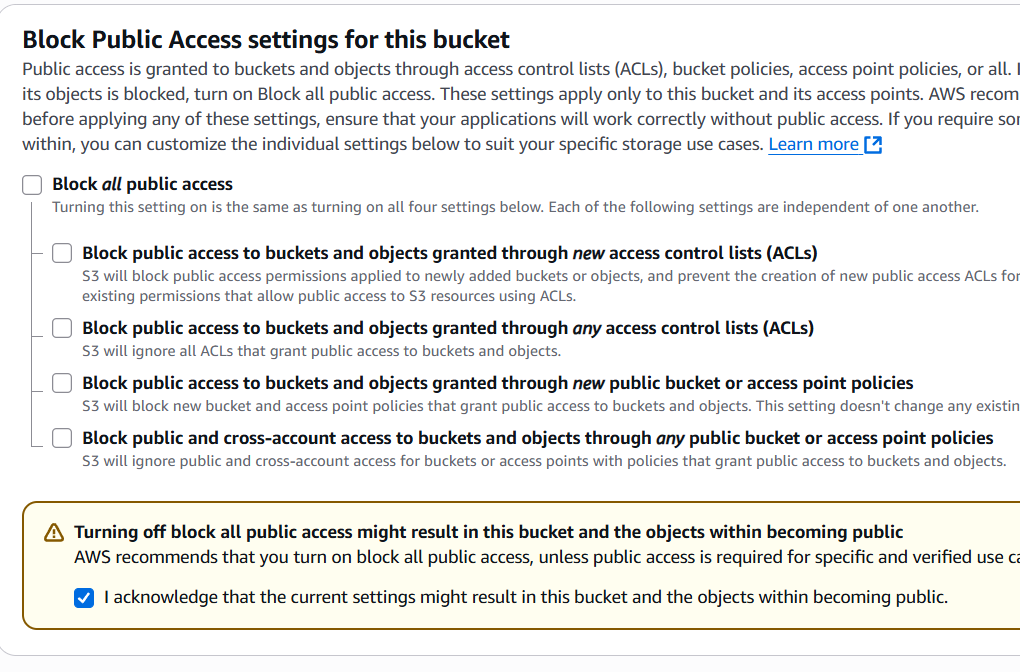
1. **source Bucket: For storing uploaded images.**
2. **destination Bucket: For storing processed images.**
3. **Go to S3 console and click Create bucket.**
4. **Enter bucket name as 'serverless-bucket-uploaded-images'.**

**Choose any AWS region as 'ap south-1'.**

****

**Step 2 - Configuring S3 bucket policy**

**In 'Block Public Access settings for this bucket' sec on 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).**

****

**Step 3 – Creating policy in IAM**

1. **Go to AWS IAM console. Navigate to policies section.**
2. **Click Create policies in (JSON) and name it “ImageBucketpolicy”.**
3. **Leave all other settings as default. Create the policy.**

**Policy:**

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

**"Statement": [**

**{**

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

**"Action": [**

**"logs:PutLogEvents",**

**"logs:CreateLogGroup",**

**"logs:CreateLogStream"**

**],**

**"Resource": "arn:aws:logs:\*:\*:\*"**

**},**

**{**

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

**"Action": ["s3:GetObject"],**

**"Resource": "arn:aws:s3:::BUCKET\_NAME/\*"**

**},**

**{**

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

**"Action": ["s3:PutObject"],**

**"Resource": "arn:aws:s3:::DEST\_BUCKET/\*"**

**}**

**]**

**}**

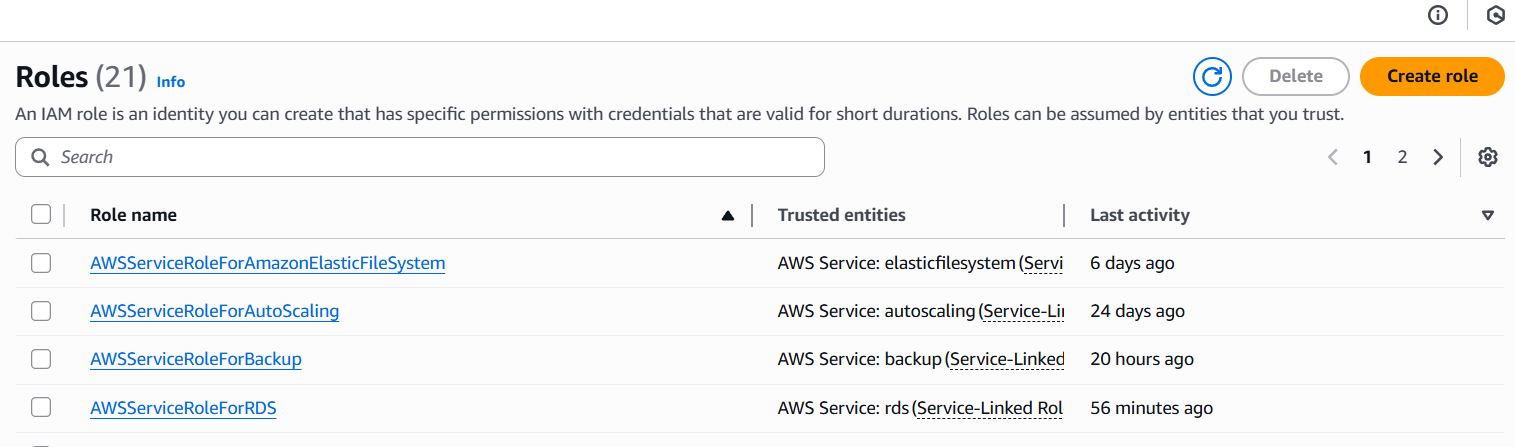
**\*BUCKET\_NAME = SOURSE BUCKET NAME**

**\*DEST\_BUCKET = DESTINATION BUCKET NAME**

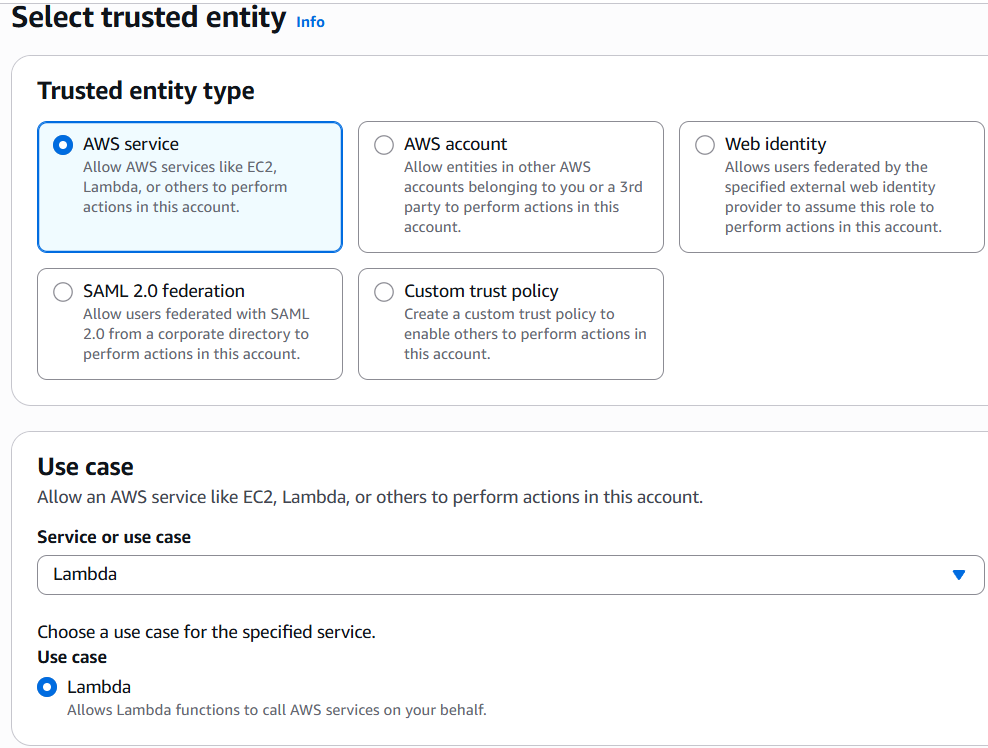
**Step 4 –creating role in IAM:**

**Following Steps are Follows:**

1. **Go to AWS IAM console**
2. **Create role.**



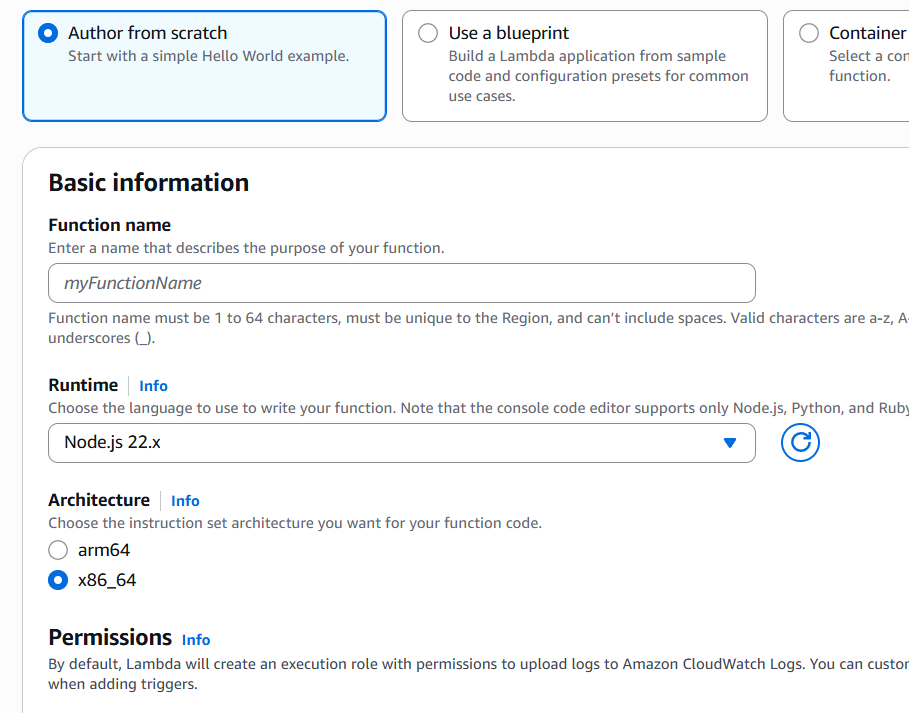
1. **Name: imglambdarole**
2. **Use case – Lambda**

****

1. **Select-ImageBucketPolicy**
2. **Then create role.**

**Step 3 – Creating Lambda function:**

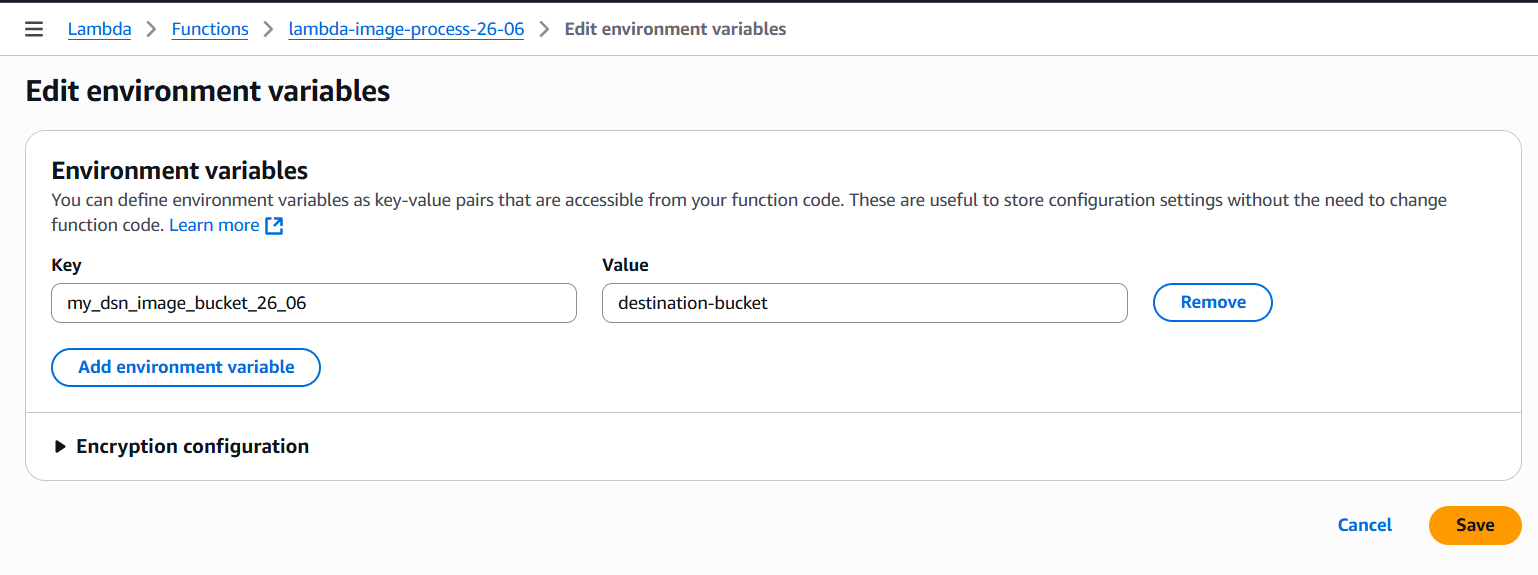
1. **Go to AWS Lambda console. Navigate to Functions section.**
2. **Click Create Function and name it “ImageProcess”.**

****

1. **Select runtime as “NodeJS 16.x” and architecture as “x86\_64”. Leave all other settings as default. Create the function.**
2. **Change the default Execution role.**

**Step 4 -upload zip file in Lambda function:**

**Step 5 - Edit environment variables**

****

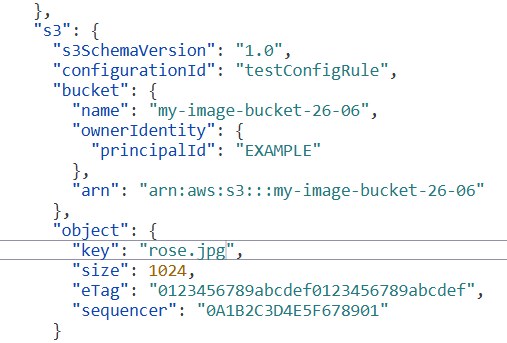
**Step 6 – Test Lambda Function**

1. **Go to AWS Lambda console. Navigate to Functions section.**
2. **open function then will be created**
3. **open test console**
4. **template=s3-put**

**EVENT JSON**

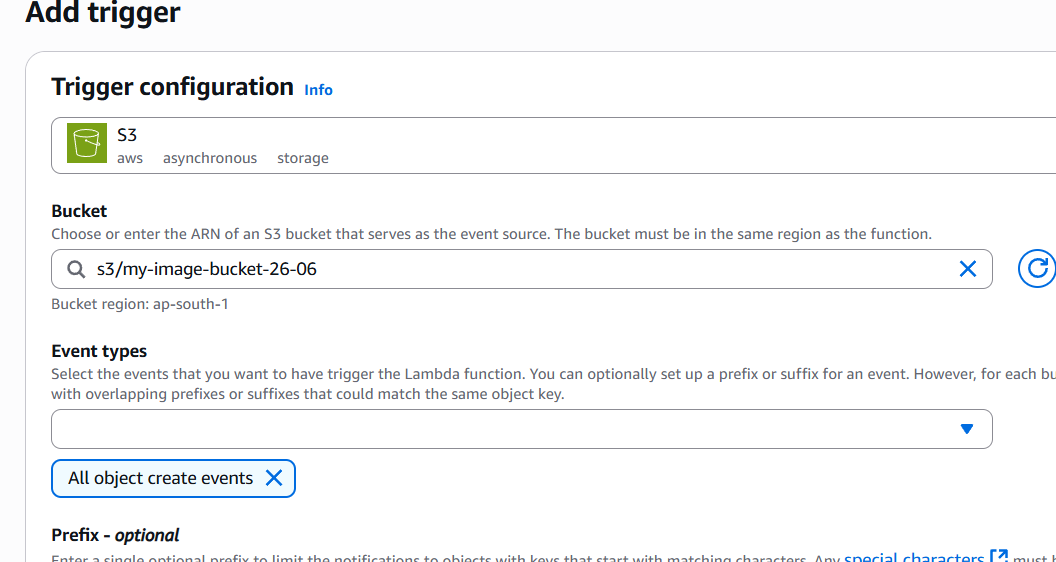
**In event Json we can change only 3 values**

**Name, ARN, key**

****

**Now we can test**

1. **Step 5 – Creating S3 trigger**
2. **Add trigger**
3. **Select s3**
4. **choose source Bucket name**
5. **Now Add**

****

1. **Upload the image in source bucket.**
2. **Now you can see the resize image.**

**Resize Image**

****