**Project-1**

**Creating serverless image processing with aws lambda and s3**

**Table of content**

* Introduction
* Architecture overview
* Folder structure
* Step-by-step guide

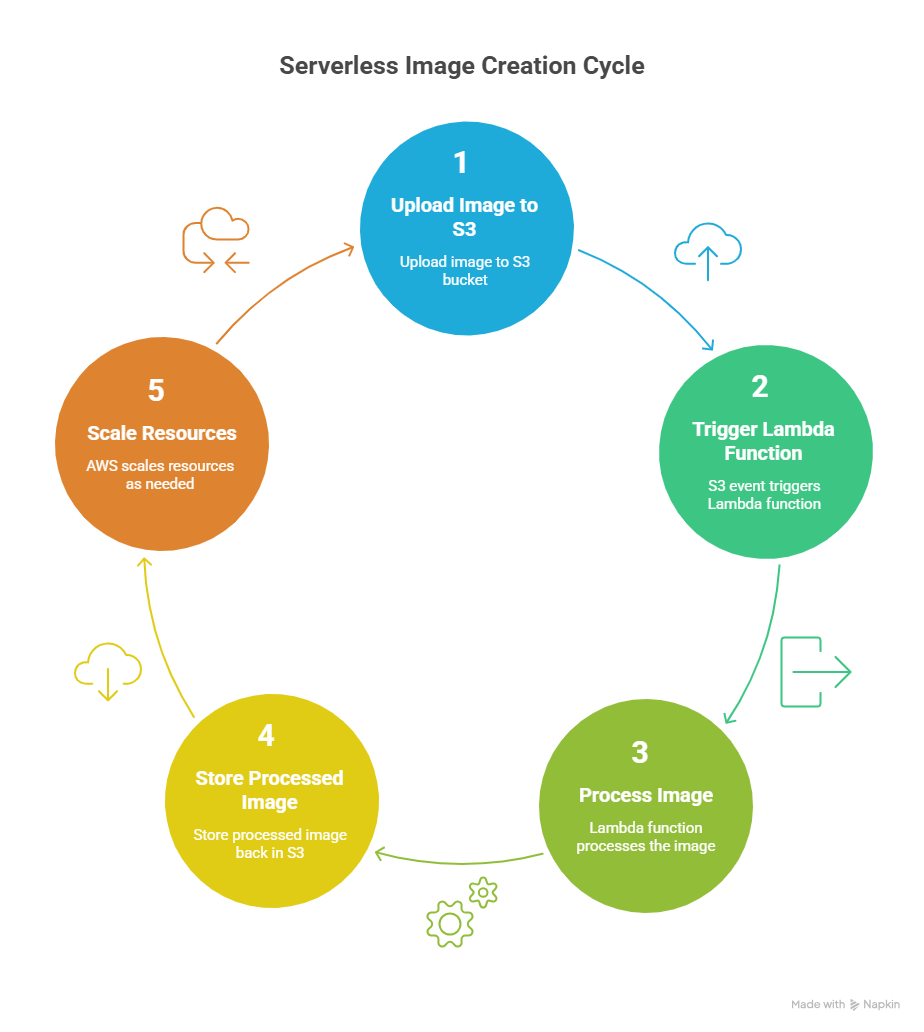
**Introduction: Serverless Image Processing with AWS Lambda and S3**

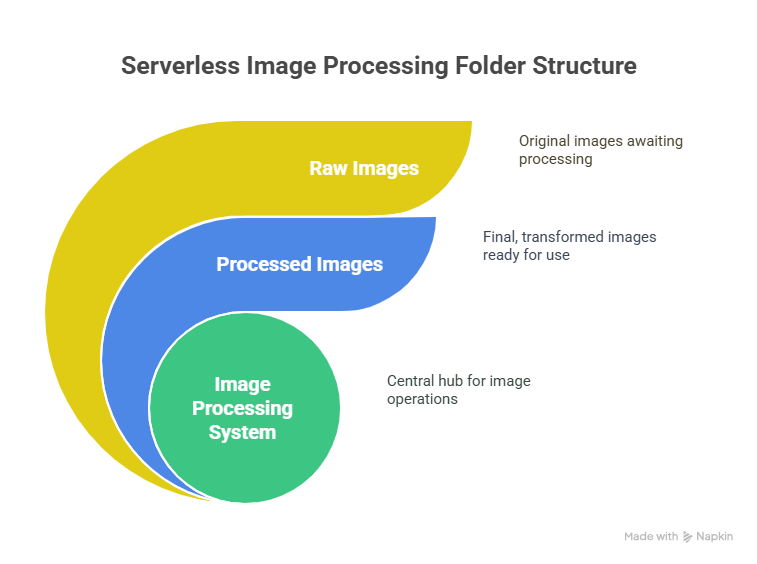
In today's digital landscape, image processing tasks—such as resizing, compressing, or converting formats—are essential for web and mobile applications. Traditionally, these tasks were handled by dedicated servers, but that approach is often costly, hard to scale, and requires constant maintenance.

**Serverless computing** provides a modern, efficient alternative. By combining **Amazon S3 (Simple Storage Service)** for storage and **AWS Lambda** for compute, developers can create a scalable, cost-effective, and fully managed image processing pipeline. With this serverless setup, images uploaded to an S3 bucket can automatically trigger a Lambda function that processes the image in real-time—without provisioning or managing any servers.

This guide introduces you to the basics of building such a solution, covering how to set up S3 buckets, write Lambda functions, and configure event-driven triggers for fully automated image handling.

**Architecture overview**

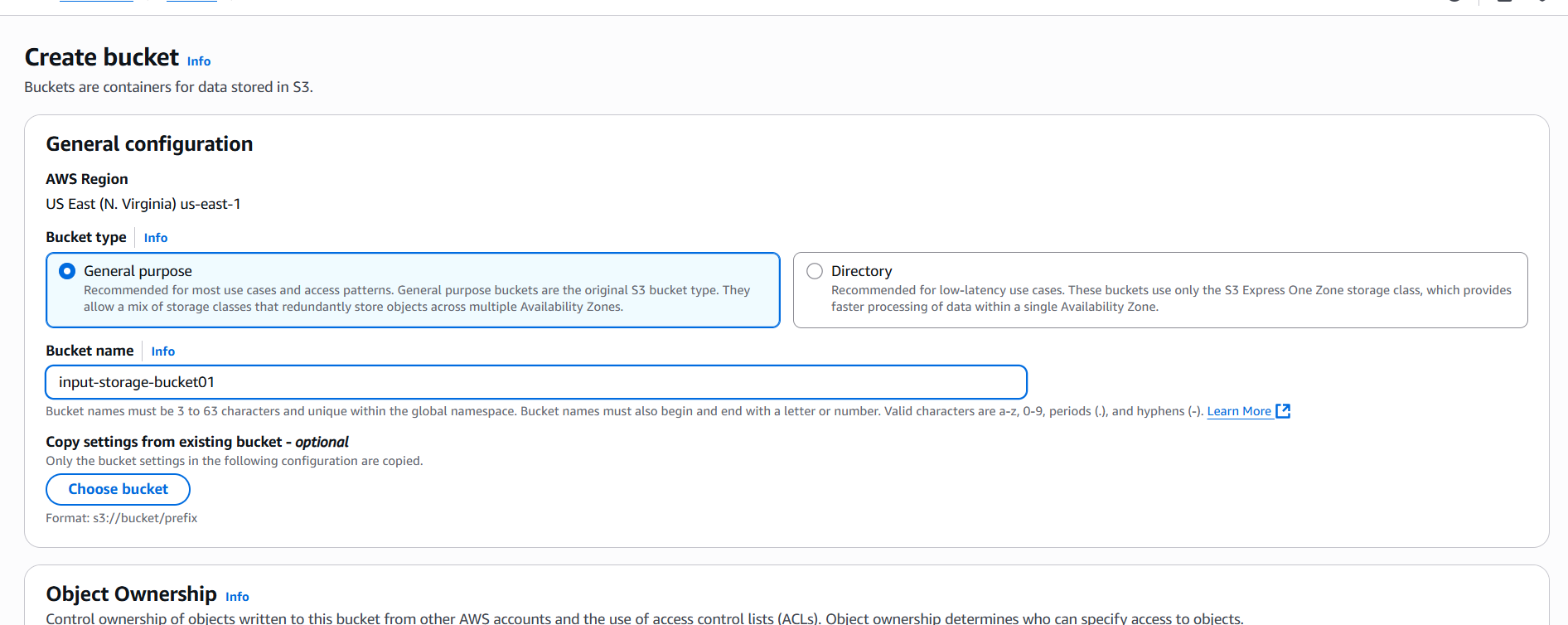
****

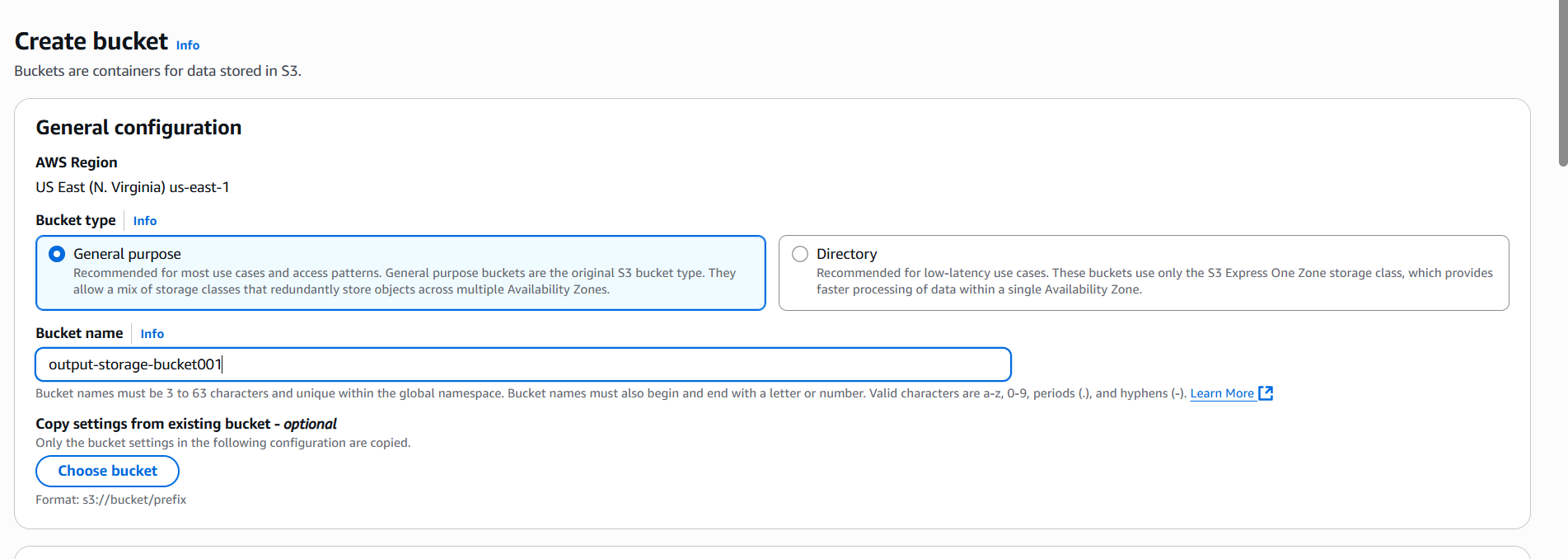
**Folder structure**

**Step-by-step guide**

**Task-1) Setting up s3 bucket.**

* Create two S3 buckets: input-image-bucket for storing the original images and output-image-bucket for storing the processed images. Alternatively, use a single bucket with different prefixes (e.g., original/ and processed/).

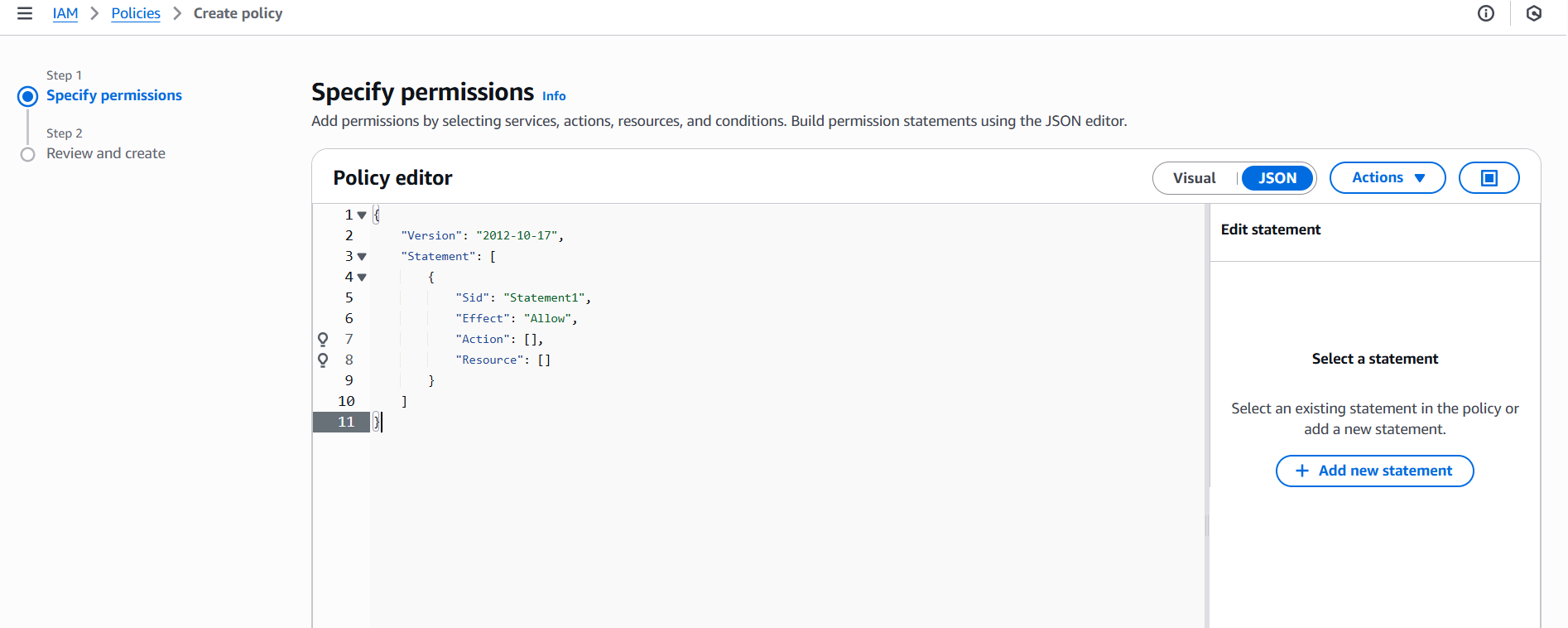


****

* Leave all the settings in default.

**Task-2) Setting up of IAM policy**

* In IAM go to the IAM policy.
* Click on create policy.
* In policy editor go to the JSON.



Replace the given code with this code.

{

  "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/\*"

    }

  ]

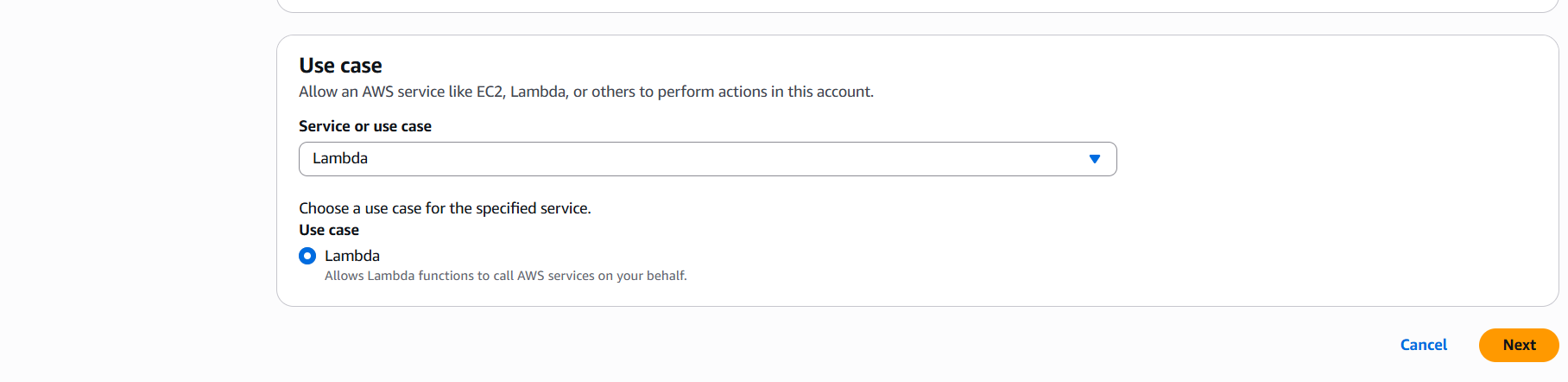
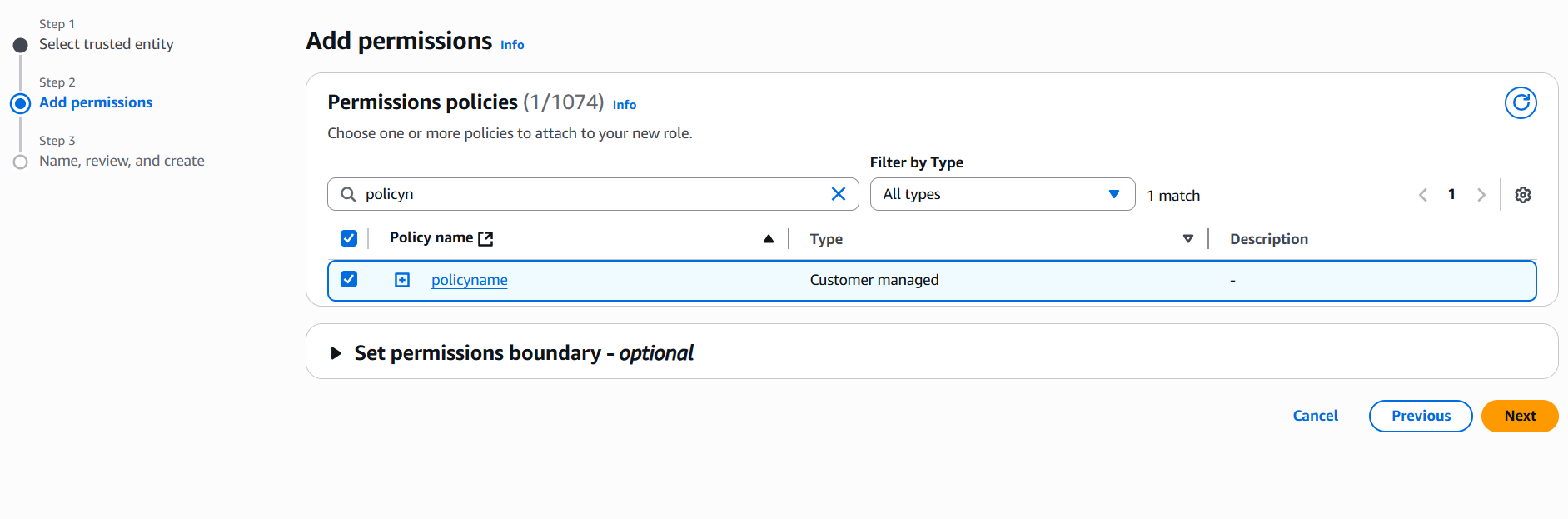
}

* Change the bucket name with your bucket name where you have uploaded your pic and DEST\_BUCKET with the image resizer bucket.



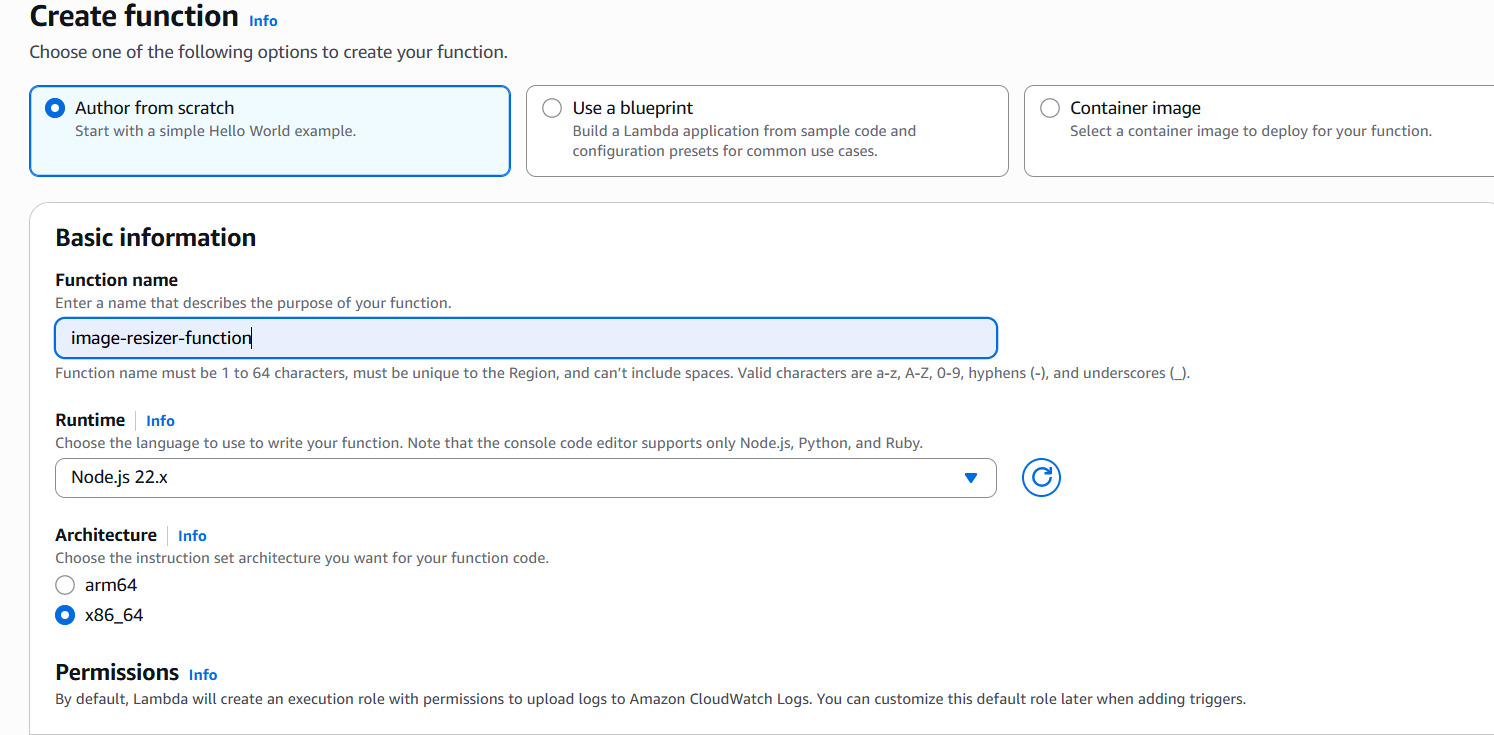
* Click on create policy

**Task-3) Setting up of IAM role**

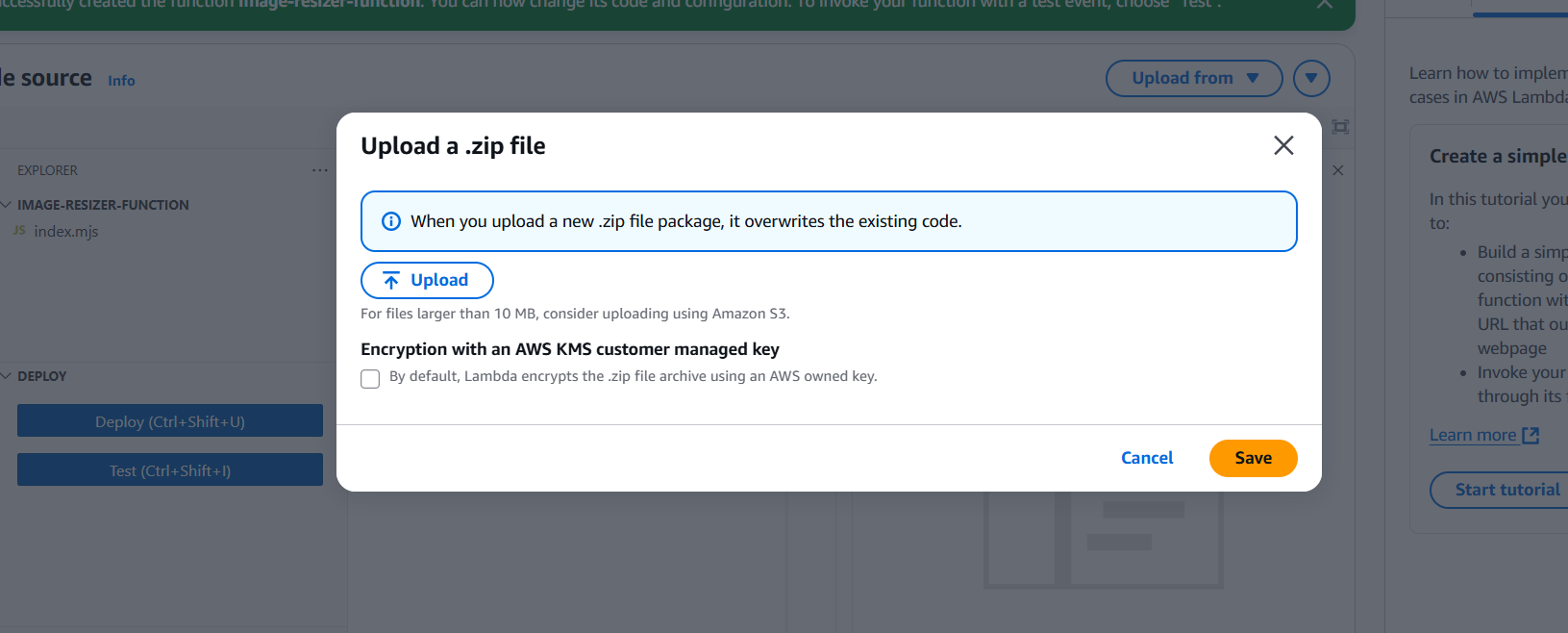
* Now in IAM go to the IAM role and click on create role.
* ****In use case choose lambda.
* ****Now search the policy which you have created earlier.
* H
* Now create the role.

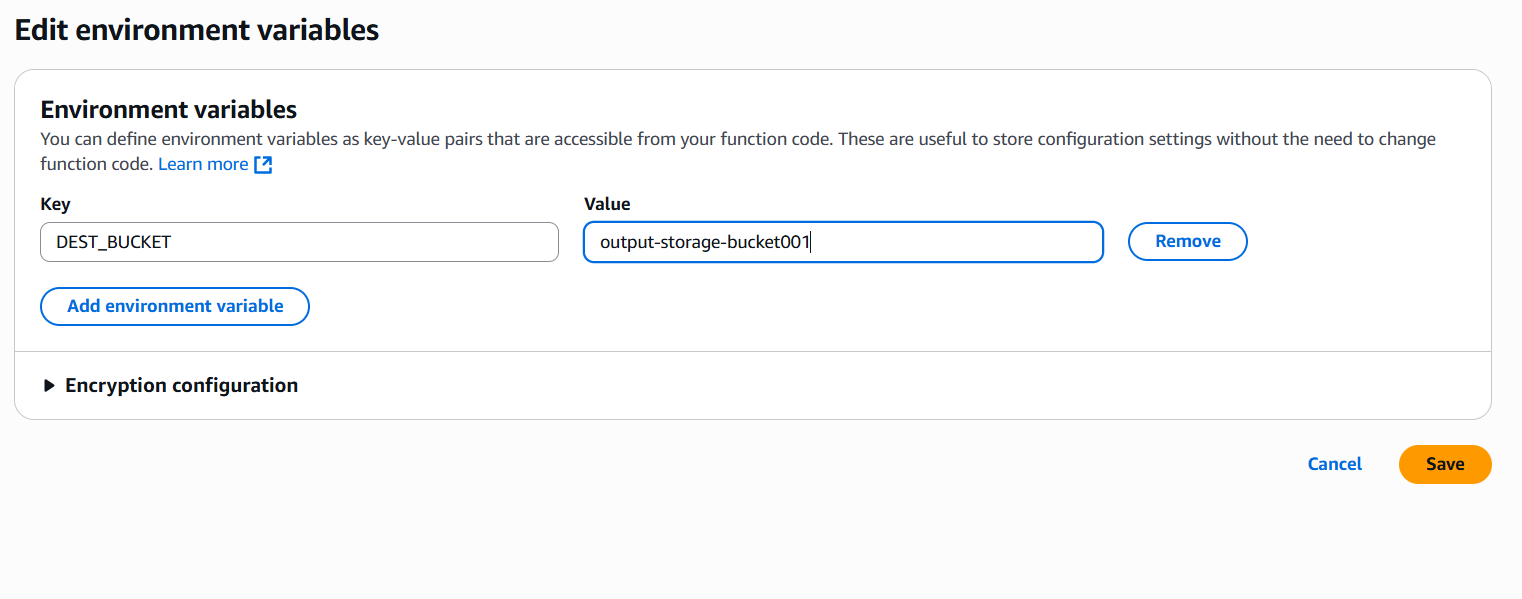
**Task-4) Setting up of lambda function**

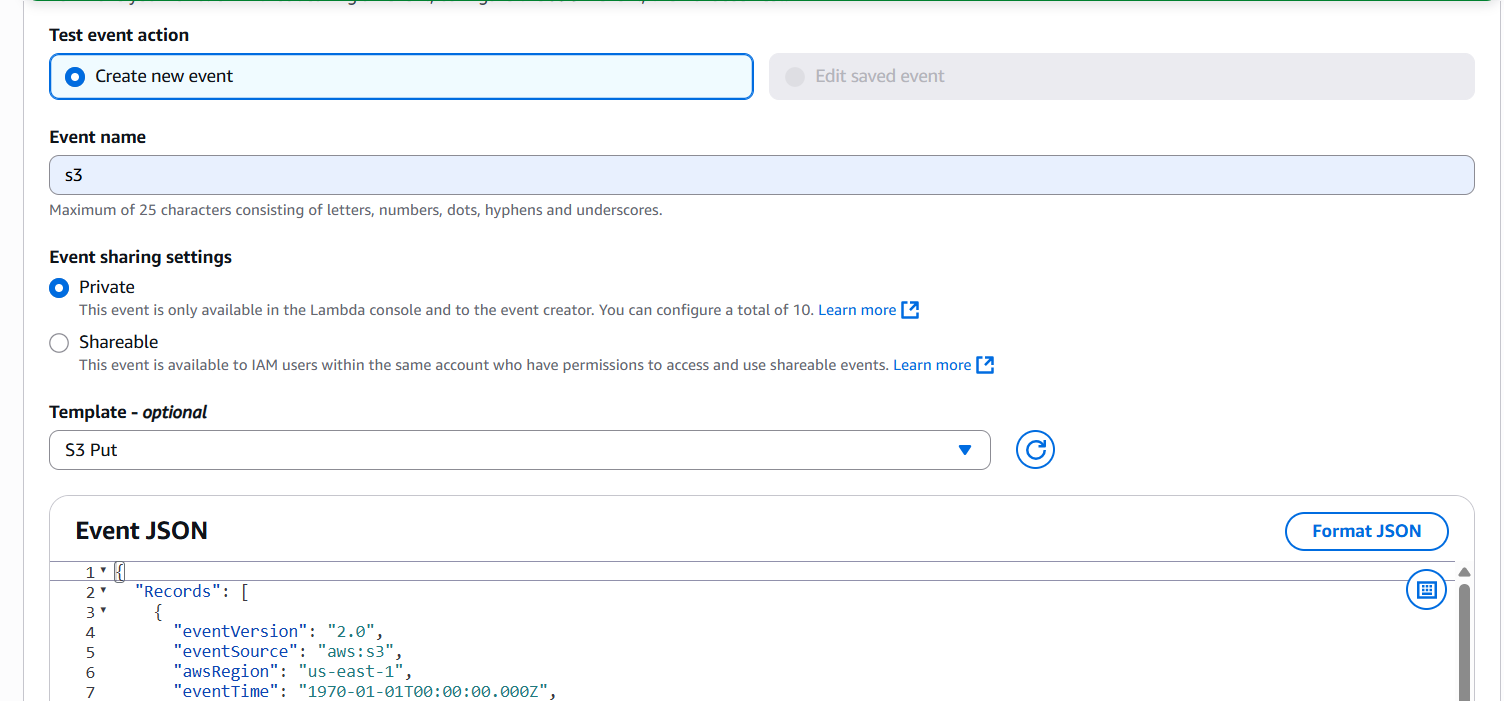
* In lambda function click on create function.
* While creating function leave all the settings in default.
* In change default execution role choose existing role and choose the role which you have created earlier.

****

* Create the function.
* Now your function is created , go to the code source and click upload from.

****

* Now upload the file.
* In configuration click on environmental variables.
* In environmental variables click on edit.
* In key you will write DEST\_BUCKET and in value you have to add the bucket where the resized image is stored.
* Now click on save.
* Now go to test event

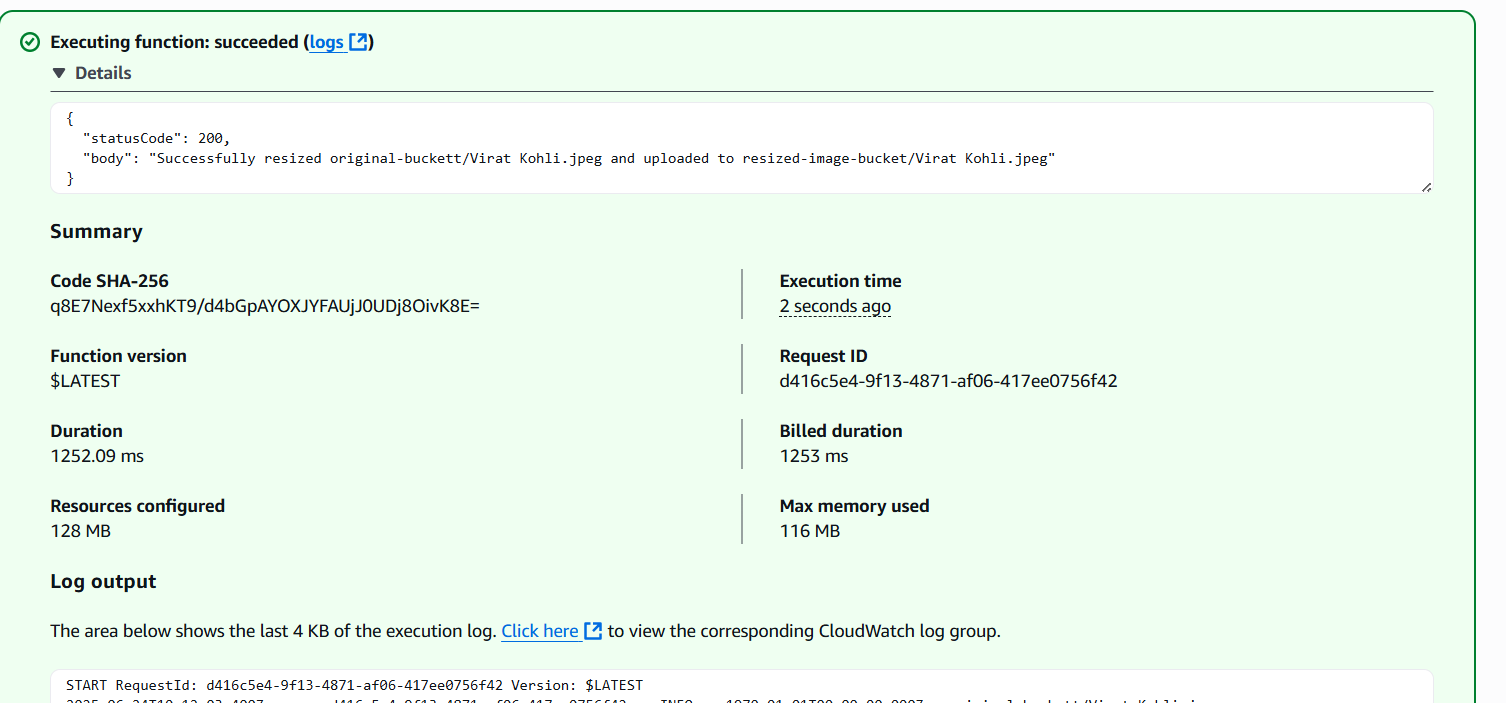


* In this you have to name the event s3.
* And choose the template s3 put.
* In event JSON you have to edit the code and replace the name of bucket with your bucket name where you have uploaded the pic.



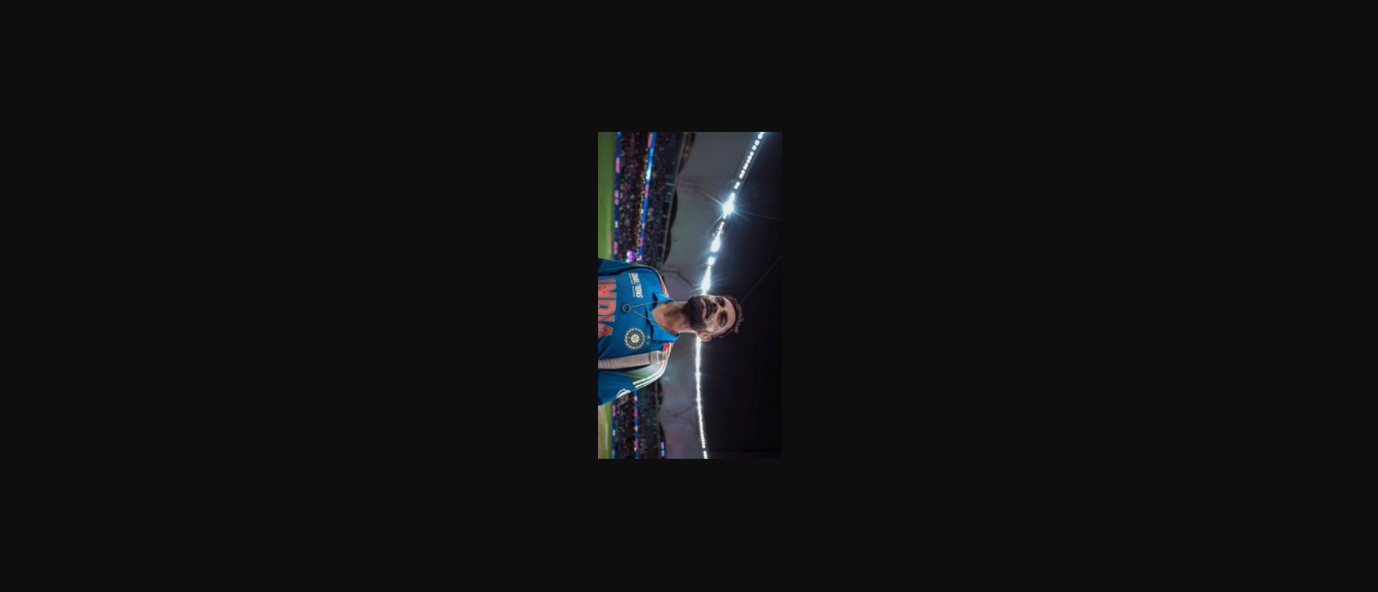


* In key name put name of your pic which you have uploaded in the bucket.
* Now click on test.
* Your image is resized.





This is your original image.



This is your resized image