PROJECT -1

Serverless Image Processing

- Create a serverless image processing application that automatically resizes and optimizes images uploaded to an Amazon S3 bucket.

Introduction

Overview:-

The purpose of this document is to provide a comprehensive guide for creating a serverless image processing application using Amazon Web Services (AWS). This application automatically resizes and optimizes images uploaded to an Amazon S3 bucket. By leveraging the power of AWS Lambda, S3, and other related services, the solution aims to demonstrate the efficiency and scalability of serverless architectures for common image processing task

Objective:-

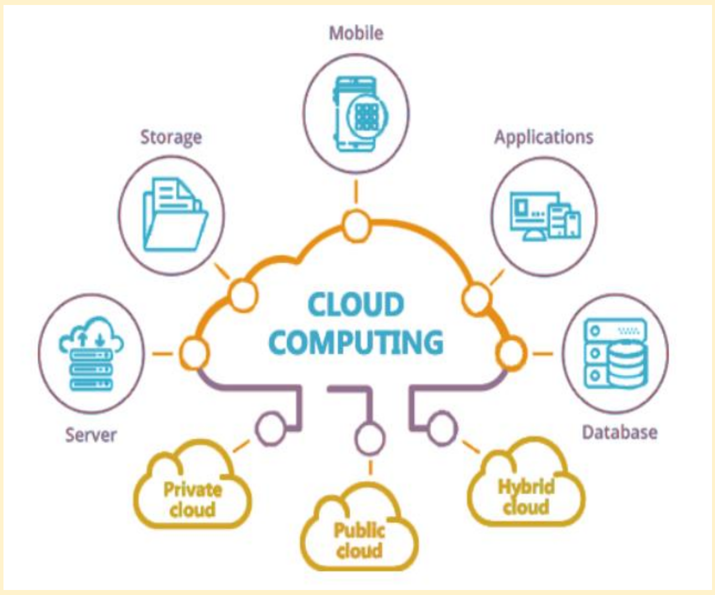
The primary goal of the serverless image processing application is to simplify and automate the handling of image uploads by resizing and optimizing images without the need for managing servers. This involves creating an end-to-end solution where images uploaded to an S3 bucket trigger a Lambda function that processes the images. The processed images are then stored back in the S3 bucket, ready for use in various applications such as websites, mobile apps, and content management systems. Key objectives include:

- Automating the resizing and optimization of images to various dimensions.
- Reducing storage costs by optimizing image sizes.
- Ensuring scalability to handle varying loads without manual intervention.
- Providing a cost-effective solution by leveraging the pay-per-use model of serverless computing

Background

Cloud Computing:-

Cloud computing is a paradigm shift in the IT industry, offering on-demand delivery of compute power, storage, applications, and other IT resources through the internet with a pay-as-you-go pricing model. It enables businesses to avoid upfront infrastructure costs, scale their operations according to demand, and improve their agility and innovation. Major cloud service providers, such as Amazon Web Services (AWS), Microsoft Azure, and Google Cloud Platform (GCP), offer a wide range of services that cater to various computing needs, from data storage and machine learning to content delivery and database management.



Serverless Architecture:-

Serverless architecture represents a further evolution of cloud computing, where the cloud provider dynamically manages the allocation and provisioning of servers. Instead of provisioning, scaling, and managing servers, developers focus solely on writing code. The infrastructure automatically scales up and down based on demand, and users are billed only for the execution time and resources consumed by their code. This model simplifies deployment and reduces operational overhead, making it ideal for microservices, APIs, and event-driven applications. AWS Lambda, Google Cloud Functions, and Azure Functions are leading examples of serverless computing services.

Project introduction

Topic:-Create a serverless image processing application that automatically resizes and optimizes images uploaded to an Amazon S3 bucket.

AWS S3 (Simple Storage Service) is a cloud data storage service. It is one of the most popular services of AWS. It has high scalability, availability, security and is cost effective. S3 has different storage tiers depending on the use case. Some common use cases of AWS S3 are:

1. **Storage:** It can be used for storing large amounts of data.
2. **Backup and Archive:** S3 has different storage tiers based on how frequent the data is accessed which can be used to backup critical data at low costs.
3. **Static website:** S3 offers static website hosting through HTML files stored in S3.
4. **Data lakes and big data analytics:** Companies can use AWS S3 as a data lake and then run analytics on it for getting business insights and take critical decisions.

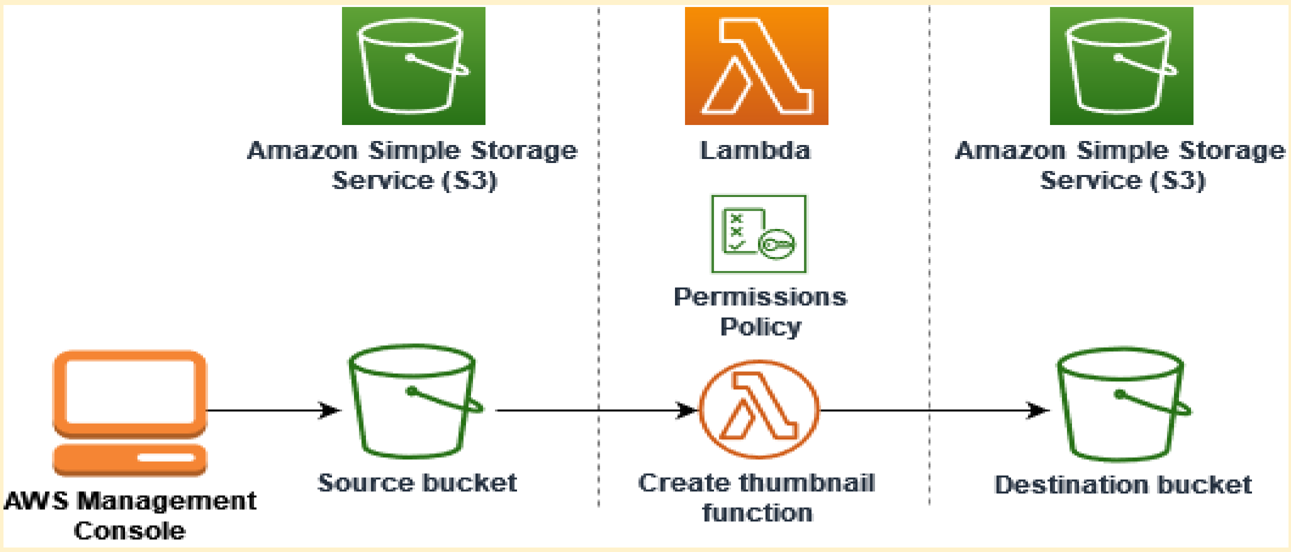
AWS Lambda is a serverless, event-driven compute service that lets you run code for virtually any type of application or backend service without provisioning or managing servers. Lambda functions run on demand i.e. they execute only when needed and you pay only for what you compute. Lambda is well integrated with may other AWS services. It supports a wide variety of programming languages.

Some common use cases for AWS Lambda are:-

1.You can use Lambda for processing files as they are uploaded in an S3 bucket or whenever some event triggers the function.

2. Lambda can also be used for creating websites. This is cost effective because you are charged only for the time when the servers are running.

3. You can pass a data stream to your Lambda function and then create analysis from that.


Amazon Simple Storage Service (S3)
Lambda
Amazon Simple Storage Service (S3)
Permissions Policy
AWS Management Console
Source bucket
Create thumbnail function
Destination bucket

To complete this task, you carry out the following steps:

1. Create source and destination Amazon S3 buckets and upload a sample image.
2. Create a Lambda function that resizes an image and outputs a thumbnail to an Amazon S3 bucket.
3. Configure a Lambda trigger that invokes your function when objects are uploaded to your source bucket.
4. Test your function, first with a dummy event, and then by uploading an image to your source bucket.

Task 1: Sign in to AWS Management Console

1. Console to the IAM Click on the Open Console button, and you will get redirected to AWS Console in a new browser tab.
2. On the AWS sign-in page,

- Leave the Account ID as default. Never edit/remove the 12 digit Account ID present in the AWS Console. otherwise, you cannot proceed with the lab.
- Now copy your User Name and Password in the Lab Username and Password in AWS Console and click on the Sign in button.

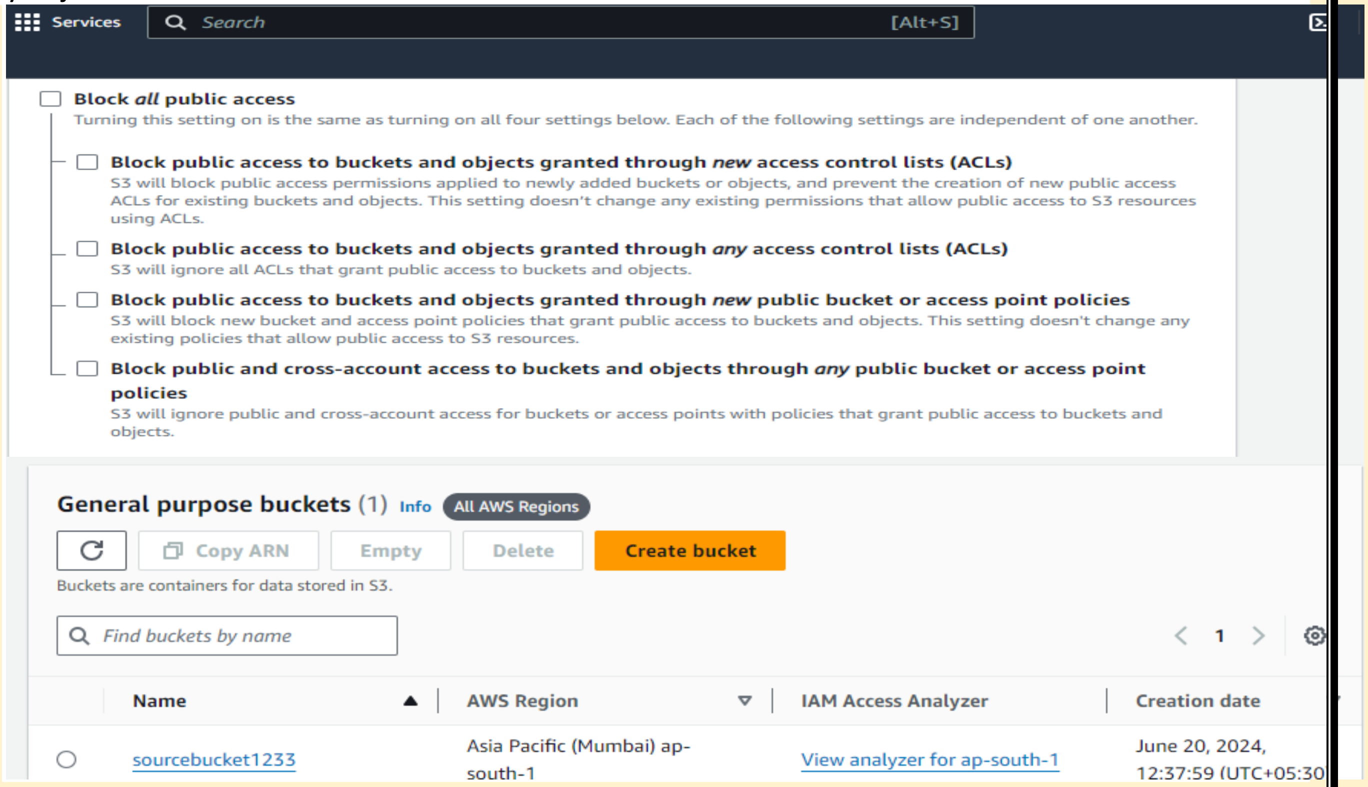
Once Signed In to the AWS Management Console, Make the default AWS Region as US East (N. Virginia) us-east-1.

•

Task 2:-Create two s3 bucket\

To create the Amazon S3 buckets (console)

1. Open the Buckets page of the Amazon S3 console.
2. Choose **Create bucket**.
3. Under **General configuration**, do the following:
   a. For **Bucket name**, enter a globally unique name that meets the Amazon S3 Bucket naming rules. Bucket names can contain only lower case letters, numbers, dots (.), and hyphens (-).
   b. For **AWS Region**, choose the AWS Region closest to your geographical location. Later in the tutorial, you must create your Lambda function in the same AWS Region, so make a note of the region you chose.
4. Leave all other options set to their default values and choose **Create bucket**.
5. Repeat steps 1 to 4 to create your destination bucket. For **Bucket name**, enter **DOC-EXAMPLE-SOURCE-BUCKET-resized**, where **DOC-EXAMPLE-SOURCE-BUCKET** is the name of the source bucket you just created.



☐ **Block *all* public access**
Turning this setting on is the same as turning on all four settings below. Each of the following settings are independent of one another.

☐ **Block public access to buckets and objects granted through *new* access control lists (ACLs)**
S3 will block public access permissions applied to newly added buckets or objects, and prevent the creation of new public access ACLs for existing buckets and objects. This setting doesn't change any existing permissions that allow public access to S3 resources using ACLs.

☐ **Block public access to buckets and objects granted through *any* access control lists (ACLs)**
S3 will ignore all ACLs that grant public access to buckets and objects.

☐ **Block public access to buckets and objects granted through *new* public bucket or access point policies**
S3 will block new bucket and access point policies that grant public access to buckets and objects. This setting doesn't change any existing policies that allow public access to S3 resources.

☐ **Block public and cross-account access to buckets and objects through *any* public bucket or access point policies**
S3 will ignore public and cross-account access for buckets or access points with policies that grant public access to buckets and objects.

General purpose buckets (1) Info All AWS Regions

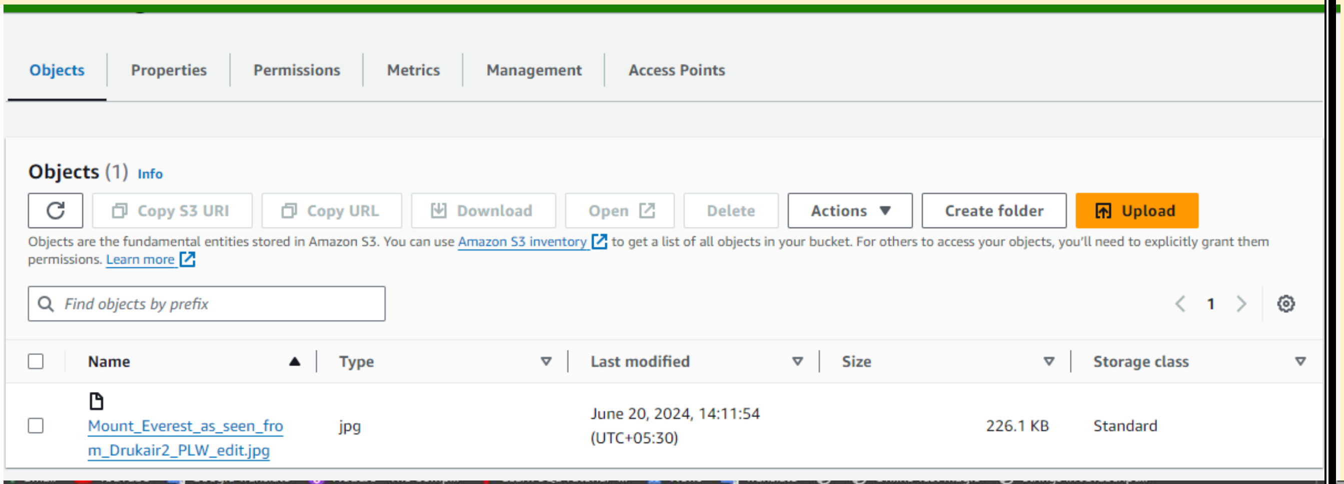
Buckets are containers for data stored in S3.

	Name	AWS Region	IAM Access Analyzer	Creation date
○	sourcebucket1233	Asia Pacific (Mumbai) ap-south-1	View analyzer for ap-south-1	June 20, 2024, 12:37:59 (UTC+05:30

Task 3:-upload a test image to your source bucket

To upload a test image to your source bucket (console)

1. Open the Buckets page of the Amazon S3 console.
2. Select the source bucket you created in the previous step.
3. Choose **Upload**.
4. Choose **Add files** and use the file selector to choose the object you want to upload.
5. Choose **Open**, then choose **Upload**.


Objects | Properties | Permissions | Metrics | Management | Access Points

Objects (1) Info

Copy S3 URI | Copy URL | Download | Open | Delete | Actions | Create folder | Upload

Objects are the fundamental entities stored in Amazon S3. You can use Amazon S3 inventory to get a list of all objects in your bucket. For others to access your objects, you'll need to explicitly grant them permissions. Learn more

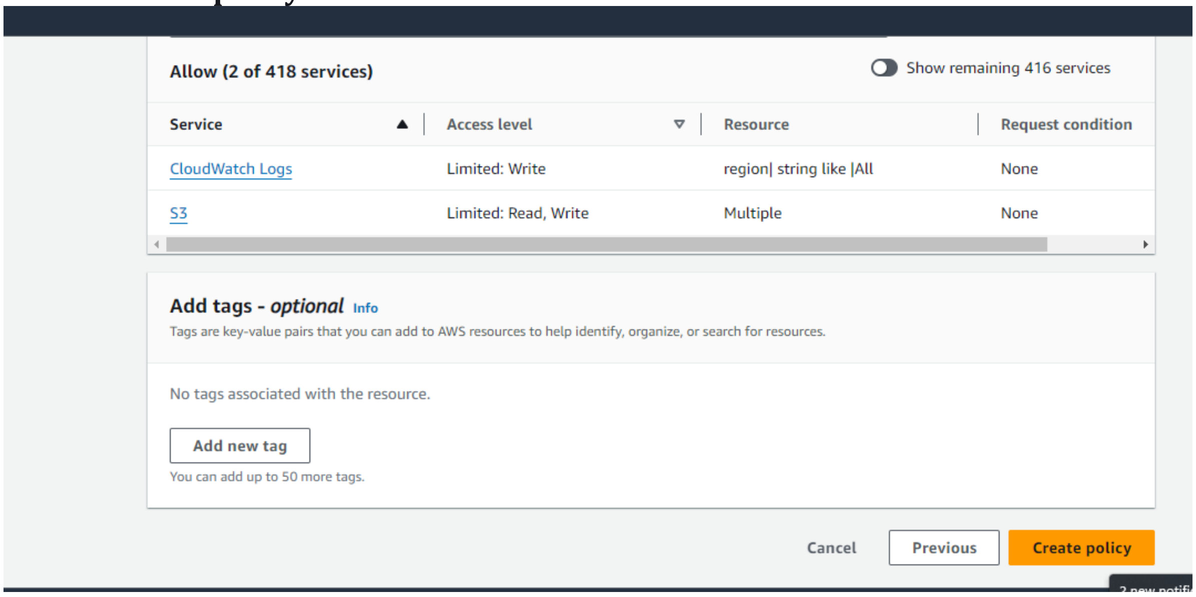
Find objects by prefix

Name	Type	Last modified	Size	Storage class
Mount_Everest_as_seen_from_Drukair2_PLW_edit.jpg	jpg	June 20, 2024, 14:11:54 (UTC+05:30)	226.1 KB	Standard

Task 4:-create the policy

1. Open the Policies page of the AWS Identity and Access Management (IAM) console.
2. Choose **Create policy**.
3. Choose the **JSON** tab, and then paste the following custom policy into the JSON editor.

```
{
 "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/*"
  }
]
```

6. Choose **Next**.
7. Under **Policy details**, for **Policy name**, enter **`LambdaS3Policy`**.
8. Choose **Create policy**
9. 

Allow (2 of 418 services)

Show remaining 416 services

Service	Access level	Resource	Request condition
CloudWatch Logs	Limited: Write	region| string like |All	None
S3	Limited: Read, Write	Multiple	None

Add tags - *optional* Info

Tags are key-value pairs that you can add to AWS resources to help identify, organize, or search for resources.

No tags associated with the resource.

Add new tag

You can add up to 50 more tags.

Cancel | Previous | Create policy

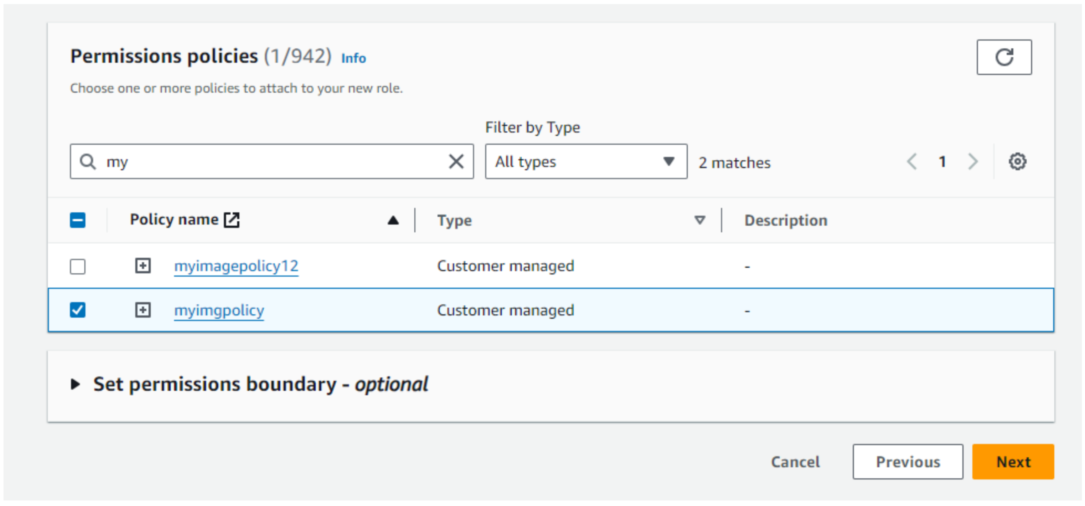
Task 5:-create an execution role

An execution role is an IAM role that grants a Lambda function permission to access AWS services and resources. To give your function read and write access to an Amazon S3 bucket, you attach the permissions policy you created in the previous step.

To create an execution role and attach your permissions policy (console)

1. Open the Roles page of the (IAM) console.
2. Choose **Create role**.
3. For **Trusted entity type**, select **AWS service**, and for **Use case**, select **Lambda**.
4. Choose **Next**.
5. Add the permissions policy you created in the previous step by doing the following:

a. In the policy search box, enter **`LambdaS3Policy`**.
b. In the search results, select the check box for `LambdaS3Policy`.
c. Choose **Next**.

6. Under **Role details**, for the **Role name** enter **`LambdaS3Role`**.
7. Choose **Create role**.
8. 

Permissions policies (1/942) Info
Choose one or more policies to attach to your new role.

Filter by Type: All types — 2 matches

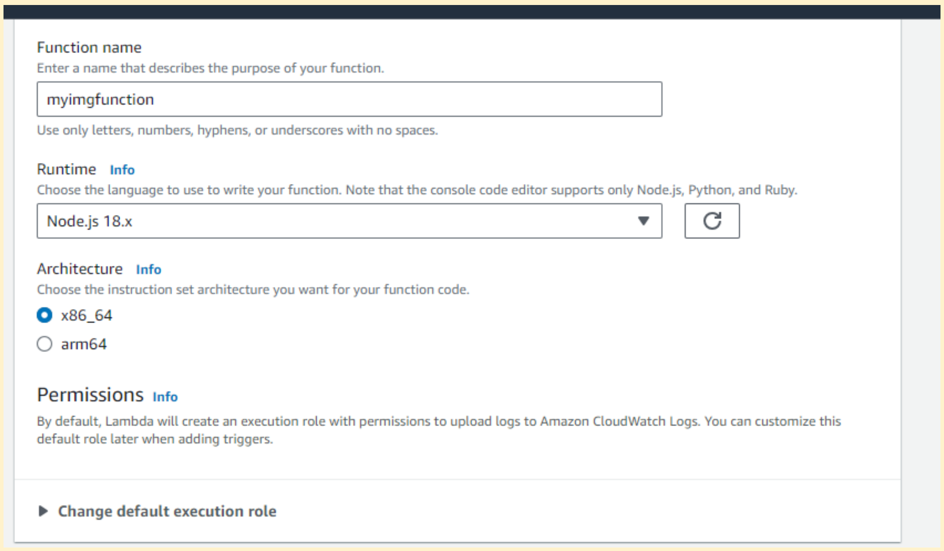
	Policy name	Type	Description
☐	myimagepolicy12	Customer managed	-
☑	myimgpolicy	Customer managed	-

Task 6:-To create the function (console)

To create your Lambda function using the console, you first create a basic function containing some 'Hello world' code. You then replace this code with your own function code by uploading the.zip or JAR file you created in the previous step.

1. Open the Functions page of the Lambda console.
2. Make sure you're working in the same AWS Region you created your Amazon S3 bucket in. You can change your region using the drop-down list at the top of the screen.

1. Choose **Create function**.
2. Choose **Author from scratch**.
3. Under **Basic information**, do the following:
   a. For **Function name**, enter **`CreateThumbnail`**.
   b. For **Runtime**choose either **Node.js 18.x** or **Python 3.9** according to the language you chose for your function.
   c. For **Architecture**, choose **x86_64**.
4. In the **Change default execution role** tab, do the following:
   a. Expand the tab, then choose **Use an existing role**.
   b. Select the `LambdaS3Role` you created earlier.
5. Choose **Create function**.


Function name

Enter a name that describes the purpose of your function.

myimgfunction

Use only letters, numbers, hyphens, or underscores with no spaces.

Runtime Info

Choose the language to use to write your function. Note that the console code editor supports only Node.js, Python, and Ruby.

Node.js 18.x

Architecture Info

Choose the instruction set architecture you want for your function code.

- (●) x86_64
- ( ) arm64

Permissions Info

By default, Lambda will create an execution role with permissions to upload logs to Amazon CloudWatch Logs. You can customize this default role later when adding triggers.

▶ Change default execution role

Task 7:-To upload the function code (console)

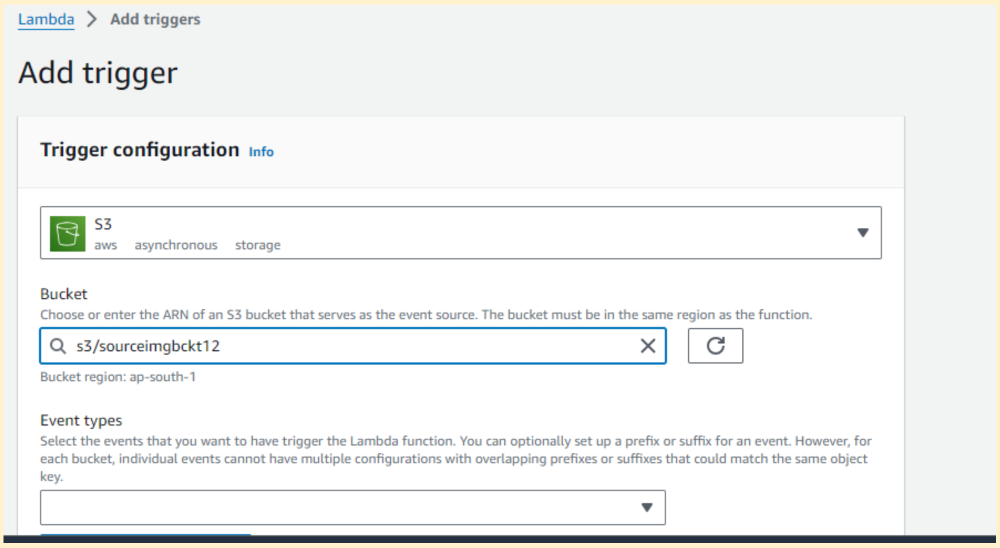
1. In the **Code source** pane, choose **Upload from**.
2. Choose **.zip file**.
3. Choose **Upload**.
4. In the file selector, select your .zip file and choose **Open**.
5. Choose **Save**.

For your Lambda function to run when you upload an image to your source bucket, you need to configure a trigger for your function. You can configure the Amazon S3 trigger using either the console or the AWS CLI.

Task 8:-To configure the Amazon S3 trigger (console)

1. Open the Functions page of the Lambda console and choose your function (`CreateThumbnail`).
2. Choose **Add trigger**.
3. Select **S3**.
4. Under **Bucket**, select your source bucket.
5. Under **Event types**, select **All object create events**.
6. Under **Recursive invocation**, select the check box to acknowledge that using the same Amazon S3 bucket for input and output is not recommended. You can learn more about recursive invocation patterns in Lambda by reading Recursive patterns that cause run-away Lambda functions in Serverless Land.
7. Choose **Add**.

When you create a trigger using the Lambda console, Lambda automatically creates a resource based policy to give the service you select permission to invoke your function.


Lambda > Add triggers

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.

s3/sourceimgbckt12

Bucket region: ap-south-1

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.

Before you test your whole setup by adding an image file to your Amazon S3 source bucket, you test that your Lambda function is working correctly by invoking it with a dummy event. An event in Lambda is a JSON-formatted document that contains data for your function to process. When your function is invoked by Amazon S3, the event sent to your function contains information such as the bucket name, bucket ARN, and object key.

- AWS Management Console
- AWS CLI

To test your Lambda function with a dummy event (console)

1. Open the Functions page of the Lambda console and choose your function (`CreateThumbnail`).
2. Choose the **Test** tab.
3. To create your test event, in the **Test event** pane, do the following:
   a. Under **Test event action**, select **Create new event**.
   b. For **Event name**, enter **`myTestEvent`**.
   c. For **Template**, select **S3 Put**.
   d. Replace the values for the following parameters with your own values.
      - For `awsRegion`, replace `us-east-1` with the AWS Region you created your Amazon S3 buckets in.
      - For `name`, replace `DOC-EXAMPLE-BUCKET` with the name of your own Amazon S3 source bucket.
      - For `key`, replace `test%2Fkey` with the filename of the test object you uploaded to your source bucket in the step Upload a test image to your source bucket.

```
{
  "Records": [
    {
      "eventVersion": "2.0",
      "eventSource": "aws:s3",
      "awsRegion": "us-east-1",
      "eventTime": "1970-01-01T00:00:00.000Z",
      "eventName": "ObjectCreated:Put",
      "userIdentity": {
         "principalId": "EXAMPLE"
      },
      "requestParameters": {
         "sourceIPAddress": "127.0.0.1"
      },
      "responseElements": {
         "x-amz-request-id": "EXAMPLE123456789",
         "x-amz-id-2":
"EXAMPLE123/5678abcdefghijklambdaisawesome/mnopqrstuvwxyzABCDEFGH"
      },
      "s3": {
         "s3SchemaVersion": "1.0",
         "configurationId": "testConfigRule",
         "bucket": {
            "name": "DOC-EXAMPLE-BUCKET",
            "ownerIdentity": {
               "principalId": "EXAMPLE"
            },
            "arn": "arn:aws:s3:::DOC-EXAMPLE-BUCKET"
         },
         "object": {
            "key": "test%2Fkey",
            "size": 1024,
            "eTag": "0123456789abcdef0123456789abcdef",
            "sequencer": "0A1B2C3D4E5F678901"
         }
      }
    }
  ]
```

```
}
```

e. Choose **Save**.
f. In the **Test event** pane, choose **Test**.
g. To check the your function has created a resized verison of your image and stored it in your target Amazon S3 bucket, do the following:
h. Open the Buckets page of the Amazon S3 console.
i. Choose your target bucket and confirm that your resized file is listed in the **Objects** pane.

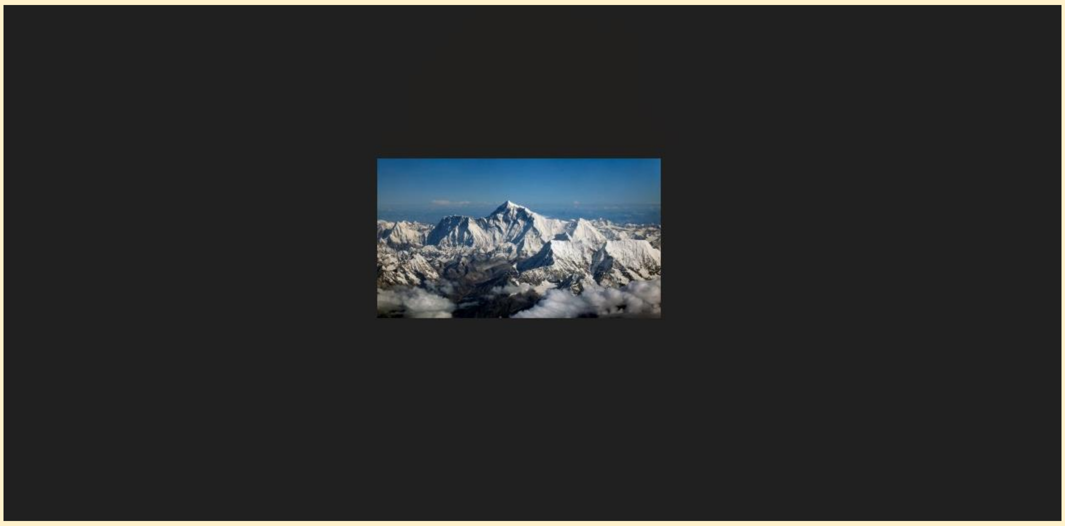
Test 9:-Test your Lambda function with a dummy event

Now that you've confirmed your Lambda function is operating correctly, you're ready to test your complete setup by adding an image file to your Amazon S3 source bucket. When you add your image to the source bucket, your Lambda function should be automatically invoked. Your function creates a resized version of the file and stores it in your target bucket.

- AWS Management Console
- AWS CLI

To test your Lambda function using the Amazon S3 trigger (console)

1. To upload an image to your Amazon S3 bucket, do the following:
    a. Open the Buckets page of the Amazon S3 console and choose your source bucket.
    b. Choose **Upload**.
    c. Choose **Add files** and use the file selector to choose the image file you want to upload. Your image object can be any .jpg or .png file.
    d. Choose **Open**, then choose **Upload**.
2. Verify that Lambda has saved a resized version of your image file in your target bucket by doing the following:
    a. Navigate back to the Buckets page of the Amazon S3 console and choose your destination bucket.
    b. In the **Objects** pane, you should now see two resized image files, one from each test of your Lambda function. To download your resized image, select the file, then choose **Download**.



Delete the resources

You can now delete the resources that you created for this tutorial, unless you want to retain them. By deleting AWS resources that you're no longer using, you prevent unnecessary charges to your AWS account.

To delete the Lambda function

1. Open the Functions page of the Lambda console.
2. Select the function that you created.
3. Choose **Actions**, **Delete**.
4. Type **`delete`** in the text input field and choose **Delete**.

To delete the policy that you created

1. Open the Policies page of the IAM console.
2. Select the policy that you created (**AWSLambdaS3Policy**).
3. Choose **Policy actions**, **Delete**.
4. Choose **Delete**.

To delete the execution role

1. Open the Roles page of the IAM console.
2. Select the execution role that you created.
3. Choose **Delete**.
4. Enter the name of the role in the text input field and choose **Delete**.

To delete the S3 bucket

1. Open the Amazon S3 console.
2. Select the bucket you created.
3. Choose **Delete**.
4. Enter the name of the bucket in the text input field.
5. Choose **Delete bucket**.