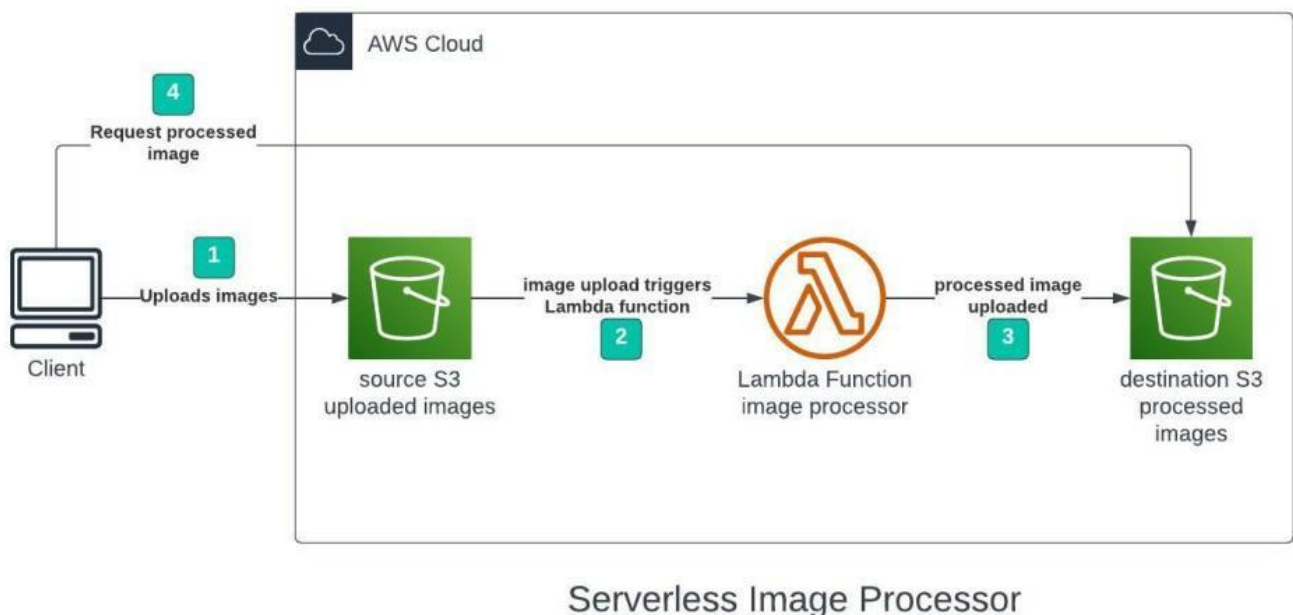


PROJECT-1

Serverless Image Processing

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

❖ The Serverless Image Handler solution helps you embed images on your websites and mobile applications to drive user engagement. It uses the SHARP Node.js library to provide high-speed image processing without sacrificing image quality. To minimize your costs of image optimization, manipulation, and processing, this solution automates version control and provides flexible storage and compute options for file reprocessing.



● LAB STEPS:-

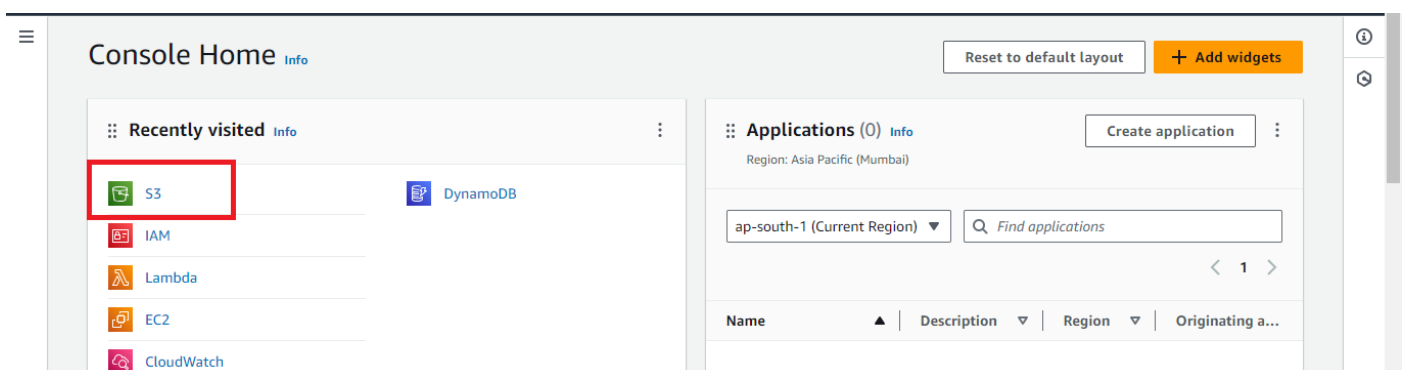
Task 1: Sign in to AWS Management Console

1. 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 Console to the IAM Username and Password in AWS Console and click on the Sign in button.
3. Once Signed In to the AWS Management Console, Make the default AWS Region as US East (N. Virginia) us-east-1.

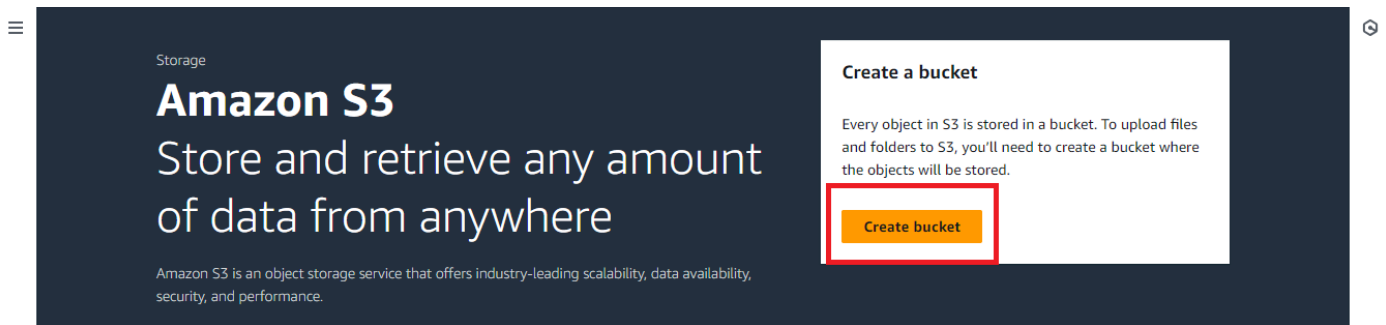
Task 2: Create Two Amazon S3 Buckets

In this task, we will create two AWS S3 buckets i.e the source bucket and the destination bucket by providing the required configurations like name, region etc.

1. Navigate to the **Services** menu in the Top, then click on **S3** in the storage section.



2. Click on Create Bucket button.



3. Create Source Bucket

A screenshot of the 'General configuration' section of the 'Create bucket' page. The 'AWS Region' is set to 'Asia Pacific (Mumbai) ap-south-1'. The 'Bucket name' field is highlighted with a red rectangle and contains the text 'yashsource25'. Below the field, there is a link to 'See rules for bucket naming'. At the bottom, there is a 'Choose bucket' button and a format example: 'Format: s3://bucket/prefix'.

4. Leave Other settings as Default and click on the **Create Bucket** button

5. Once the Bucket is created successfully, Select your S3 bucket.

- Click on the Copy ARN button to copy the ARN.
- Save the source bucket ARN in a text file for later use.
- arn:aws:s3:::yashsource25

6. Create Destination Bucket

General configuration

AWS Region
Asia Pacific (Mumbai) ap-south-1

Bucket name [Info](#)
yashdestination25

Bucket name must be unique within the global namespace and follow the bucket naming rules. [See rules for bucket naming](#)

Copy settings from existing bucket - *optional*
Only the bucket settings in the following configuration are copied.

[Choose bucket](#)

Format: s3://bucket/prefix

7. Leave Other settings as Default and click on the **Create Bucket** button

8. Once the Bucket is created successfully, Select your S3 bucket.

- Click on the Copy ARN button to copy the ARN.
- Save the source bucket ARN in a text file for later use.
- arn:aws:s3:::yashdestination25

General purpose buckets (2) [Info](#) [All AWS Regions](#)

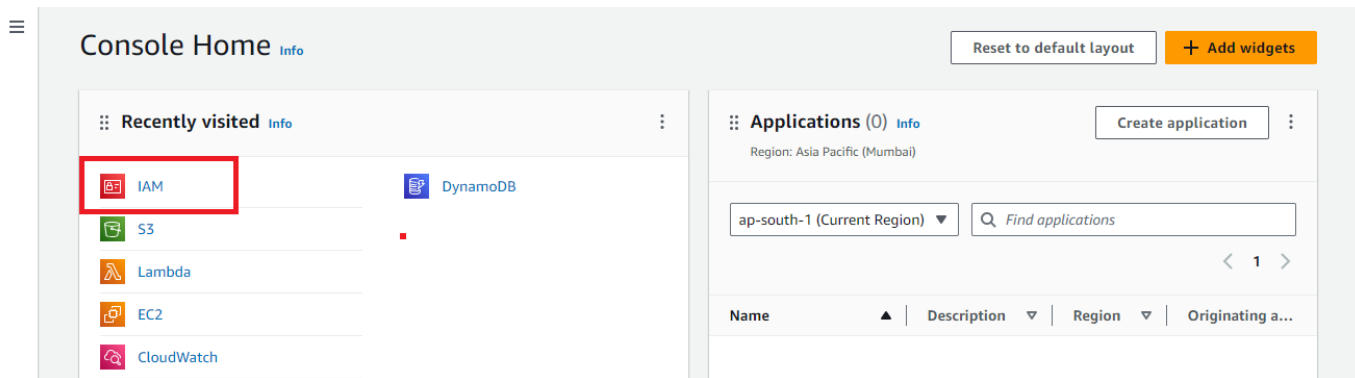
Buckets are containers for data stored in S3.

[Refresh](#) [Copy ARN](#) [Empty](#) [Delete](#) [Create bucket](#)

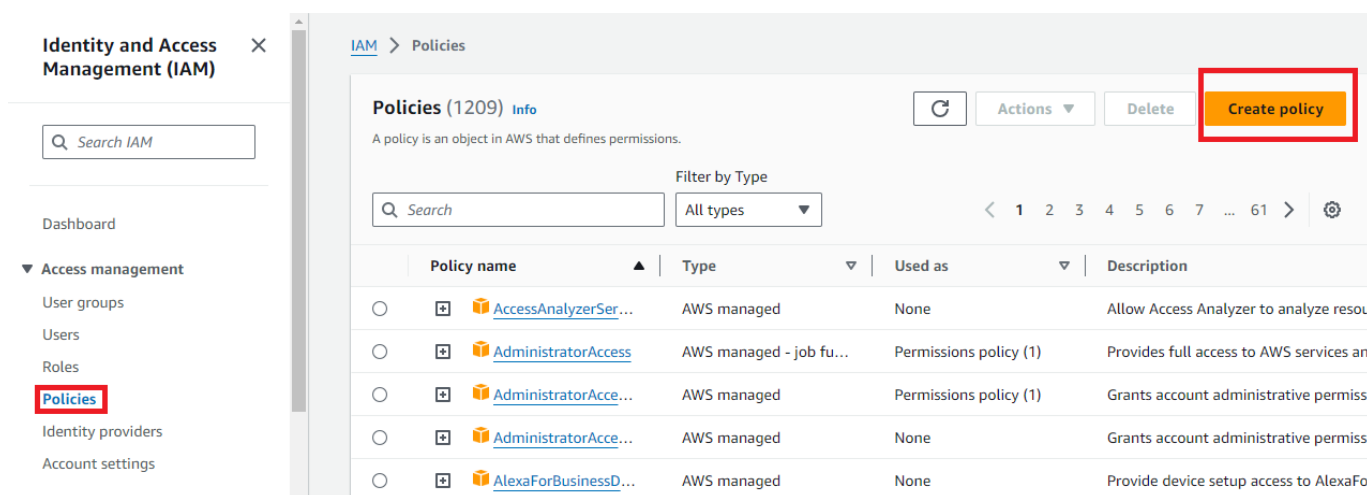
	Name	AWS Region	IAM Access Analyzer	Creation date
<input type="radio"/>	yashdestination25	Asia Pacific (Mumbai) ap-south-1	View analyzer for ap-south-1	June 18, 2024, 21:10:25 (UTC+05:30)
<input type="radio"/>	yashsource25	Asia Pacific (Mumbai) ap-south-1	View analyzer for ap-south-1	June 18, 2024, 21:07:38 (UTC+05:30)

Task 3: Create an IAM Policy

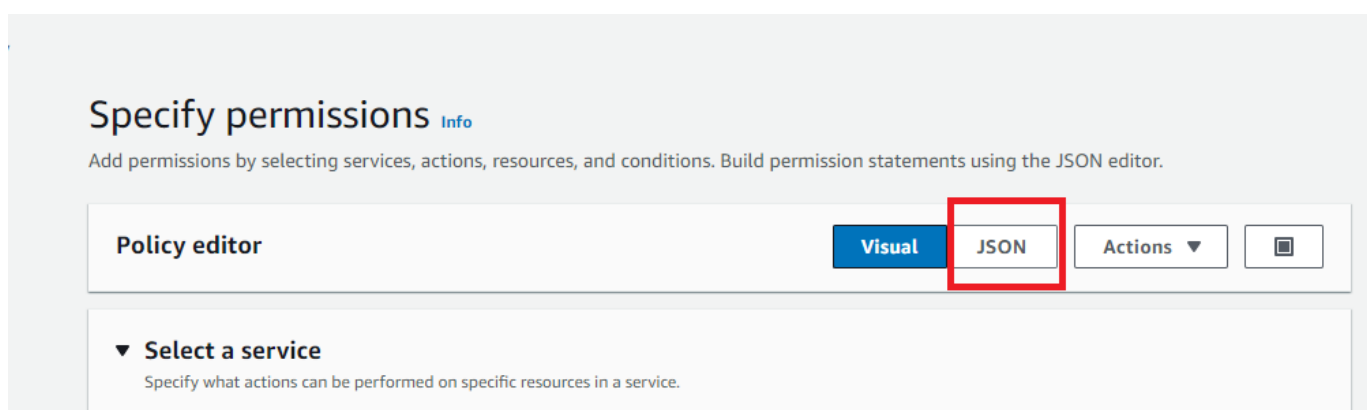
1. Go to **Services** and Select **IAM** under **Security, Identity and Compliance**.



2. Click on **Policies** in the left navigation bar and click on the **Create policy** button.



3. 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": [
        "logs:PutLogEvents",
        "logs:CreateLogGroup",
        "logs:CreateLogStream"
      ],
      "Resource": "arn:aws:logs:*:*:*"
    },
    {
      "Effect": "Allow",
      "Action": ["s3:GetObject"],
      "Resource": "arn:aws:s3:::yashsource25/*"
    },
    {
      "Effect": "Allow",
      "Action": ["s3:PutObject"],
      "Resource": "arn:aws:s3:::yashdestination25/*"
    }
  ]
}
```

4. Leave Everything as default and click on **Next** button.

5. On the Review Policy page:

6. Enter **Policy Name** and Click on the **Create policy** button

The screenshot shows the 'Review and create' step of the AWS IAM console. The breadcrumb trail is 'IAM > Policies > Create policy'. The left sidebar shows 'Step 1: Specify permissions' and 'Step 2: Review and create'. The main heading is 'Review and create' with an 'Info' link. Below the heading is the instruction 'Review the permissions, specify details, and tags.' The 'Policy details' section contains a 'Policy name' field with the instruction 'Enter a meaningful name to identify this policy.' and a red arrow pointing to it. Below the field is the text 'Maximum 128 characters. Use alphanumeric and '+=,.-_@' characters.' The 'Add tags - optional' section shows 'No tags associated with the resource.' and an 'Add new tag' button. At the bottom right, the 'Create policy' button is highlighted with a red box, along with 'Cancel' and 'Previous' buttons.

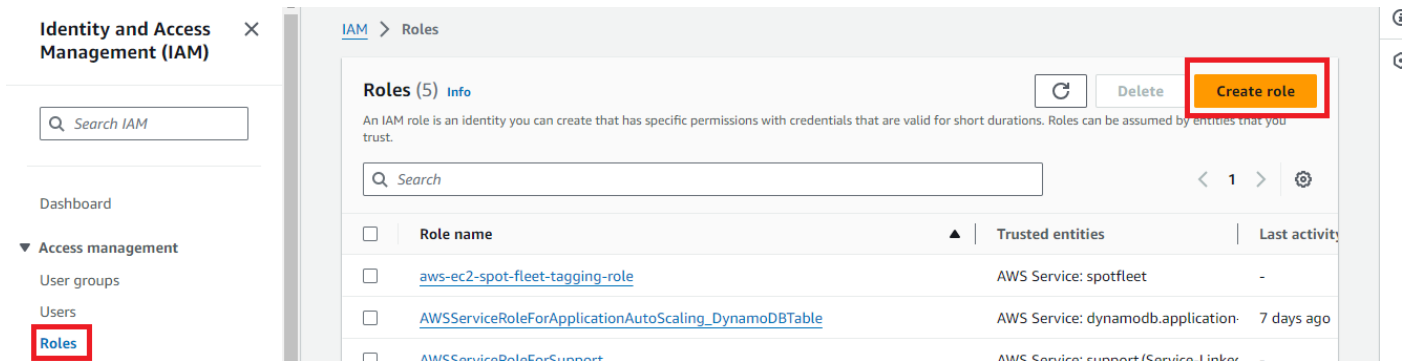
7. An **IAM** Policy with the name **yashpolicy** is created.

The screenshot shows the 'Policies' list page in the AWS IAM console. The breadcrumb trail is 'IAM > Policies'. The heading is 'Policies (1210)' with an 'Info' link. Below the heading is the instruction 'A policy is an object in AWS that defines permissions.' The top right has buttons for 'Actions', 'Delete', and 'Create policy'. The search bar contains 'yashp' and the 'Filter by Type' dropdown is set to 'All types'. The table below shows the results of the search.

Policy name	Type	Used as	Description
yashpolicy	Customer managed	None	-

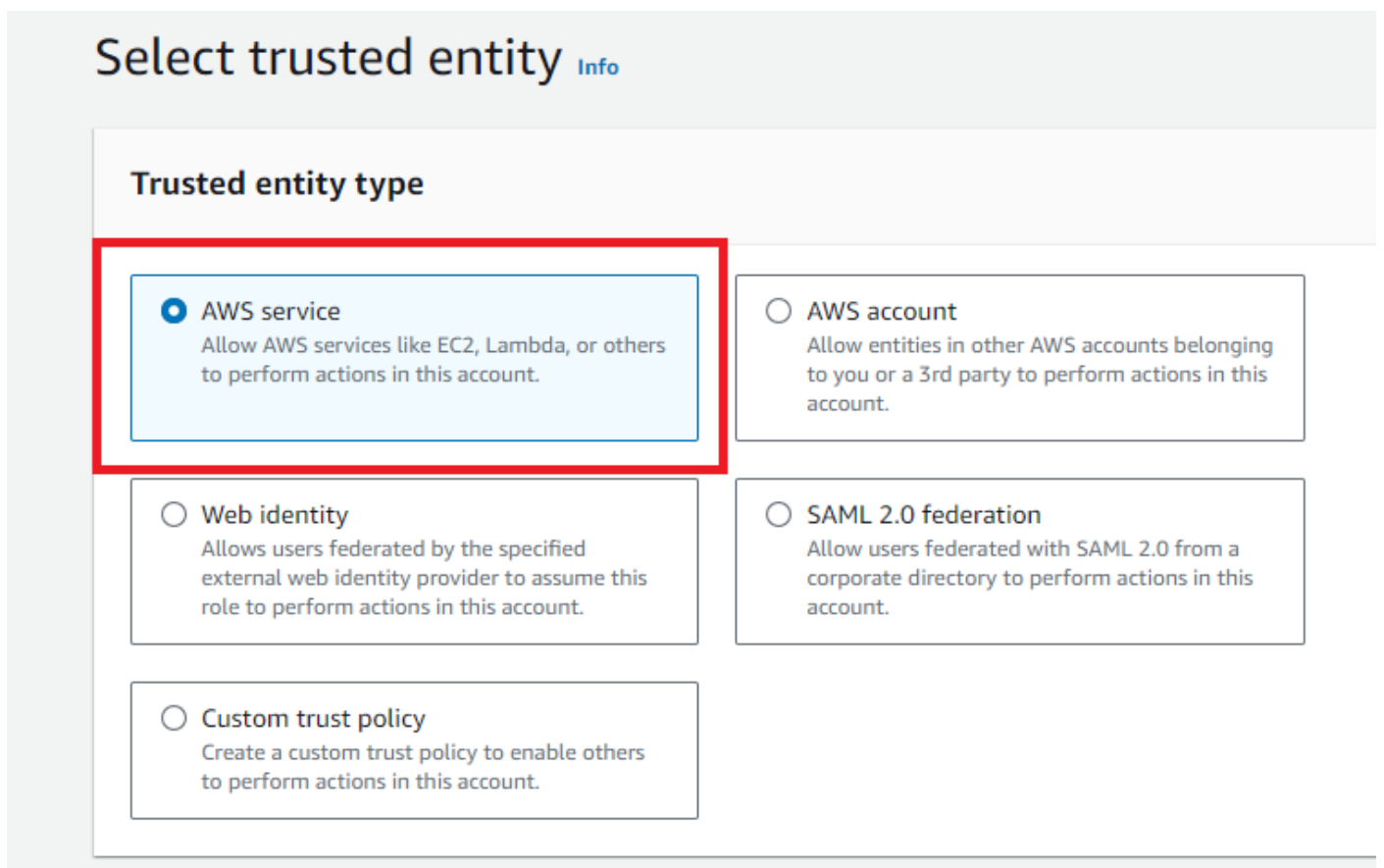
Task 4: Create an IAM Role

1. In the left menu, click on **Roles** and 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 Next button.



Use case
Allow an AWS service like EC2, Lambda, or others to perform actions in this account.

Service or use case

Lambda

Choose a use case for the specified service.

Use case

☒ Lambda
Allows Lambda functions to call AWS services on your behalf.

Cancel **Next**

3. Select your **policy** and click on the **Next** button.

Add permissions [Info](#)

Permissions policies (1/937) [Info](#)

Choose one or more policies to attach to your new role.

Filter by Type

Q yash X All types 2 matches < 1 > ⚙

	Policy name ↗	Type	Description
<input type="checkbox"/>	s3crr_for_yash2507_ac330b	Customer managed	-
<input checked="" type="checkbox"/>	yashpolicy	Customer managed	-

► Set permissions boundary - optional

Cancel Previous **Next**

4. **Role Name:** Enter **cloudrole**

5. Click on the **Create Role** button.

- You have successfully created an IAM role by name cloudrole.

Roles (6) [Info](#)

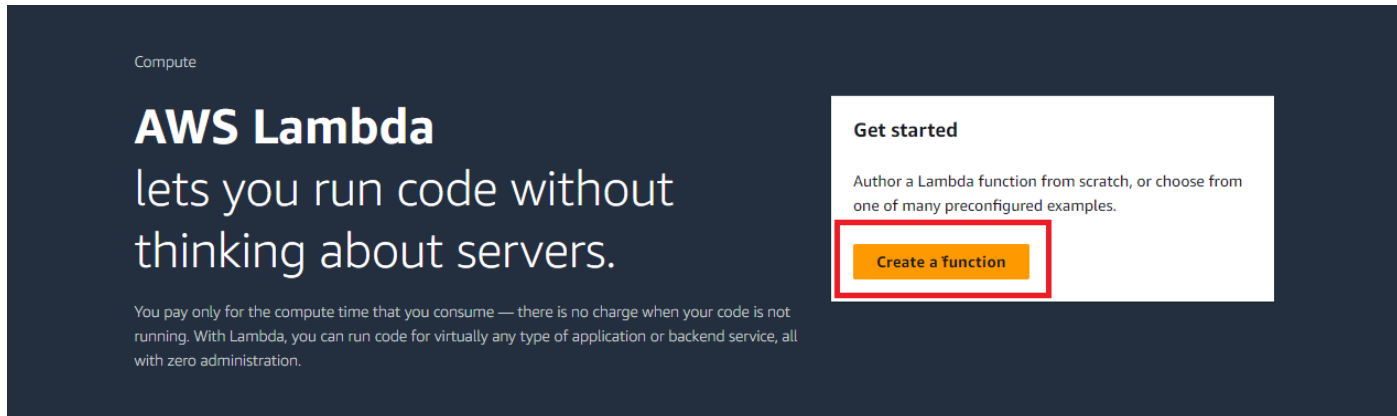
An IAM role is an identity you can create that has specific permissions with credentials that are valid for short durations. Roles can be assumed by entities that you trust.

Q cloudrole X 1 match < 1 > ⚙

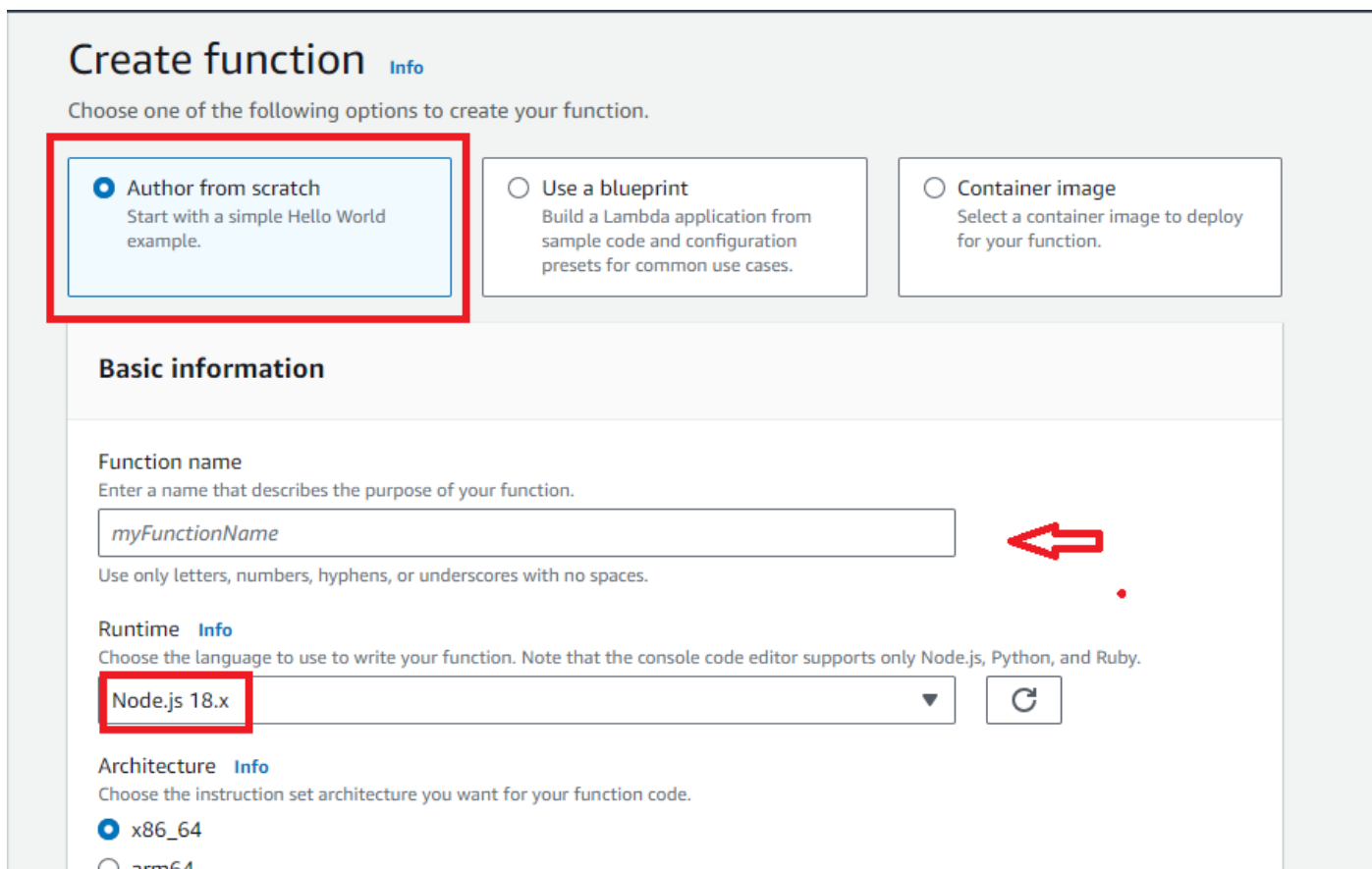
	Role name	Trusted entities	Last activity
<input type="checkbox"/>	cloudrole	AWS Service: lambda	-

Task 5: Creating Lambda function

1. Go to AWS Lambda Console, Navigate to functions section . Click **Create function**



2. Name it and select runtime and Leave all other settings as default.

A screenshot of the 'Create function' console in the AWS Lambda service. The page title is 'Create function' with an 'Info' link. Below the title, it says 'Choose one of the following options to create your function.' There are three radio button options: 'Author from scratch' (selected and highlighted with a red box), 'Use a blueprint', and 'Container image'. Below these is the 'Basic information' section. It has a 'Function name' field with the placeholder 'myFunctionName' and a red arrow pointing to it. Below the name field is a note: 'Use only letters, numbers, hyphens, or underscores with no spaces.' The 'Runtime' section has a dropdown menu with 'Node.js 18.x' selected and highlighted with a red box. To the right of the dropdown is a refresh icon. The 'Architecture' section has two radio button options: 'x86_64' (selected) and 'arm64'.

3. Change Default execution role and create function

▼ Change default execution 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.

[View the cloudrole role](#) on the IAM console.

► Advanced settings

Cancel **Create function**

4. Edit Environment Variables

Code | Test | Monitor | **Configuration** | Aliases | Versions

General configuration

Triggers

Permissions

Destinations

Function URL

Environment variables

Tags

General configuration [Info](#)

Description	Memory
-	128 MB
Timeout	SnapStart Info
0 min 3 sec	None

☰ [Lambda](#) > [Functions](#) > [yash_lambda-function](#) > Edit environment variables

Edit environment variables

Environment variables

You can define environment variables as key-value pairs that are accessible from your function code. These are useful to store configuration settings without the need to change function code. [Learn more](#) [🔗](#)

Key	Value	
<input type="text" value="DEST_BUCKET"/>	<input type="text" value="yashdestination25"/>	<input type="button" value="Remove"/>

▶ Encryption configuration

5. Upload Zip file in Lambda function

▼ **Function overview** [Info](#)

yash_function25

Layers (0)

Description

-

Last modified

7 minutes ago

Function ARN

[arn:aws:lambda:ap-south-1:905418447105:function:yash_function25](#)

Function URL [Info](#)

-

[Code](#) [Test](#) [Monitor](#) [Configuration](#) [Aliases](#) [Versions](#)

Code source [Info](#)

▲ File Edit Find View Go Tools Window 🔍 Go to Anything (Ctrl-P) index.mjs Environment Var × + 🗪 ⚙️

***Zip file link-**

<https://github.com/OneLightWebDev/image-resizer-lambda>

Task 6: Test Lambda Function

*Go to AWS Lambda console. Navigate to Functions section.

*open function then will be created

*open test console

*template=s3-put

The screenshot shows the 'Test event' interface in the AWS Lambda console. At the top, there are 'Save' and 'Test' buttons. Below this, a message states: 'To invoke your function without saving an event, configure the JSON event, then choose Test.' The 'Test event action' section has two options: 'Create new event' (selected) and 'Edit saved event'. The 'Event name' field contains 'cloudevent' with a note: 'Maximum of 25 characters consisting of letters, numbers, dots, hyphens and underscores.' The 'Event sharing settings' section has two radio buttons: 'Private' (selected) and 'Shareable'. Below 'Private' is a note: 'This event is only available in the Lambda console and to the event creator. You can configure a total of 10. [Learn more](#)'. Below 'Shareable' is a note: 'This event is available to IAM users within the same account who have permissions to access and use shareable events. [Learn more](#)'. The 'Template - optional' dropdown menu is set to 's3-put'.

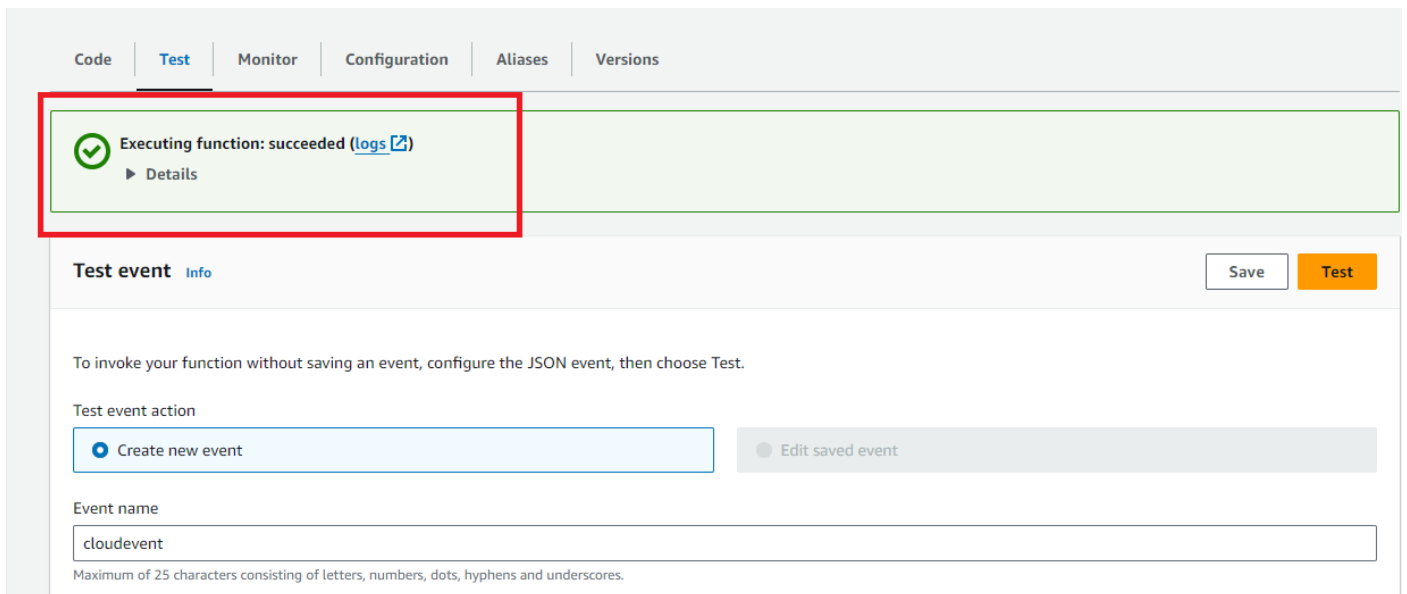
EVENT JSON:

```
{  
  "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":  
"EXAMPLE123/5678abcdefghijklmbdaisawesome/mnopqrstuvwxyzAB  
CDEFGH"  
},  
"s3": {  
  "s3SchemaVersion": "1.0",  
  "configurationId": "testConfigRule",  
  "bucket": {  
    "name": "yashsource25",  
    "ownerIdentity": {  
      "principalId": "EXAMPLE"  
    },  
    "arn": "arn:aws:s3:::yashsource25"  
  },  
}
```

```
"object": {  
  "key": "18981044.jpg",  
  "size": 1024,  
  "eTag": "0123456789abcdef0123456789abcdef",  
  "sequencer": "0A1B2C3D4E5F678901" } } } ] }
```

Now We can Test:



The screenshot shows the AWS Lambda console interface. At the top, there are tabs for 'Code', 'Test', 'Monitor', 'Configuration', 'Aliases', and 'Versions'. The 'Test' tab is selected. Below the tabs, a green notification box with a checkmark icon states 'Executing function: succeeded' with a link to 'logs'. Below this, there is a 'Test event' section with an 'Info' link. It includes a 'Save' button and a 'Test' button. The 'Test event action' section has two options: 'Create new event' (selected) and 'Edit saved event'. The 'Event name' field contains the text 'cloudevent'. A note at the bottom states: 'Maximum of 25 characters consisting of letters, numbers, dots, hyphens and underscores.'

Task 7: Creating S3 Trigger

- *Add trigger
- *Select s3
- *choose source Bucket name
- *Now Add

yash_lambda-function

Throttle

Copy ARN

Actions ▼


▼ Function overview [Info](#)


Export to Application Composer

Download ▼

Diagram

Template

 yash_lambda-function

 Layers (0)

+ Add trigger

+ Add destination


Description

-

Last modified

6 minutes ago

Function ARN

 arn:aws:lambda:ap-south-1:905418447105:function:yash_lambda-function


Function URL [Info](#)

-



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.



Q s3/yashsource25



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.

All object create events ✕

Prefix - optional

Enter a single optional prefix to limit the notifications to objects with keys that start with matching characters.

e.g. images/

Task 8: Upload image in Source Bucket



[Amazon S3](#) > [Buckets](#) > yashsource25

yashsource25 [Info](#)

[Objects](#) | [Properties](#) | [Permissions](#) | [Metrics](#) | [Manage](#)

Objects (2) [Info](#) [Refresh](#) [Copy S3 URI](#)

Objects are the fundamental entities stored in Amazon S3. You can use [Amazon S3 im](#)

<input type="checkbox"/>	Name	Type
<input type="checkbox"/>	 mountain.jpeg	jpeg
<input checked="" type="checkbox"/>	 tiger.jpg	jpg

Original Image



Destination Bucket

Amazon S3 > Buckets > yashdestination25

yashdestination25

Info

Objects

Properties

Permissions

Metrics

Management

Access Points

Objects (2)

Info

Refresh

Copy S3 URI

Copy URL

Download

Open

Delete

Actions

Create folder

Upload

Find objects by prefix

<

1

>

Settings

	Name	Type	Last modified	Size	Storage class
<input type="checkbox"/>	mountain.jpeg	jpeg	June 19, 2024, 22:50:49 (UTC+05:30)	1.8 MB	Standard
<input checked="" type="checkbox"/>	tiger.jpg	jpg	June 19, 2024, 22:50:50 (UTC+05:30)	86.8 KB	Standard

Resize Image



