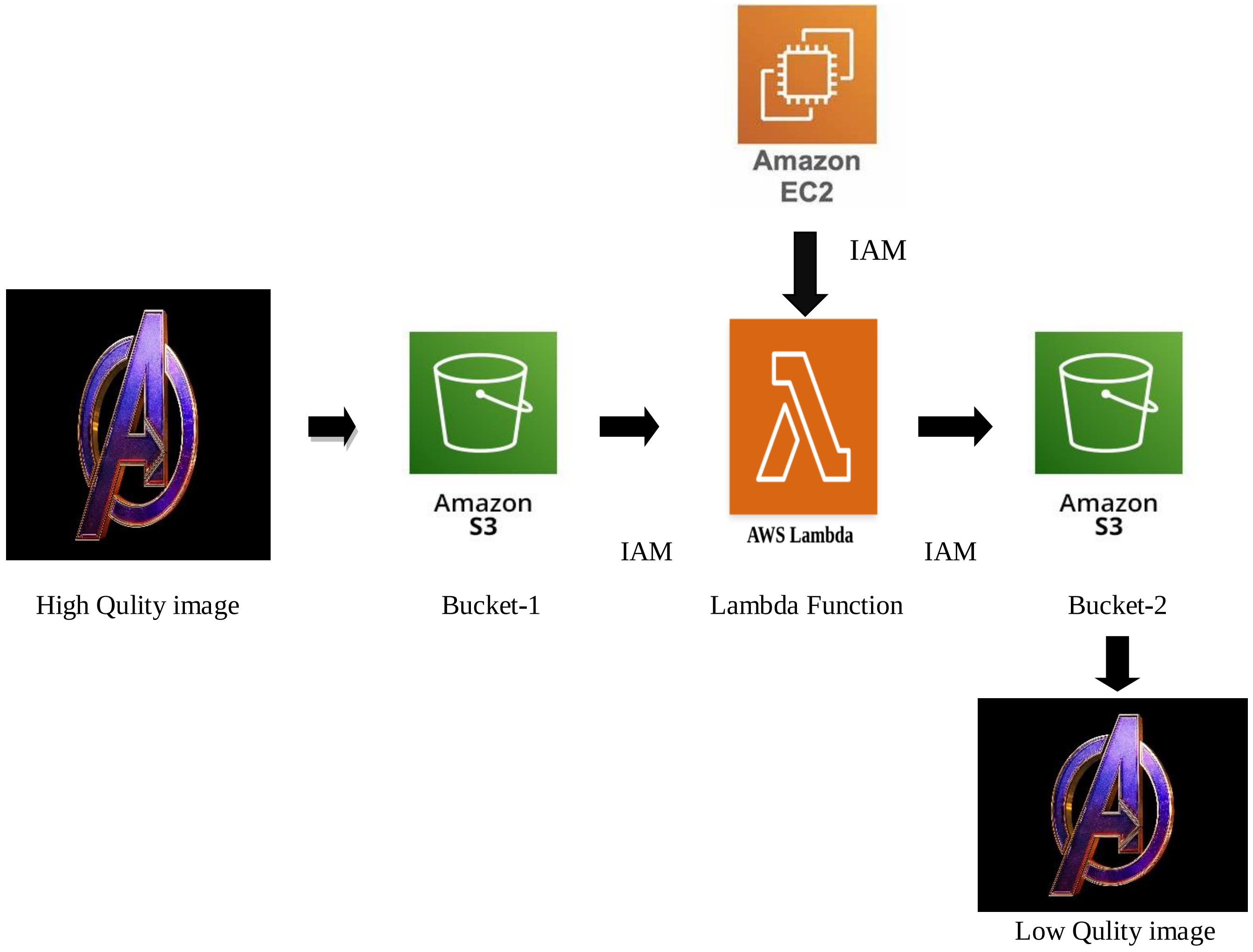
Image Compression Project

Aim : To create a system that automatically compresses images uploaded to an S3 bucket using AWS Lambda, saving the compressed images in another S3 bucket.

Using AWS Lambda for image compression is highly beneficial across various industries due to improved performance, cost savings, and enhanced user experience.

Block Diagram :



Note : We should perform this project step by step. There are many steps included .

In this pdf I will not show how to create instance or s3 I will only be showing you the process.

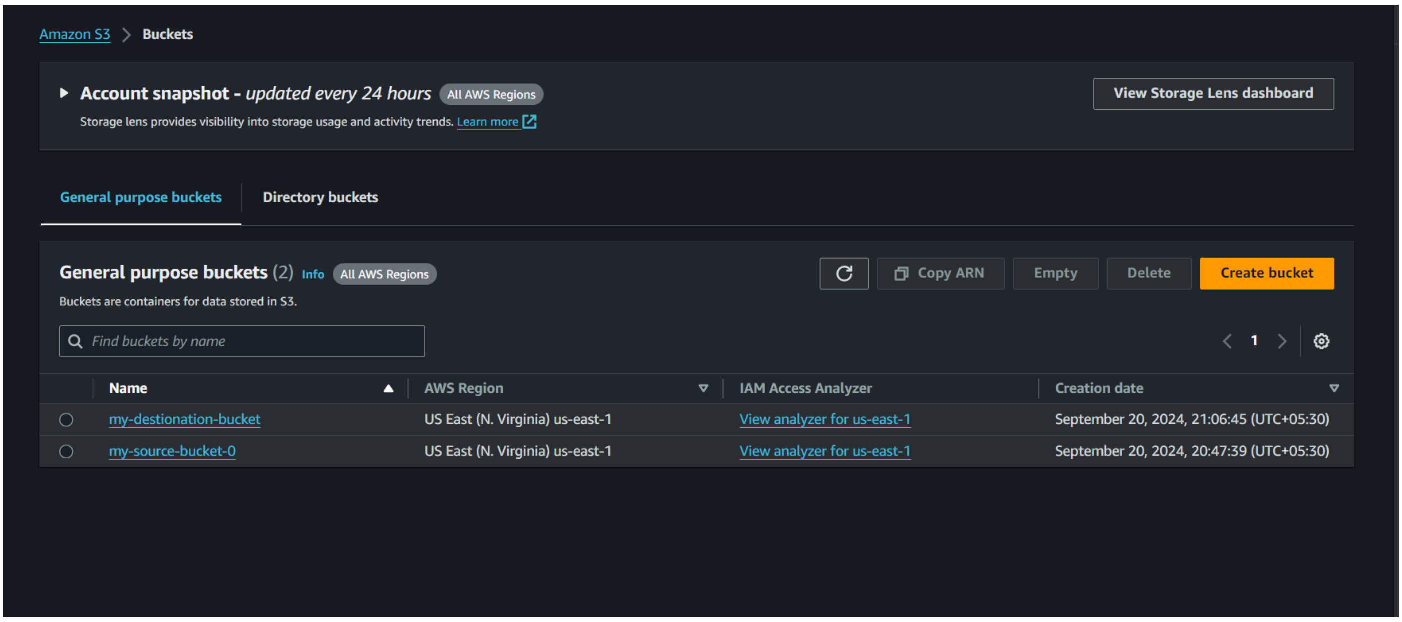
Requirements :

- Two S3 buckets (source and destination)
- Two roles (role-1 {s3 to lambda and lambda to s3}full access)
  (role-2 Ec2 to lambda full access)
- One Lambda function add layer to lambda function

Step-by-Step Instructions :

Step-1 : Login with your AWS account and navigate to s3

Create two s3 buckets



Storage lens provides visibility into storage usage and activity trends. Learn more

General purpose buckets (2) Info All AWS Regions

Buckets are containers for data stored in S3.

Find buckets by name

	Name	AWS Region	IAM Access Analyzer	Creation date
○	my-destionation-bucket	US East (N. Virginia) us-east-1	View analyzer for us-east-1	September 20, 2024, 21:06:45 (UTC+05:30)
○	my-source-bucket-0	US East (N. Virginia) us-east-1	View analyzer for us-east-1	September 20, 2024, 20:47:39 (UTC+05:30)

As you can see I have two buckets

Source : my-source-bucket-0

Destination : my-destionation-bucket (wrong spelling for unique name)

Step-2 : Now lets create lambda function

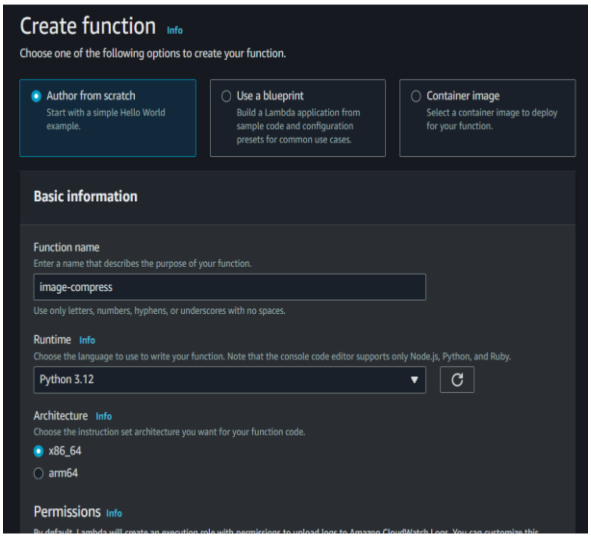
Select : Author from scratch

Name : image-compress

Runtime : Python 3.12

Architecture : x86_64

Don’t do anything just click on create function .



Create function Info

Choose one of the following options to create your function.

● Author from scratch
Start with a simple Hello World example.

○ Use a blueprint
Build a Lambda application from sample code and configuration presets for common use cases.

○ Container image
Select a container image to deploy for your function.

Basic information

Function name
Enter a name that describes the purpose of your function.
image-compress
Use only letters, numbers, hyphens, or underscores with no spaces.

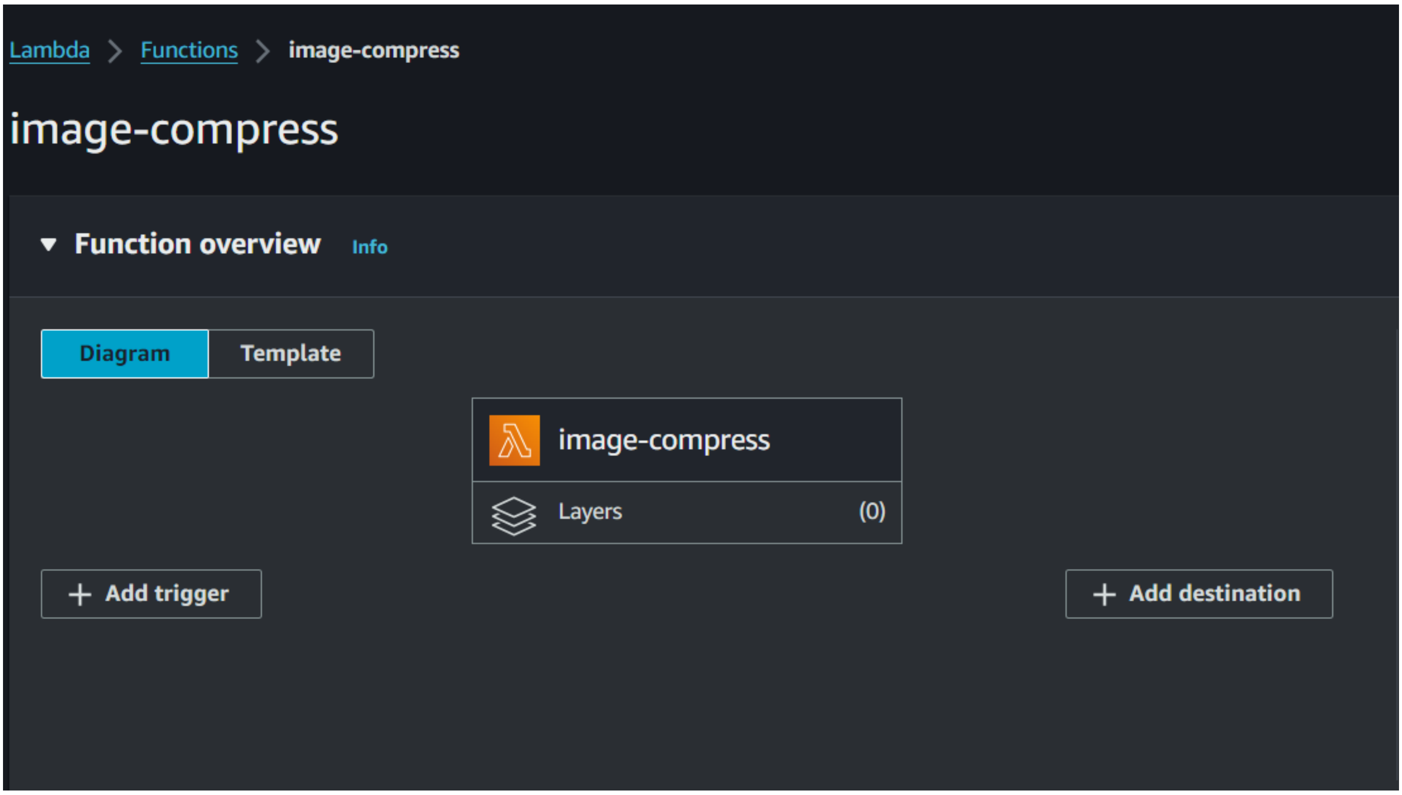
Runtime Info
Choose the language to use to write your function. Note that the console code editor supports only Node.js, Python, and Ruby.
Python 3.12

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

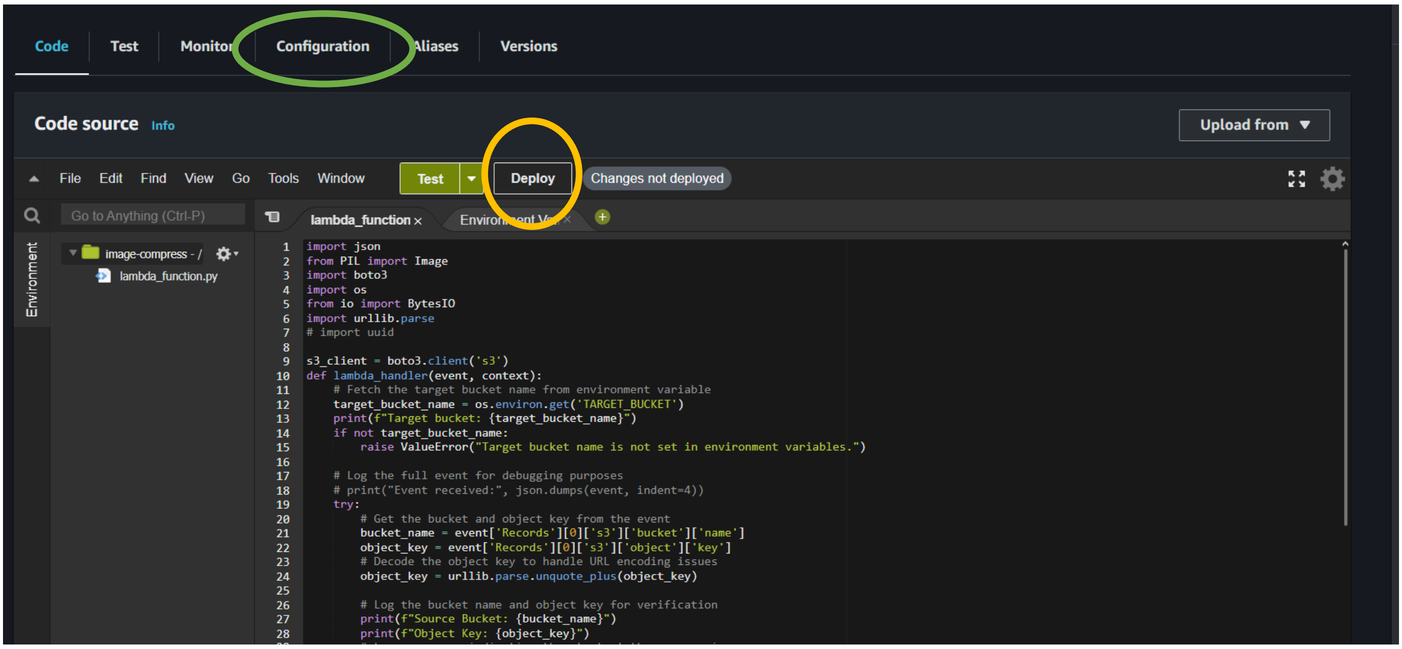
As you can see lambda function is created .



Now for the function we need to have function code.

```
import json
from PIL import Image
import boto3
import os
from io import BytesIO
import urllib.parse
# import uuid

s3_client = boto3.client('s3')
def lambda_handler(event, context):
    # Fetch the target bucket name from environment variable
    target_bucket_name = os.environ.get('TARGET_BUCKET')
    print(f"Target bucket: {target_bucket_name}")
    if not target_bucket_name:
        raise ValueError("Target bucket name is not set in environment variables.")

    # Log the full event for debugging purposes
    # print("Event received:", json.dumps(event, indent=4))
    try:
        # Get the bucket and object key from the event
        bucket_name = event['Records'][0]['s3']['bucket']['name']
        object_key = event['Records'][0]['s3']['object']['key']
        # Decode the object key to handle URL encoding issues
        object_key = urllib.parse.unquote_plus(object_key)
        # Log the bucket name and object key for verification
        print(f"Source Bucket: {bucket_name}")
        print(f"Object Key: {object_key}")
        # Log a message indicating the start of the compression process
        print(f"Starting compression for object: {object_key}")
        if not object_key.startswith("resized_"):
            # Fetch the image from the S3 bucket
            print(f"Fetching object: {object_key} from bucket: {bucket_name}")
            s3_response = s3_client.get_object(Bucket=bucket_name, Key=object_key)
            image_data = s3_response['Body'].read()
            # Log that image data has been successfully fetched
            print(f"Image data fetched for object: {object_key}")
            # Resize the image
            image = Image.open(BytesIO(image_data))
            width, height = image.size
            resized_image = image.resize((width // 2, height // 2))
            # Log the image resize process
            print(f"Resized image from {width}x{height} to {width // 2}x{height // 2}")
            # Save resized image to buffer
            output_buffer = BytesIO()
            resized_image.save(output_buffer, format='PNG')
            output_buffer.seek(0)
            # Log the size of the buffer
            print(f"Size of resized image buffer: {output_buffer.getbuffer().nbytes}")
            # Create a unique key to avoid collisions
            # new_object_key = f"resized_{uuid.uuid4()}_{object_key}"
            new_object_key = f"resized_{object_key}"
            # Upload the resized image to the target bucket
            print(f"Uploading resized image to {target_bucket_name}/{new_object_key}")
            response = s3_client.put_object(Bucket=target_bucket_name, Key=new_object_key,
Body=output_buffer)
            # Log the response from the put_object call
            print(f"PutObject response: {response}")
        else:
            print(f"Object {object_key} is already resized. Skipping processing.")
    except s3_client.exceptions.NoSuchKey as e:
        print(f"Error: The object key '{object_key}' does not exist in the bucket '{bucket_name}'")
        raise e
    except Exception as e:
        print(f"Error processing object {object_key} from bucket {bucket_name}: {str(e)}")
        raise e
    return {
        'statusCode': 200,
        'body': json.dumps('Compression Complete!')
    }
```

As you can see I have copied and pasted the code . now deploy it

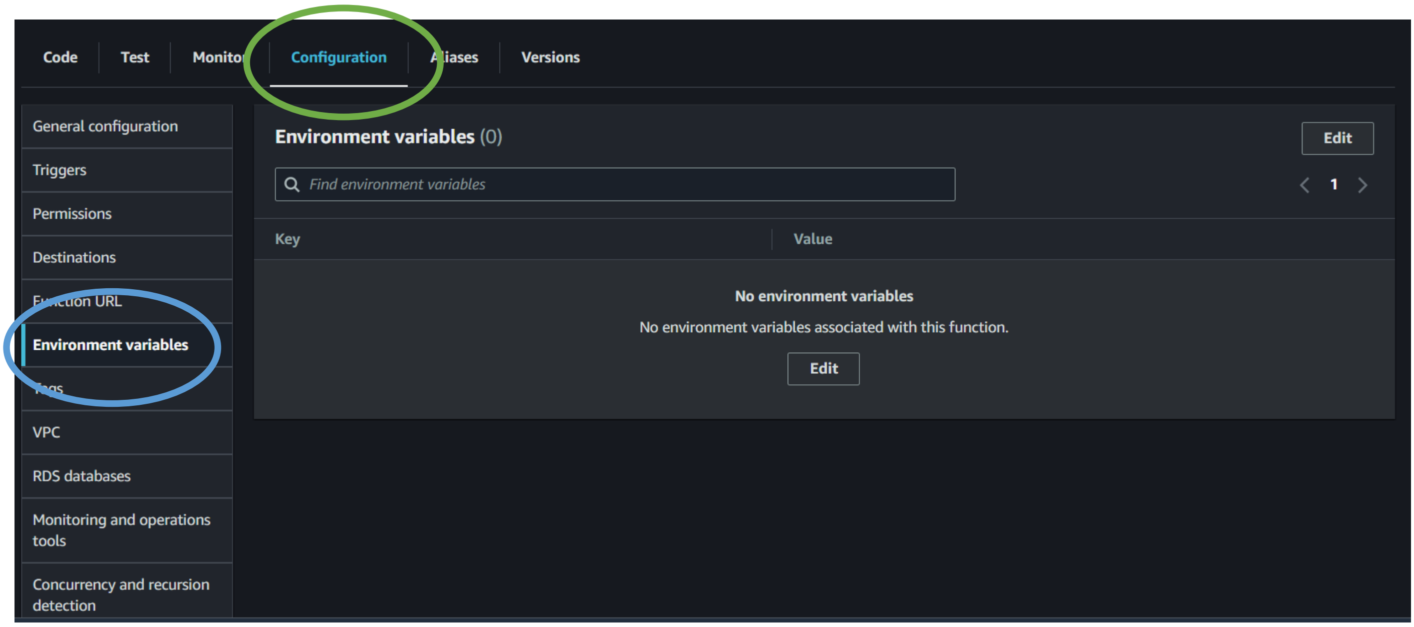


```python
import json
from PIL import Image
import boto3
import os
from io import BytesIO
import urllib.parse
# import uuid

s3_client = boto3.client('s3')
def lambda_handler(event, context):
    # Fetch the target bucket name from environment variable
    target_bucket_name = os.environ.get('TARGET_BUCKET')
    print(f"Target bucket: {target_bucket_name}")
    if not target_bucket_name:
        raise ValueError("Target bucket name is not set in environment variables.")

    # Log the full event for debugging purposes
    # print("Event received:", json.dumps(event, indent=4))
    try:
        # Get the bucket and object key from the event
        bucket_name = event['Records'][0]['s3']['bucket']['name']
        object_key = event['Records'][0]['s3']['object']['key']
        # Decode the object key to handle URL encoding issues
        object_key = urllib.parse.unquote_plus(object_key)

        # Log the bucket name and object key for verification
        print(f"Source Bucket: {bucket_name}")
        print(f"Object Key: {object_key}")
```

Now if you read carefully the above code . its asking target_bucket_name

So this is the reason we need to create **Environment variables**

To set Environment variables
go to configuration it in green circled .

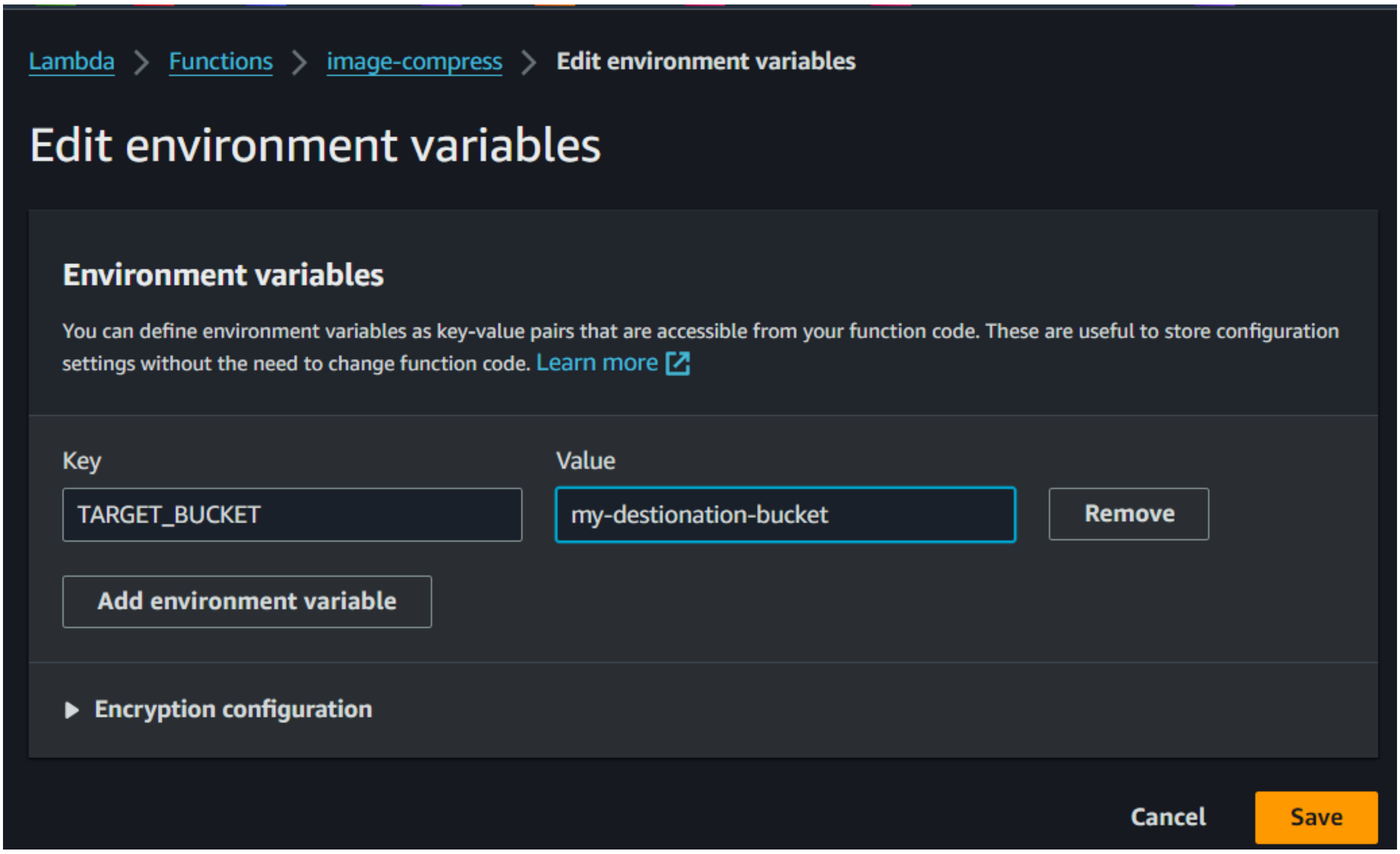


Environment variables (0)

No environment variables

No environment variables associated with this function.

Click on edit



Lambda > Functions > image-compress > 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: TARGET_BUCKET

Value: my-destionation-bucket

Remove
Add environment variable
▶ Encryption configuration
Cancel
Save

IN **key** write it as TARGET_BUCKET

IN Value write your destination Bucket name : my-destionation-bucket

Save the changes

Step-3 : Lets start creating layers for my lambda function

What does layers do ?

AWS Lambda layers let you keep your code and libraries separate. This makes it easier to reuse code, manage libraries, and keep your function size smaller. You can also update libraries without changing your main code .

To create layers we have to install Python and pip for that we are using Ec2

Use the below commands in Ec2

#installing pip

```
sudo dnf install python3-pip
```

#creating Dir

```
mkdir -p lambda-layer/python
```

# Entering into the Dir

```
cd lambda-layer/python
```

#downloading pillow files

pip3 install --platform manylinux2014_x86_64 --target . --python-version 3.12 --only-binary=:all: Pillow

#exit from the python Dir

cd ..

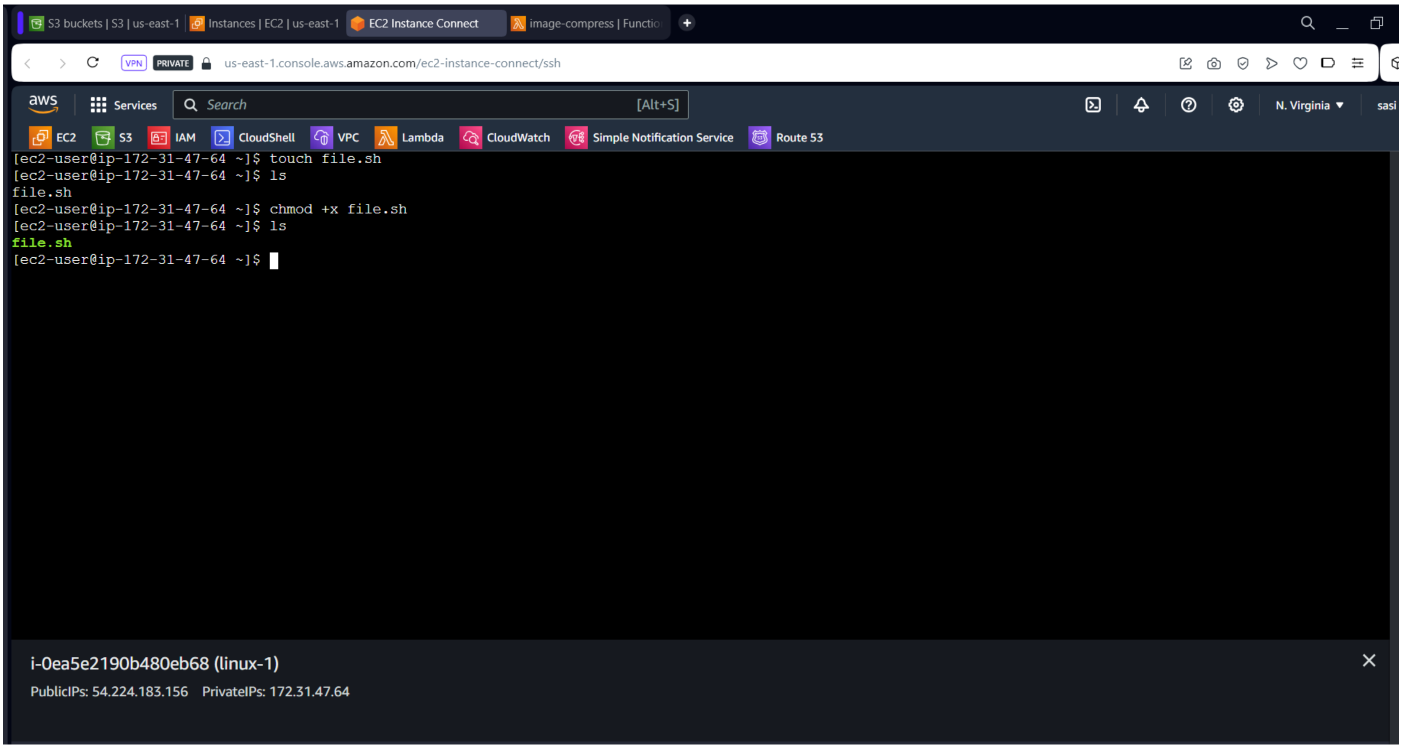
#creating zip file from python Dir

zip -r layer.zip python

#pushing the zip file lambda-function Layer

aws lambda publish-layer-version --layer-name pillow-layer --zip-file fileb://layer.zip --compatible-runtimes python3.12 --region <region-name>

Now if you don't want to waste time just create shell script and the above commands to script please change your region-name


S3 buckets | S3 | us-east-1
Instances | EC2 | us-east-1
EC2 Instance Connect
image-compress | Functio
us-east-1.console.aws.amazon.com/ec2-instance-connect/ssh
Services
Search
[Alt+S]
N. Virginia
EC2
S3
IAM
CloudShell
VPC
Lambda
CloudWatch
Simple Notification Service
Route 53

```
[ec2-user@ip-172-31-47-64 ~]$ touch file.sh
[ec2-user@ip-172-31-47-64 ~]$ ls
file.sh
[ec2-user@ip-172-31-47-64 ~]$ chmod +x file.sh
[ec2-user@ip-172-31-47-64 ~]$ ls
file.sh
[ec2-user@ip-172-31-47-64 ~]$
```

i-0ea5e2190b480eb68 (linux-1)

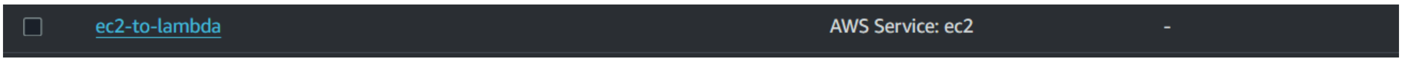
PublicIPs: 54.224.183.156   PrivateIPs: 172.31.47.64

Now before Running the file we need to assign the role for ec2 to access lambda.

Step-4 : Creating roles and attaching policies

i. Go to IAM create role. Create 1 role

ii. Ec2 to lambda



□	ec2-to-lambda	AWS Service: ec2	-

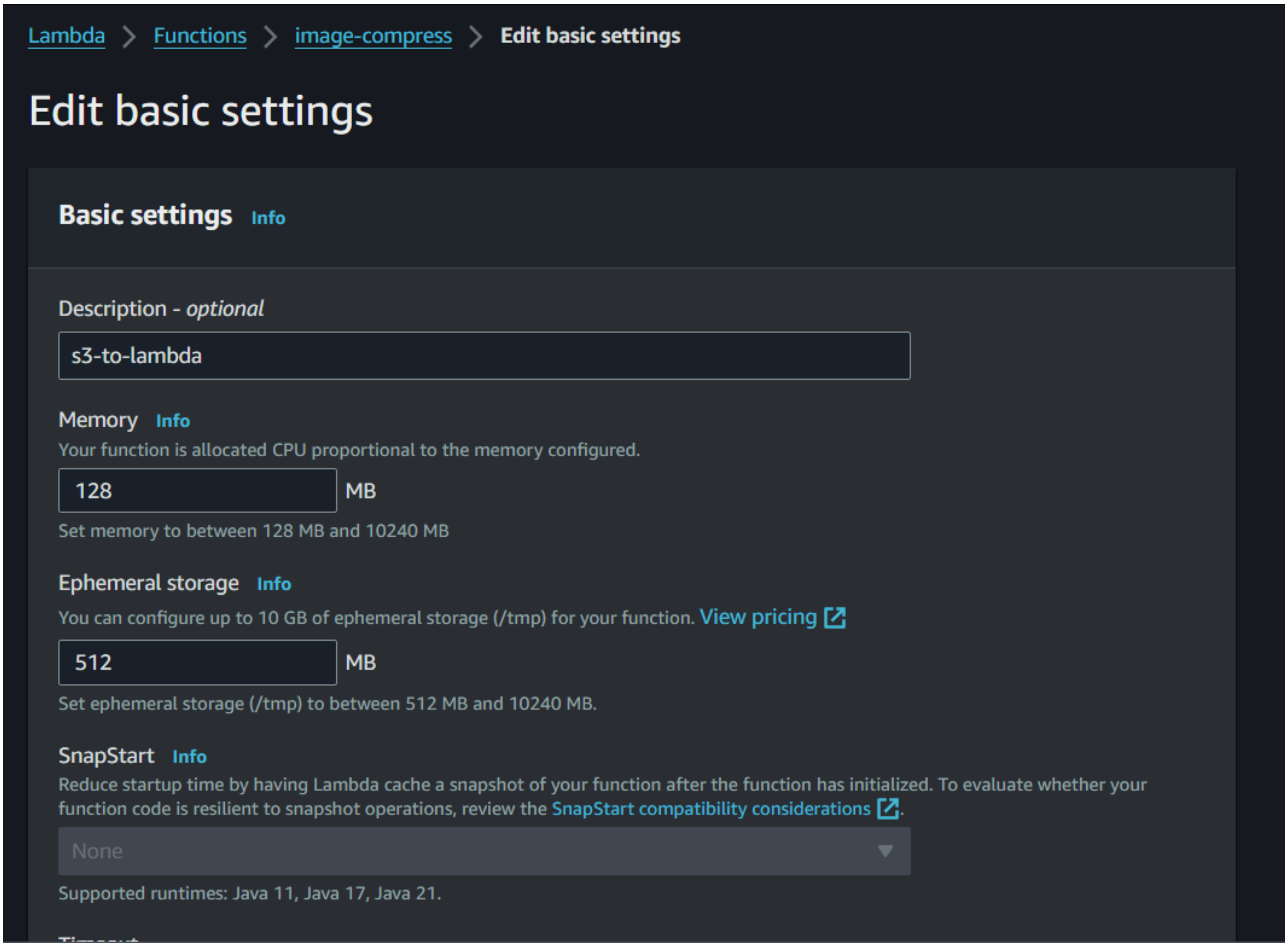
As you can see I have created 1 roles

Attach the role for Ec2 .

Now we have to attach the role to lambda to that it can access the s3 buckets

How to attach the role for lambda ?

Configuration → permissions → execution role → edit



Lambda > Functions > image-compress > Edit basic settings

Edit basic settings

Basic settings Info

Description - *optional*

s3-to-lambda

Memory Info

Your function is allocated CPU proportional to the memory configured.

128 MB

Set memory to between 128 MB and 10240 MB

Ephemeral storage Info

You can configure up to 10 GB of ephemeral storage (/tmp) for your function. View pricing

512 MB

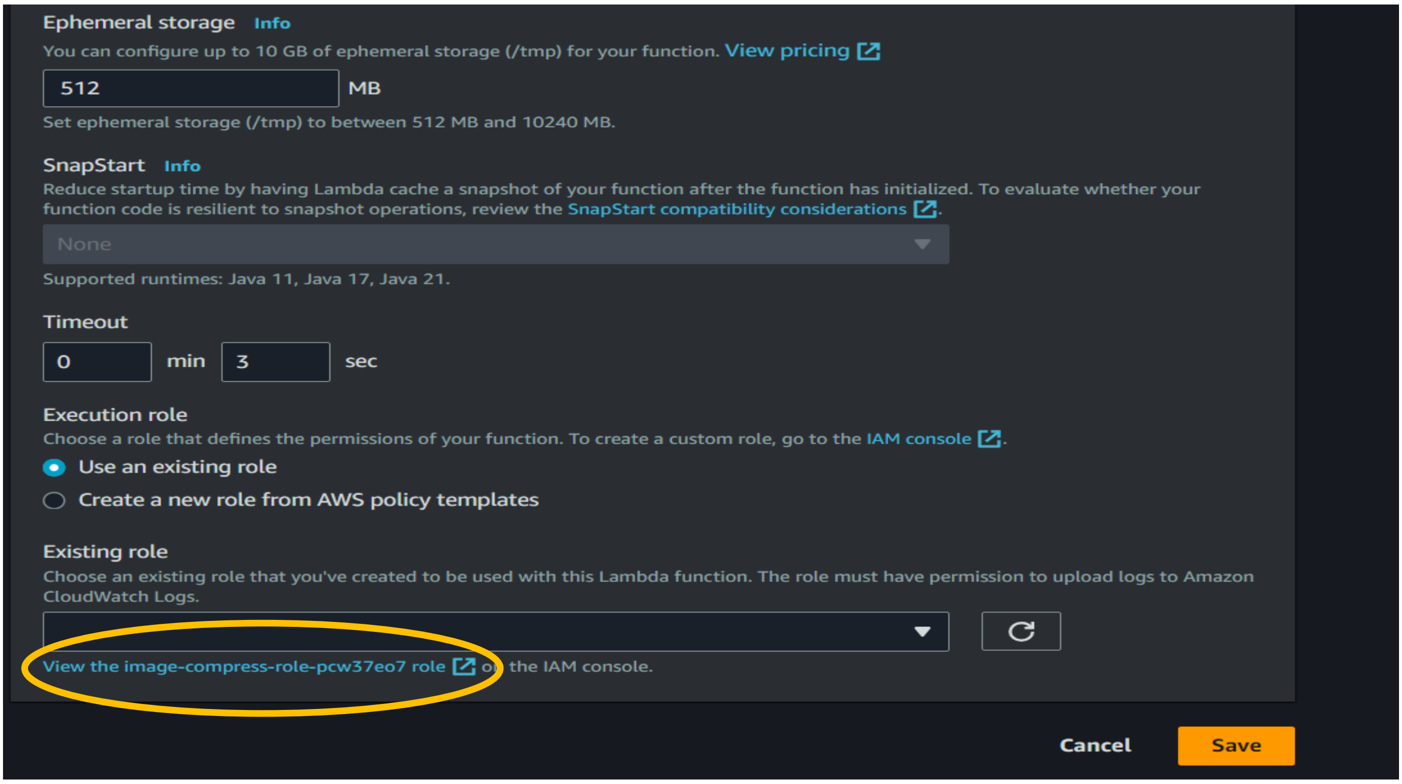
Set ephemeral storage (/tmp) to between 512 MB and 10240 MB.

SnapStart Info

Reduce startup time by having Lambda cache a snapshot of your function after the function has initialized. To evaluate whether your function code is resilient to snapshot operations, review the SnapStart compatibility considerations.

None

Supported runtimes: Java 11, Java 17, Java 21.



Ephemeral storage Info

You can configure up to 10 GB of ephemeral storage (/tmp) for your function. View pricing

512 MB

Set ephemeral storage (/tmp) to between 512 MB and 10240 MB.

SnapStart Info

Reduce startup time by having Lambda cache a snapshot of your function after the function has initialized. To evaluate whether your function code is resilient to snapshot operations, review the SnapStart compatibility considerations.

None

Supported runtimes: Java 11, Java 17, Java 21.

Timeout

0 min 3 sec

Execution role

Choose a role that defines the permissions of your function. To create a custom role, go to the IAM console.

● Use an existing role

○ Create a new role from AWS policy templates

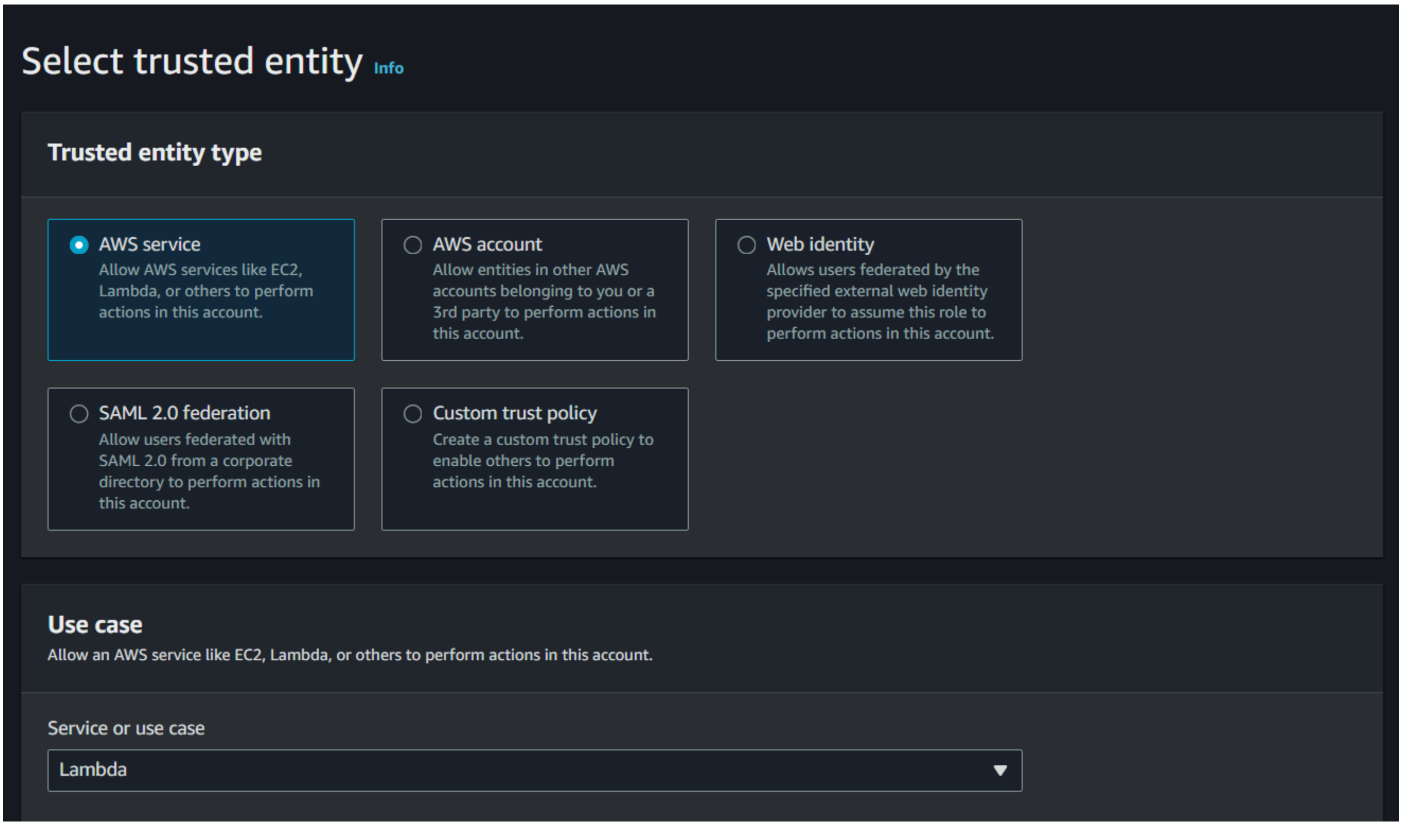
Existing role

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 image-compress-role-pcw37eo7 role on the IAM console.

Click on the link which is circled

Create a role for lambda



Select trusted entity Info

Trusted entity type

● AWS service
Allow AWS services like EC2, Lambda, or others to perform actions in this account.

○ AWS account
Allow entities in other AWS accounts belonging to you or a 3rd party to perform actions in this account.

○ Web identity
Allows users federated by the specified external web identity provider to assume this role to perform actions in this account.

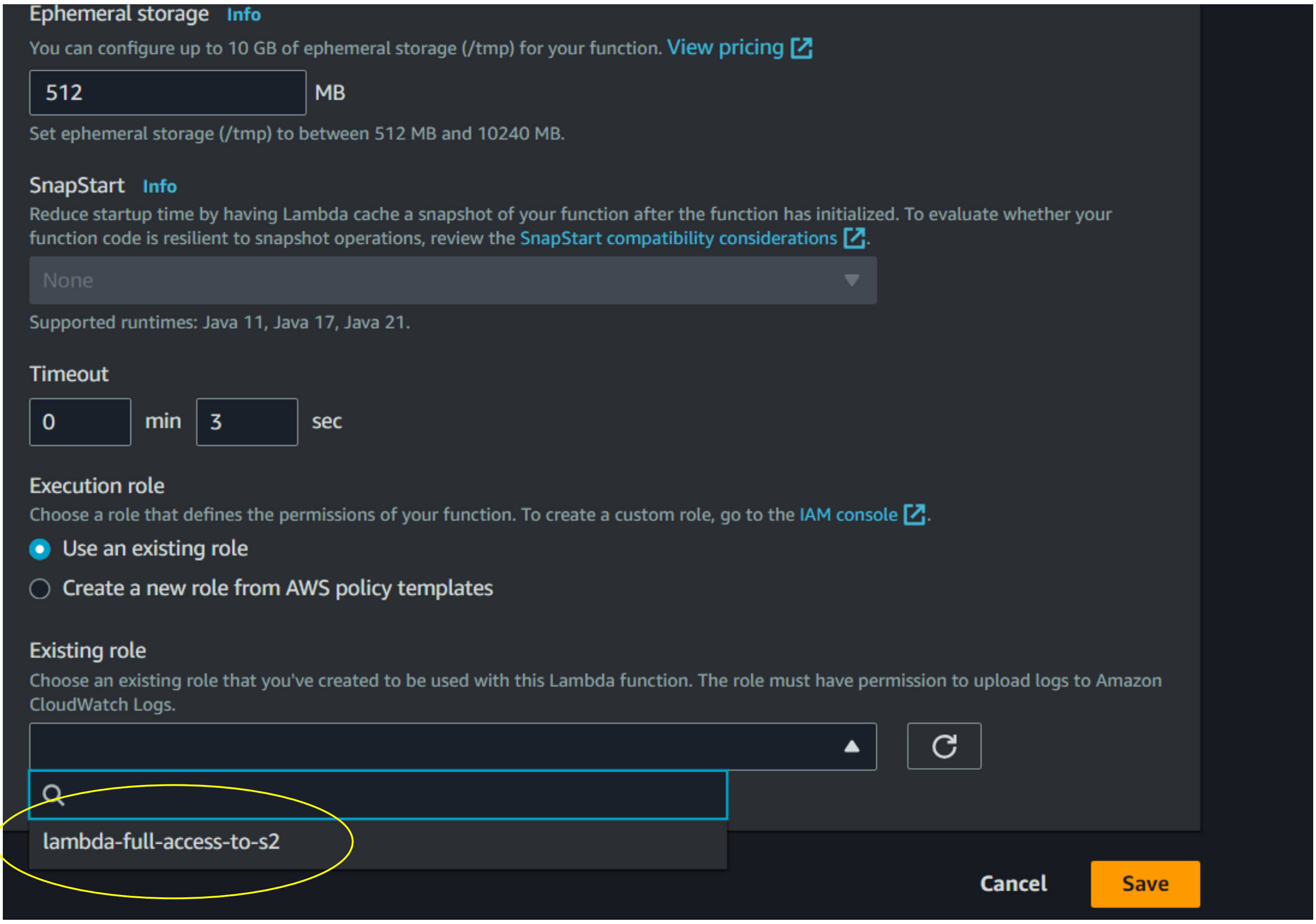
○ SAML 2.0 federation
Allow users federated with SAML 2.0 from a corporate directory to perform actions in this account.

○ Custom trust policy
Create a custom trust policy to enable others to perform actions in this account.

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

Service or use case
Lambda

After that we need to select the policy s3 full access and give the name and create it .



Ephemeral storage Info
You can configure up to 10 GB of ephemeral storage (/tmp) for your function. View pricing

512 MB

Set ephemeral storage (/tmp) to between 512 MB and 10240 MB.

SnapStart Info
Reduce startup time by having Lambda cache a snapshot of your function after the function has initialized. To evaluate whether your function code is resilient to snapshot operations, review the SnapStart compatibility considerations.

None

Supported runtimes: Java 11, Java 17, Java 21.

Timeout
0 min 3 sec

Execution role
Choose a role that defines the permissions of your function. To create a custom role, go to the IAM console.

● 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.

lambda-full-access-to-s2

As you can see the role is created attach that and save it

Summary
Till now we have create 2 s3 buckets and lambda

Added code to lambda

Created Ec2 inside that we have a shell script

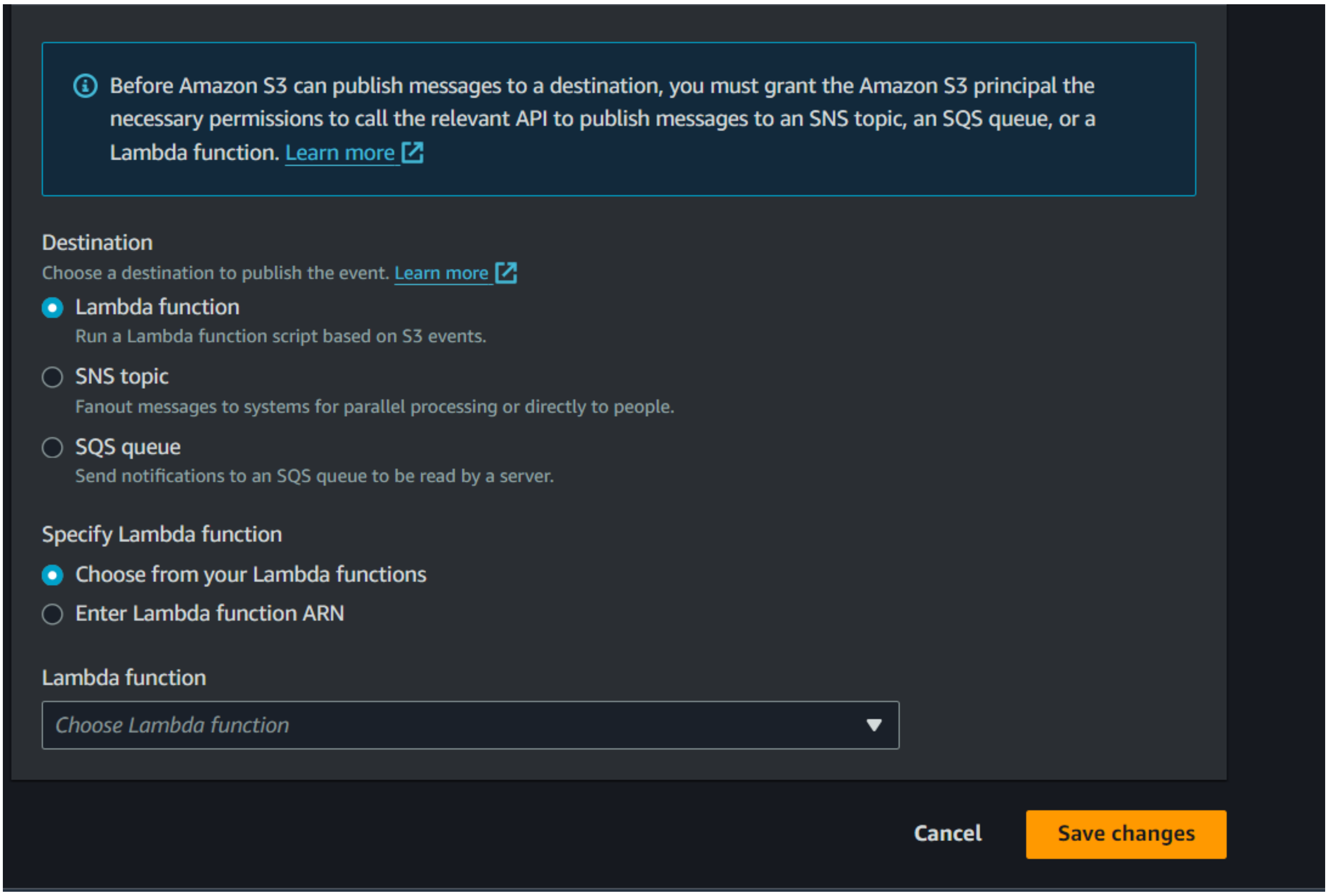
And we have attach the roles and policy for the ec2 to access lambda and

Lambda to access S3.

Step-5 : Now we need to have a triggering event for lambda for that we need to attach the s3 source bucket

How to do that

Go to s3 source bucket **my-source-bucket-0 → properties → Event notifications→Create event notification →event name → event types(select All object create events) → lambda function → choose lambda function**



ⓘ Before Amazon S3 can publish messages to a destination, you must grant the Amazon S3 principal the necessary permissions to call the relevant API to publish messages to an SNS topic, an SQS queue, or a Lambda function. Learn more

Destination

Choose a destination to publish the event. Learn more

● Lambda function
Run a Lambda function script based on S3 events.

○ SNS topic
Fanout messages to systems for parallel processing or directly to people.

○ SQS queue
Send notifications to an SQS queue to be read by a server.

Specify Lambda function

● Choose from your Lambda functions

○ Enter Lambda function ARN

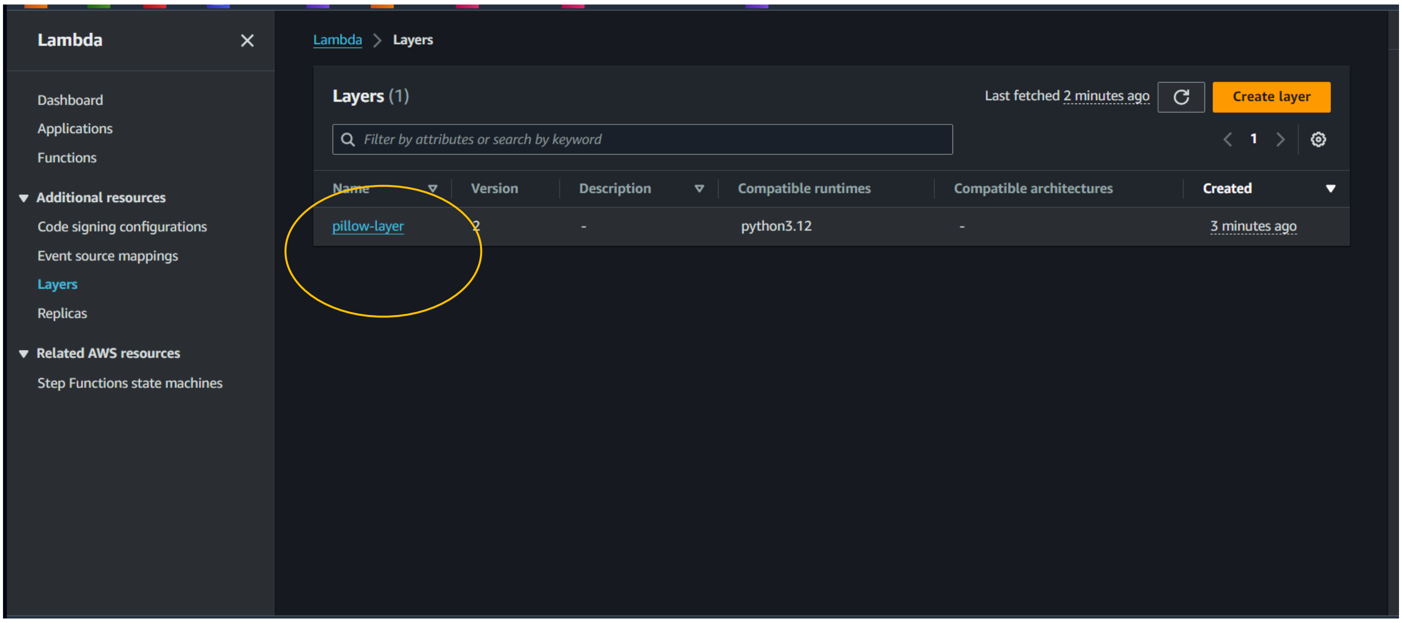
Lambda function

Choose Lambda function ▼

Save changes

Now run the shell script which we have created in Ec2

After running the script we will get layer In layers



Layers (1)

Name	Version	Description	Compatible runtimes	Compatible architectures	Created
pillow-layer	2	-	python3.12	-	3 minutes ago

Now go to lambda click on attach layers

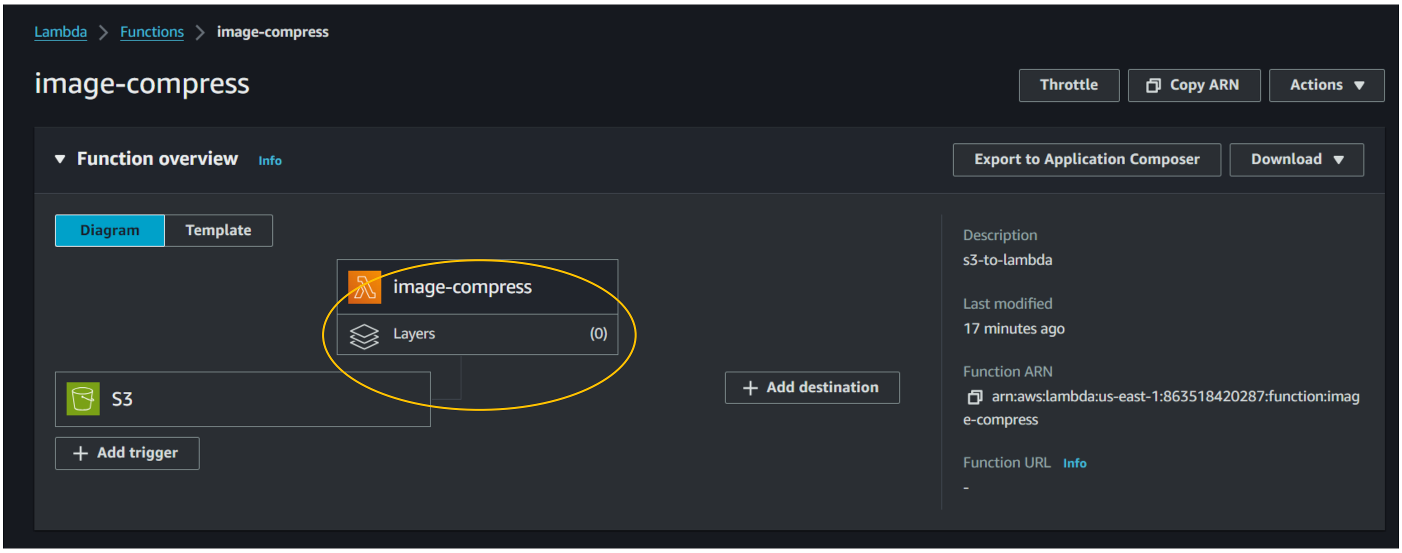


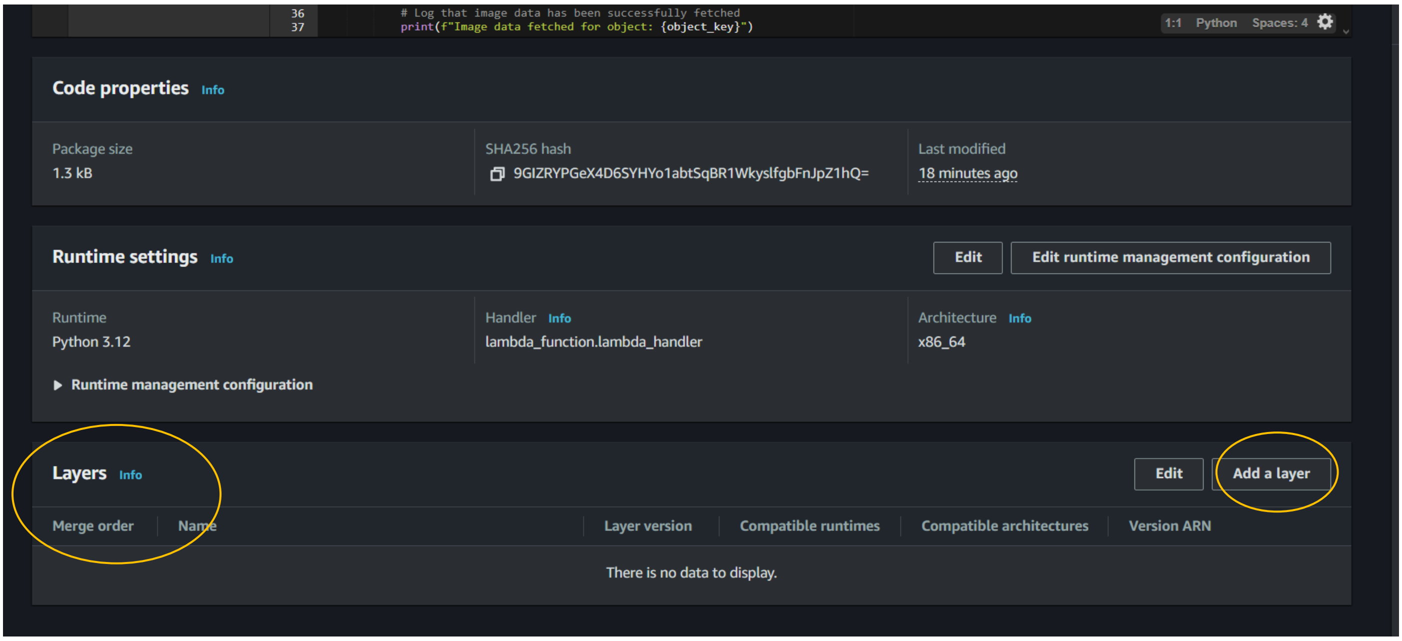
image-compress

Description
s3-to-lambda

Last modified
17 minutes ago

Function ARN
arn:aws:lambda:us-east-1:863518420287:function:image-compress

Function URL
-



```
        # Log that image data has been successfully fetched
        print(f"Image data fetched for object: {object_key}")
```

1:1 Python Spaces: 4
Info
Edit
Edit runtime management configuration
Runtime management configuration
Add a layer

Code properties

Package size	SHA256 hash	Last modified
1.3 kB	9GIZRYPGeX4D6SYHYo1abtSqBR1WkyslfgbFnJpZ1hQ=	18 minutes ago

Runtime settings

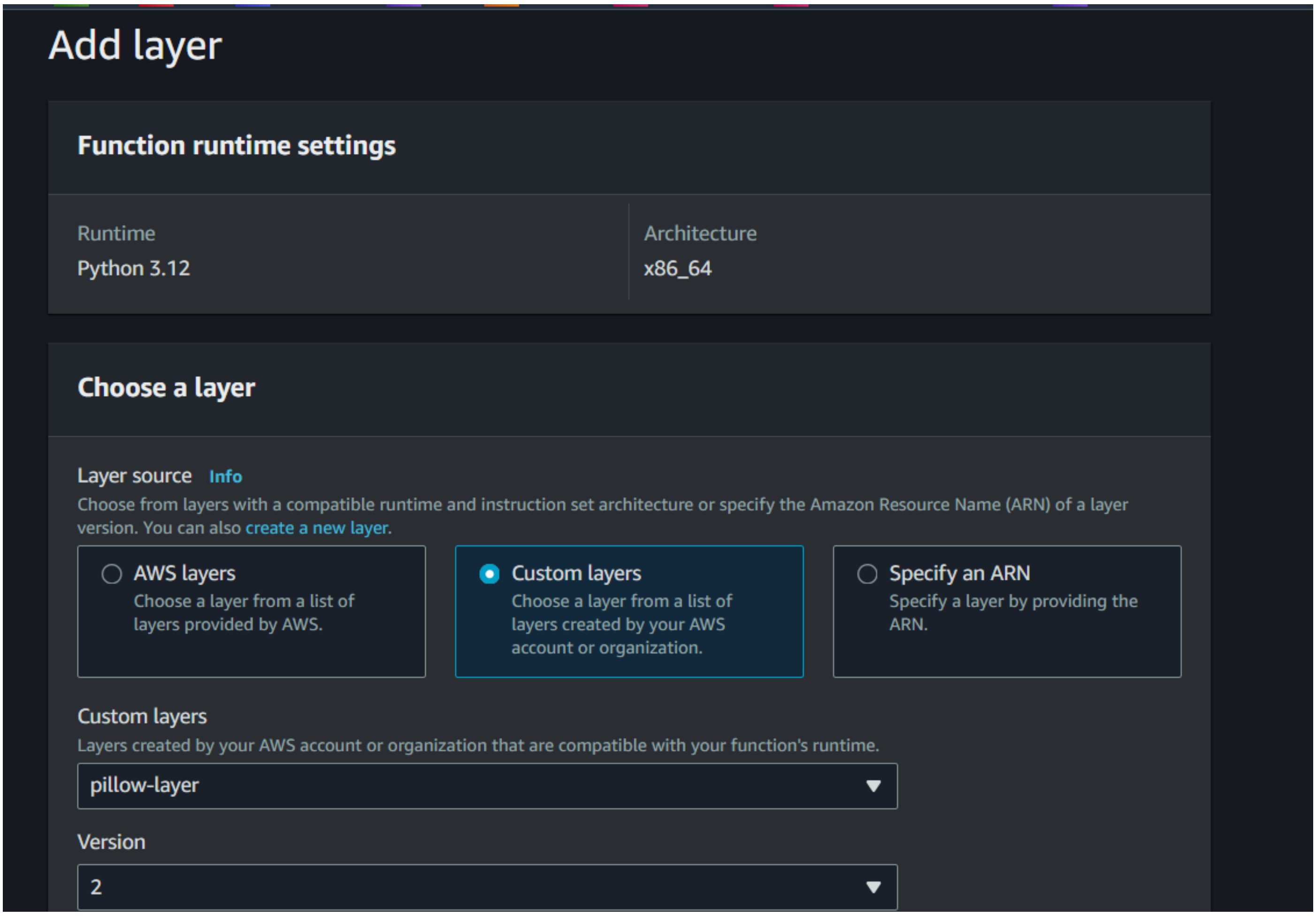
Runtime	Handler	Architecture
Python 3.12	lambda_function.lambda_handler	x86_64

Layers

Merge order	Name	Layer version	Compatible runtimes	Compatible architectures	Version ARN

There is no data to display.

Add a layer



Add layer

Function runtime settings

Runtime
Python 3.12

Architecture
x86_64

Choose a layer

Layer source Info
Choose from layers with a compatible runtime and instruction set architecture or specify the Amazon Resource Name (ARN) of a layer version. You can also create a new layer.

○ AWS layers
Choose a layer from a list of layers provided by AWS.

● Custom layers
Choose a layer from a list of layers created by your AWS account or organization.

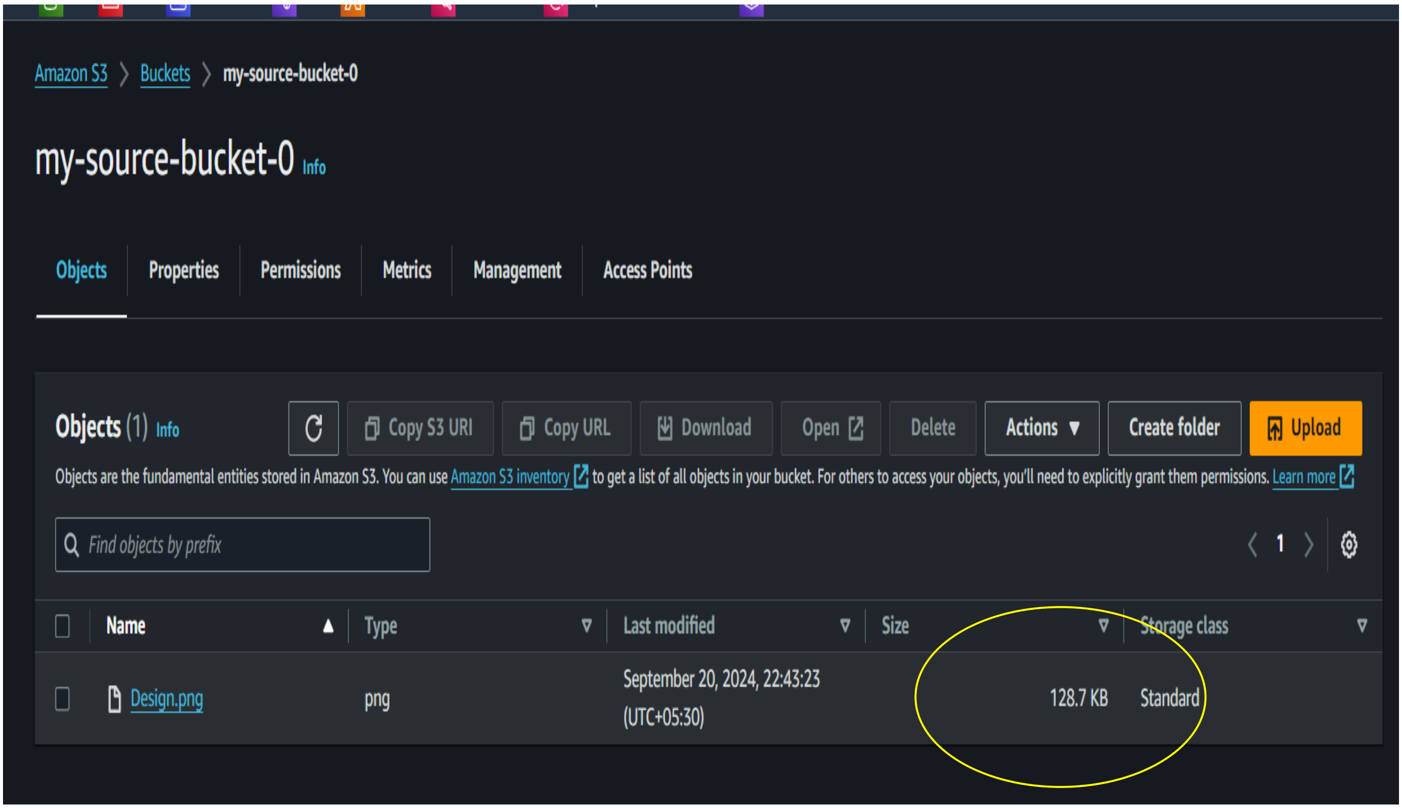
○ Specify an ARN
Specify a layer by providing the ARN.

Custom layers
Layers created by your AWS account or organization that are compatible with your function's runtime.
pillow-layer ▼

Version
2 ▼

Then click add .

Now all set lets Test the project. Go to source bucket and add image and you can see the compressed image in the destination .



Amazon S3 〉 Buckets 〉 my-source-bucket-0

my-source-bucket-0 Info

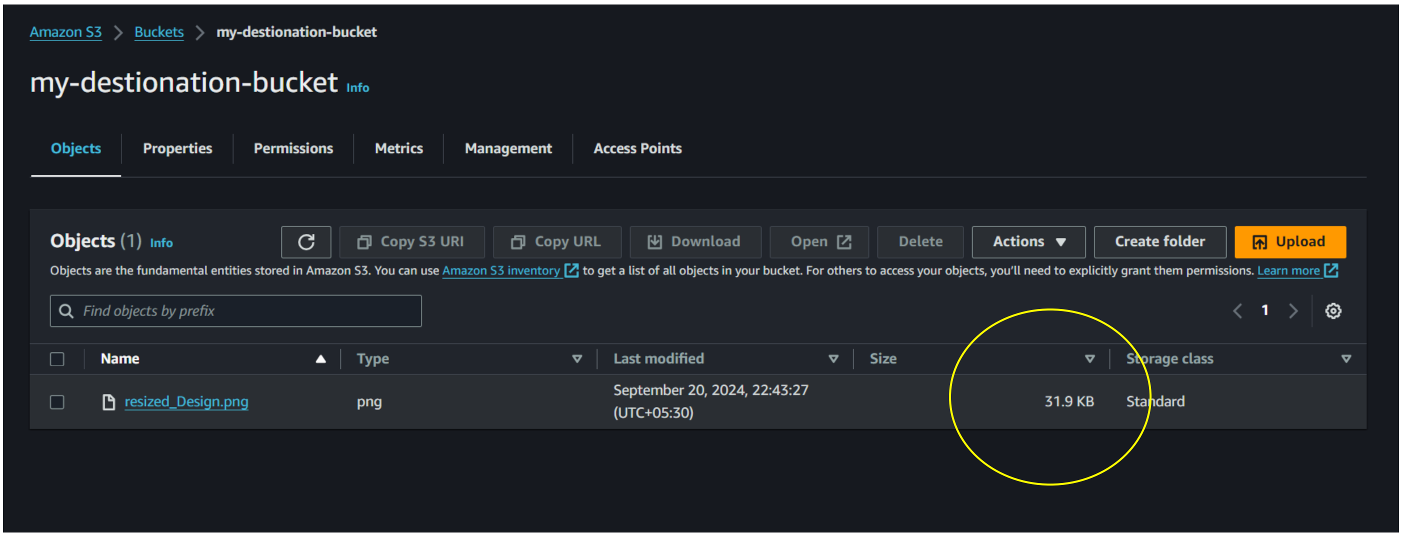
Objects (1) Info

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

〈 1 〉

□	Name ▲	Type ▼	Last modified ▼	Size ▼	Storage class ▼
□	Design.png	png	September 20, 2024, 22:43:23 (UTC+05:30)	128.7 KB	Standard



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

□	Name	Type	Last modified	Size	Storage class
□	resized_Design.png	png	September 20, 2024, 22:43:27 (UTC+05:30)	31.9 KB	Standard

As you can see that we have compressed the image

Now what all to delete

S3 buckets

Lambda function

Ec2

Layers

Roles

And all set ..