

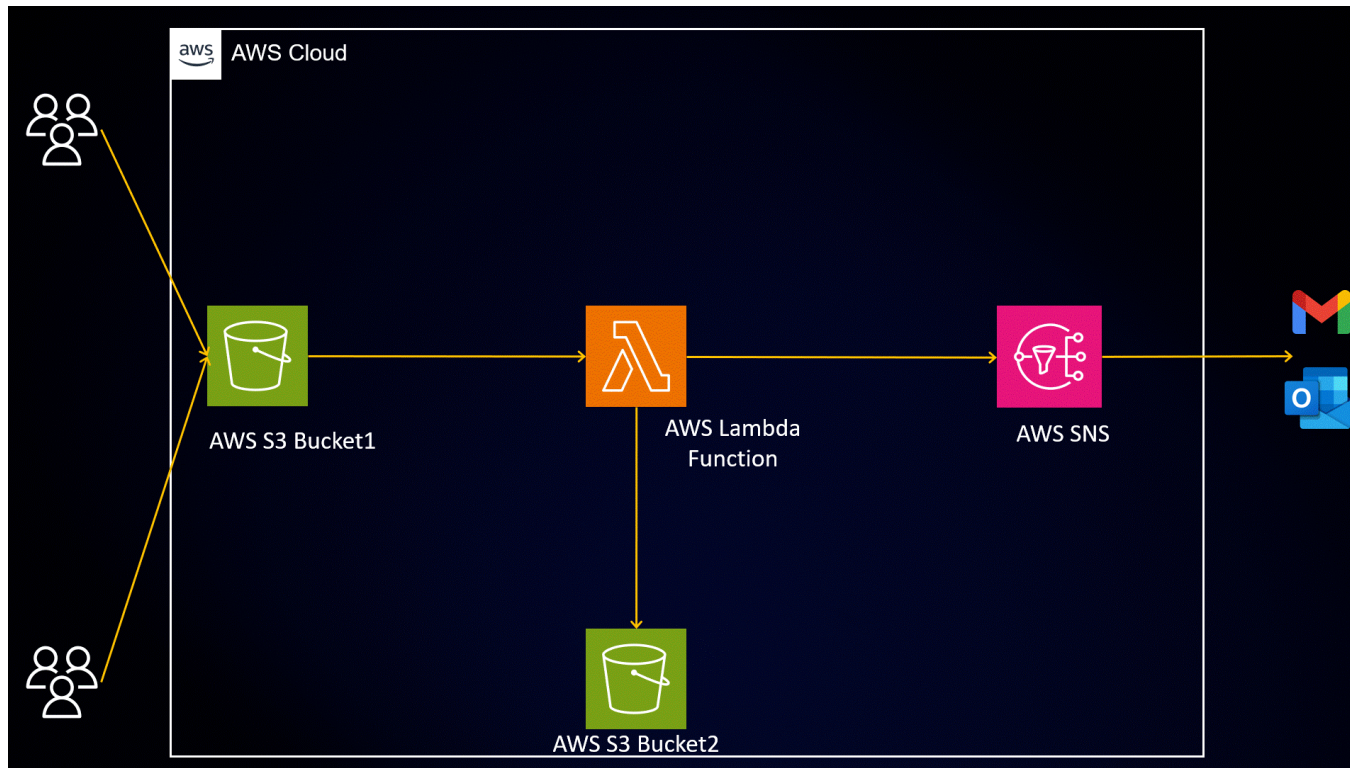
End-to-End Image Resizing Pipeline with AWS S3, Lambda, and SNS: A Step-by-Step Guide



Aman Pathak · [Follow](#)

Published in DevOps.dev · 9 min read · Jul 2





AWS Infra Diagram

Introduction:

In today's digital age, images play a crucial role in various applications and websites. However, managing and serving images of different sizes can be a challenging task. To overcome this challenge, we can leverage the power of cloud computing and automation to resize images on-the-fly. In this blog post, we will explore how to automate image resizing using AWS S3, Lambda, and SNS, enabling efficient image management and delivery.

Prerequisites:

Before diving into the implementation, let's ensure we have the following prerequisites in place:

AWS Account: To follow along with the steps outlined in this blog post, you will need an AWS account. If you don't have one, you can sign up for free on the AWS website.

Basic Knowledge of AWS Services: Familiarity with AWS S3 (Simple Storage Service), Lambda (Serverless Compute), and SNS (Simple Notification Service) will be helpful. If you are new to these services, don't worry — we will cover the necessary concepts and steps in detail.

Implementation Options:

In this blog post, we will cover only implementation options: manual setup through the AWS Management Console and the other approach is infrastructure-as-code using Terraform.

1. Manual Setup: We will guide you through the step-by-step configuration process using the AWS Management Console. This approach is suitable for

users who prefer a hands-on approach and want to understand the individual components and their configurations.

2. Terraform: For users who prefer infrastructure-as-code and want to automate the provisioning of resources, we will provide instructions for implementing the solution using Terraform. This approach allows for repeatable and version-controlled deployments of the Image Resize Automation.

To follow along with the manual setup implementation, please refer to the detailed instructions provided in this blog post. If you prefer the Terraform approach, you can access the complete Terraform code and instructions on our GitHub repository: <https://github.com/AmanPathak-DevOps/Terraform-for-AWS/blob/master/Non-Modularized/Resize-Image-Using-LambdaFunction-S3-SNS/>. The repository includes the necessary Terraform scripts and configuration files to set up the Image Resize Automation.

By following the upcoming sections of this blog, you'll gain insights into both manual and Terraform-based implementations of the automated AWS Image Resize Automation and email notification system. Whether you prefer a hands-on or automated approach, this solution will empower you to effortlessly stay resized your images and optimize your storage.

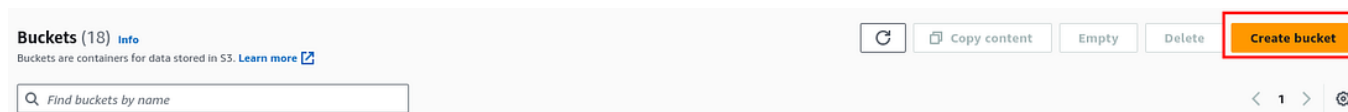
Now that we have our prerequisites in order, let's dive into the implementation details and automate the image resizing process step by step. By the end of this blog post, you will have a fully functional system that can automatically resize images and store them in a separate bucket, while notifying a subscribed email address about the successful resizing process.

. . .

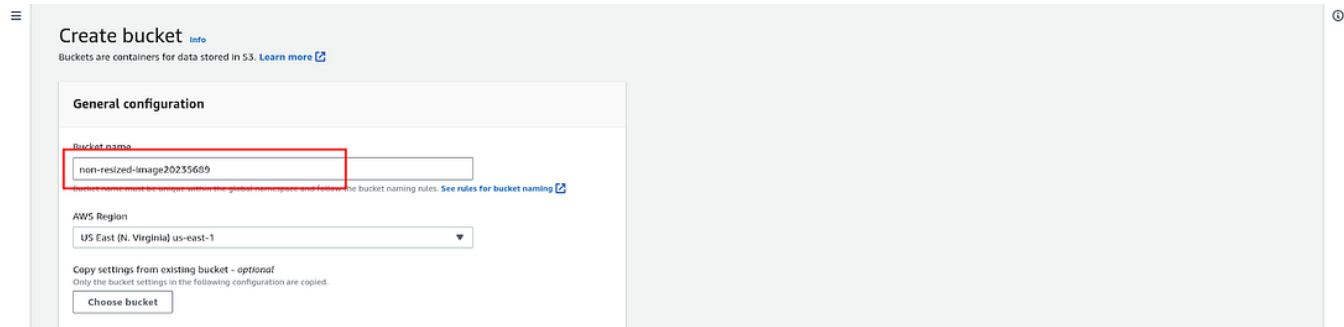
Let's get started!

1. In this project, we will be creating two buckets. The first bucket will be used to store the original images that we want to resize. The second bucket will be dedicated to storing the resized images. The resizing process will be automated through an AWS Lambda function, which will automatically upload the resized images to the second bucket.

Creating the first S3 bucket by clicking on **Create bucket**.



2. Enter the name of Bucket 1 and click on **Create bucket**.



Create bucket [Info](#)
Buckets are containers for data stored in S3. [Learn more](#)

General configuration

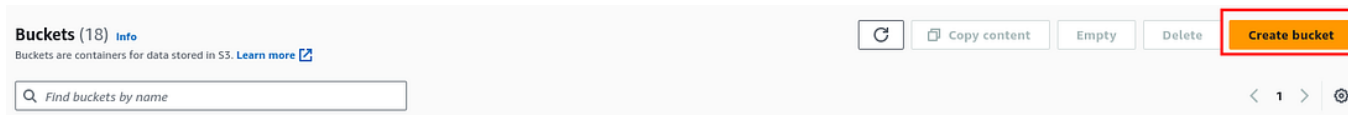
Bucket name
non-resized-image20235689
Bucket names must be unique within the global namespace and follow the bucket naming rules. [See rules for bucket naming](#)

AWS Region
US East (N. Virginia) us-east-1

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

[Choose bucket](#)

3. Create the second S3 bucket by clicking on **Create bucket**.

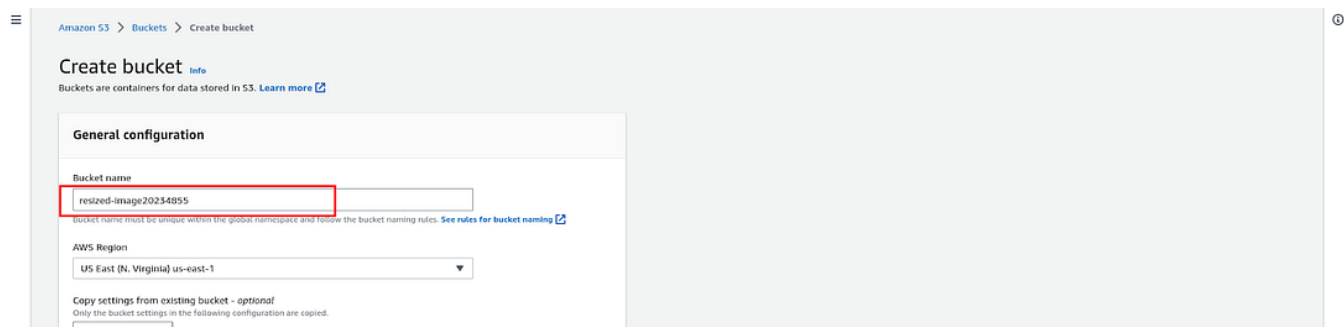


Buckets (18) [Info](#)
Buckets are containers for data stored in S3. [Learn more](#)

[Refresh](#) [Copy content](#) [Empty](#) [Delete](#) **Create bucket**

< 1 > [Settings](#)

4. Enter the name of **Bucket2** and click on **Create bucket**.



Create bucket [Info](#)
Buckets are containers for data stored in S3. [Learn more](#)

General configuration

Bucket name
resized-image20234855
Bucket names must be unique within the global namespace and follow the bucket naming rules. [See rules for bucket naming](#)

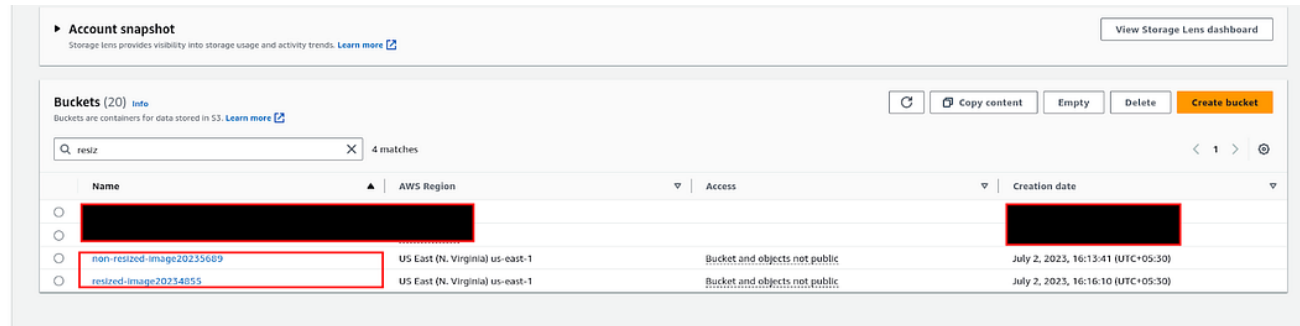
AWS Region
US East (N. Virginia) us-east-1

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

[Choose bucket](#)

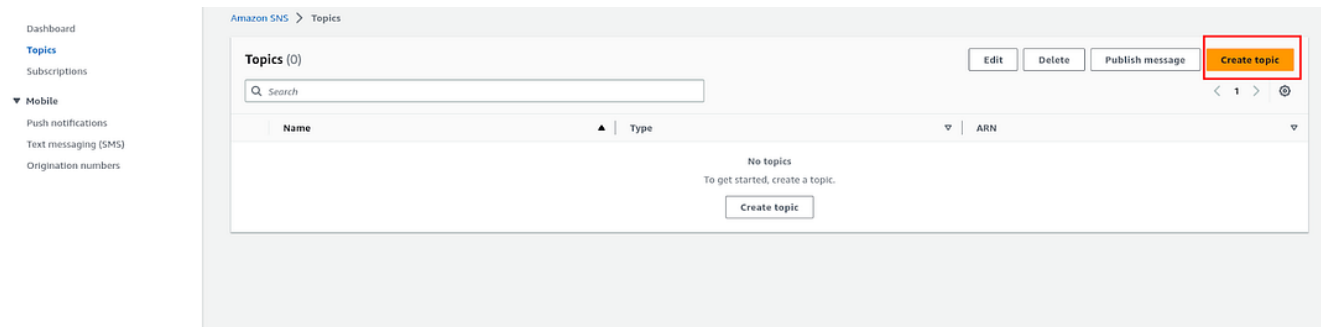
5. So, Here we have created two Buckets.

You can check the below screenshot.

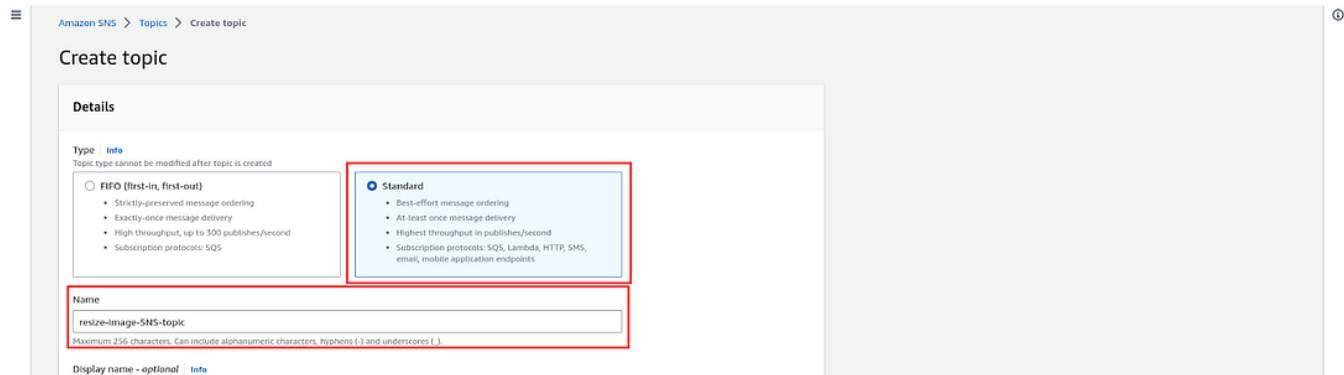


6. Now, let's create an SNS topic and an SNS subscription before creating a lambda function.

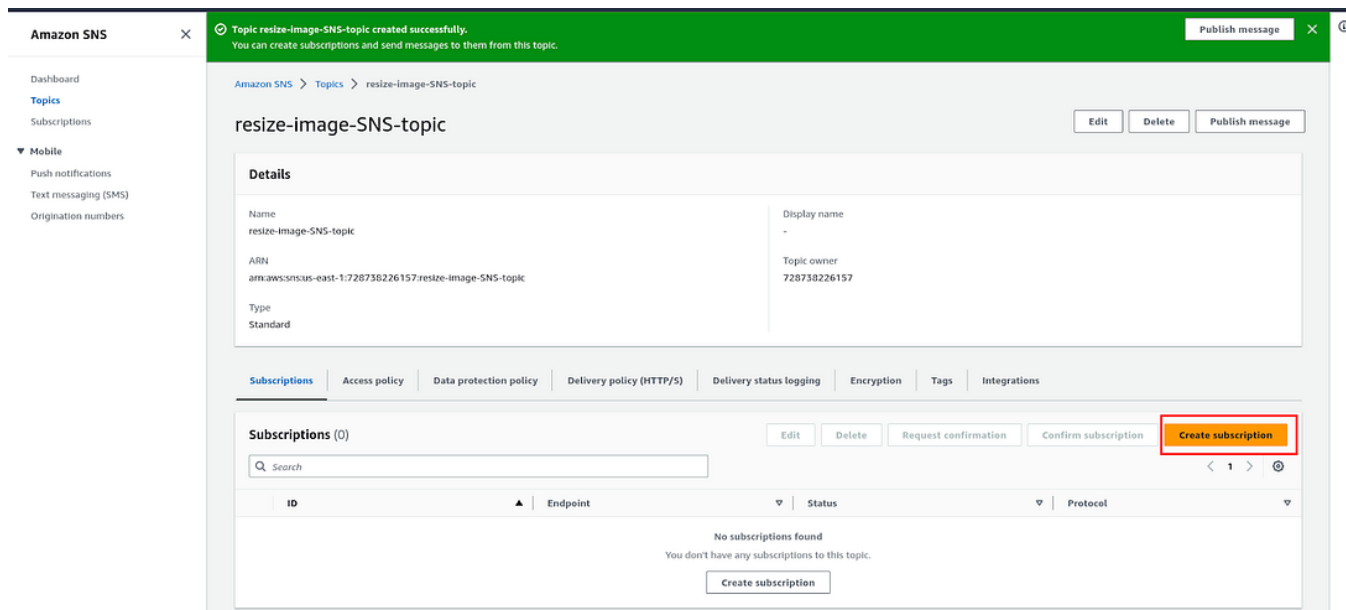
Click on **Create topic**.



7. Select **Standard** type SNS topic and enter the suitable name for your SNS topic and click on **Create topic**.



8. As you can see in the below screenshot, we have configured the SNS topic. Now, click on **Create subscription** to add your email address.



9. Select the **Protocol** type as **email** and add your mail address in **Endpoint** and click on **Create Subscription**.

must register your toll-free number by September 30, 2022 or you will no longer be able to use the toll-free number. [Learn more](#)

Amazon SNS > Subscriptions > Create subscription

Create subscription

Details

Topic ARN

Protocol
The type of endpoint to subscribe to.

Endpoint
An email address that can receive notifications from Amazon SNS.

After your subscription is created, you must confirm it. [Info](#)

Subscription filter policy - optional [Info](#)
This policy filters the messages that a subscriber receives.

Redrive policy (dead-letter queue) - optional [Info](#)
Send undeliverable messages to a dead-letter queue.

Cancel [Create subscription](#)

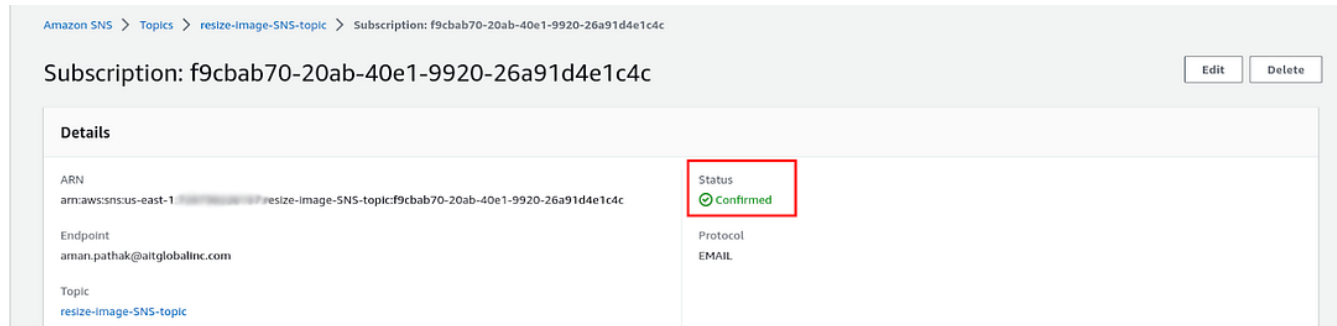
10. After creating **Subscription**, you will get a notification on your given mail address if not, please check your **spam box**. The mail will look like the below screenshot.

Click on **Confirm subscription** to get the notification for the resized image.



11. Once you confirm your **subscription**. The **status** should be **Confirmed** from **Pending status**.

Here, we configured the **SNS topic** and **subscription**.



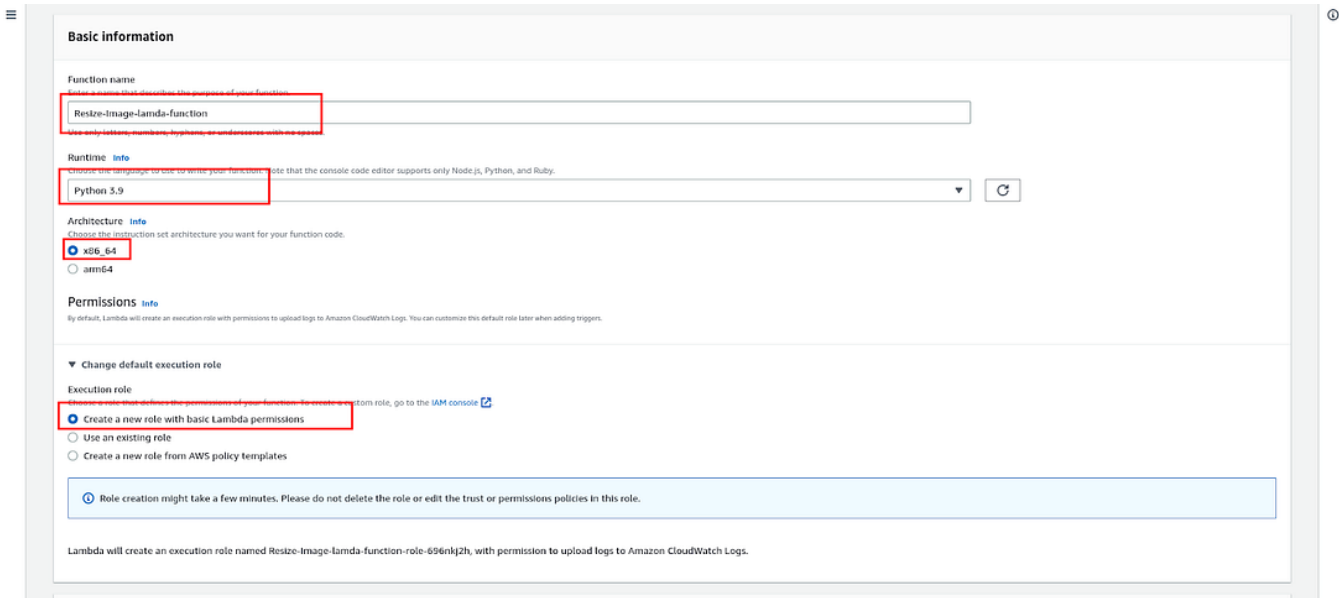
12. Now, Let's create a **Lambda function** that will help us to resize the images and store the images.

Click on **Create function**.



13. Enter the **function name**, **runtime**(I am using Python 3.9), and **architecture**, and for **role** select the first option, So we can configure from

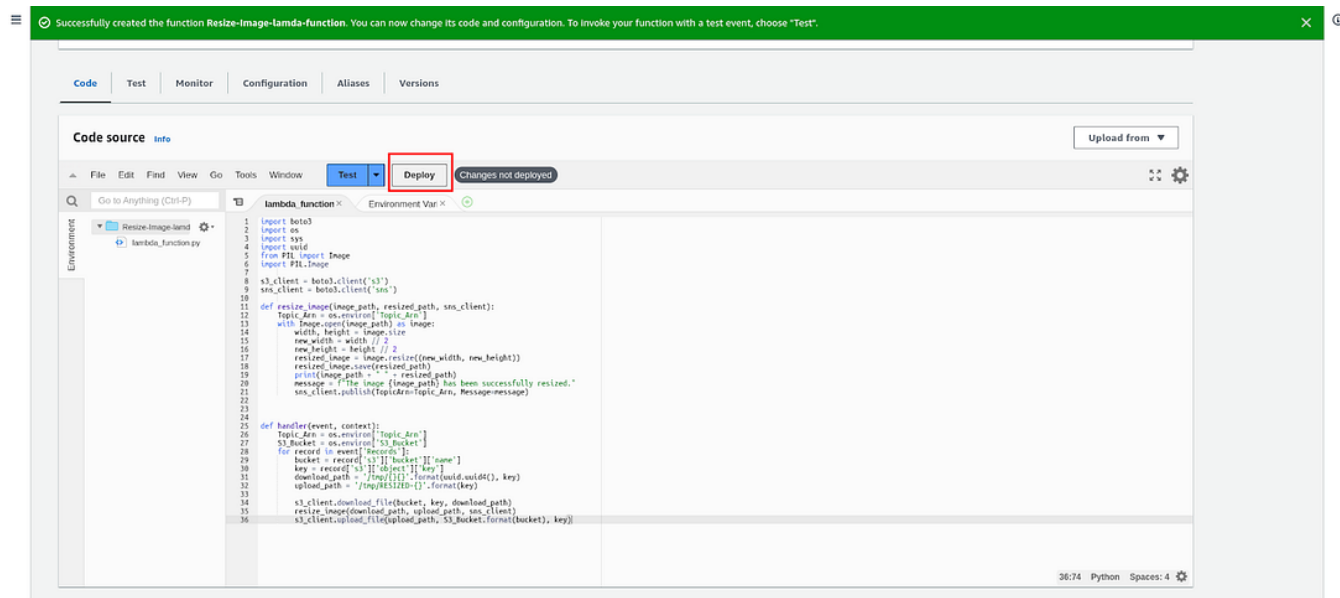
our own all the access for the services and click on **Create function**.



The screenshot shows the 'Basic information' tab of the AWS Lambda 'Create function' wizard. The 'Function name' field is set to 'Resize-Image-lambda-function'. The 'Runtime' is set to 'Python 3.9'. The 'Architecture' is set to 'x86_64'. Under the 'Permissions' section, the 'Change default execution role' dropdown is expanded, and the option 'Create a new role with basic Lambda permissions' is selected. A blue information box at the bottom states: 'Role creation might take a few minutes. Please do not delete the role or edit the trust or permissions policies in this role.' Below this, a note says: 'Lambda will create an execution role named Resize-Image-lambda-function-role-6596nkj2h, with permission to upload logs to Amazon CloudWatch Logs.'

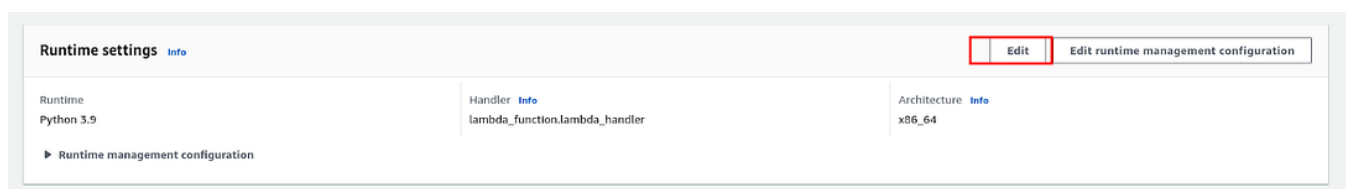
14. First of all, replace the code with our code that will resize the image

Code link- [Lambda-Code](#), After replacing the code, **deploy** it by clicking on **Deploy**.



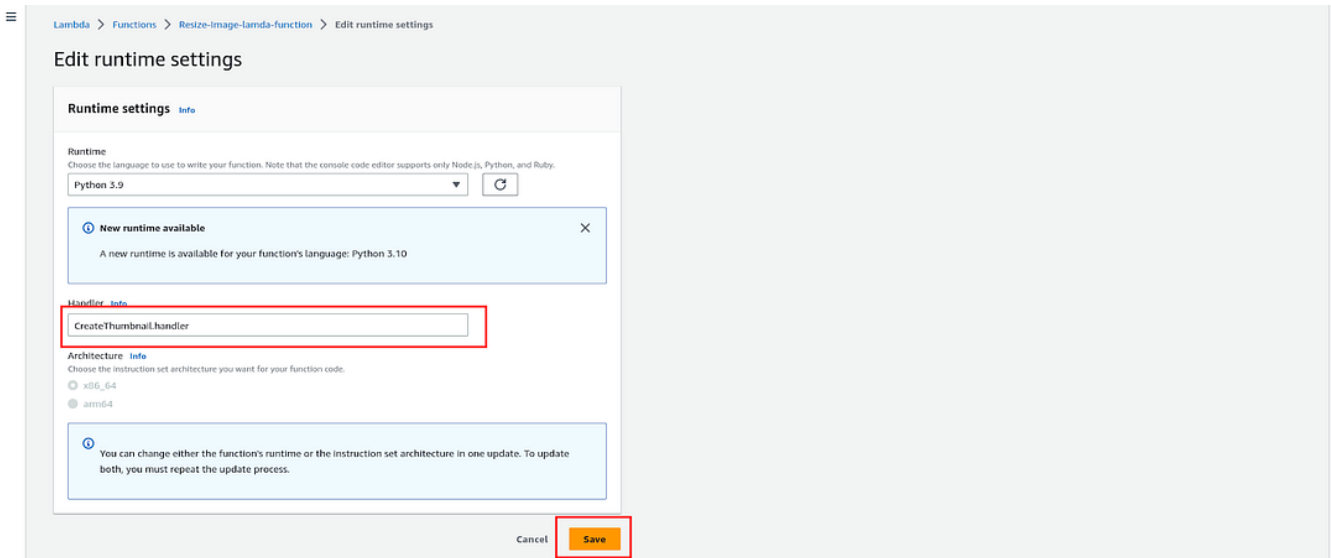
15. Now, we need to configure lots of things in the **lambda function**. Let's start with **runtime settings**, that is, exactly under the code itself.

Click on Edit



16. Now, add the **handler** name according to your **Python code name** and click on **Save**.

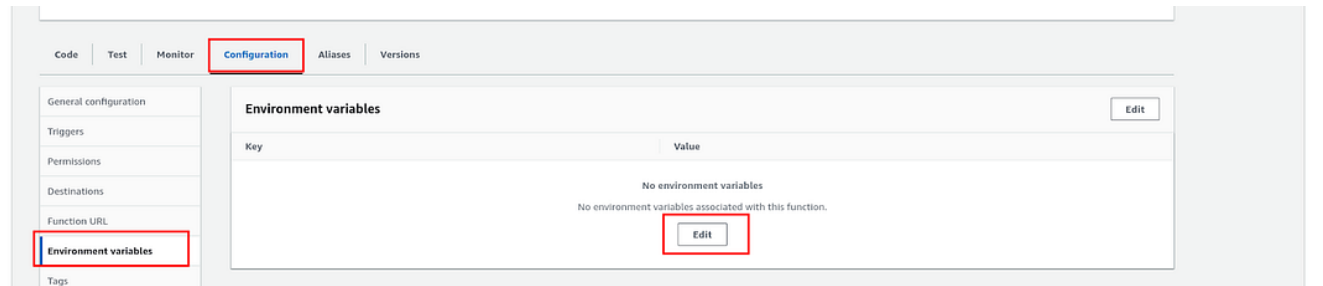
My lambda function code name is `CreateThumbnail` and handler name is `handler`. So, I have written `CreateThumbnail.handler`.



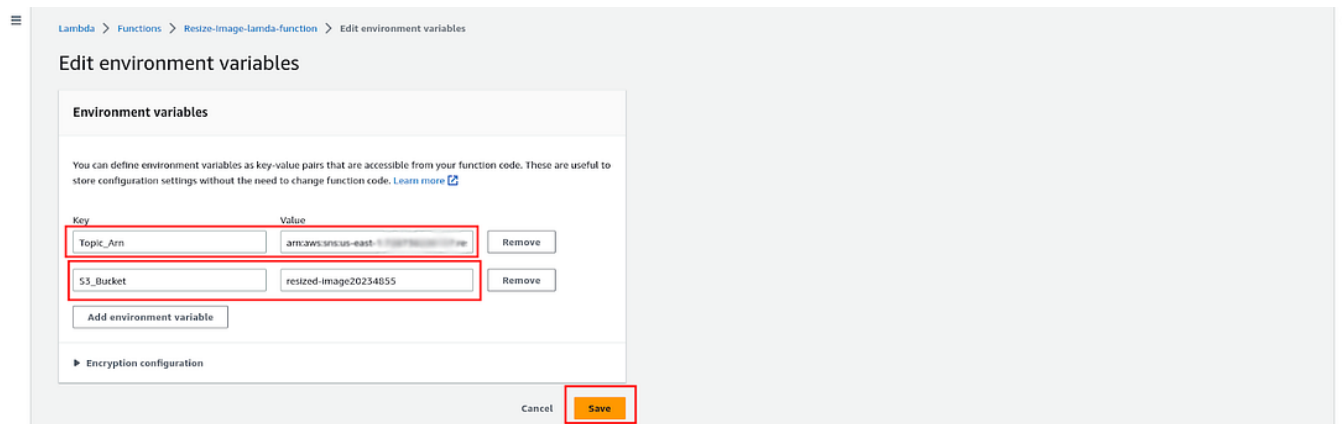
17. In our Python code, we have to provide *Bucket2- resized-bucket name* and **SNS topic Arn**. But instead of hardcoding it, we will use Lambda

[Sign up](#)[Sign In](#)[Write](#)

variables, and click on edit.



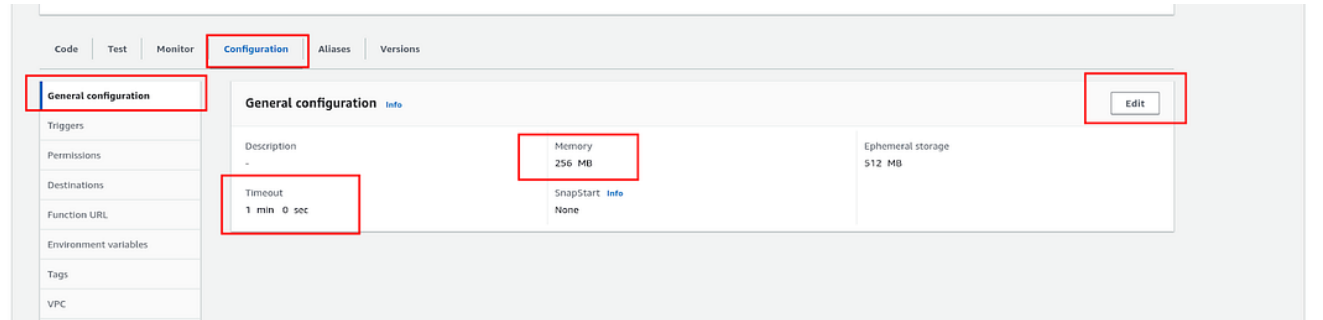
18. Add the SNS topic arn and S3 bucket name of Bucket2 according to your service name and on and click on Save.



19. In the lambda function, we have to increase the **timeout** and **Memory size** because the lambda function needs more time and memory to perform the resizing task.

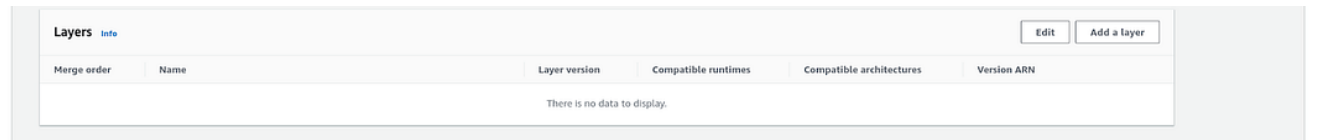
To do that, Go to the **Configuration** tab-> **General configuration** section -> **click on edit**. Add the timeout 1 min and give memory 256MB. It would be

enough to perform our task.



20. Now, we need to add one **PIL** layer for our Python code.

Click on **Add a layer**.



21. Add the ARN of the layer **arn:aws:lambda:us-east-1:770693421928:layer:Klayers-p39-pillow:1** and click on **Save**.

Add layer

Function runtime settings

Runtime: Python 3.9
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.

Specify an ARN
Specify a layer by providing the Amazon Resource Name (ARN).

arn:aws:lambda:us-east-1:770693421928:layer:pillow-p39:1 [Verify](#)

Description
pillow==9.2.0 | 9e649f6351046301b0a50b0fc2c419678e5577424d00534e402bb7c40727e74

Compatible runtimes
Python 3.6, Python 3.7, Python 3.8, Python 3.9

Compatible architectures
-

[Cancel](#) [Add](#)

22. Let's do the IAM role configuration for our Lambda function. Go to IAM -> roles -> 'search for your lambda name role' and click on the role.

Identity and Access Management (IAM)

Search IAM

Dashboard

Access management

- User groups
- Users
- Roles**
- Policies
- Identity providers
- Account settings

Access reports

- Access analyzer
- Archive rules
- Analyzers
- Settings

Roles (Selected 1/80) [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.

Search: Resize-image-lambda X 1 match

Role name	Trusted entities	Last activity
<input checked="" type="checkbox"/> Resize-image-lambda-function-role-696nk2jh	AWS Service: lambda	-

[Manage](#)

Roles Anywhere [Info](#)
Authenticate your non-AWS workloads and securely provide access to AWS services.

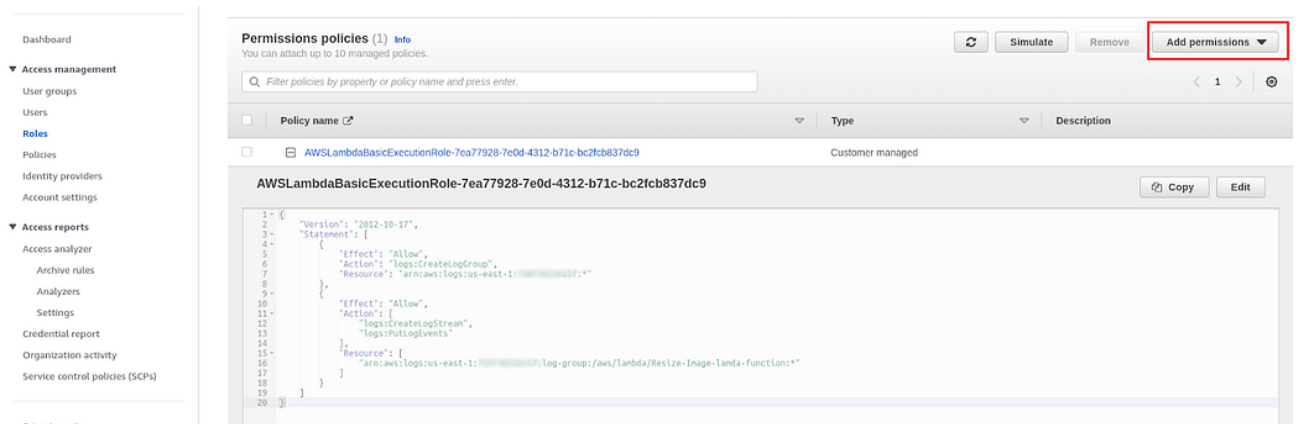
Access AWS from your non-AWS workloads
Operate your non-AWS workloads using the same authentication and authorization strategy that you use within AWS.

X.509 Standard
Use your own existing PKI infrastructure or use [AWS Certificate Manager Private Certificate Authority](#) to authenticate identities.

Temporary credentials
Use temporary credentials with ease and benefit from the enhanced security they provide.

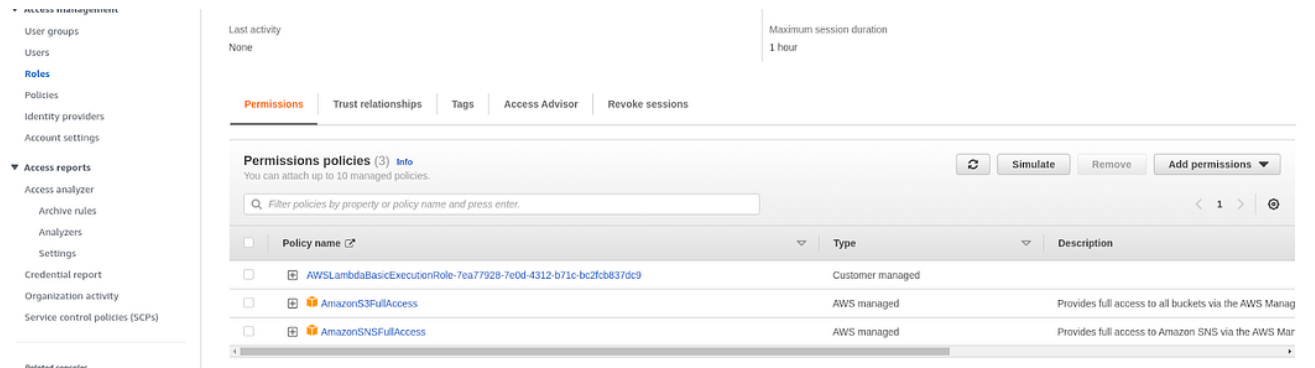
23. Here we have only one **permission** that is related to see the **logs** of our **lambda function** in loggroup. But we need two more permissions which is **SNSFullAccess** to send the mail to the person and **S3FullAccess** to get the images and store the images in the respective buckets.

To do that, Click on **Add permissions** and select **Attach policy**.

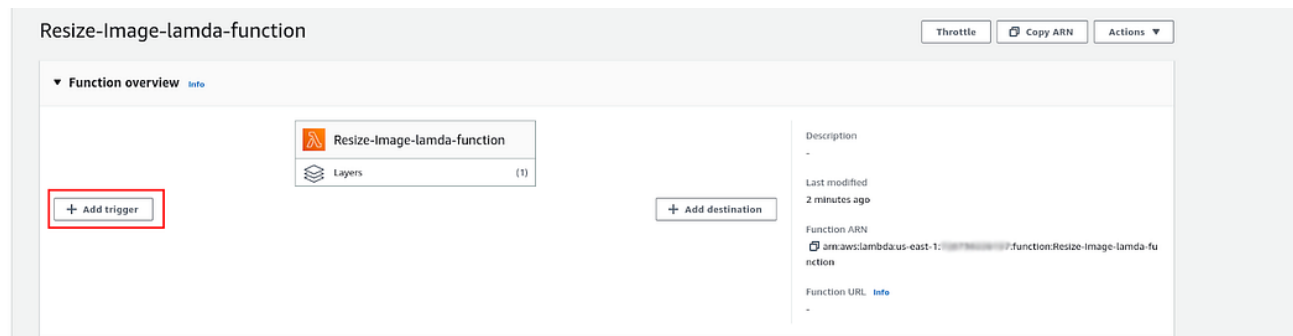


24. Now, find the two permissions one by one **SNSFullAccess** and **S3FullAccess**, and check the box on the left of the permissions. After doing both, click on **Add permissions**.

The final IAM role has the below permissions list.



25. Finally, We have to add a trigger for the Lambda function. So whenever someone uploads an image to Bucket1. The lambda function will trigger and start the process of resizing the image and doing the other respective things.



26. Add the S3 as source, Bucket1 which will trigger the Lambda function, and click on Add.

Trigger configuration Info

S3
aws storage

Bucket
Please select the S3 bucket that serves as the event source. The bucket must be in the same region as the function.
s3/non-resized-image20235689

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/

Suffix - optional
Enter a single optional suffix to limit the notifications to objects with keys that end with matching characters.
e.g. .jpg

Recursive invocation
If your function writes objects to an S3 bucket, ensure that you are using different S3 buckets for input and output. Writing to the same bucket increases the risk of creating a recursive invocation, which can result in increased Lambda usage and increased costs. [Learn more](#)

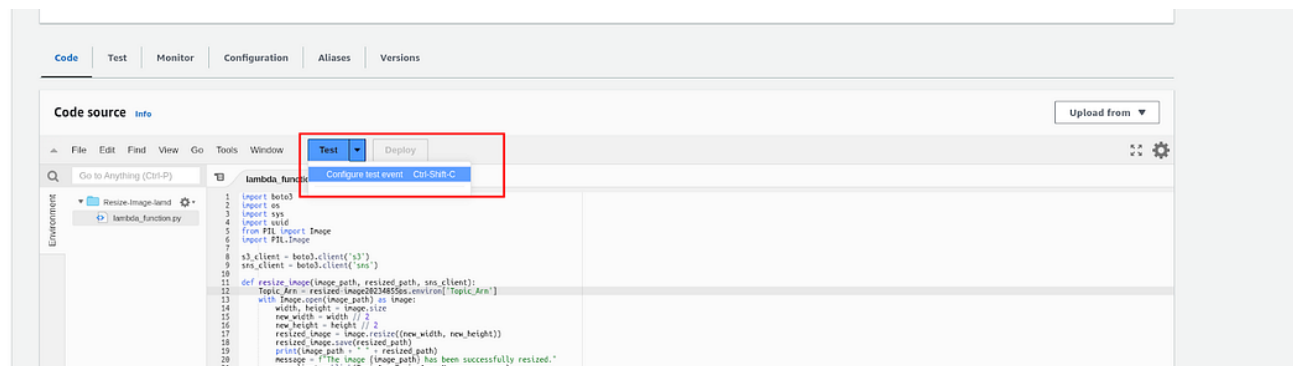
☒ I acknowledge that using the same S3 bucket for both input and output is not recommended and that this configuration can cause recursive invocations, increased Lambda usage, and increased costs.

Lambda will add the necessary permissions for AWS S3 to invoke your Lambda function from this trigger. [Learn more](#) about the Lambda permissions model.

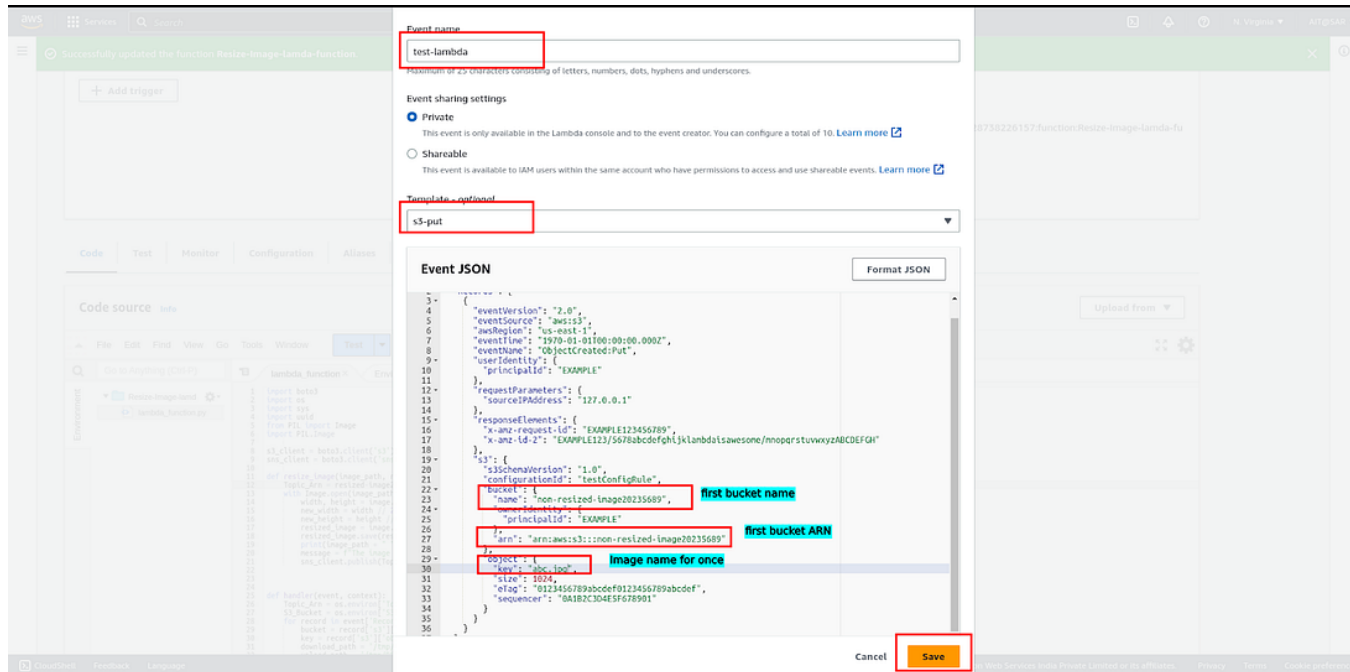
Cancel **Add**

27. Now, we have configured all the things related to the **Lambda function**. It's time to test it.

To do that click on **Test -> Configure test event**.



28. Add the **Event-name**, the template should be **s3 put** replace the few things that are given in the below screenshot, and click on **Save**.



29. Now, don't run the lambda manually. Just upload an image to **Bucket1** and see the magic below.

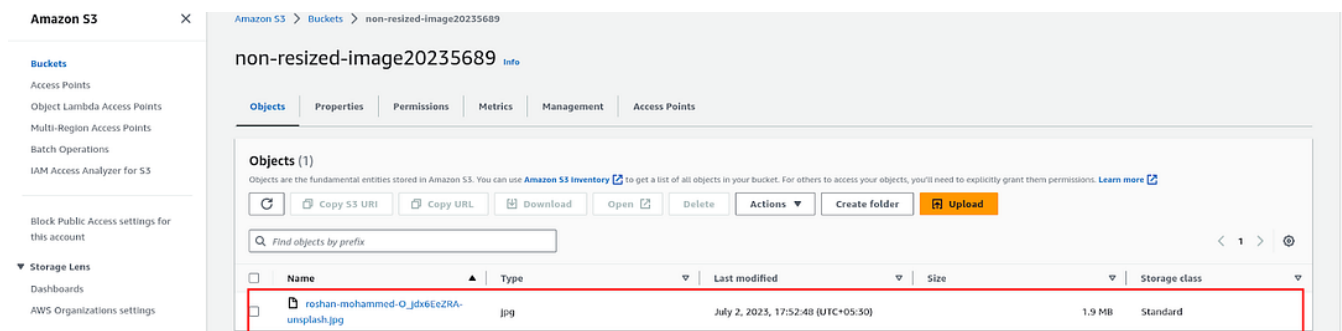
...

Outputs:

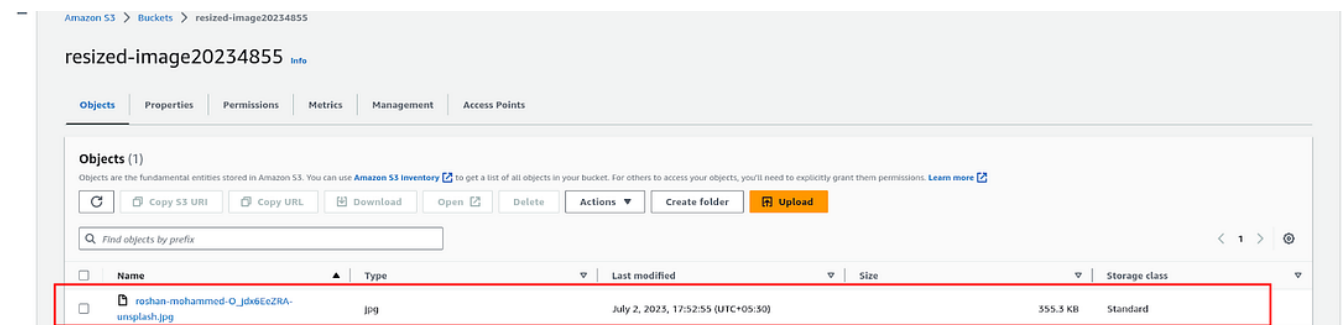
Uploading an Image to the Bucket1.

```
amanpathak@pop-os:~/Downloads$ aws s3 cp roshan-mohammed-0_jdx6EeZRA-unsplash.jpg s3://non-resized-image20235689
upload: ./roshan-mohammed-0_jdx6EeZRA-unsplash.jpg to s3://non-resized-image20235689/roshan-mohammed-0_jdx6EeZRA-unsplash.jpg
amanpathak@pop-os:~/Downloads$
```

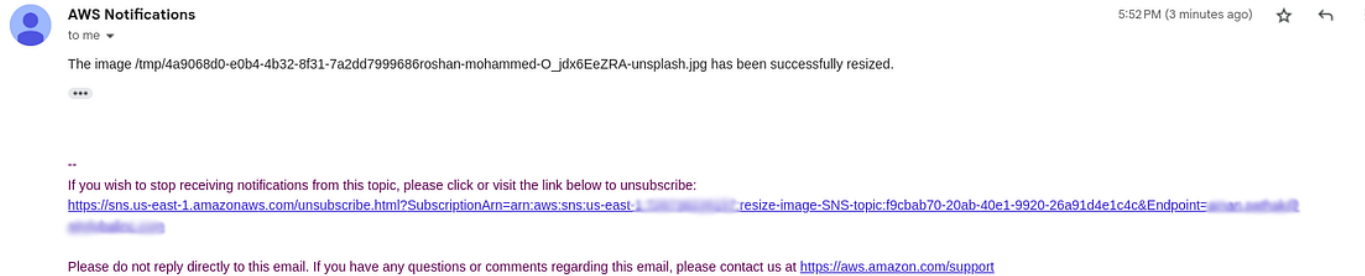
The Image is of 1.9MB in the Bucket 1.



The Image has been resized automatically from 1.9MB to 355.3KB.



Hurray, We have received the mail as well.



. . .

In conclusion, we have successfully implemented an automated image resizing system using AWS S3, Lambda, and SNS. This solution allows us to store the original images in one bucket and automatically resize and store them in another bucket. The integration of Lambda and SNS provides real-time notifications about the resizing process. By leveraging these powerful AWS services, we have simplified the image management workflow, improved efficiency, and enhanced user experience. Stay tuned for more exciting projects and follow us on social media for further updates and engaging content. Join us in the cloud revolution and unlock the potential of automation for your applications.

To stay updated with more exciting tutorials, AWS best practices, and cloud-related content, make sure to follow our blog and subscribe to our

newsletter. By staying connected, you will unlock a wealth of knowledge and discover new ways to leverage cloud technologies for your projects.

Remember, the power of automation and cloud computing is at your fingertips. Embrace it, innovate, and unlock the full potential of your applications and services.

Feel free to reach out to me if you have any doubts or queries.

Stay tuned for more exciting content and happy cloud computing!

Follow us on:

LinkedIn- <https://www.linkedin.com/in/aman-devops/>

GitHub- <https://github.com/AmanPathak-DevOps/>

Don't miss out on future updates and exclusive content. Subscribe to our newsletter now and be part of the cloud revolution.

Together, let's transform the way we build and deploy applications in the cloud!