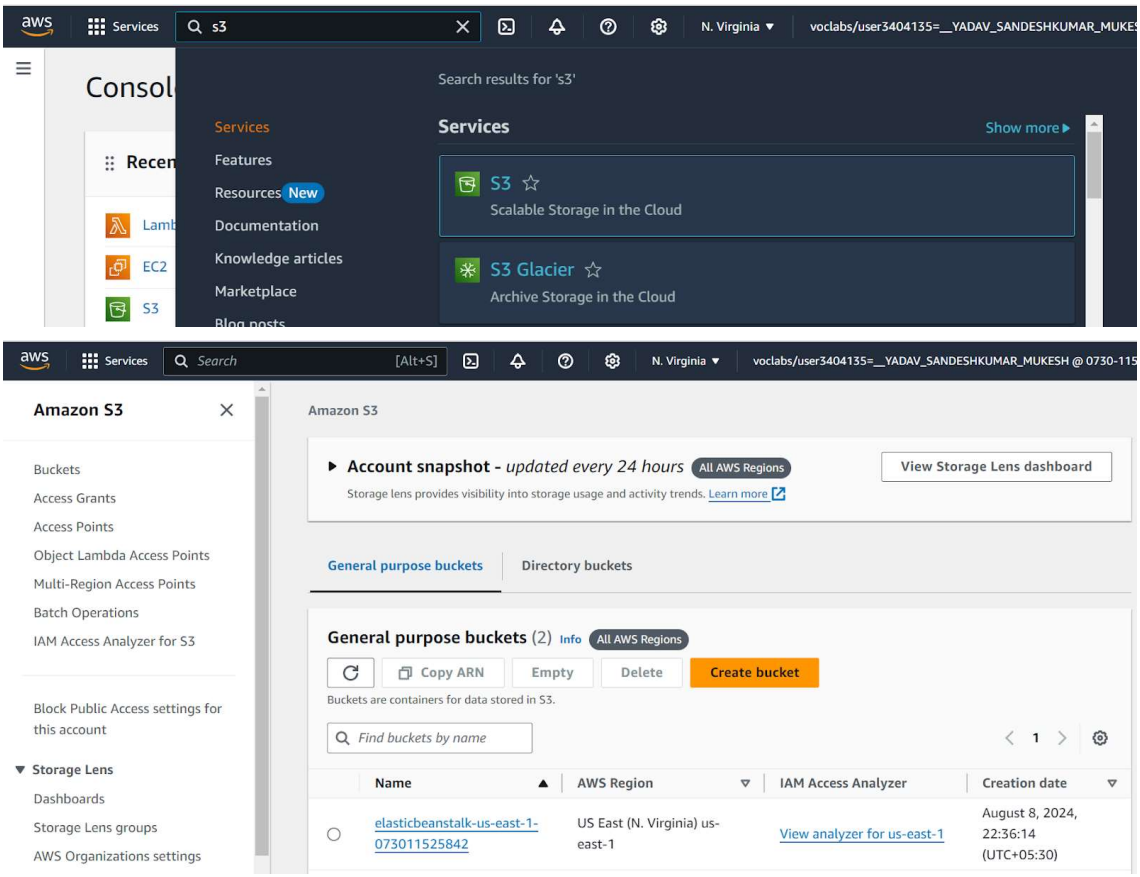
Name : Sandeshkumar.M.Yadav **Div : D15C** **Roll : 61**

Experiment No :12

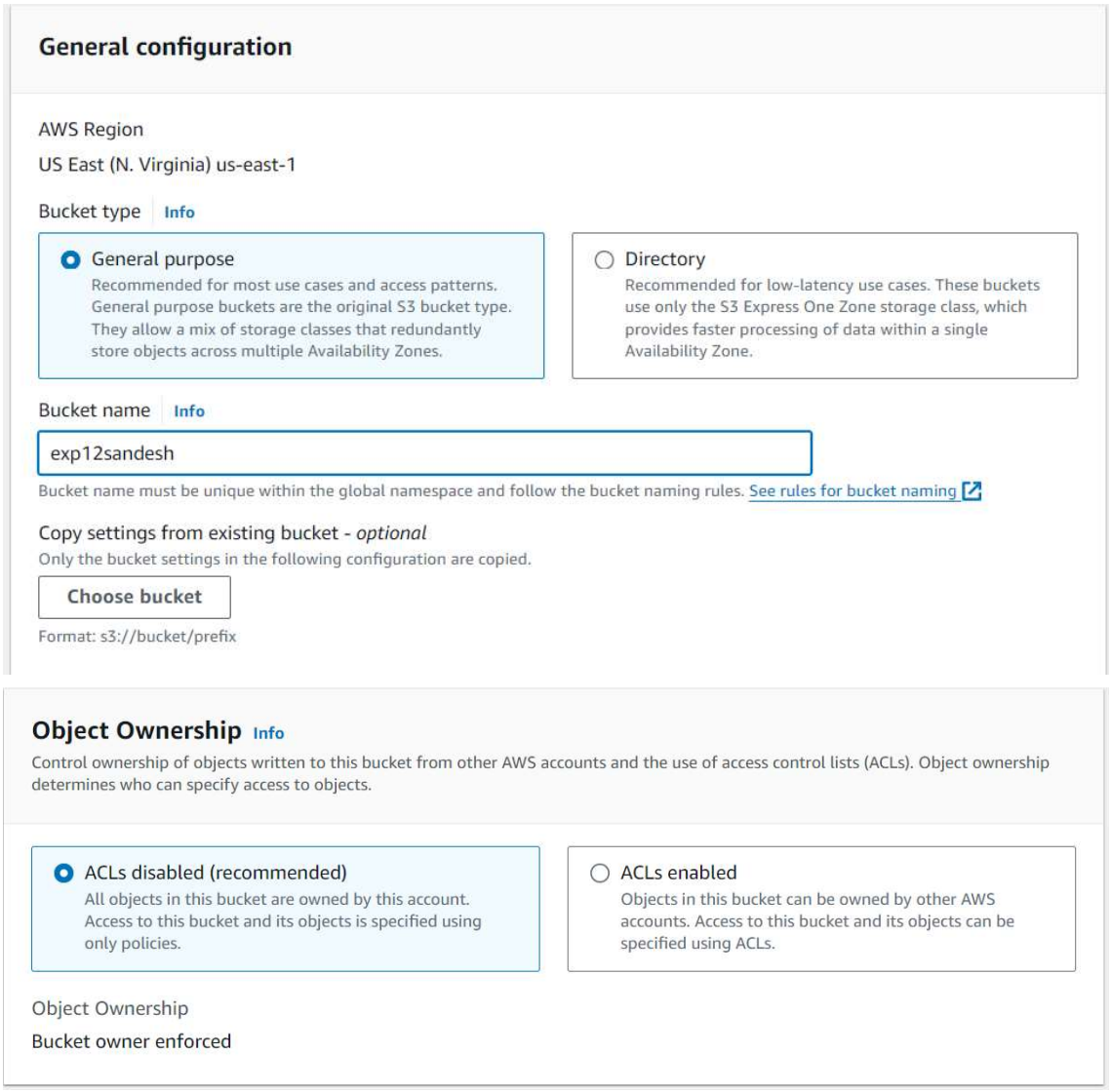
AIM : To create a Lambda function which will log “An Image has been added” once you add an object to a specific bucket in S3

CREATING LAMBDA FUNCTION :

Step 1: Log in to your AWS Personal account. Then go to S3 in the services menu and click on "Create S3 Bucket."



Step 2: Give your bucket a name, select "General purpose project," then uncheck "Block public access." Keep the other settings as they are.



Block Public Access settings for this bucket

Public access is granted to buckets and objects through access control lists (ACLs), bucket policies, access point policies, or all. In order to ensure that public access to this bucket and its objects is blocked, turn on Block all public access. These settings apply only to this bucket and its access points. AWS recommends that you turn on Block all public access, but before applying any of these settings, ensure that your applications will work correctly without public access. If you require some level of public access to this bucket or objects within, you can customize the individual settings below to suit your specific storage use cases. Learn more

- ☐ **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.

⚠ **Turning off block all public access might result in this bucket and the objects within becoming public**

AWS recommends that you turn on block all public access, unless public access is required for specific and verified use cases such as static website hosting.

☐ I acknowledge that the current settings might result in this bucket and the objects within becoming public.

✓ **Successfully created bucket "exp12sandesh"**
To upload files and folders, or to configure additional bucket settings, choose **View details**.

View details ×

Amazon S3 > Buckets

▸ **Account snapshot** - *updated every 24 hours* All AWS Regions
Storage lens provides visibility into storage usage and activity trends. Learn more

View Storage Lens dashboard

General purpose buckets | Directory buckets

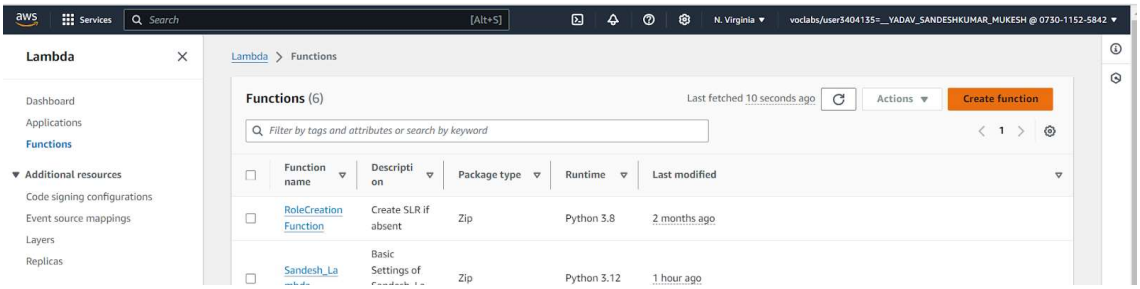
General purpose buckets (3) Info All AWS Regions
Buckets are containers for data stored in S3.

Copy ARN | Empty | Delete | Create bucket

Find buckets by name

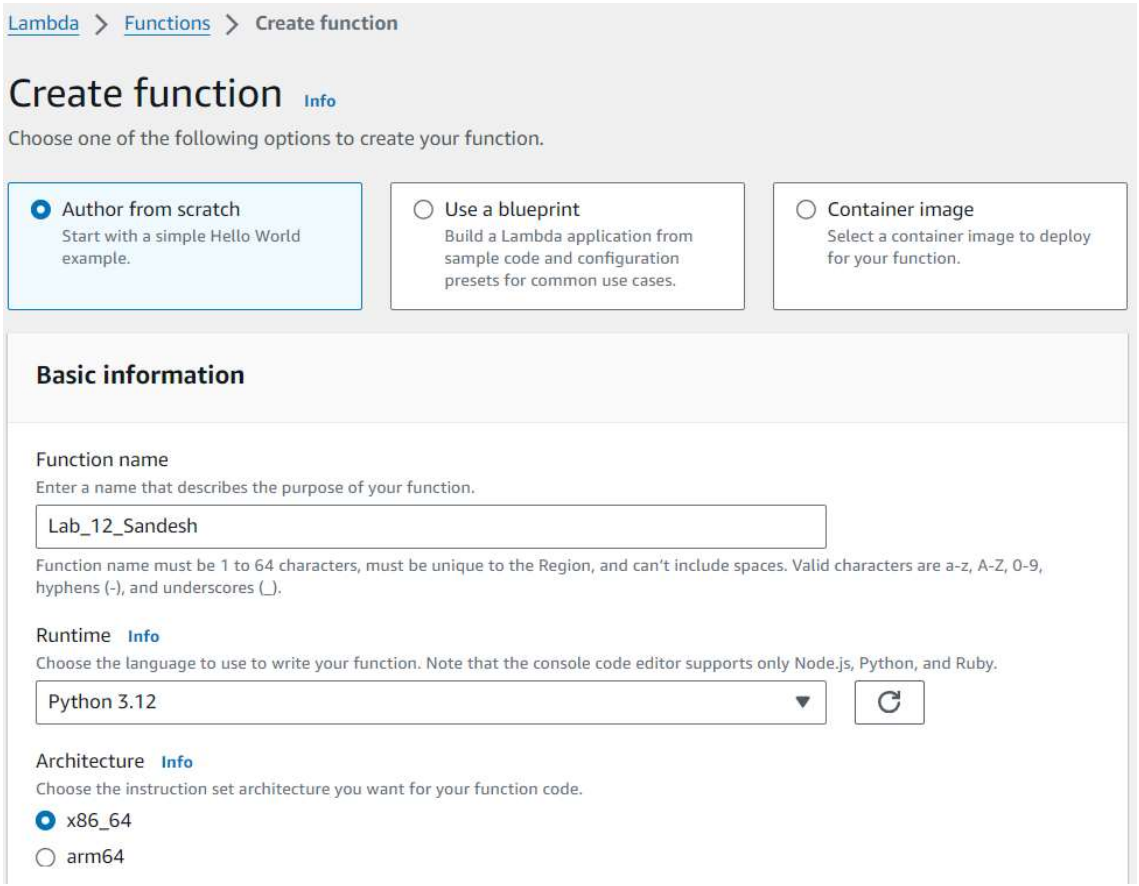
Name	AWS Region	IAM Access Analyzer	Creation date
elasticbeanstalk-us-east-1-073011525842	US East (N. Virginia) us-east-1	View analyzer for us-east-1	August 8, 2024, 22:36:14 (UTC+05:30)
exp12sandesh	US East (N. Virginia) us-east-1	View analyzer for us-east-1	October 10, 2024, 20:30:38 (UTC+05:30)
website-for-video	US East (N. Virginia) us-east-1	View analyzer for us-east-1	September 26, 2024, 20:03:00 (UTC+05:30)

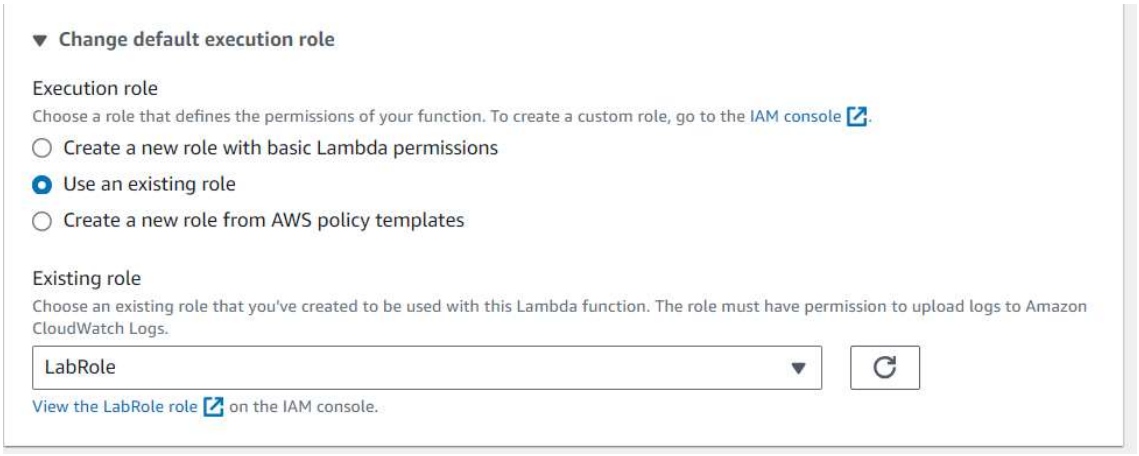
Step 3: Open lambda console and click on create function button.

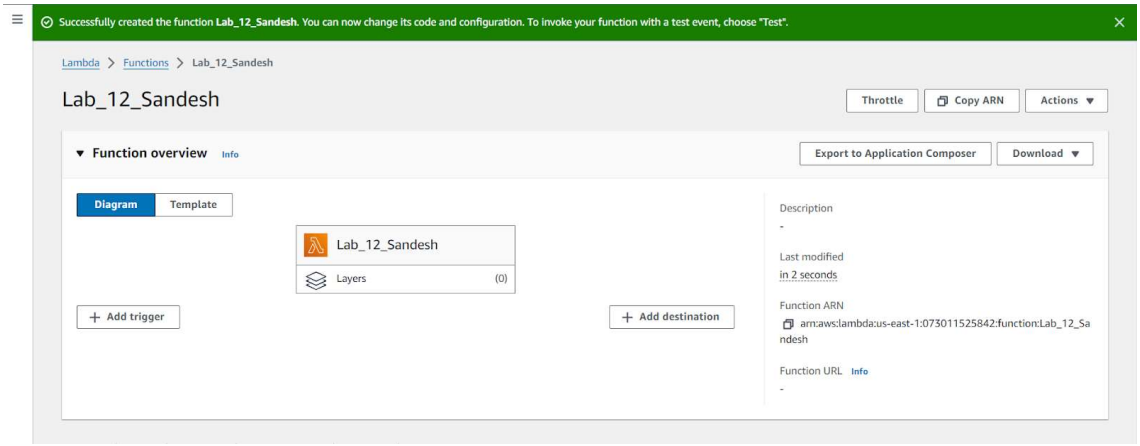


Step 4: Give your Lambda function a name and choose a programming language. The code editor only supports Node.js, Python, and Ruby, so in my case I have chosen Python 3.12. Set the architecture to x86. For the execution role, select 'Use an existing role,' then pick 'Lab role' from the dropdown menu under existing roles .

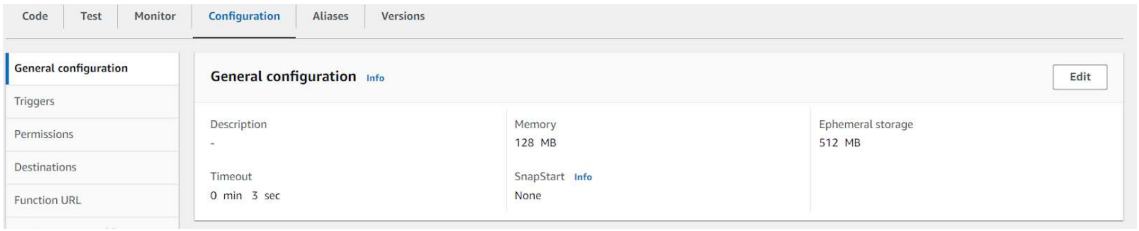
(This is because the Lab role already has the permissions needed for Lambda to run properly, so you don't need to create a new role from scratch. It's a quicker and more convenient option)

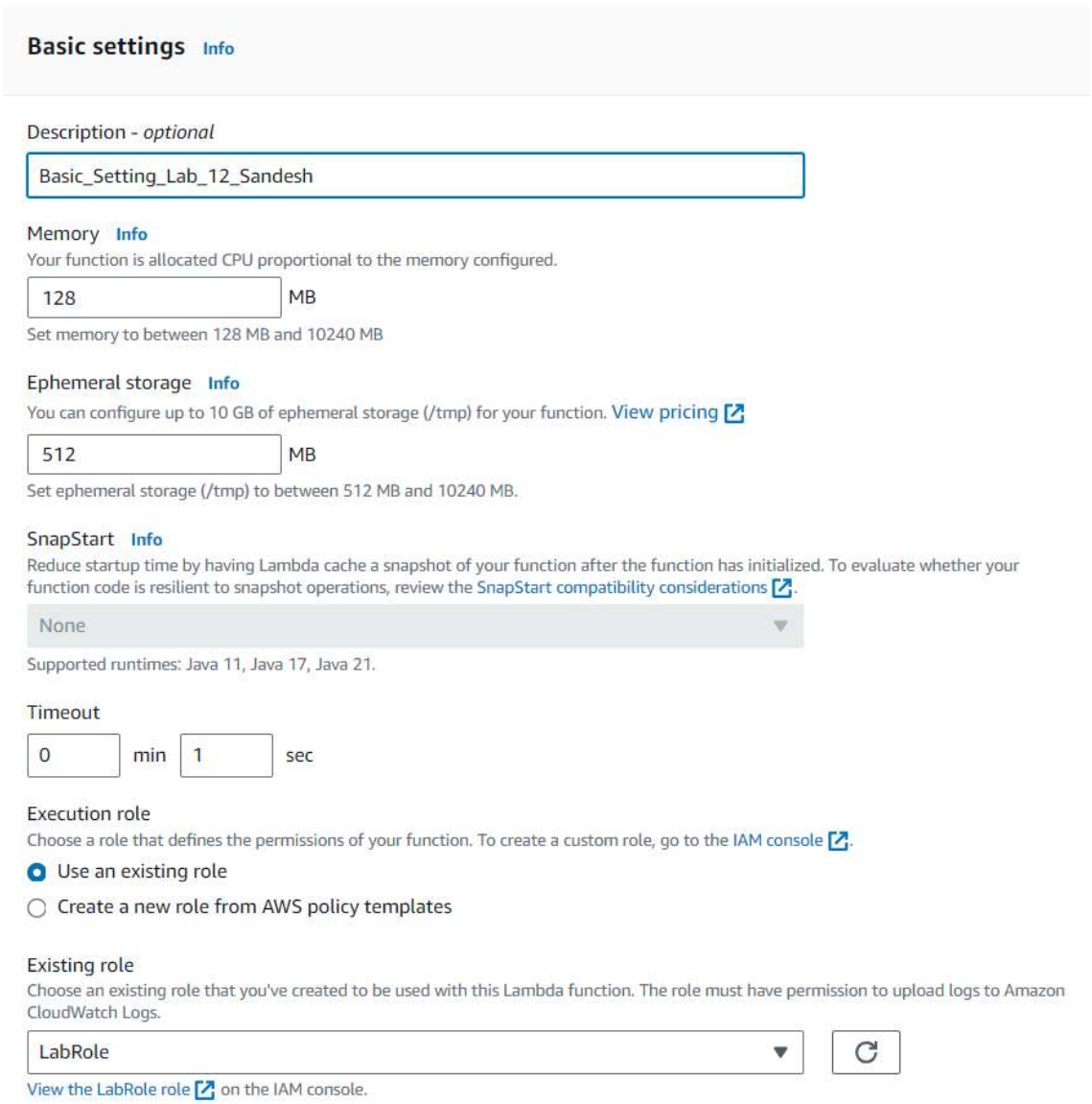



Change default execution role
Execution role
Choose a role that defines the permissions of your function. To create a custom role, go to the IAM console.
Create a new role with basic Lambda permissions
Use an existing role
Create a new role from AWS policy templates
Existing role
Choose an existing role that you've created to be used with this Lambda function. The role must have permission to upload logs to Amazon CloudWatch Logs.
LabRole
View the LabRole role on the IAM console.

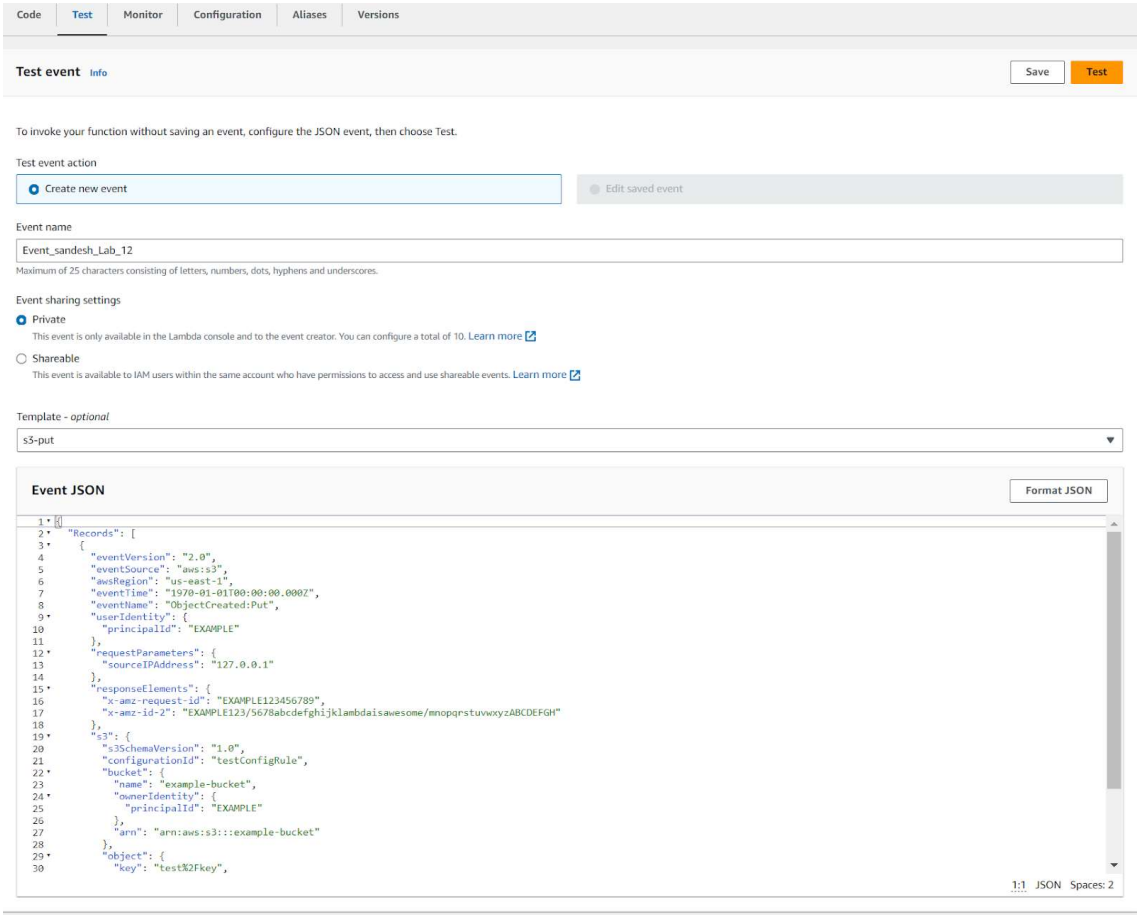

Successfully created the function Lab_12_Sandesh. You can now change its code and configuration. To invoke your function with a test event, choose "Test".
Lambda > Functions > Lab_12_Sandesh
Lab_12_Sandesh
Throttle
Copy ARN
Actions
Function overview Info
Export to Application Composer
Download
Diagram
Template
Lab_12_Sandesh
Layers (0)
Add trigger
Add destination
Description
-
Last modified
in 2 seconds
Function ARN
arn:aws:lambda:us-east-1:073011525842:function:Lab_12_Sandesh
Function URL Info
-

Step 5: To view or change the basic settings, go to the 'Configuration' tab and click 'Edit' under 'General settings.' (THIS STEP IS OPTIONAL)

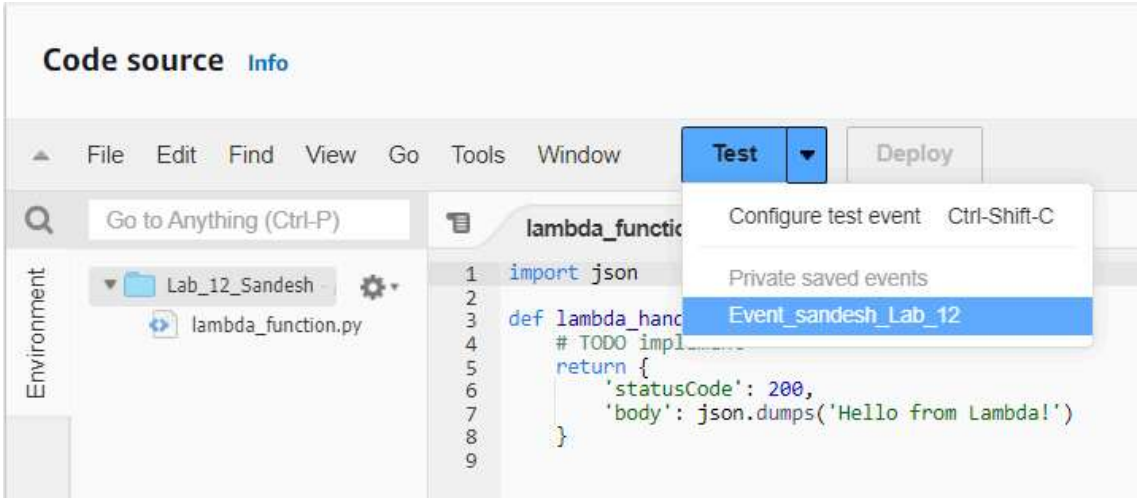




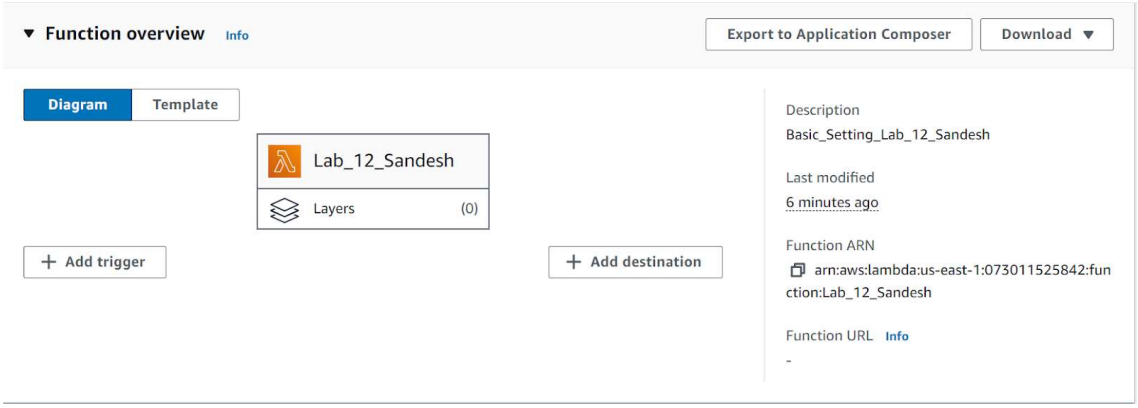
Step 6: Click on the "Test" tab, then select "Create a new event." Give the event a name, set "Event Sharing" to private, and choose the "S3 Put" template.S3 (Simple Storage Service) template allows you to test your Lambda function specifically for events related to uploading files to an S3 bucket.



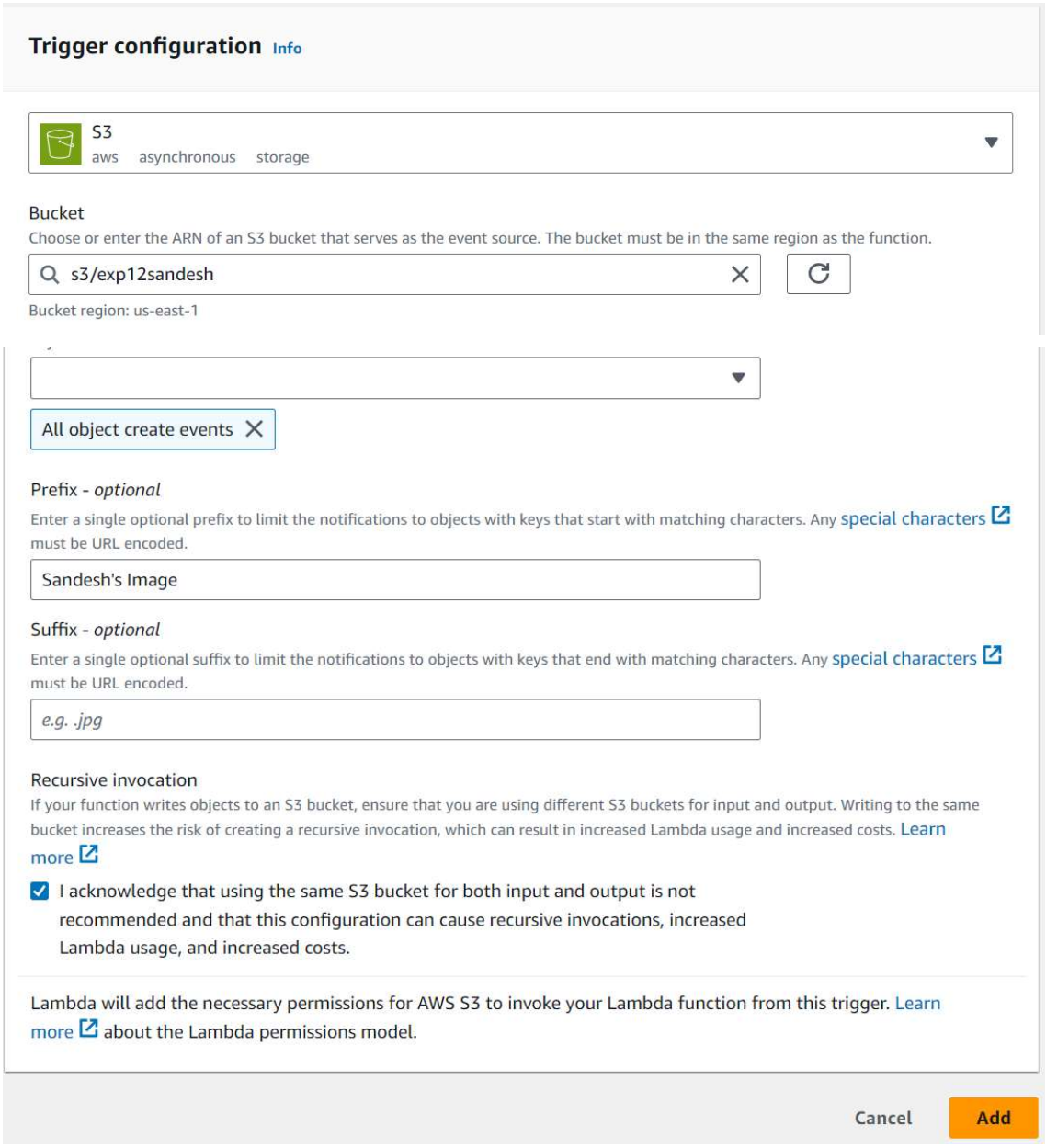
Step 7: Now In the Code section select the created event from the dropdown .

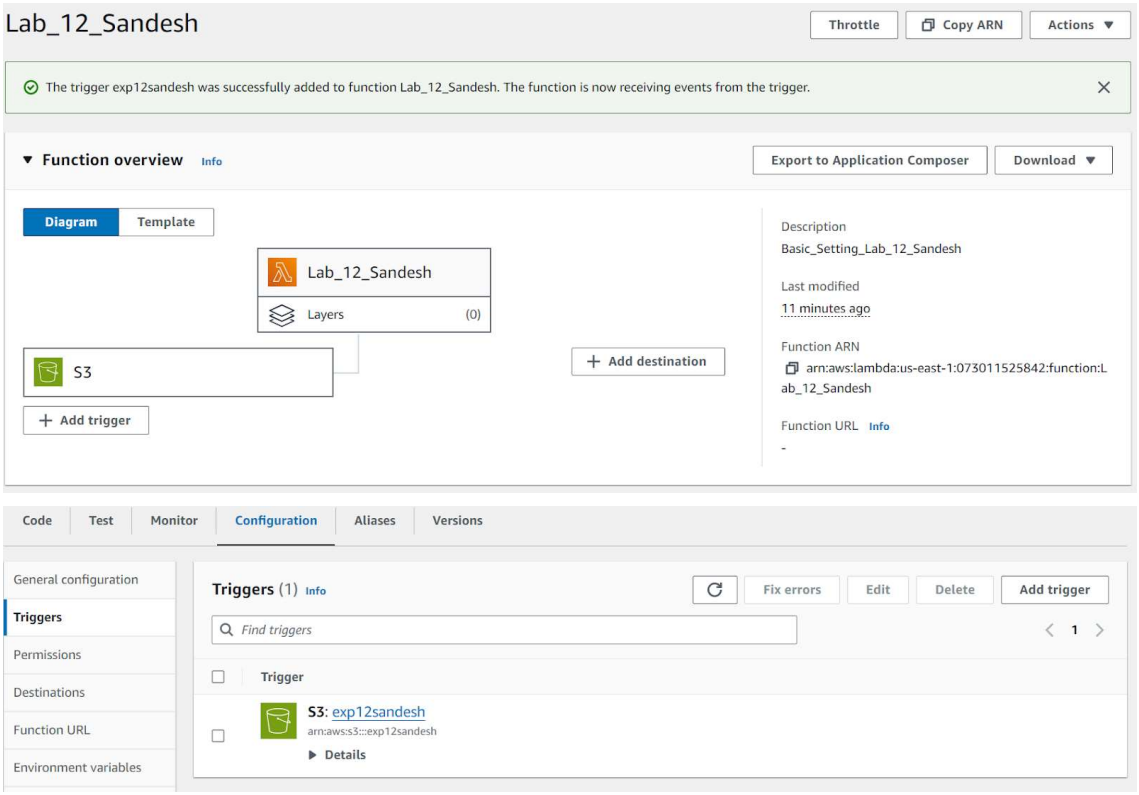


Step 8: In the Lambda function, click on "Add Trigger." Adding a trigger allows your Lambda function to automatically run in response to specific events such as uploads to an S3 bucket



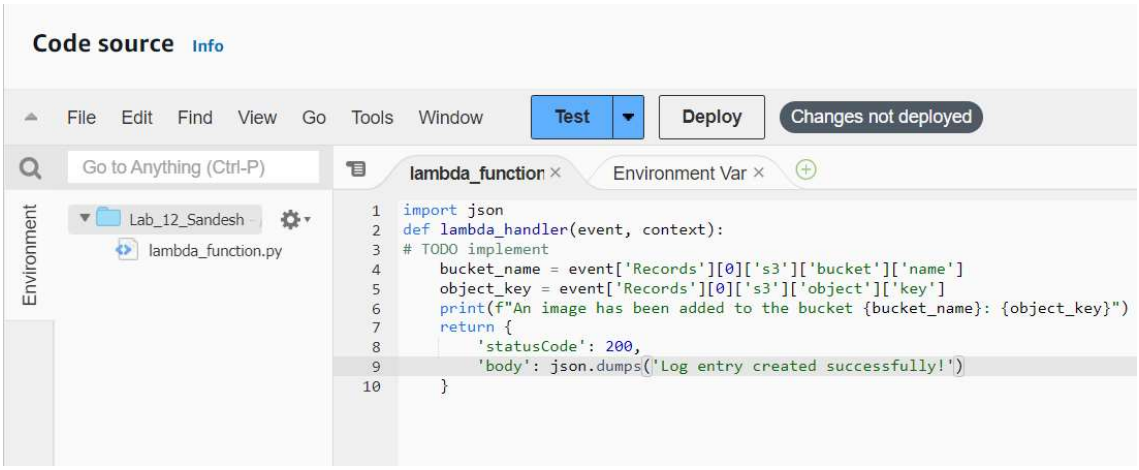
Now select the source as S3, then choose the bucket name from the dropdown menu. Keep the other settings as default, and you can also add a prefix for the image if you want.A prefix for an image (or any file) in S3 is a string that you can use to organize or filter files within a bucket. It acts like a folder name, helping to categorize your files.

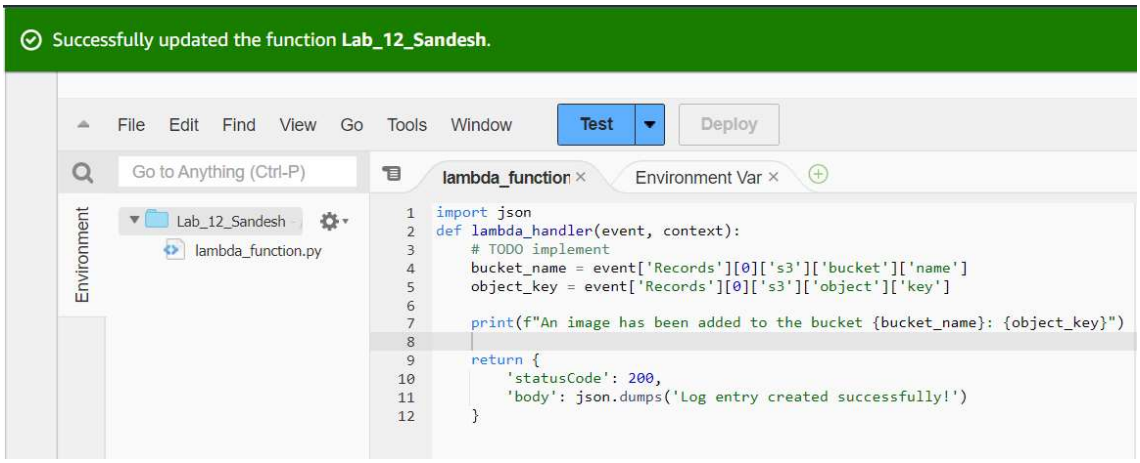

Trigger configuration Info
S3
aws asynchronous storage
Bucket
Choose or enter the ARN of an S3 bucket that serves as the event source. The bucket must be in the same region as the function.
s3/exp12sandesh
Bucket region: us-east-1
All object create events
Prefix - optional
Enter a single optional prefix to limit the notifications to objects with keys that start with matching characters. Any special characters must be URL encoded.
Sandesh's Image
Suffix - optional
Enter a single optional suffix to limit the notifications to objects with keys that end with matching characters. Any special characters must be URL encoded.
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



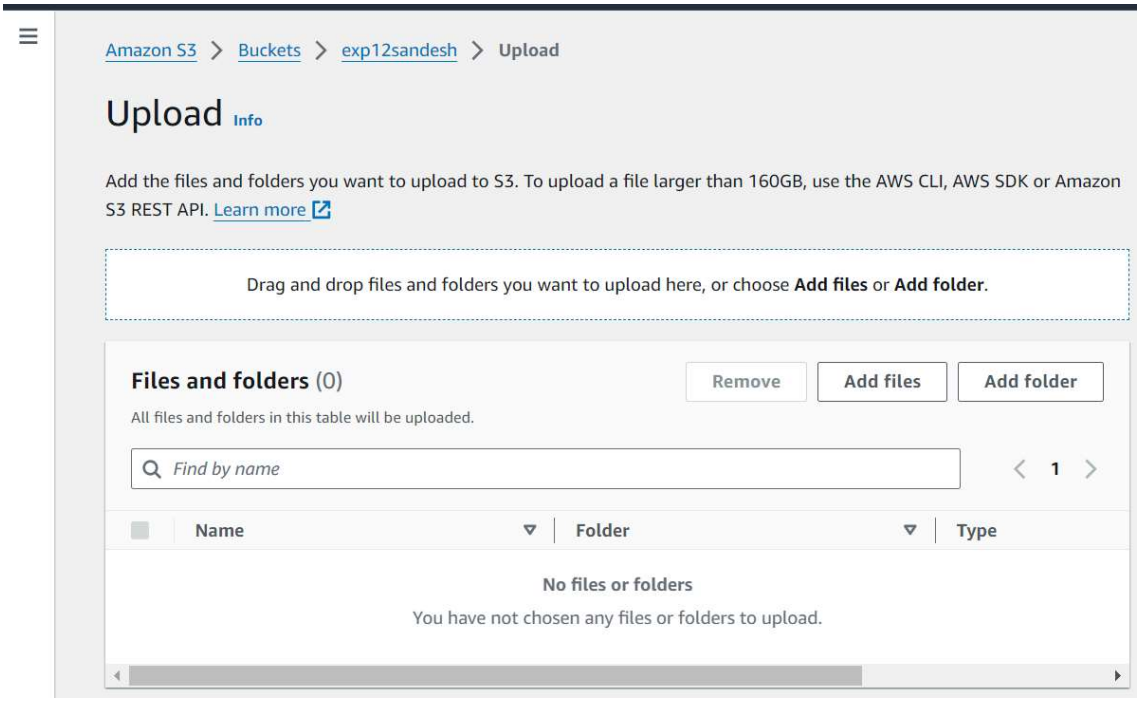
Step 9: Now Write code that logs a message like “An Image has been added” when triggered. Save the file and click on deploy.

```
import json
def lambda_handler(event, context):
# TODO implement
bucket_name = event['Records'][0]['s3']['bucket']['name']
object_key = event['Records'][0]['s3']['object']['key']
print(f"An image has been added to the bucket {bucket_name}: {object_key}")
return {
'statusCode': 200,
'body': json.dumps('Log entry created successfully!')
}
```

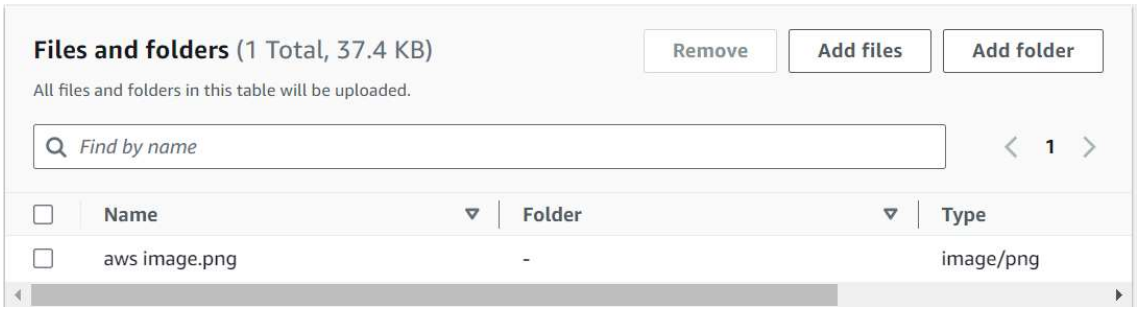




Step 10: Now we will upload any image to the bucket

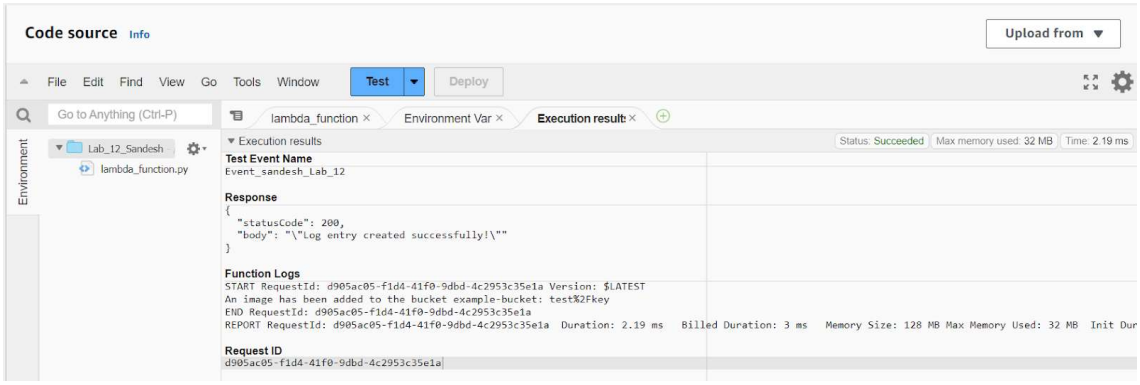


Click on add file where you can upload any image of your choice in your bucket

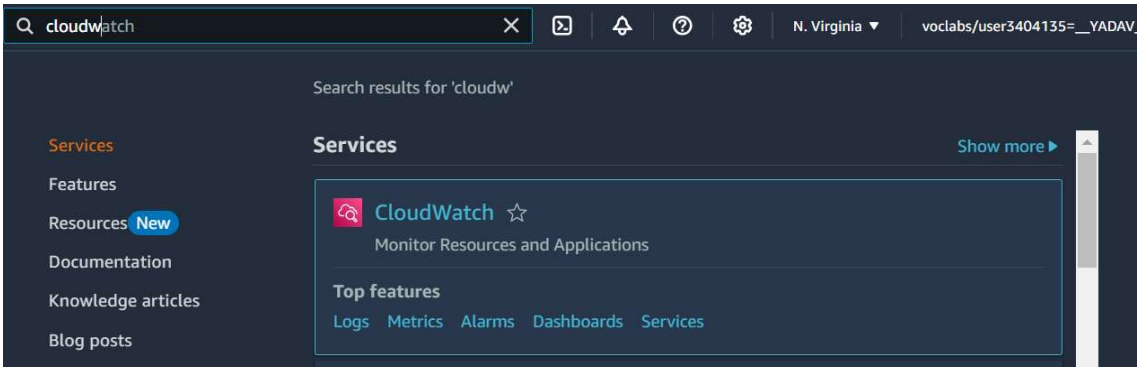


Click on Upload

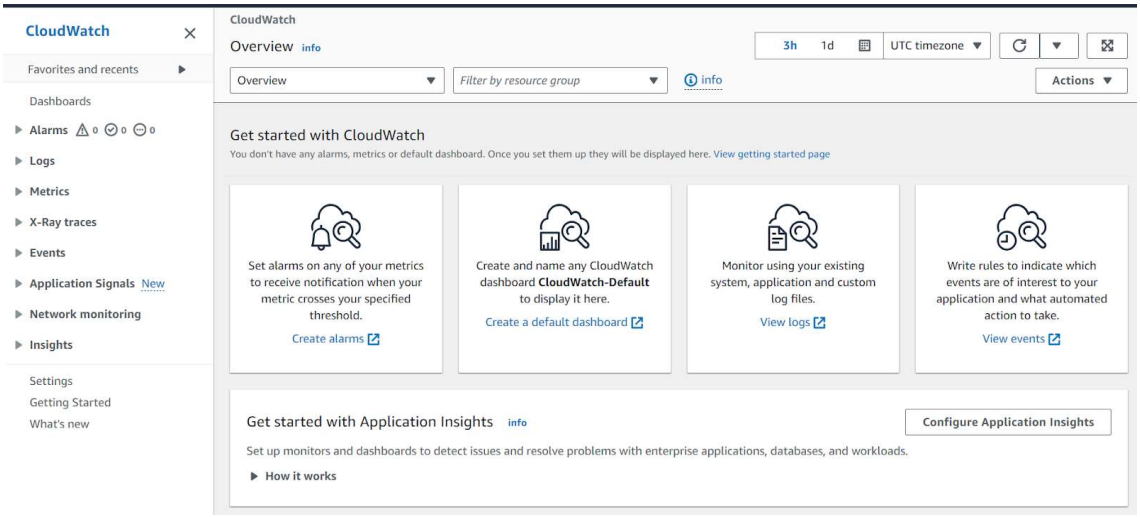
Step 11: Now click on "Test" in Lambda to see if it logs the activity when an image is added to S3.



Step 12: Now let's check the logs on CloudWatch. Go to the "Monitor" section and click on "View CloudWatch Logs.

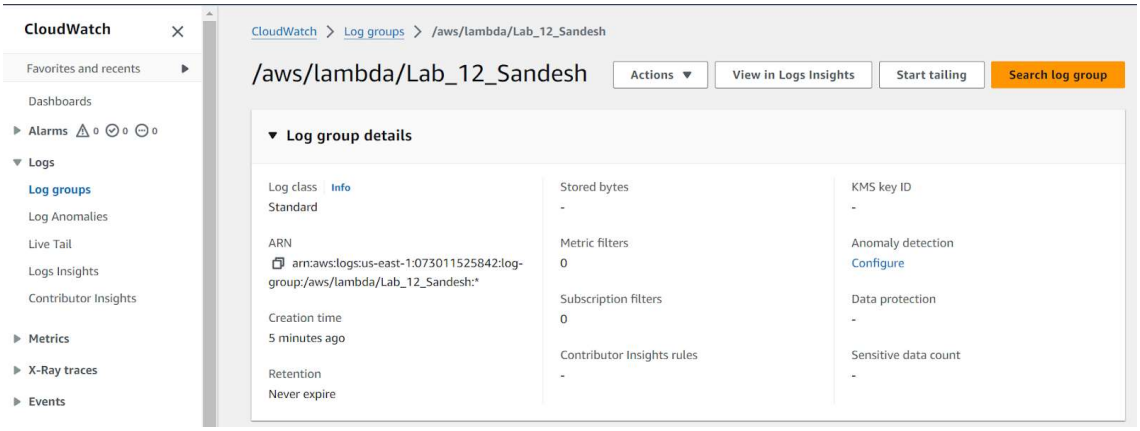


Click on the CloudWatch:

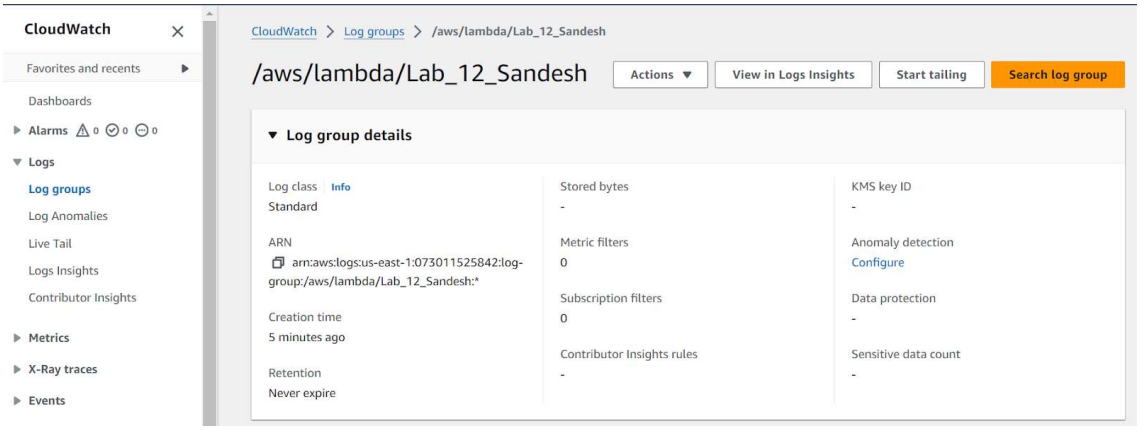


Name : Sandeshkumar.M.Yadav **Div : D15C** **Roll : 61**

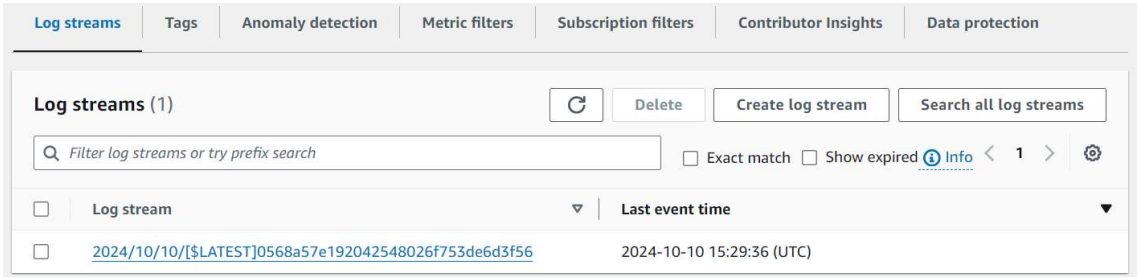
Click on the logs:

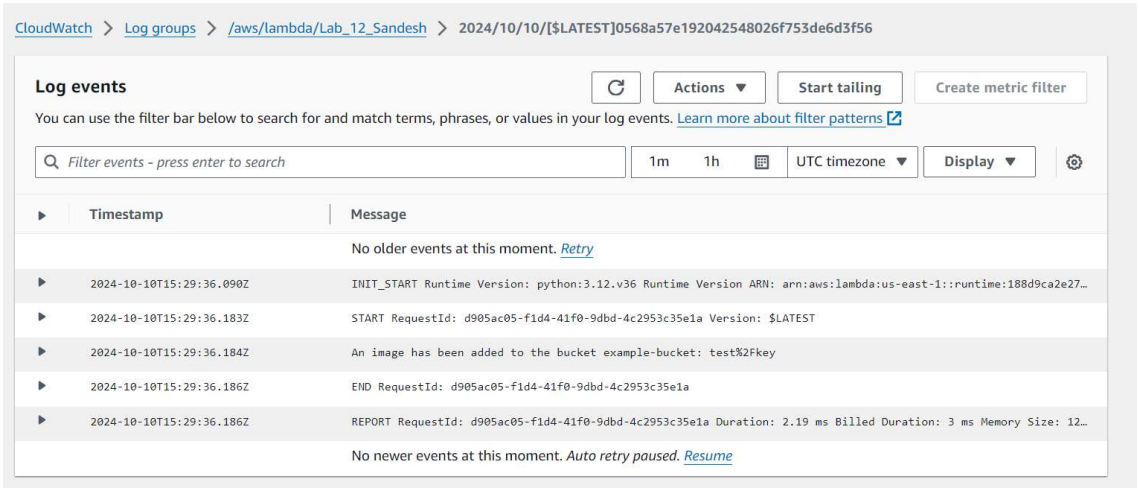


Click on the log group:



Scrolled down and click on the log stream:





CONCLUSION:

In this experiment, we successfully created an AWS Lambda function that logs a message when an image is uploaded to an S3 bucket. Using the S3 Put template ensured that the function was triggered correctly by S3 events. The experiment demonstrated Lambda's event-driven architecture, showing how it can automatically respond to file uploads in S3. Additionally, we learned how to troubleshoot common issues with event setup and verify activity logs using CloudWatch.