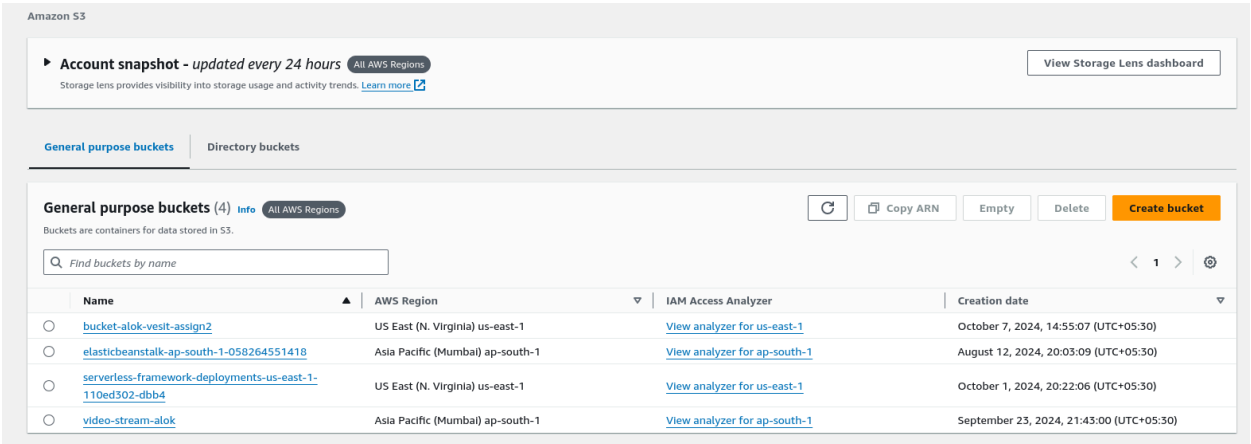
Name: Alok Yadav Div: D15C Roll No. 59

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

Steps To create the lambda function:

1. Login to your AWS Personal account. Now open S3 from services and click on create S3 bucket.



2. Now Give a name to the Bucket, select general purpose project and deselect the Block public access and keep other this to default.

Name: Alok Yadav Div: D15C Roll No. 59

Create bucket Info

Buckets are containers for data stored in S3.

General configuration

AWS Region

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

Bucket type Info

- General purpose
  Recommended for most use cases and access patterns. General purpose buckets are the original S3 bucket type. They allow a mix of storage classes that redundantly store objects across multiple Availability Zones.
- Directory
  Recommended for low-latency use cases. These buckets use only the S3 Express One Zone storage class, which provides faster processing of data within a single Availability Zone.

Bucket name Info

alok-exp-12-lambda

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

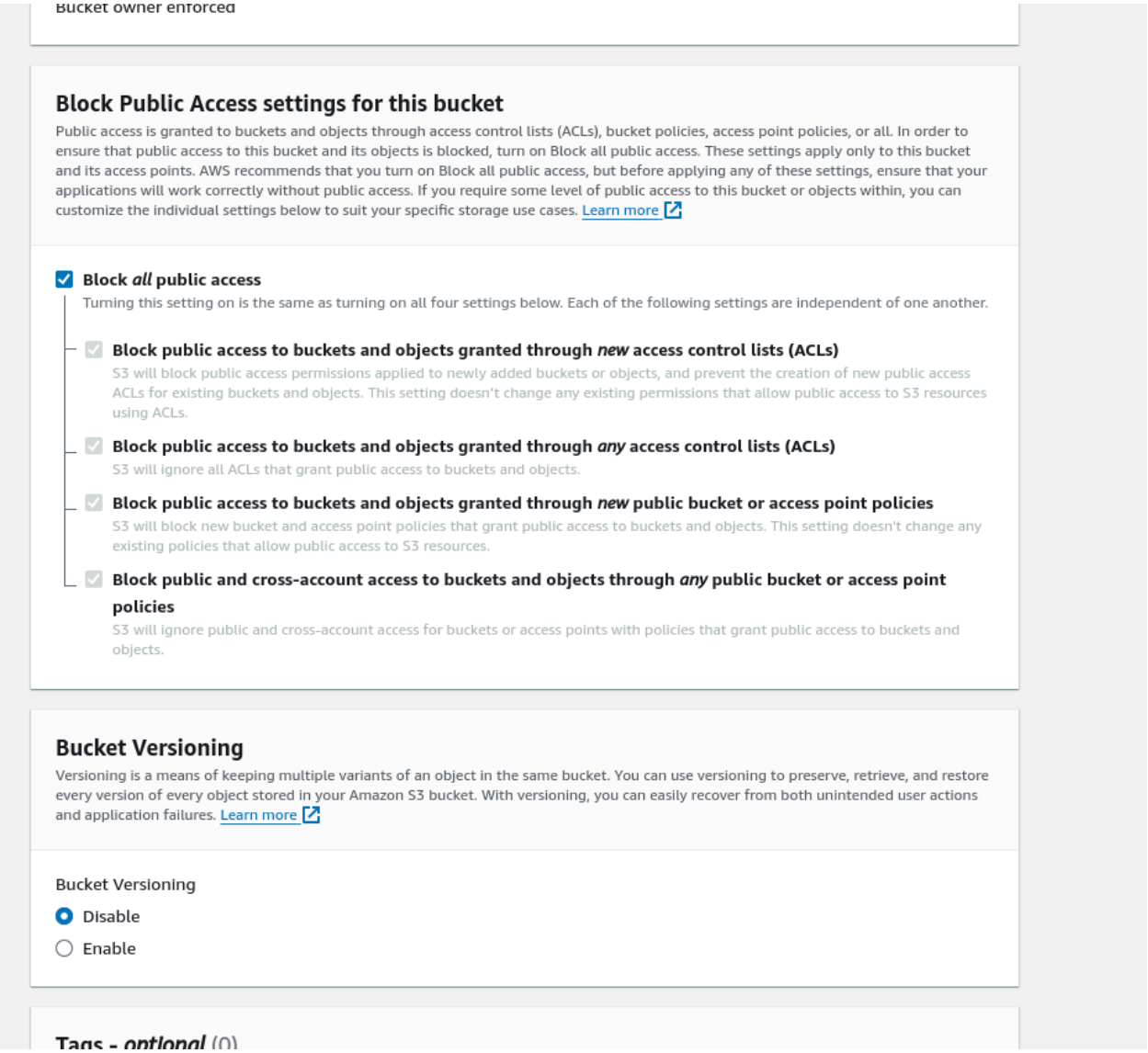
Object Ownership Info

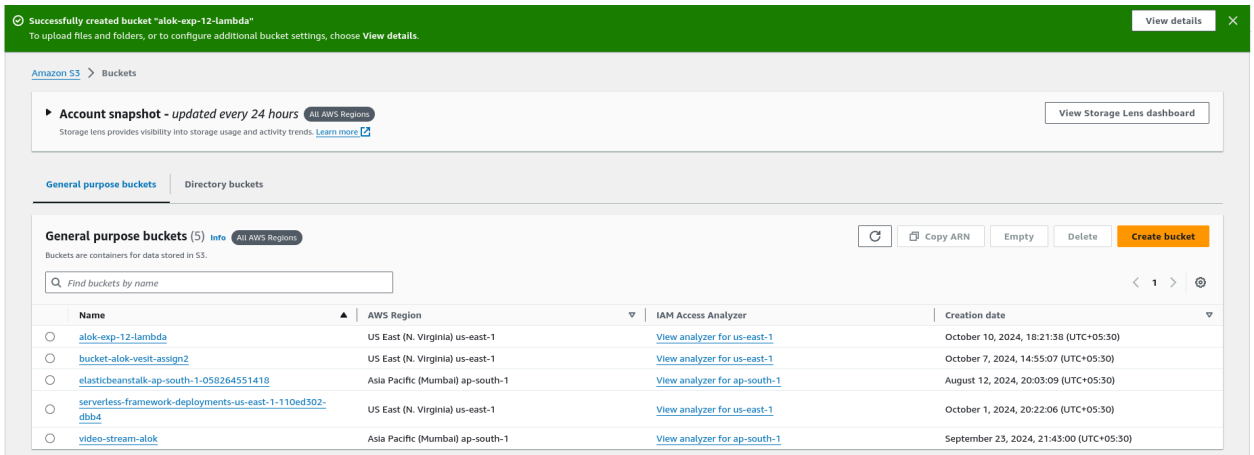
Control ownership of objects written to this bucket from other AWS accounts and the use of access control lists (ACLs). Object ownership determines who can specify access to objects.

- ACLs disabled (recommended)
  All objects in this bucket are owned by this account. Access to this bucket and its objects is specified using only policies.
- ACLs enabled
  Objects in this bucket can be owned by other AWS accounts. Access to this bucket and its objects can be specified using ACLs.

Object Ownership

Name: Alok Yadav Div: D15C Roll No. 59

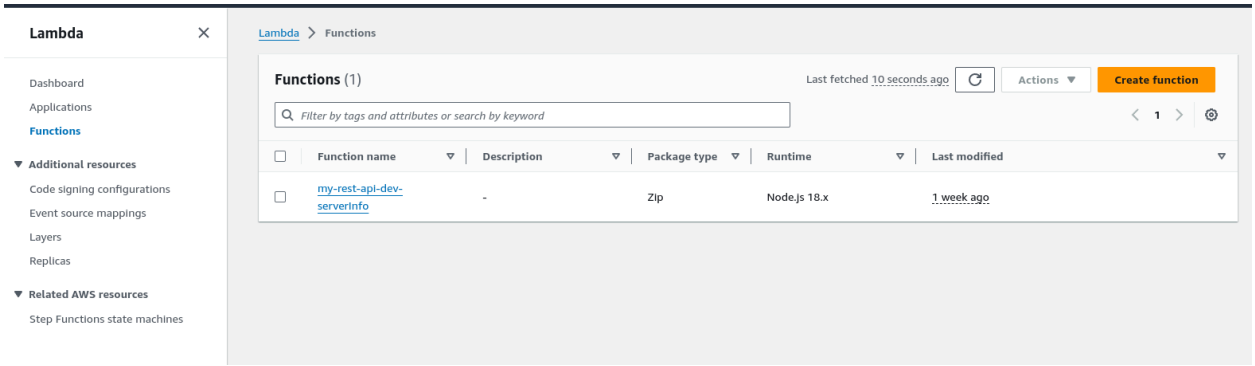




Bucket created successfully

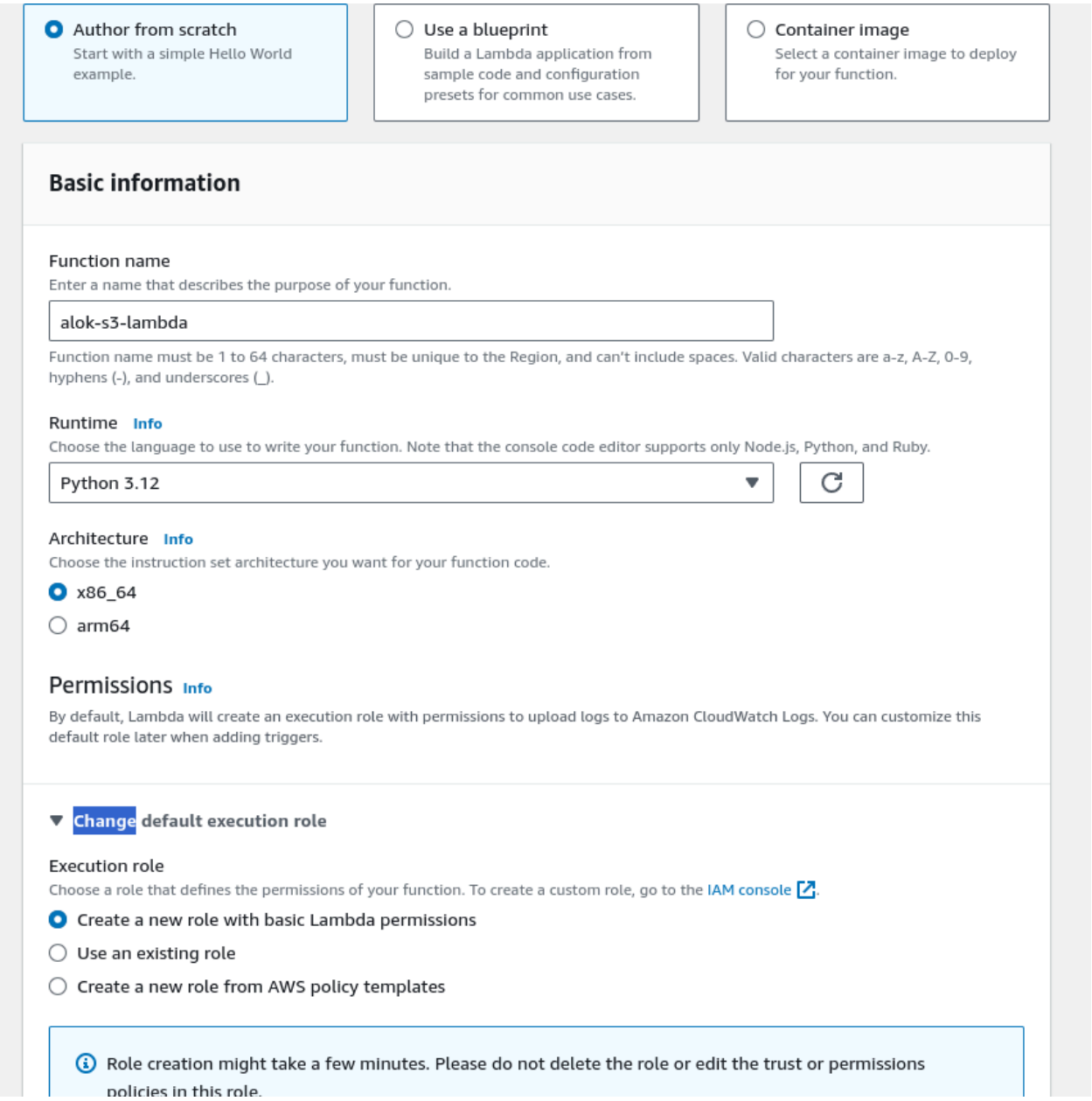
Name: Alok Yadav Div: D15C Roll No. 59

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

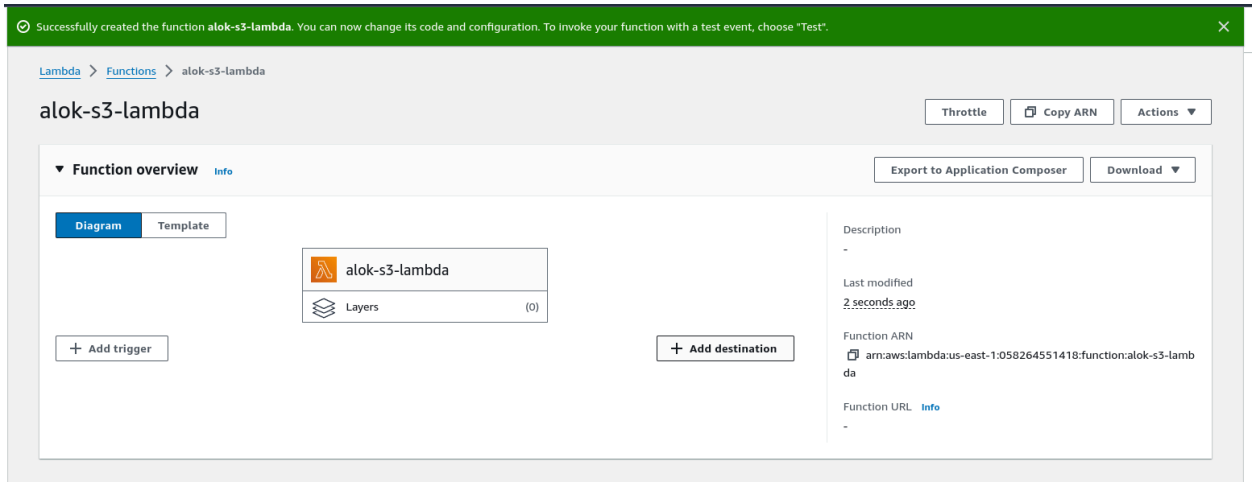


4. Now Give a name to your Lambda function, Select the language to use to write your function. Note that the console code editor supports only Node.js, Python, and Ruby. So will select Python 3.12 Architecture as x86, and Exceution role to Create a new role with basic Lambda permissions.

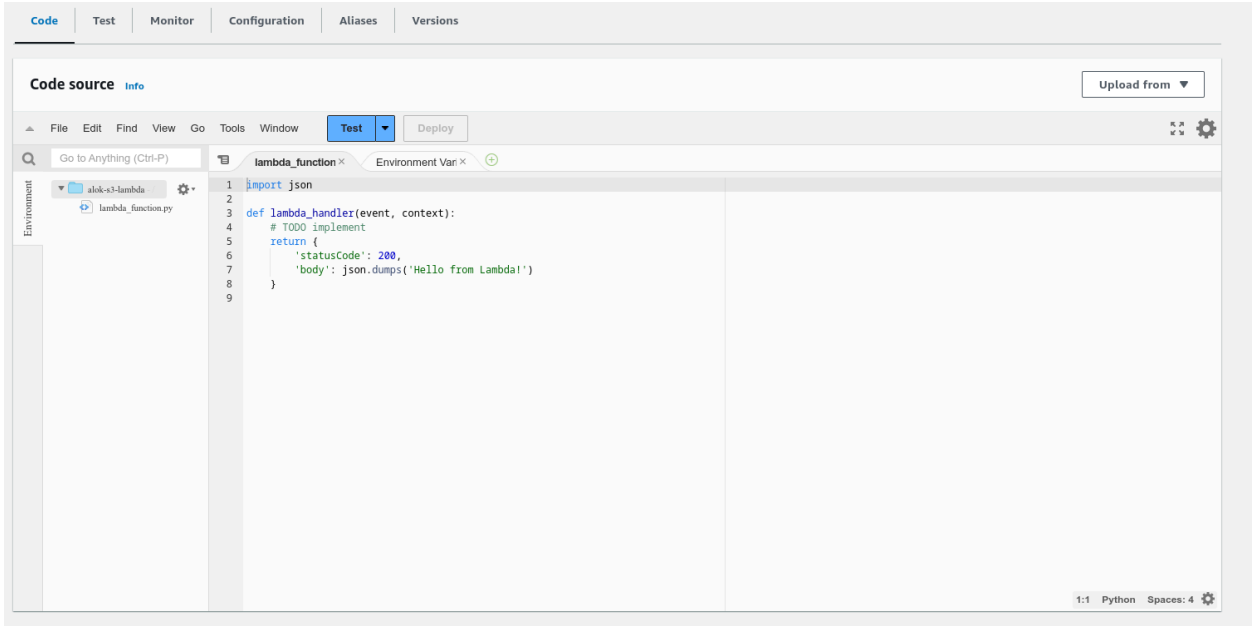
Name: Alok Yadav Div: D15C Roll No. 59



Name: Alok Yadav Div: D15C Roll No. 59

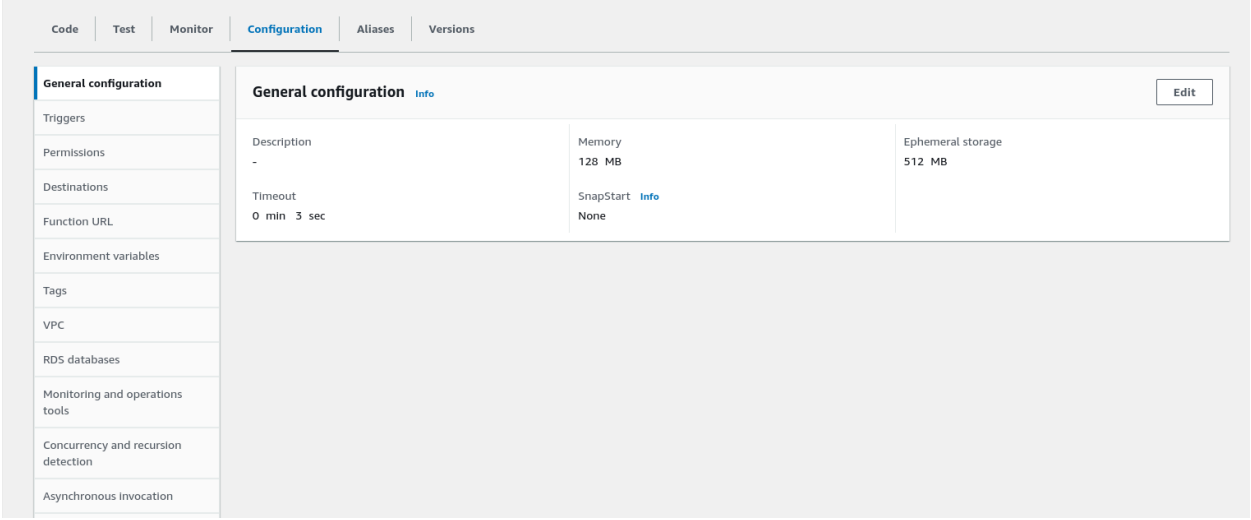


Lambda function created successfully



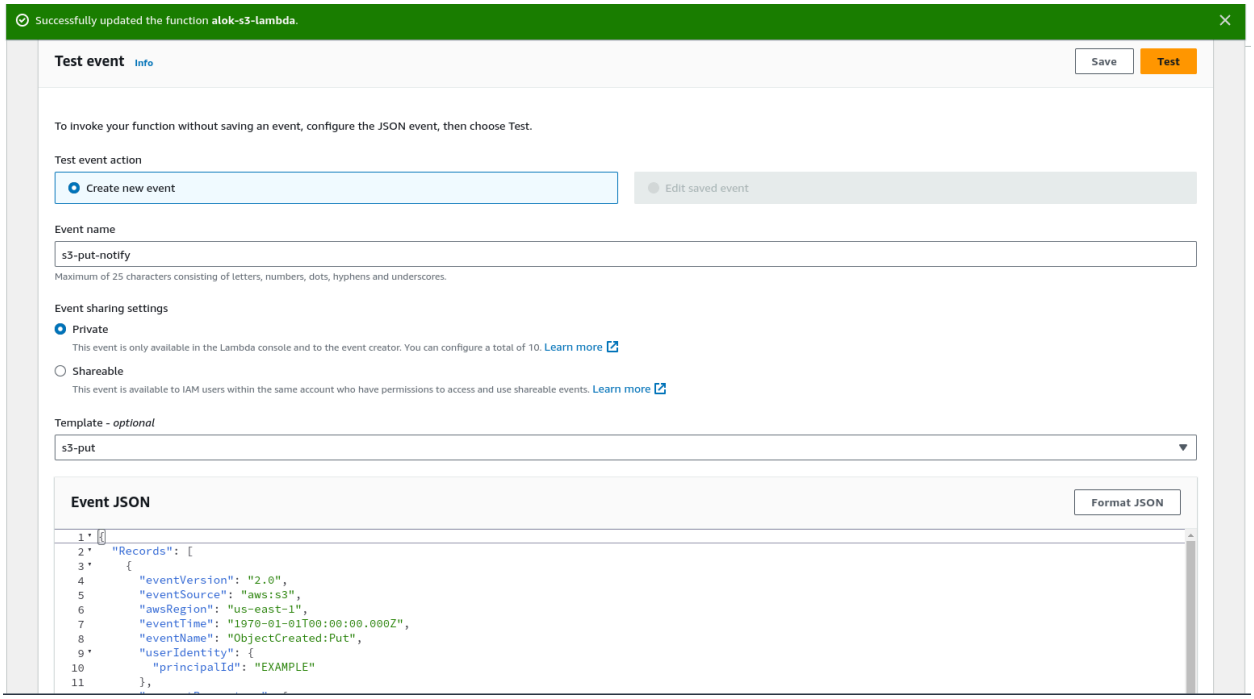
So to See or Edit the basic settings go to configuration then click on edit general setting.

Name: Alok Yadav Div: D15C Roll No. 59



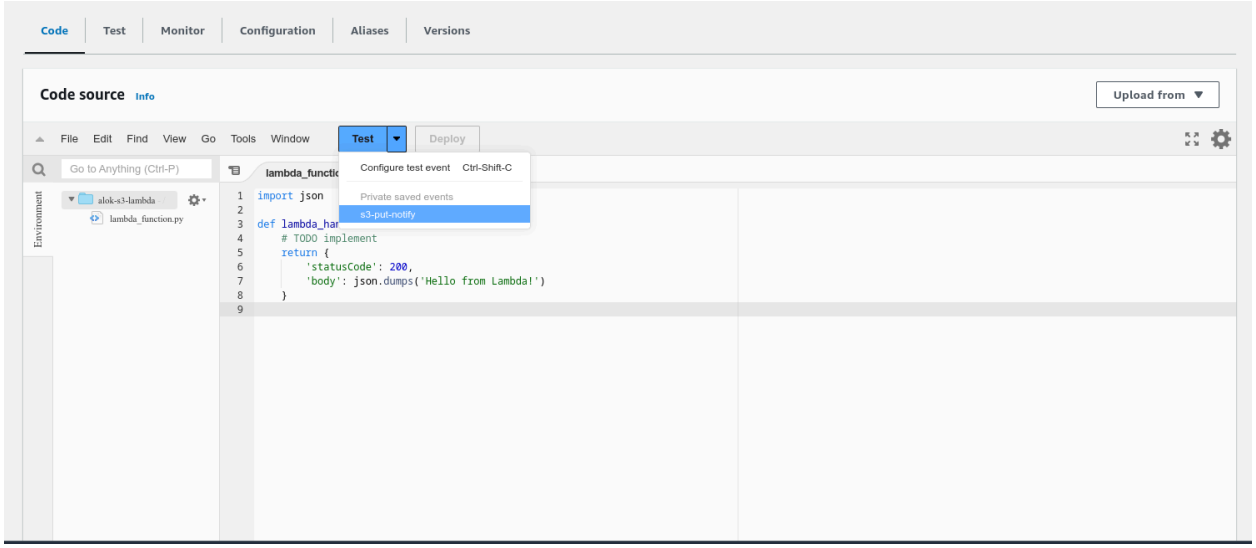
Here, you can enter a description and change Memory and Timeout. I've changed the
Timeout period to 1 sec since that is sufficient for now.

5. Now Click on the Test tab then select Create a new event, give a name to the event and select Event Sharing to private, and select s3 put template.

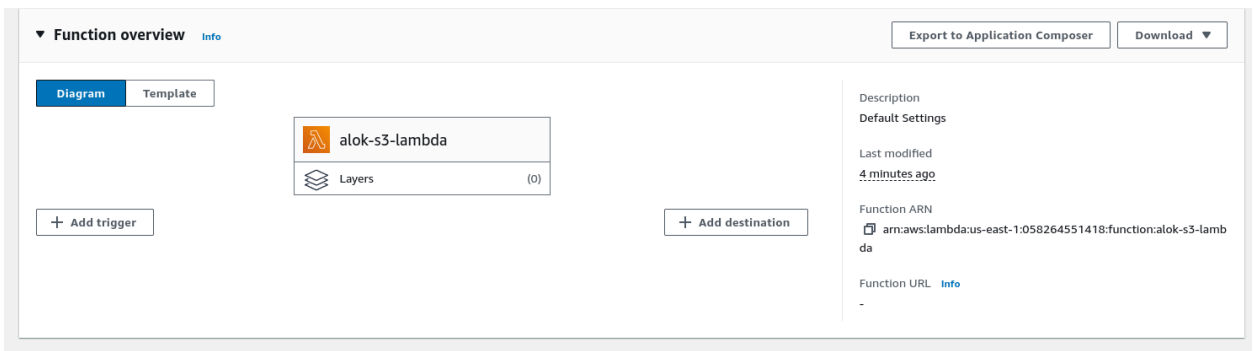


6. Now In Code section select the created event from the dropdown .

Name: Alok Yadav Div: D15C Roll No. 59



7. Now In the Lambda function click on add tigger.



Now select the source as S3 then select the bucket name from the dropdown, keep other things to default and also you can add prefix to image.

Name: Alok Yadav Div: D15C Roll No. 59

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/alok-exp-12-lambda

Bucket region: us-east-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. Any special characters must be URL encoded.

e.g. images/

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.

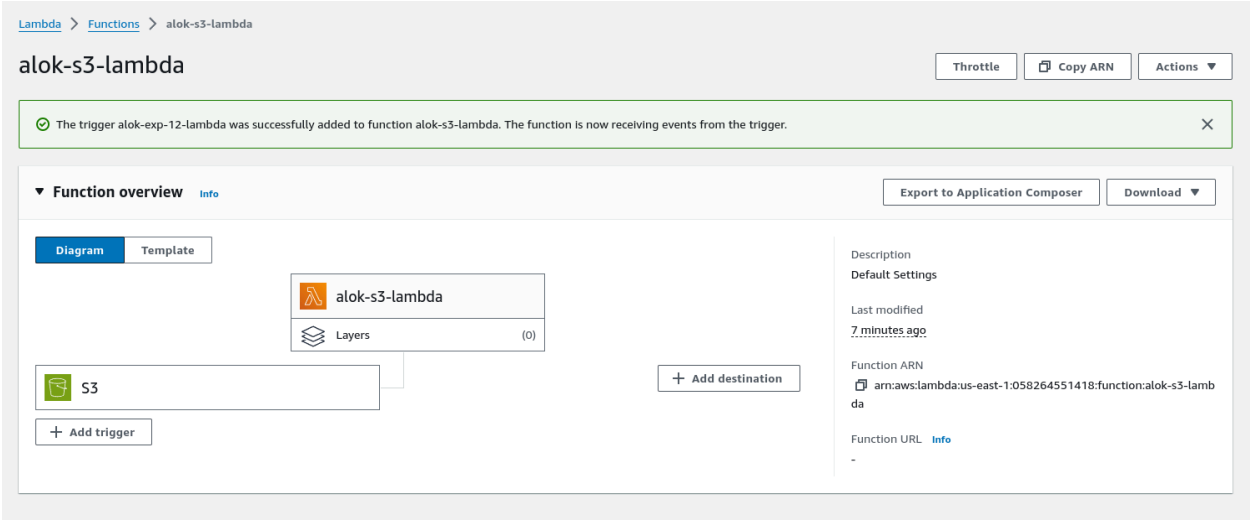
e.g. .jpg

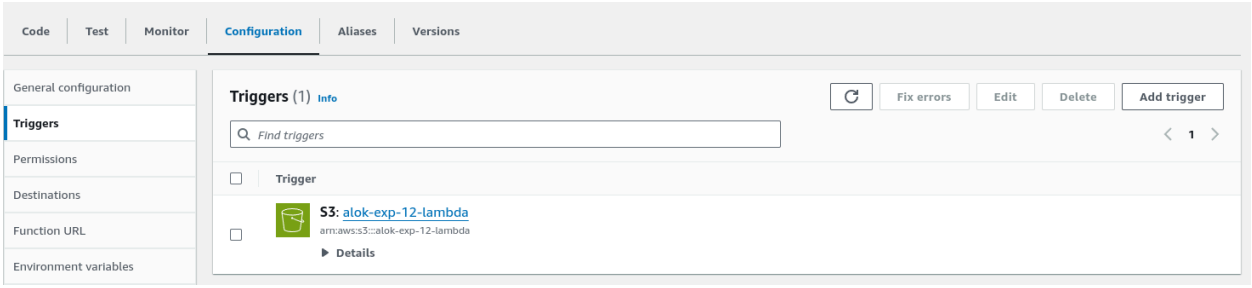
Recursive invocation
If your function writes objects to an S3 bucket, ensure that you are using different S3 buckets for input and output. Writing to the same bucket increases the risk of creating a recursive invocation, which can result in increased Lambda usage and increased costs. Learn more

☐ 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.

Name: Alok Yadav Div: D15C Roll No. 59

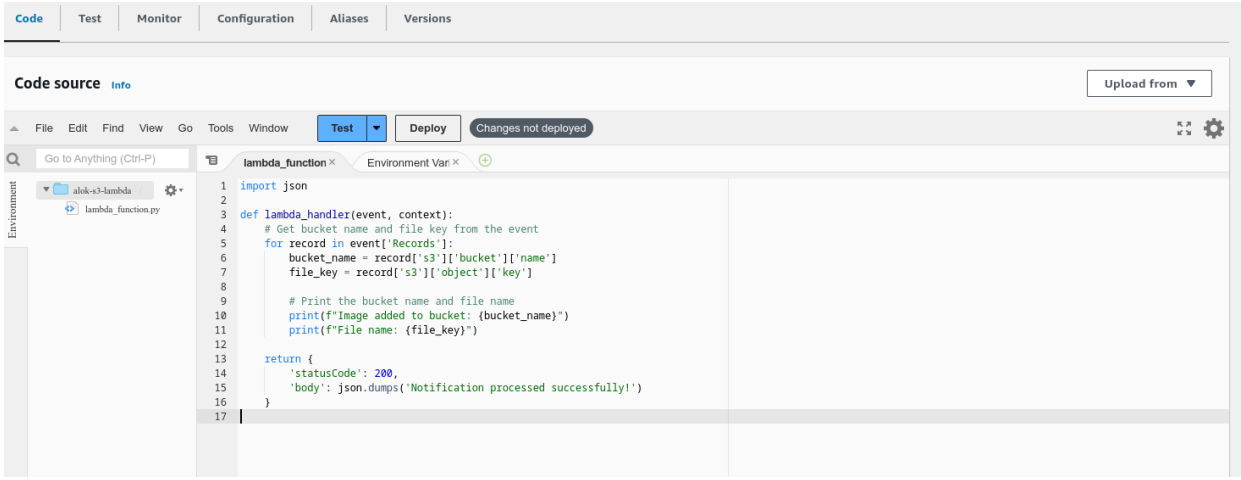




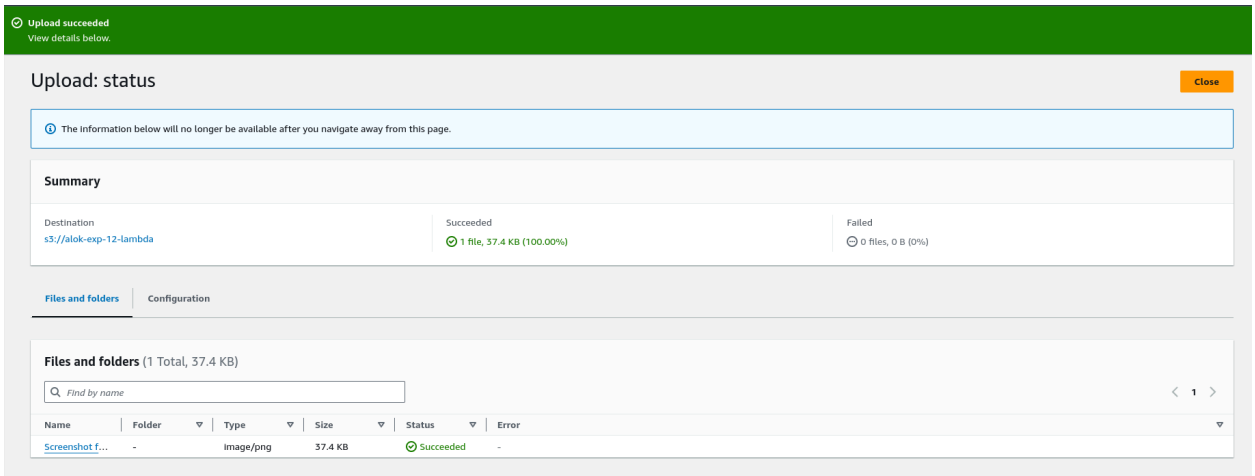
8. Now Write code that logs a message like “Image added to bucket” when triggered. Save the file and click on deploy.

```
import json

def lambda_handler(event, context):
    # Get bucket name and file key from the event
    for record in event['Records']:
        bucket_name = record['s3']['bucket']['name']
        file_key = record['s3']['object']['key']

        # Print the bucket name and file name
        print(f"Image added to bucket: {bucket_name}")
        print(f"File name: {file_key}")

    return {
        'statusCode': 200,
        'body': json.dumps('Notification processed successfully!')
    }
```

Name: Alok Yadav Div: D15C Roll No. 59

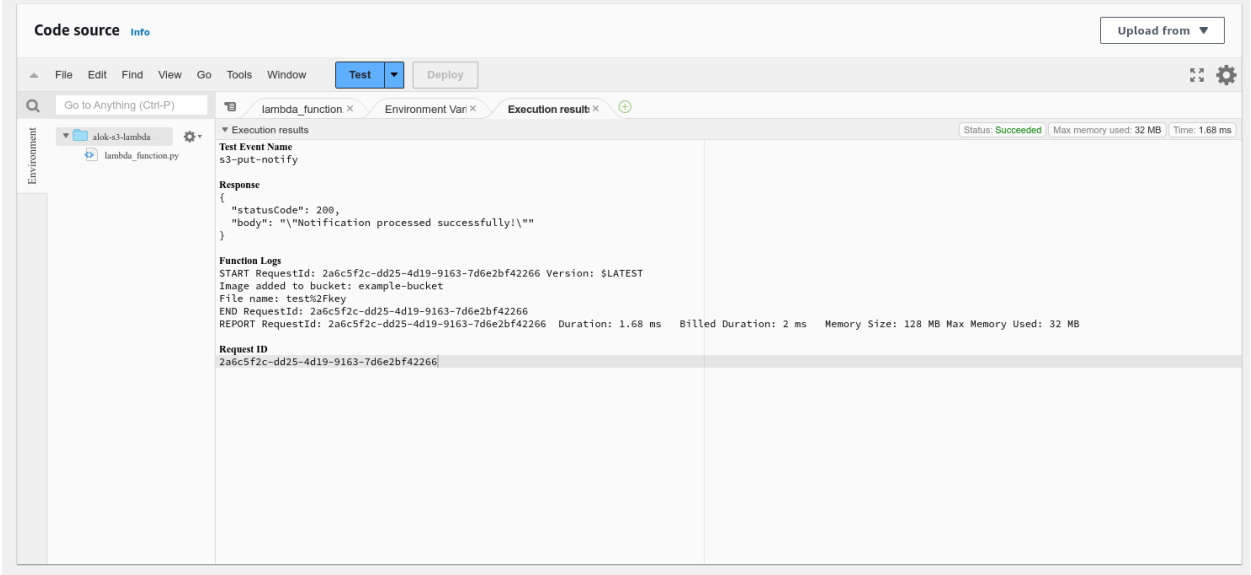


9. Now upload any image to the bucket.

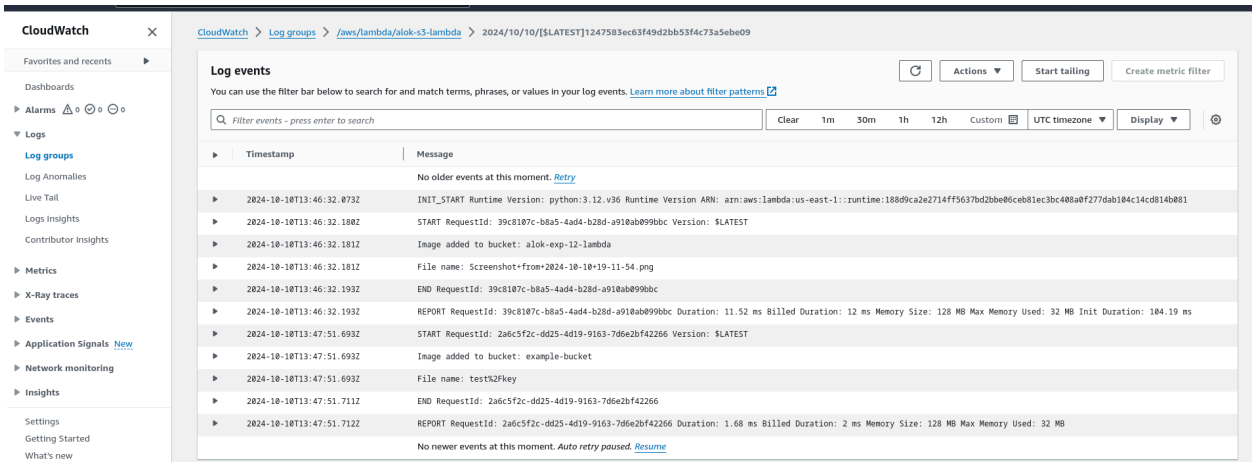


10. Now to click on test in lambda to check whether it is giving log when image is added to S3.

Name: Alok Yadav Div: D15C Roll No. 59



11. Now Lets see the log on Cloud watch.To see it go to monitor section and then click on view cloudwatch logs.



As we can see our lambda function activity is recorded by Cloudwatch

Conclusion:

In this experiment, we successfully implemented an AWS Lambda function that logs a message when an image is uploaded to an S3 bucket. A key aspect was selecting the **S3 Object Created (Put)** event template, as failing to do so would result in errors due to an incompatible event structure. The Lambda function was successfully triggered by S3 object uploads, demonstrating the functionality and efficiency of AWS Lambda's event-driven architecture. This experiment highlighted Lambda's ability to seamlessly respond to S3 events and emphasized

the importance of correctly configuring event triggers to avoid common issues with event data.