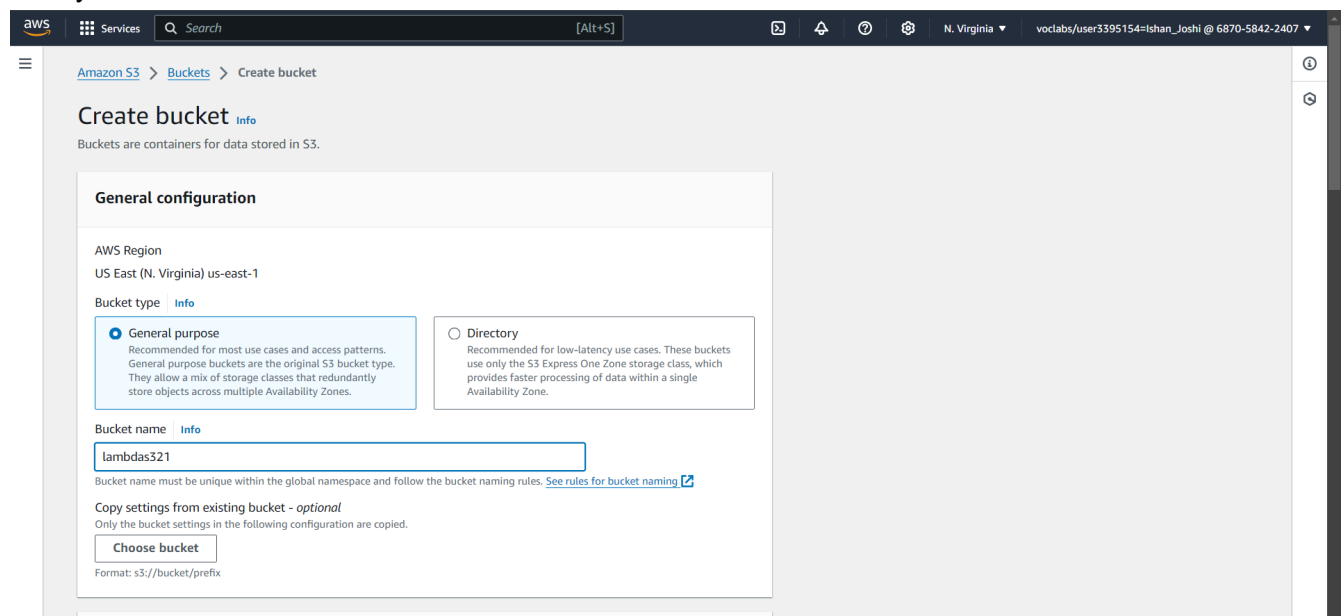


Experiment 12

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:

Step 1: On your AWS console, click on ‘S3’ in the services section and click on ‘Create bucket’. Give your bucket a name.



aws Services Search [Alt+S] N. Virginia voclabs/user3395154=Ishan_Joshi @ 6870-5842-2407

Amazon S3 > Buckets > Create bucket

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.

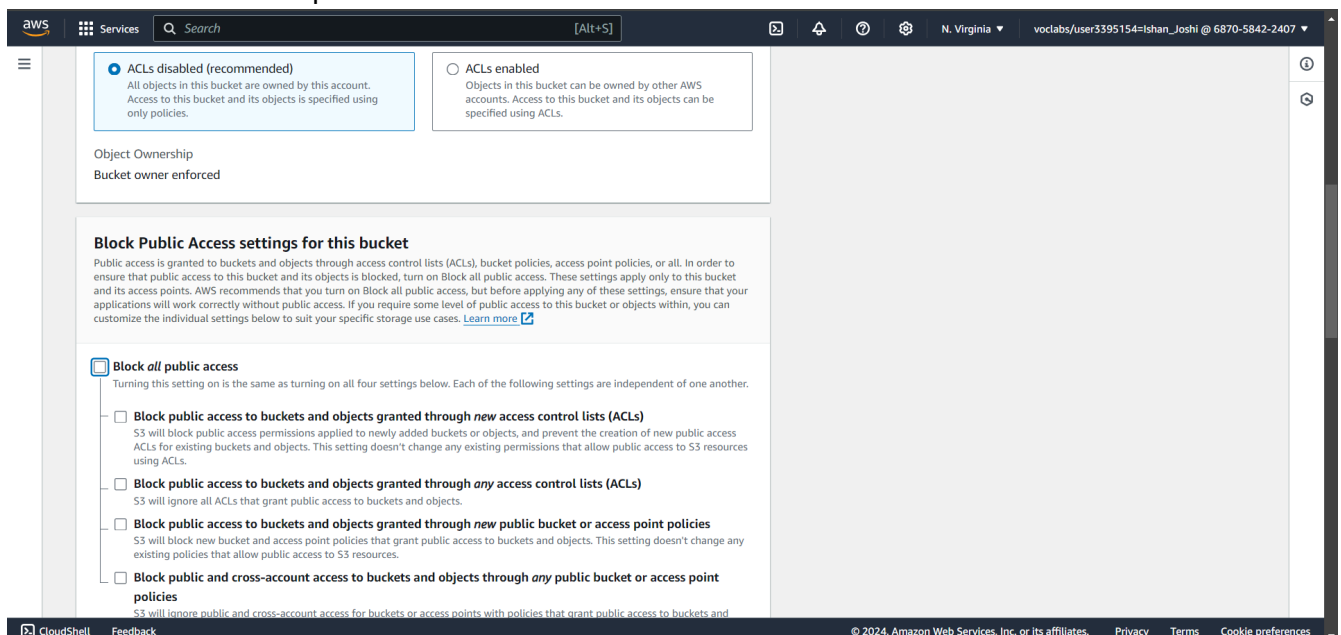
☐ **Directory**
Recommended for low-latency use cases. These buckets use only the S3 Express One Zone storage class, which provides faster processing of data within a single Availability Zone.

Bucket name Info
lambdas321
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.

Format: s3://bucket/prefix

Uncheck the ‘Block all public access’ box.



aws Services Search [Alt+S] N. Virginia voclabs/user3395154=Ishan_Joshi @ 6870-5842-2407

☒ **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
Bucket owner enforced

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.

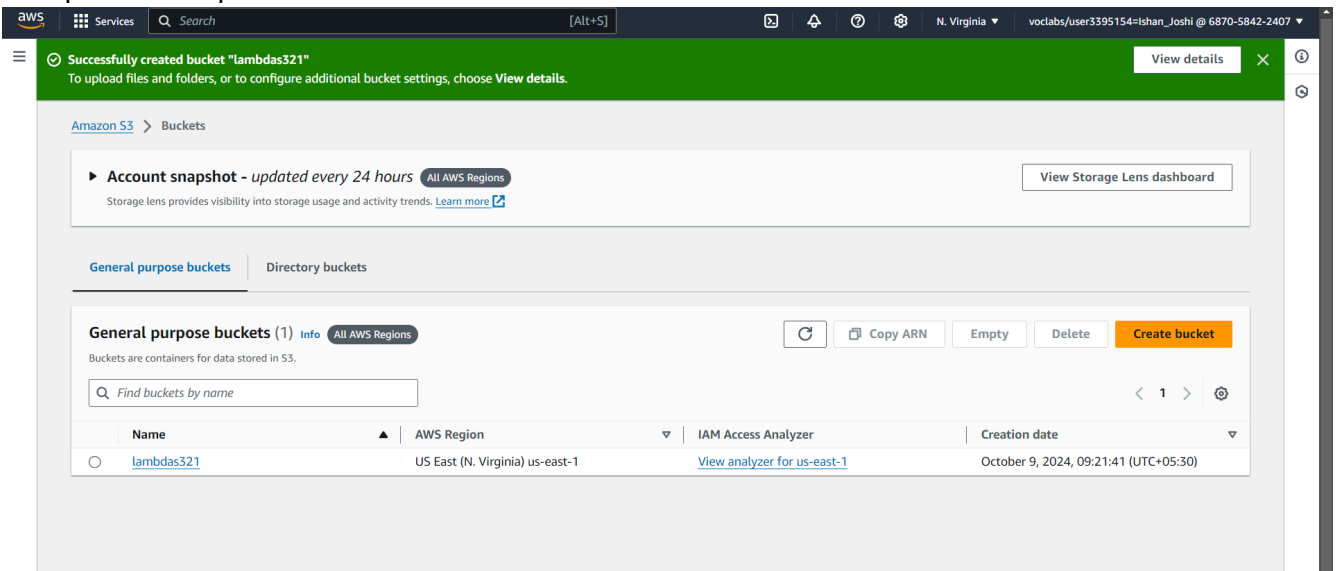
☐ **Block public access to buckets and objects granted through any access control lists (ACLs)**
S3 will ignore all ACLs that grant public access to buckets and objects.

☐ **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

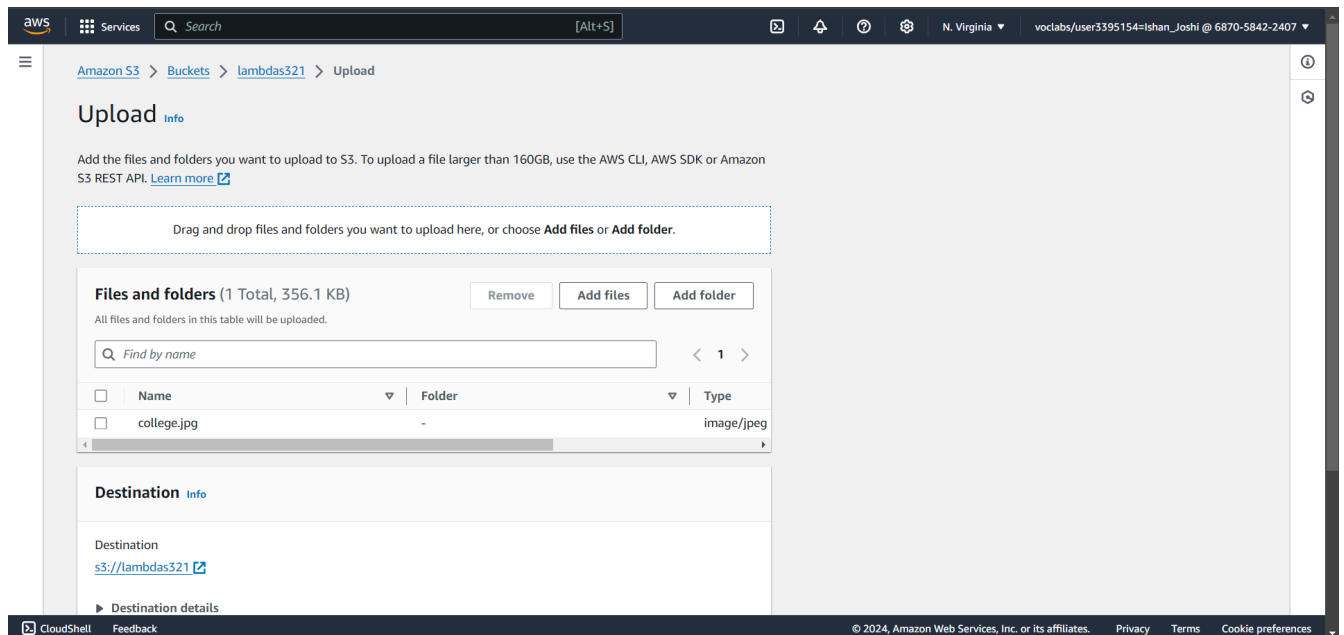
CloudShell Feedback © 2024, Amazon Web Services, Inc. or its affiliates. Privacy Terms Cookie preferences

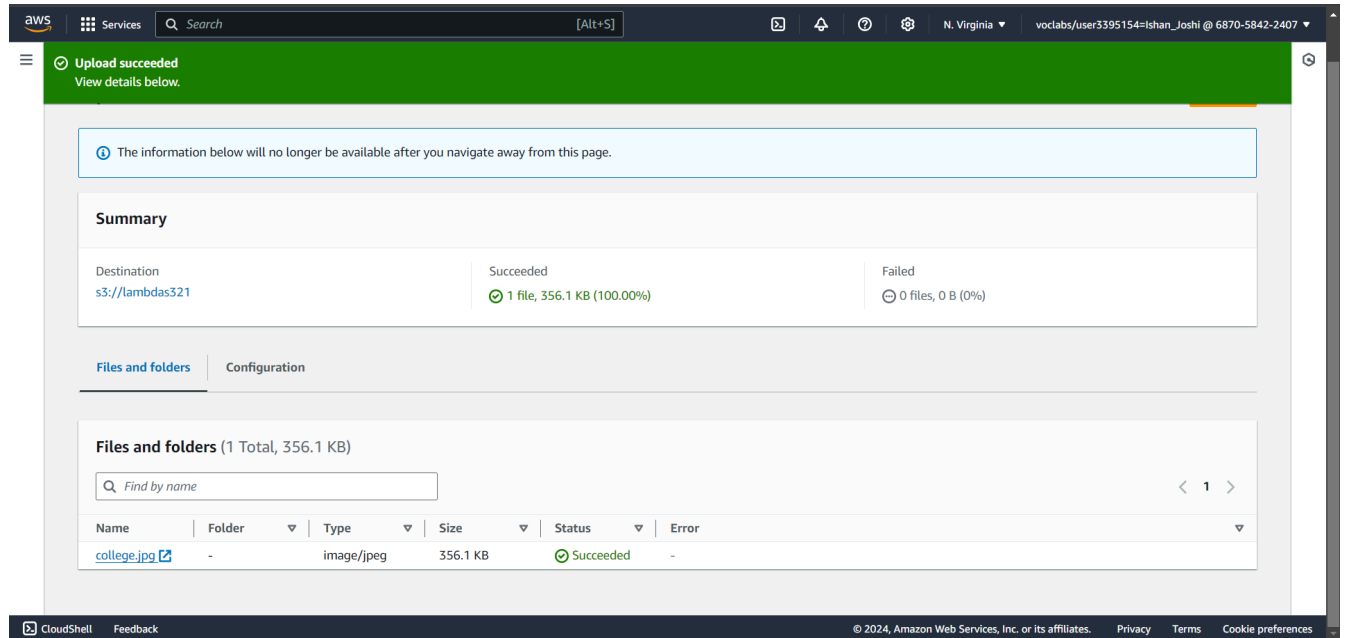
Keep all other options as default and click on 'Create bucket'.



Your bucket is created.

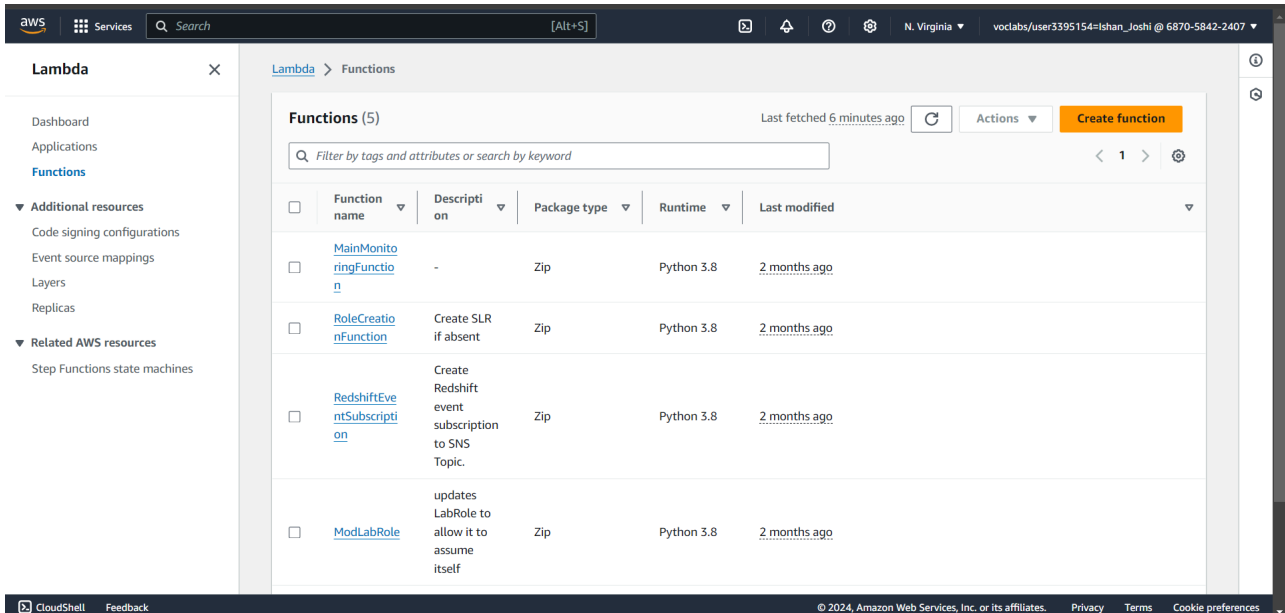
Step 2: Upload an image onto your S3 bucket by clicking on your S3 bucket, clicking on 'Upload', clicking on 'Add files', navigating to your image and selecting it.





Your image gets uploaded onto the S3 bucket.

Step 3: Navigate to the AWS Lambda console using the 'Services' section. Click on 'Create function'.



Step 4: Give your function a name and keep other settings as default.

The screenshot shows the AWS Lambda 'Create function' console page. The breadcrumb navigation is 'Lambda > Functions > Create function'. 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), 'Use a blueprint', and 'Container image'. The 'Basic information' section contains a 'Function name' field with 'Lambda21' entered, a 'Runtime' dropdown set to 'Node.js 20.x', and an 'Architecture' dropdown set to 'x86_64'. The footer of the console shows 'CloudShell', 'Feedback', and copyright information for 2024.

aws Services Search [Alt+S] N. Virginia voclabs/user3395154=Ishan_Joshi @ 6870-5842-2407

Lambda > Functions > 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.

Function name must be 1 to 64 characters, must be unique to the Region, and can't include spaces. Valid characters are a-z, A-Z, 0-9, hyphens (-), and underscores (_).

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

Architecture [Info](#)
Choose the instruction set architecture you want for your function code.
☒ **x86_64**
☐ arm64

CloudShell Feedback © 2024, Amazon Web Services, Inc. or its affiliates. Privacy Terms Cookie preferences

Under 'Execution role', choose 'Use an existing role' and in the dropdown box below, choose 'LabRole'. Then, click on 'Create function'. Your function gets created.

This screenshot shows the 'Permissions' section of the 'Create function' console page. It includes a 'Change default execution role' section with three radio button options: 'Create a new role with basic Lambda permissions', 'Use an existing role' (selected), and 'Create a new role from AWS policy templates'. Below this, the 'Existing role' dropdown is set to 'LabRole'. At the bottom, there is an 'Additional Configurations' section and a 'Create function' button.

aws Services Search [Alt+S] N. Virginia voclabs/user3395154=Ishan_Joshi @ 6870-5842-2407

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 default role later when adding triggers.

▼ 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.

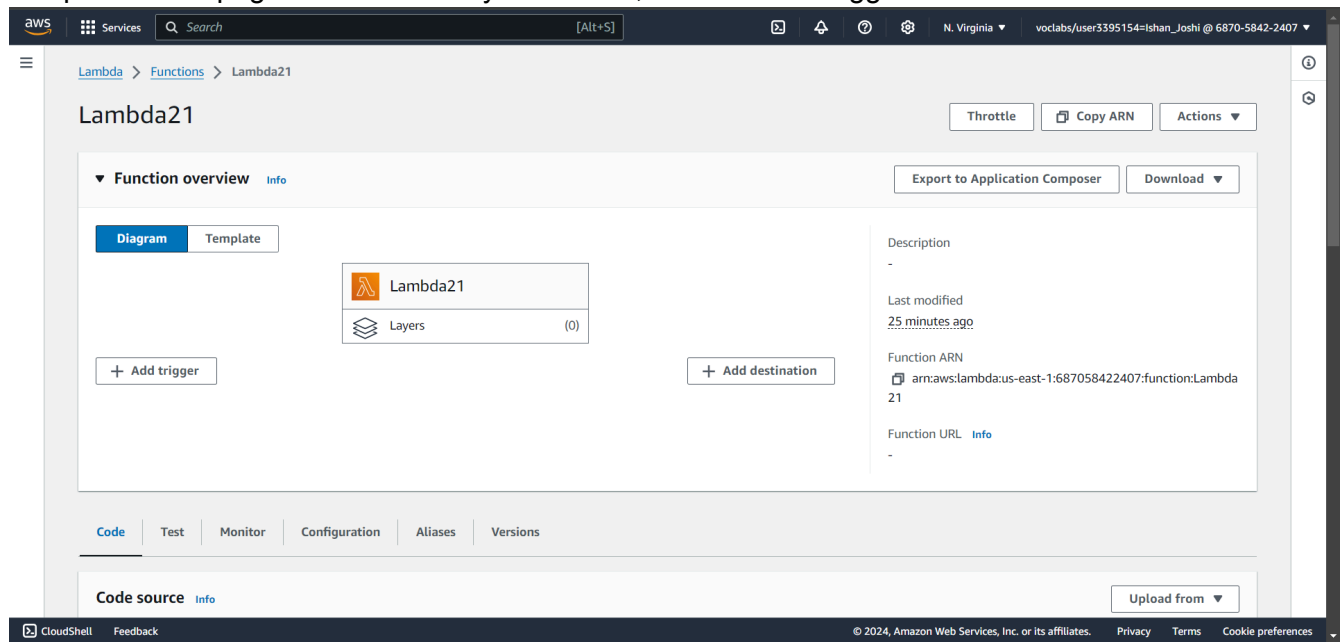
[View the LabRole role](#) on the IAM console.

► Additional Configurations
Use additional configurations to set up code signing, function URL, tags, and Amazon VPC access for your function.

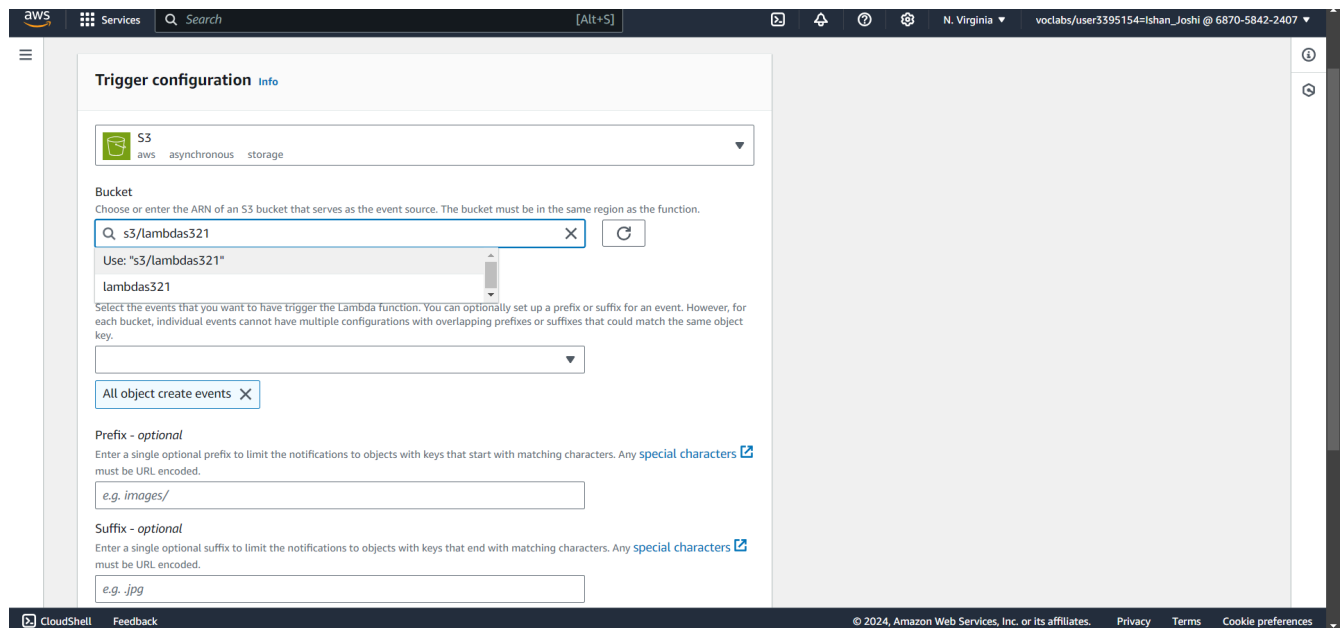
Cancel **Create function**

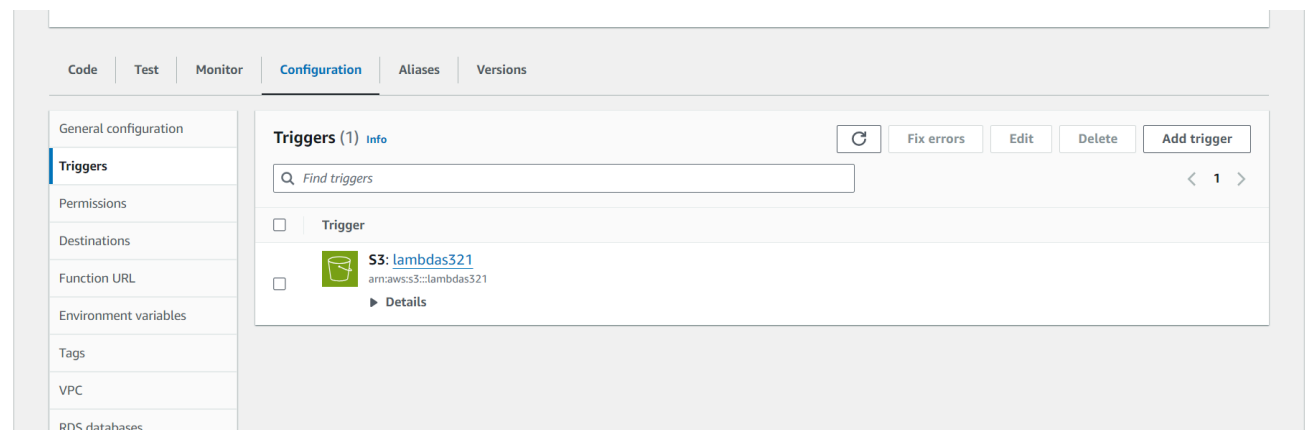
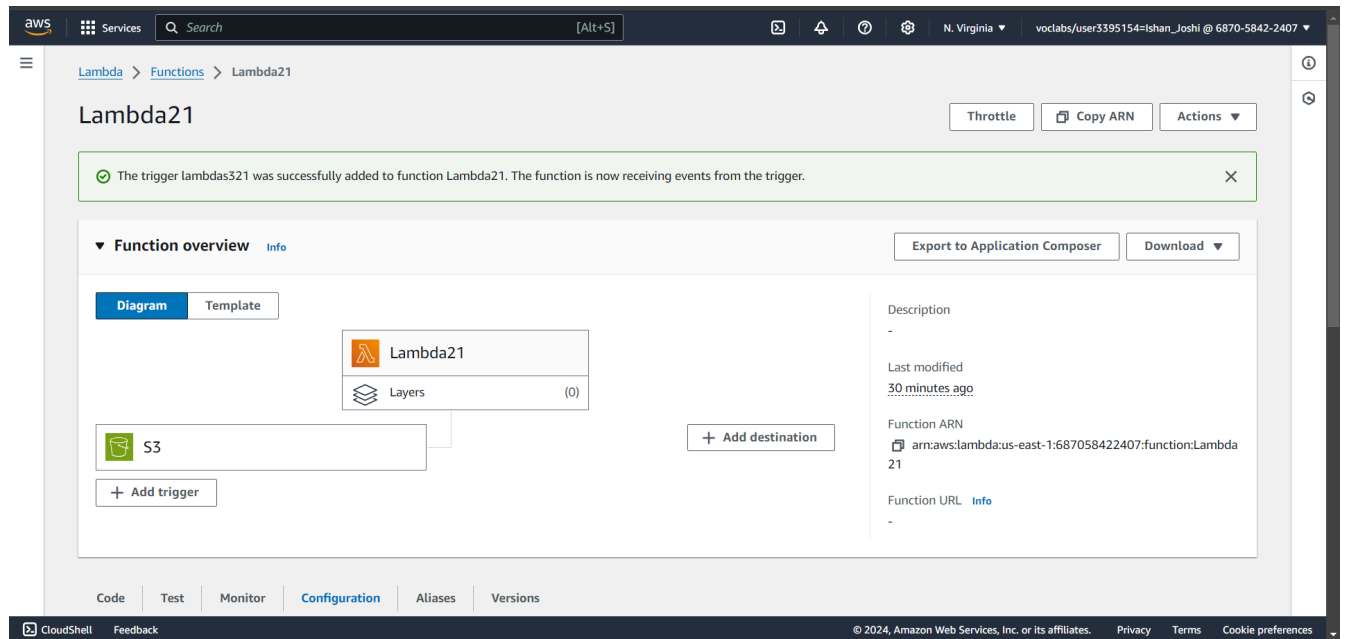
CloudShell Feedback © 2024, Amazon Web Services, Inc. or its affiliates. Privacy Terms Cookie preferences

Step 5: On the page of the function you created, click on 'Add trigger'.



Step 6: Choose 'Trigger configuration' as S3 and select the name of your bucket in the dropdown box below it. Keep other options as default and click on 'Add'.





The trigger gets successfully added to your function

Step 7: In the 'Code source' section of your function, paste the following javascript code instead of the existing code:-

```
export const handler = async (event) => {
  if (!event.Records || event.Records.length === 0) {
    console.error("No records found in the event.");
    return {
      statusCode: 400,
      body: JSON.stringify('No records found in the event')
    };
  }

  // Extract bucket name and object key from the event
  const record = event.Records[0];
  const bucketName = record.s3.bucket.name;
  const objectKey = decodeURIComponent(record.s3.object.key.replace(/\+/g, ' ')); // Handle
  encoded keys

  console.log(`An image has been added to the bucket ${bucketName}:
  ${objectKey}`); console.log(`Event Source: ${record.eventSource}`);
  console.log(`Event Source: ${record.eventSource}`);
  console.log(`Event Source: ${record.eventSource}`);
  console.log(`Event Source: ${record.eventSource}`);

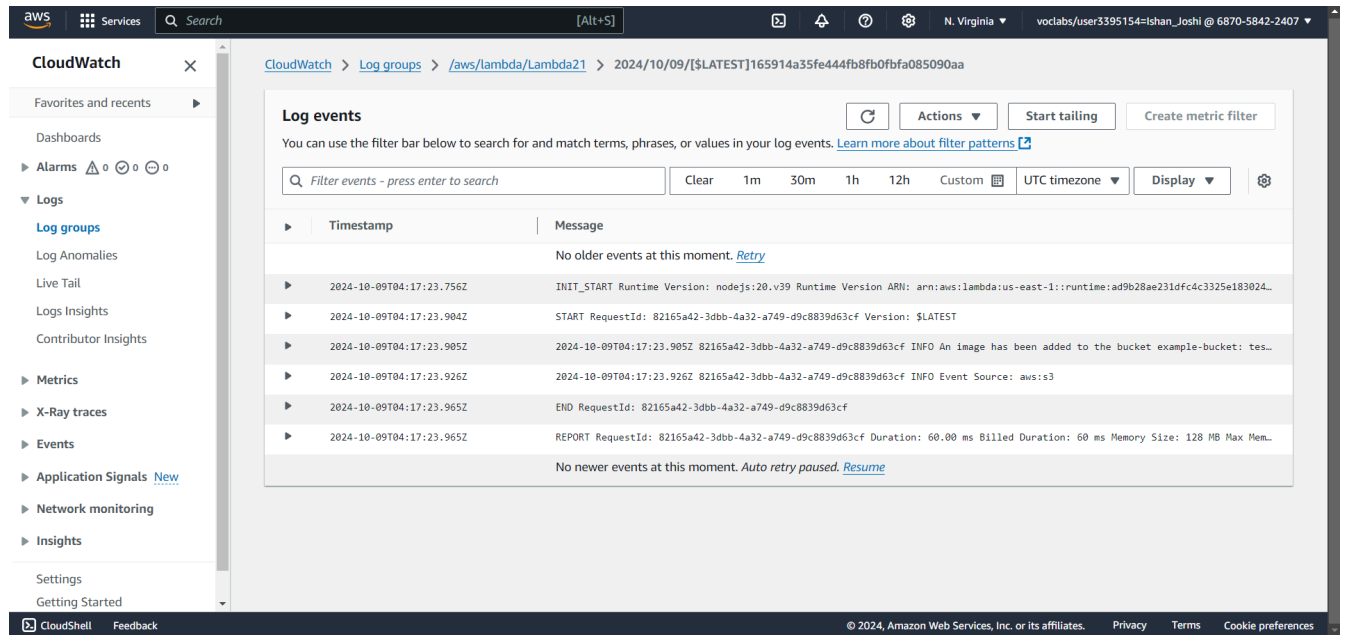
  return {
    statusCode: 200,
    body: JSON.stringify('Log entry created successfully!')
  };
};
```



Step 8: Click on the arrow next to the 'Test' button and click on 'Configure test event'. In the popup box that appears, if you have an existing event, enter the name of your event or create a new event and in the 'Event JSON' section, paste the following code:-

```
{
  "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":
"EXAMPLE123/5678abcdefghijklambdaisawesome/mnopqrstuvwxyzABCDEFGH"
      },
      "s3": {
        "s3SchemaVersion": "1.0",
        "configurationId": "testConfigRule",
        "bucket": {
          "name": "example-bucket",
          "ownerIdentity": {
            "principalId": "EXAMPLE"
          },
          "arn": "arn:aws:s3:::example-bucket"
        },
        "object": {
          "key": "test%2Fkey",
          "size": 1024,
          "eTag": "0123456789abcdef0123456789abcdef",
          "sequencer": "0A1B2C3D4E5F678901"
        }
      }
    }
  ]
}
```


Running the test gives the above output which displays that 'An Image has been added to the bucket' and that the log entry was successfully created.



Conclusion: In this experiment, we demonstrated how to set up a Lambda function that logs the message "An image has been added" whenever an image is uploaded to a specific S3 bucket. We started by creating an S3 bucket and uploading an image to it. Next, we set up a Lambda function, attached an S3 trigger, and selected the S3 bucket we created. After that, we configured the Lambda function's code and set up a test event. Upon running the test, the function logged key event details, such as the bucket name and object key, confirming the image upload. Additionally, Lambda logs confirmed that the image was successfully added to the bucket.

