

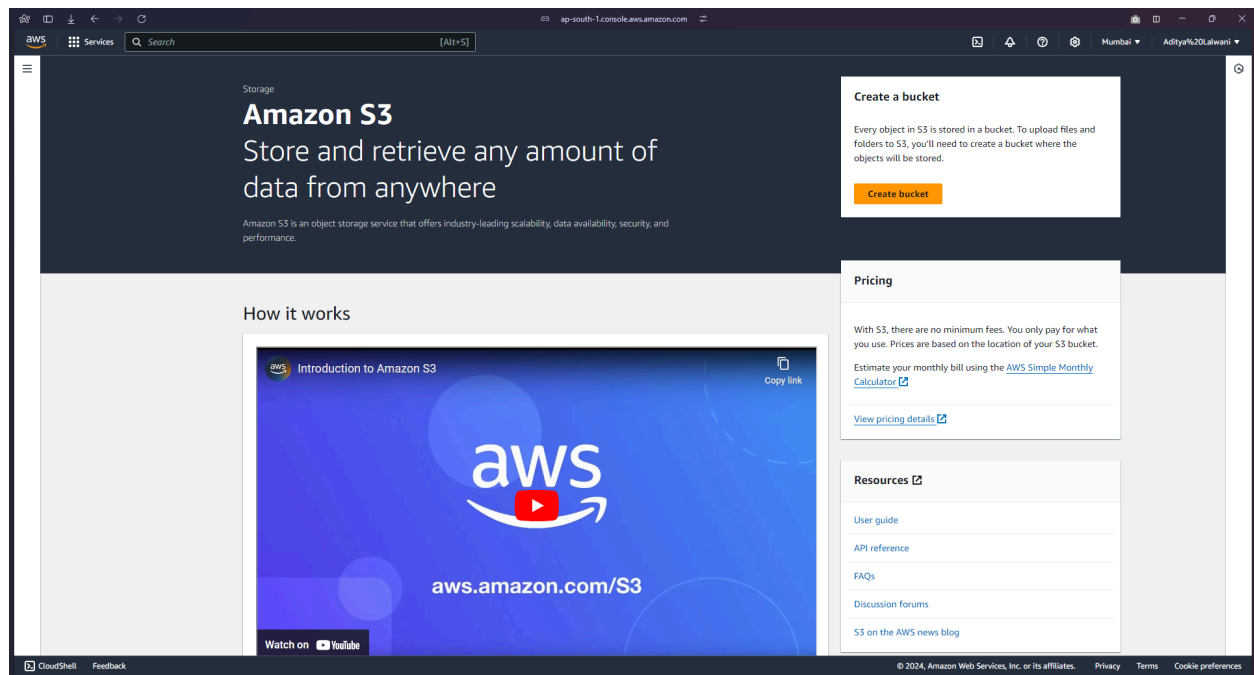
Aim: To create a Lambda function which will log “An Image has been added” once you add an object to a specific bucket in S3.

Prerequisites:

Lambda function (created in the previous experiment).

Create a s3 bucket.

Search for S3 bucket in the services search. Then click on create bucket.



Keep the bucket as a general purpose bucket. Give a name to your bucket.

Create bucket [info](#)

Buckets are containers for data stored in S3.

General configuration

AWS Region
Asia Pacific (Mumbai) ap-south-1

Bucket name [info](#)

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

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
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

Uncheck block all public access.

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.

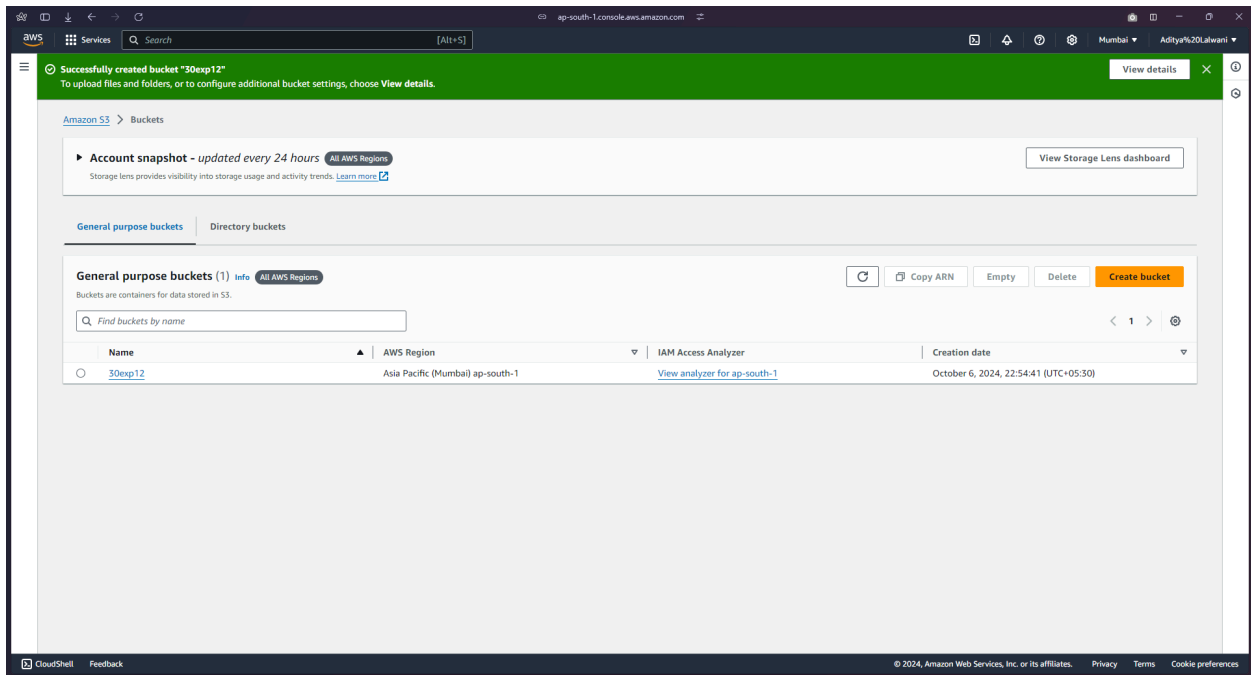
☐ I acknowledge that the current settings might result in this bucket and the objects within becoming public.

Bucket Versioning

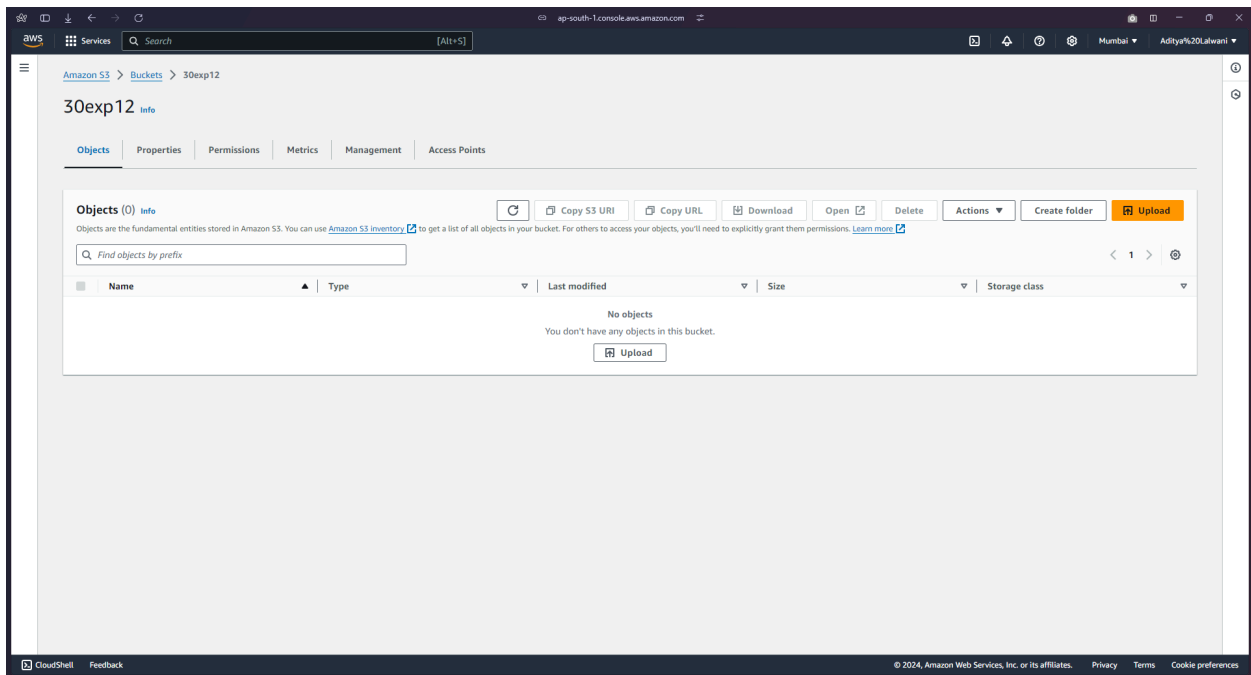
Versioning is a means of keeping multiple variants of an object in the same bucket. You can use versioning to preserve, retrieve, and restore every version of every object stored in your Amazon S3 bucket. With versioning, you can easily recover from both unintended user actions and application failures. [Learn more](#)

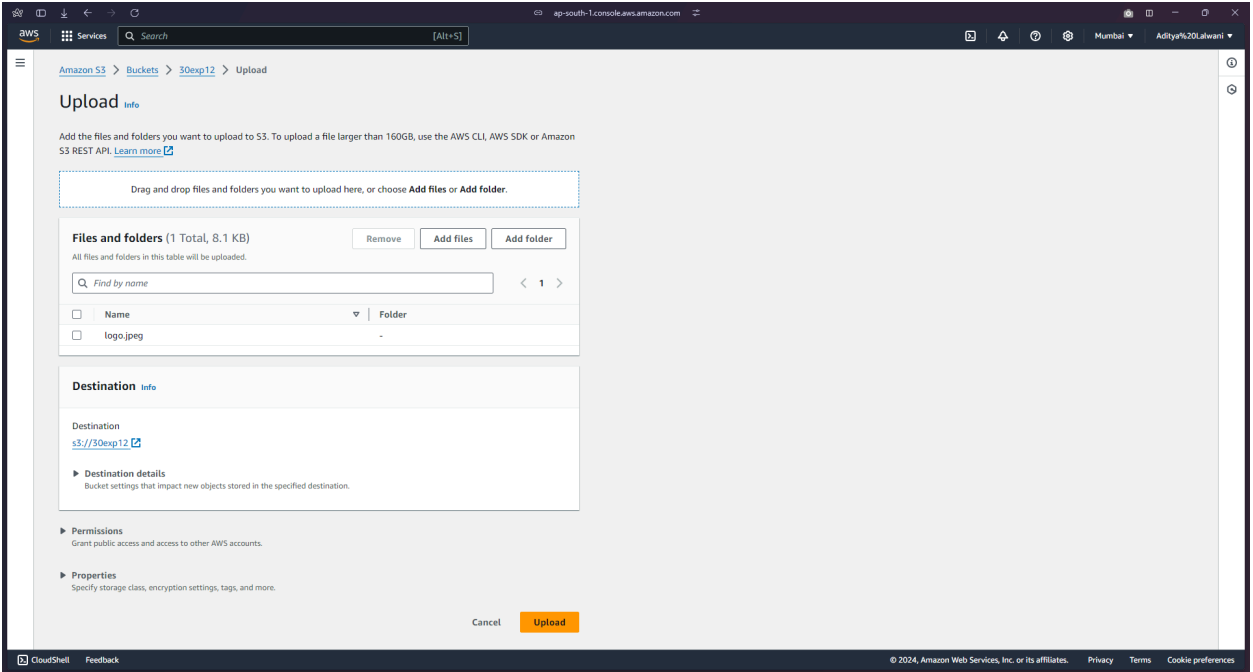
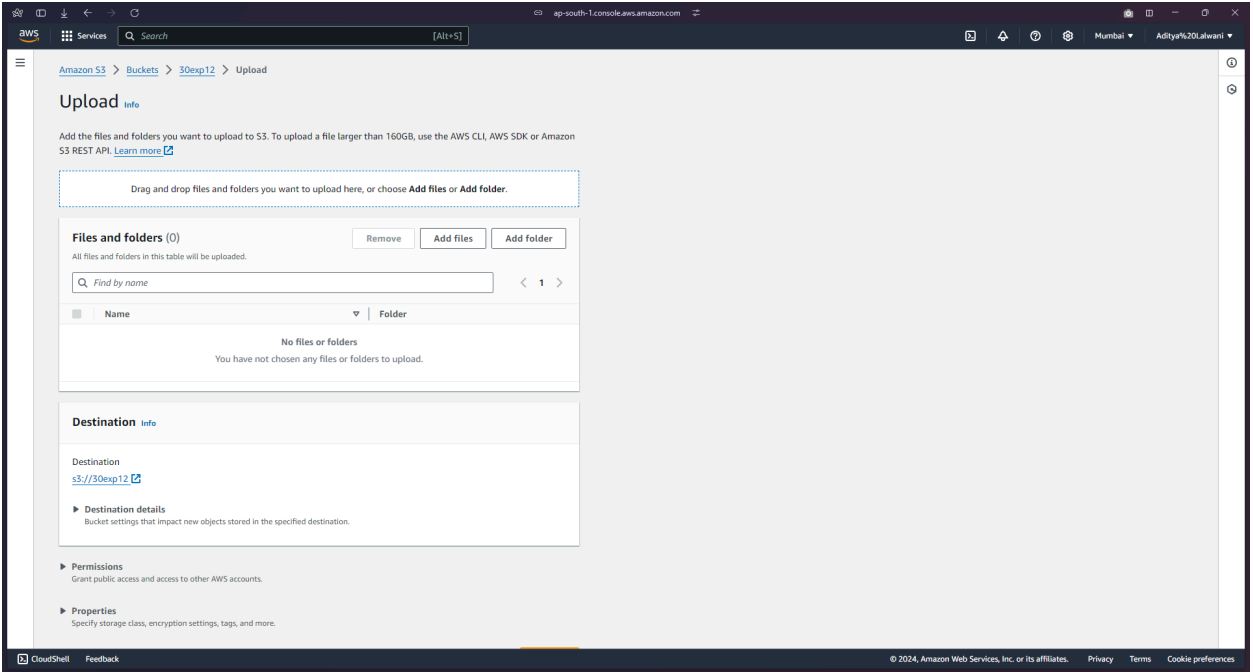
Bucket Versioning
☒ **Disable**
☐ **Enable**

click on create. This would create your bucket. Now click on the name of the bucket.

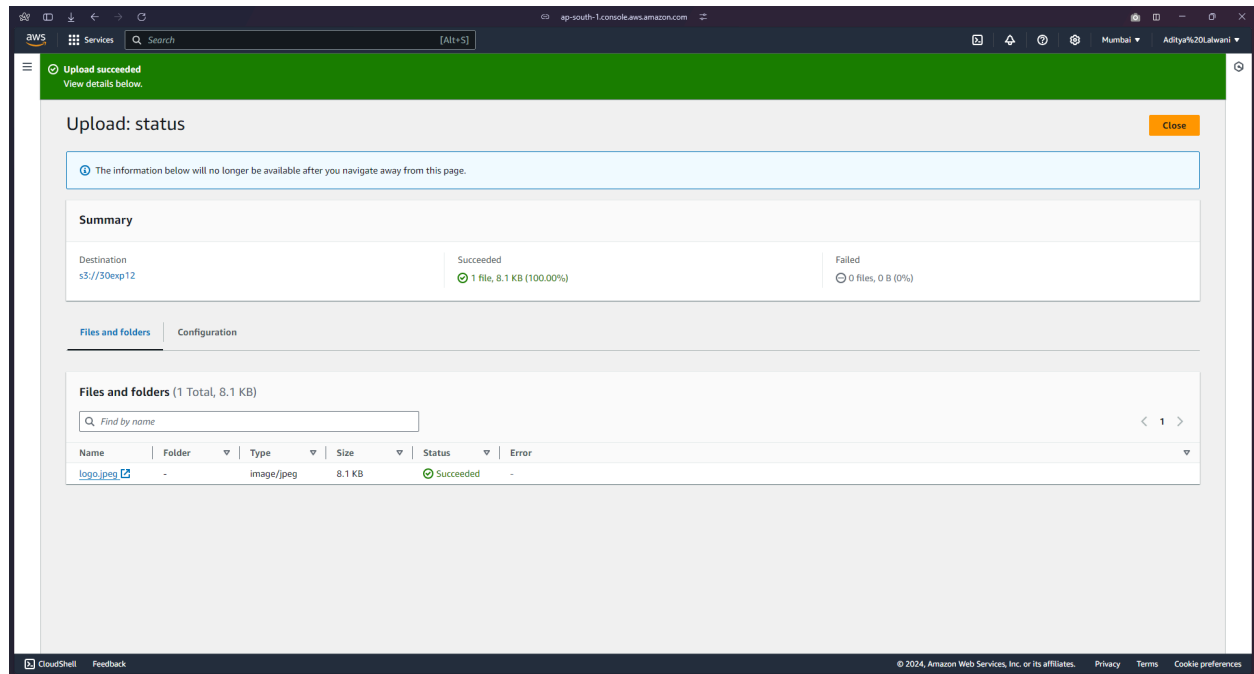


Here, click on upload, then add files. Select any image that you want to upload in the bucket and click on upload.



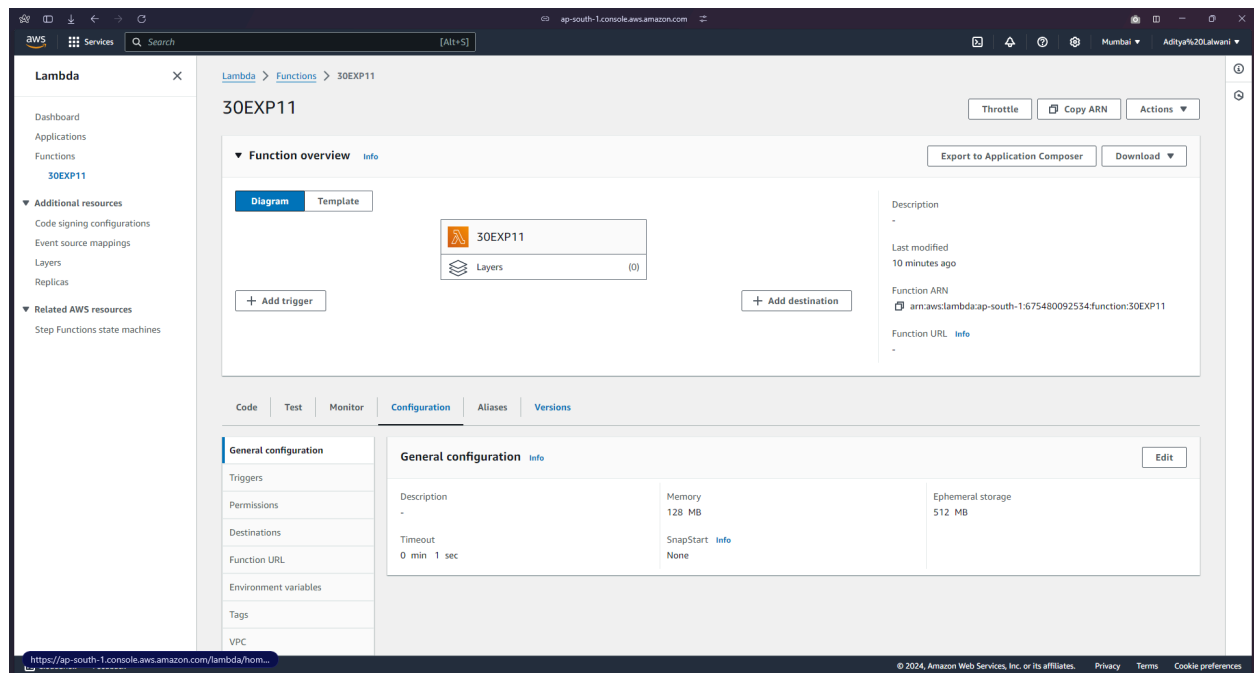


The image has been uploaded to the bucket.

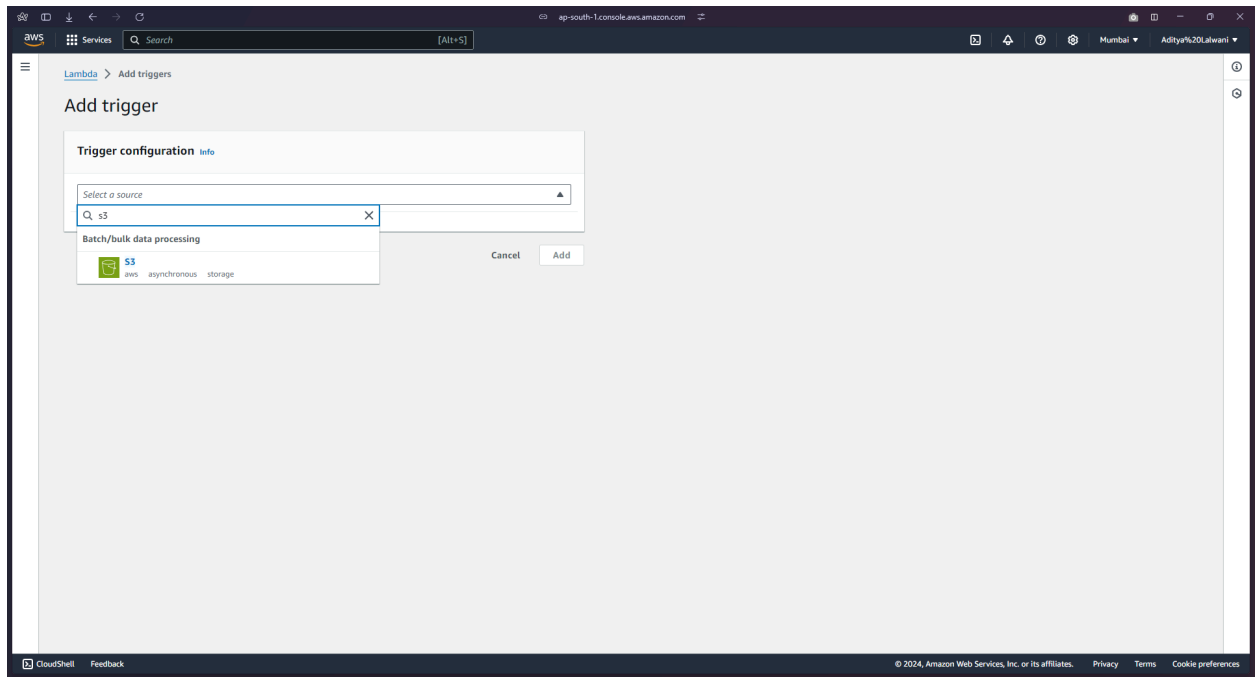


Configure Lambda function

Go to the lambda function you had created before and click on add trigger.



Under trigger configuration, search for S3 and select it.



Here, select the S3 bucket you created for this experiment. Acknowledge the condition given by AWS, then click on Add. This will add the S3 bucket trigger to your function.

The screenshot shows the 'Add trigger' dialog in the AWS Lambda console. The 'S3' service is selected. The 'Bucket' field contains 's3/30exp12'. The 'Event types' dropdown is set to 'All object create events'. The 'Prefix - optional' field is empty. The 'Suffix - optional' field is empty. The 'Recursive invocation' checkbox is checked, with a warning message: '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'. At the bottom, there are 'Cancel' and 'Add' buttons.

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.
s3/30exp12
Bucket region: ap-south-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.

Cancel Add

The screenshot shows the 'Function overview' page for the Lambda function '30EXP11'. A green notification bar at the top states: 'The trigger 30exp12 was successfully added to function 30EXP11. The function is now receiving events from the trigger.' The 'Function overview' section includes a 'Diagram' tab, a visual representation of the function with its layers, and a '+ Add destination' button. The 'Configuration' tab is active, showing 'Triggers (0)' and a table with one trigger. The 'General configuration' section on the left includes 'Triggers', 'Permissions', 'Destinations', 'Function URL', and 'Environment variables'.

Lambda > Functions > 30EXP11

30EXP11 Throttle Copy ARN Actions

The trigger 30exp12 was successfully added to function 30EXP11. The function is now receiving events from the trigger.

Function overview Info Export to Application Composer Download

Diagram Template

30EXP11
Layers (0)

Description
-
Last modified
14 minutes ago
Function ARN
arn:aws:lambda:ap-south-1:675480092534:function:30EXP11
Function URL [Info](#)

Configuration Aliases Versions

General configuration
Triggers
Permissions
Destinations
Function URL
Environment variables

Triggers (0) Info Fix errors Edit Delete Add trigger

Trigger

Scroll down to the code section of the function. Add the following javascript code to the code area by replacing the existing code.

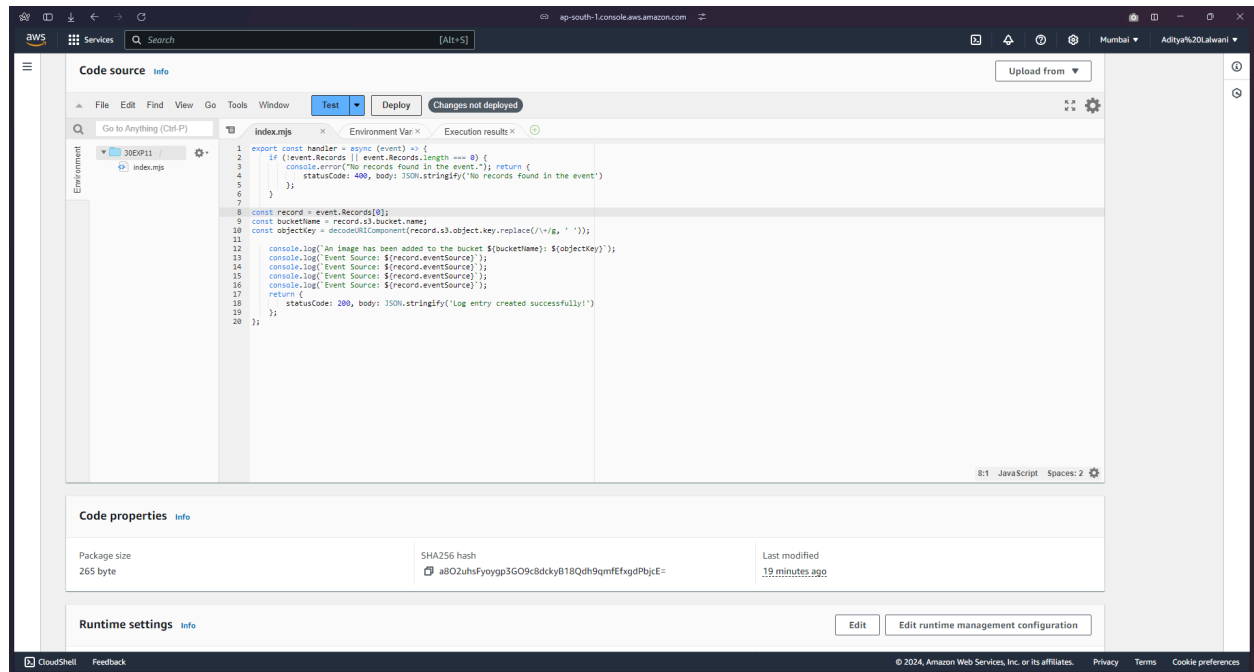
```
export const handler = async (event) => {
  if (!event.Records || event.Records.length === 0) {
    console.error("No records found in the event.");
    return {
      statusCode: 400, body: JSON.stringify('No
      records found in the event')
    };
  }

  // 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

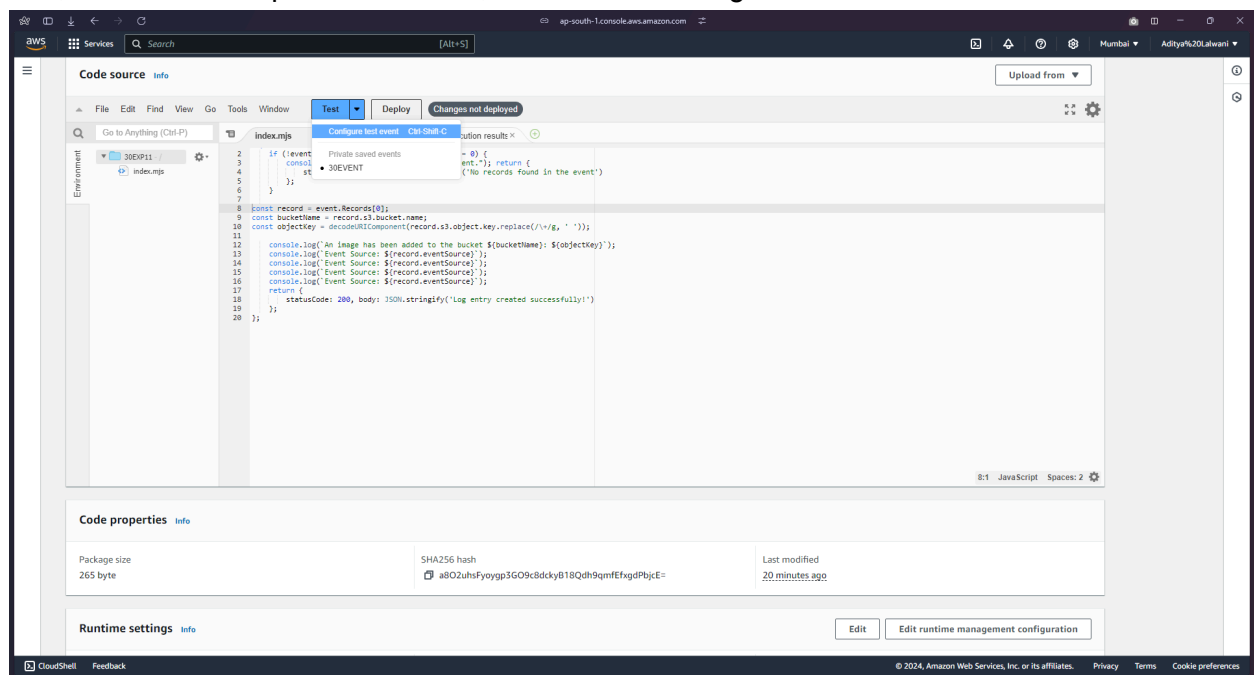
  console.log(`An image has been added to the bucket ${bucketName}:
  ${objectKey}`); console.log(`Event Source: ${record.eventSource}`);
  console.log(`Event Source: ${record.eventSource}`); console.log(`Event Source:
  ${record.eventSource}`); console.log(`Event Source: ${record.eventSource}`);

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

This code checks for records in the event, extracts the bucket name and object key, logs the details, and returns a success message if an image is added to the bucket.



Now, click on the dropdown near test, then click on configure test event.



Here, select edit saved event. Select the event taht you had created before. Under Event JSON, paste the following code.

```

{
  "Records": [

```

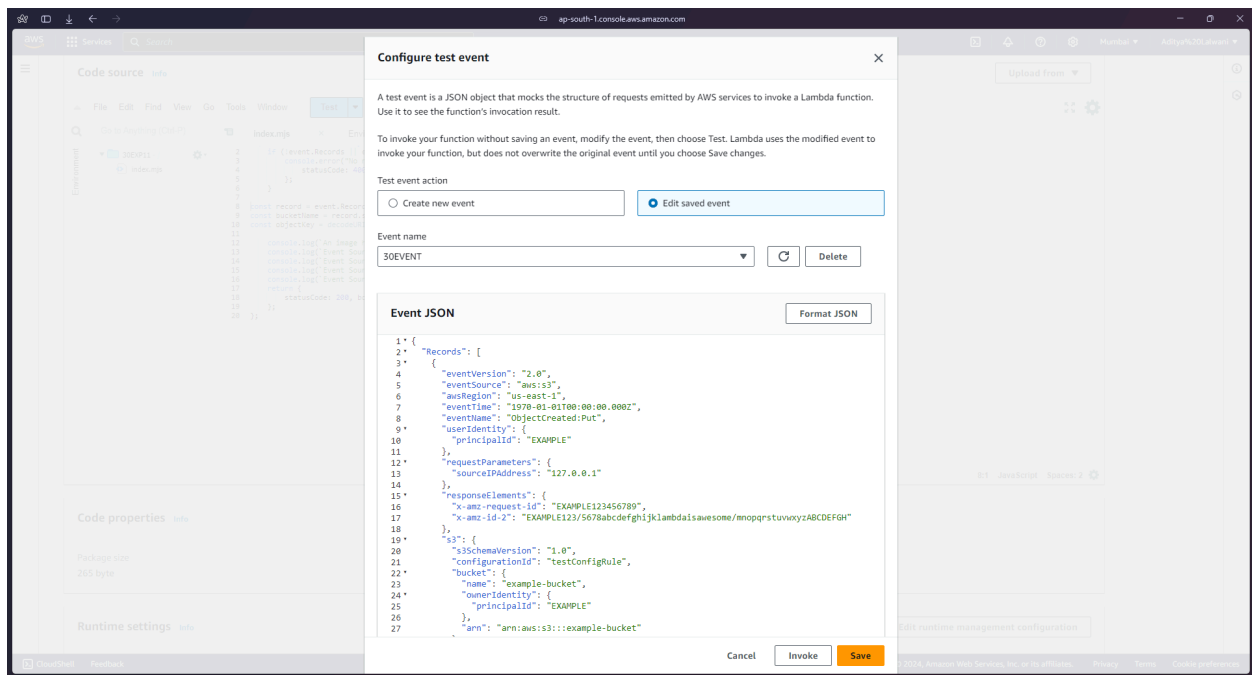
```
{
  "eventVersion": "2.0",
  "eventSource": "aws:s3",
  "awsRegion": "us-east-1",
  "eventTime": "1970-01-01T00:00:00.000Z",
  "eventName": "ObjectCreated:Put",
  "userIdentity": {
    "principalId": "EXAMPLE"
  },
  "requestParameters": {
    "sourceIPAddress": "127.0.0.1"
  },
  "responseElements": {
    "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"
}
}
}
}
]
}

```



Save the changes. Then deploy the code changes by clicking on deploy.

After deploying, click on Test. The console output shows that 'an image has been added to the bucket'

The JSON response shows that the log entry was created successfully.

The screenshot displays the AWS Lambda console interface for the function **30EXP11**. The **Test** tab is selected, showing the execution results for the **Test Event Name** (30EVENT). The status is **Succeeded**, with a maximum memory used of 64 MB and a time of 53.13 ms.

Response:

```
{
  "statusCode": 200,
  "body": "\"Log entry created successfully!\""
}
```

Function Logs:

```
START RequestId: ec0f009b-c0d7-4756-8629-fd68f490ad8f Version: SLATEST
2024-10-06T17:44:42.934Z ec0f009b-c0d7-4756-8629-fd68f490ad8f INFO An image has been added to the bucket example-bucket: test/key
2024-10-06T17:44:42.942Z ec0f009b-c0d7-4756-8629-fd68f490ad8f INFO Event Source: aws:is3
2024-10-06T17:44:42.942Z ec0f009b-c0d7-4756-8629-fd68f490ad8f INFO Event Source: aws:is3
2024-10-06T17:44:42.942Z ec0f009b-c0d7-4756-8629-fd68f490ad8f INFO Event Source: aws:is3
2024-10-06T17:44:42.942Z ec0f009b-c0d7-4756-8629-fd68f490ad8f INFO Event Source: aws:is3
END RequestId: ec0f009b-c0d7-4756-8629-fd68f490ad8f
REPORT RequestId: ec0f009b-c0d7-4756-8629-fd68f490ad8f Duration: 53.13 ms Billed Duration: 54 ms Memory Size: 128 MB Max Memory Used: 64 MB Init Duration: 133.94 ms
```

Request ID: ec0f009b-c0d7-4756-8629-fd68f490ad8f

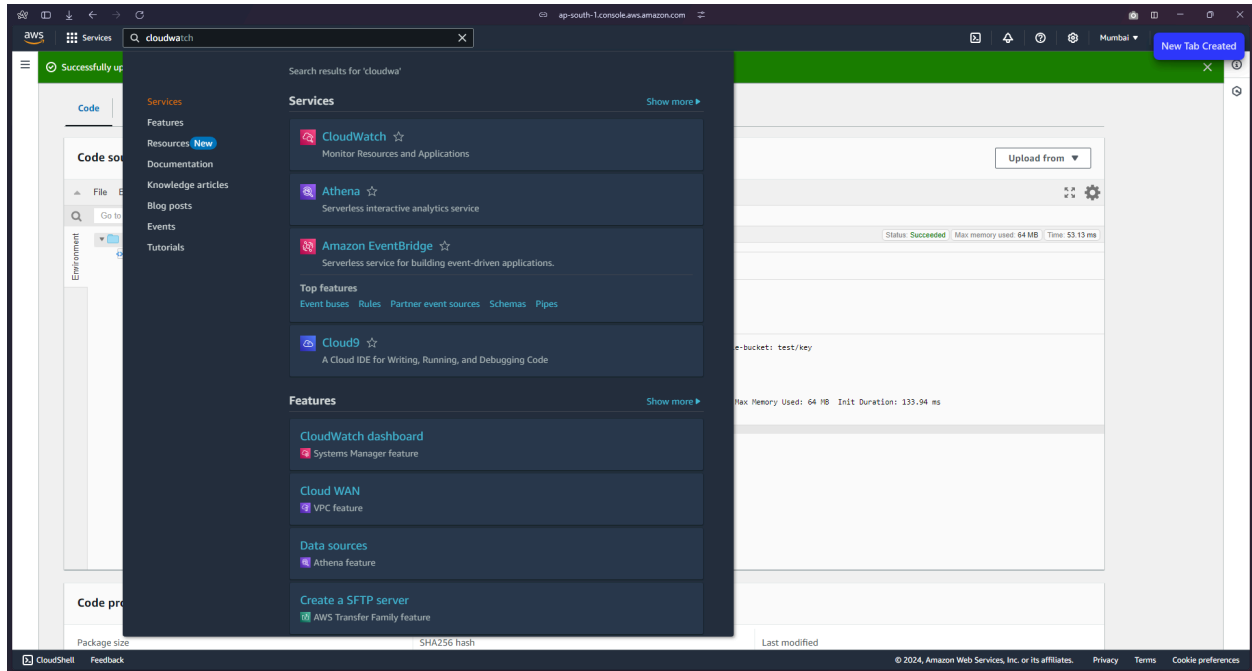
Code properties:

| Package size | SHA256 hash | Last modified |
|--------------|-------------|---------------|
| | | |

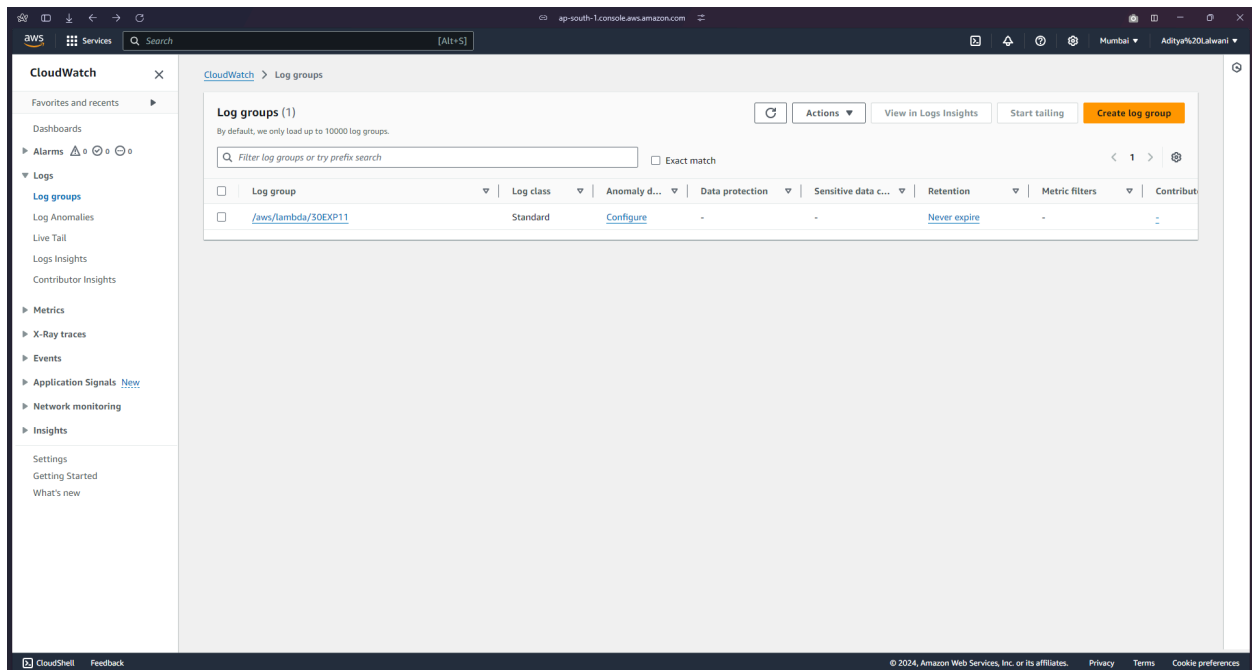
The footer of the console shows the copyright notice: © 2024, Amazon Web Services, Inc. or its affiliates. Privacy Terms Cookie preferences.

Check the logs

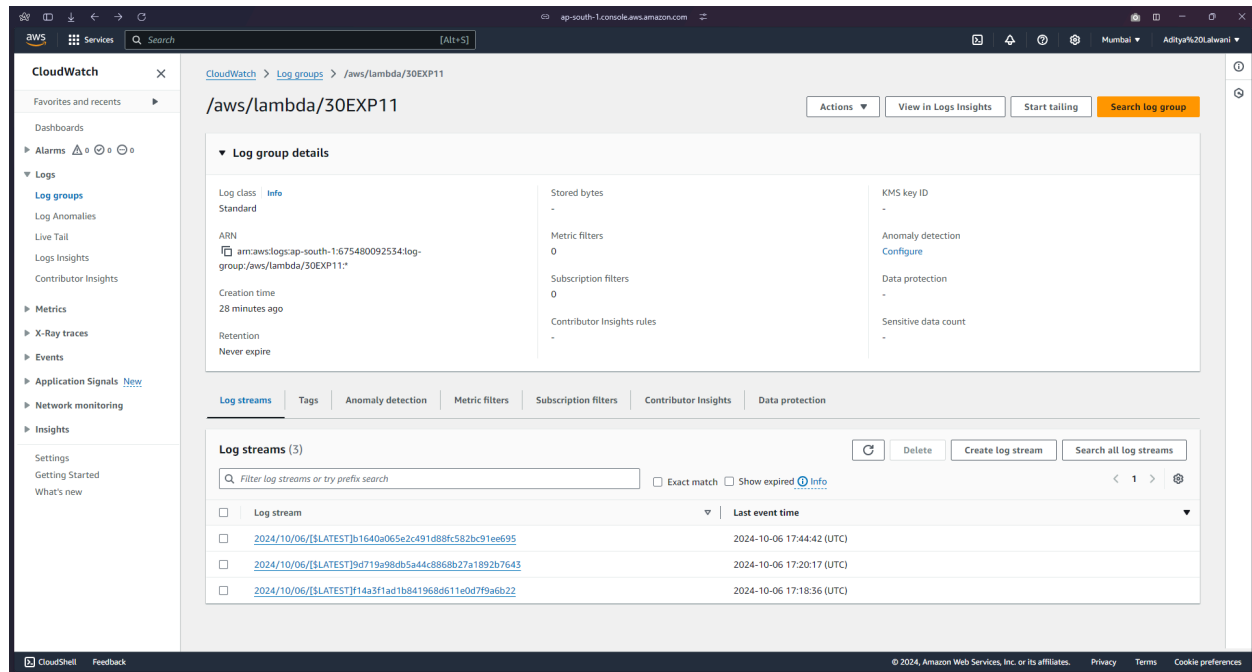
To check the logs explicitly, search for CloudWatch on services and open it in a new tab.



Select the log that has the lambda function name you just ran.



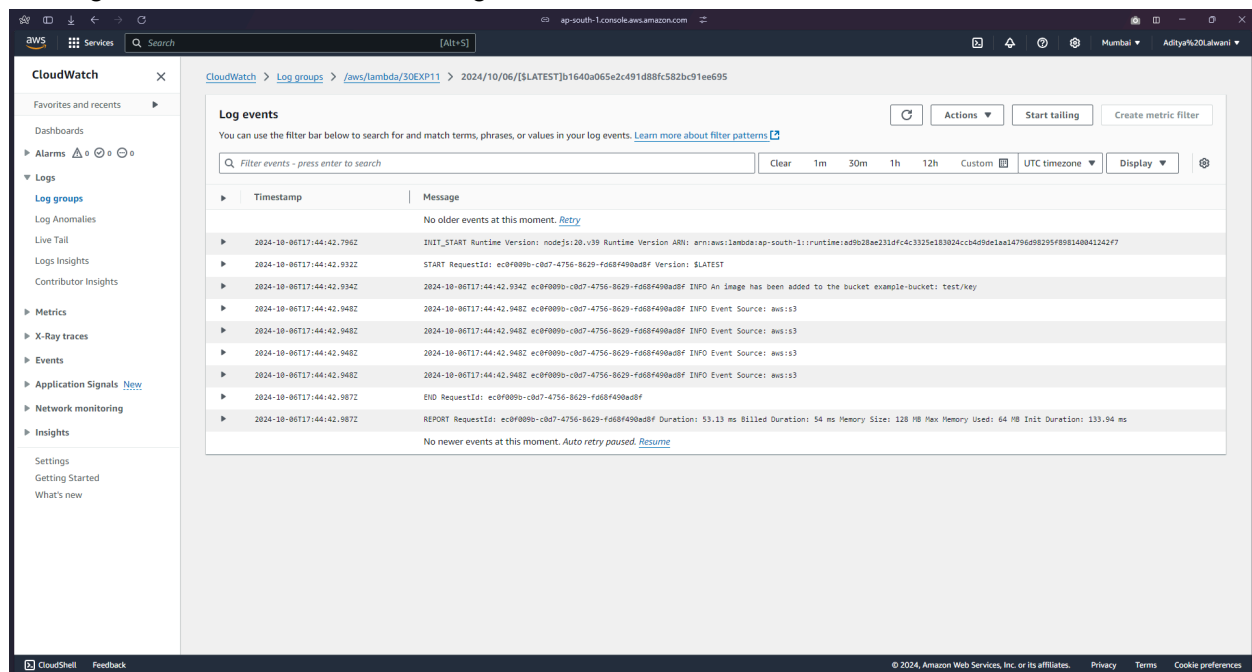
Here, under Log streams, select the log stream you want to check.



The screenshot shows the AWS CloudWatch console. On the left is a navigation menu with options like Dashboards, Alarms, Logs, Metrics, X-Ray traces, Events, Application Signals, Network monitoring, and Insights. The main content area is titled '/aws/lambda/30EXP11'. It has tabs for Log group details, Log streams, Tags, Anomaly detection, Metric filters, Subscription filters, Contributor Insights, and Data protection. The 'Log group details' tab is active, showing information about the log group. Below it, the 'Log streams' tab is selected, displaying a list of log streams with their last event times.

| Log stream | Last event time |
|-------------------------------------------------------|---------------------------|
| 2024/10/06/[\$LATEST]b1640a065e2c491d88fc582bc91ee695 | 2024-10-06 17:44:42 (UTC) |
| 2024/10/06/[\$LATEST]9d719a98db5a44c886b27a1892b7643 | 2024-10-06 17:20:17 (UTC) |
| 2024/10/06/[\$LATEST]f14a3f1ad1b841968d611e0d7f9a6b22 | 2024-10-06 17:18:36 (UTC) |

Here again, we can see that 'An image has been added to the bucket'.



The screenshot shows the AWS CloudWatch console with the 'Log events' tab selected. It displays a list of log events with their timestamps and messages. The message 'An image has been added to the bucket example-bucket: test/key' is visible in the log events.

| Timestamp | Message |
|--------------------------|----------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| 2024-10-06T17:44:42.796Z | INIT_START Runtime Version: nodejs18.v39 Runtime Version ARN: arn:aws:lambda:ap-south-1:runtime:ad9b28ae231ffc4c3325e183024cc8d8de1a14796d98295f898140841242f7 |
| 2024-10-06T17:44:42.932Z | START RequestId: ec8f0809-c8d7-4756-8629-fd68f490ad8f Version: \$LATEST |
| 2024-10-06T17:44:42.934Z | ec8f0809-c8d7-4756-8629-fd68f490ad8f INFO An image has been added to the bucket example-bucket: test/key |
| 2024-10-06T17:44:42.948Z | 2024-10-06T17:44:42.948Z ec8f0809-c8d7-4756-8629-fd68f490ad8f INFO Event Source: aws:s3 |
| 2024-10-06T17:44:42.948Z | 2024-10-06T17:44:42.948Z ec8f0809-c8d7-4756-8629-fd68f490ad8f INFO Event Source: aws:s3 |
| 2024-10-06T17:44:42.948Z | 2024-10-06T17:44:42.948Z ec8f0809-c8d7-4756-8629-fd68f490ad8f INFO Event Source: aws:s3 |
| 2024-10-06T17:44:42.948Z | 2024-10-06T17:44:42.948Z ec8f0809-c8d7-4756-8629-fd68f490ad8f INFO Event Source: aws:s3 |
| 2024-10-06T17:44:42.987Z | END RequestId: ec8f0809-c8d7-4756-8629-fd68f490ad8f |
| 2024-10-06T17:44:42.987Z | REPORT RequestId: ec8f0809-c8d7-4756-8629-fd68f490ad8f Duration: 53.13 ms Billed Duration: 54 ms Memory Size: 128 MB Max Memory Used: 64 MB Init Duration: 133.84 ms |

we created a Lambda function that logs a message when an image is added to a S3 bucket. By configuring an S3 bucket trigger for the Lambda function, we see how AWS services can work

together to automate tasks. The function logged important details about the event, including the bucket name and object key. After deploying the function and testing with a sample event, we verified the logs in CloudWatch, confirming that the Lambda function correctly detected and logged the addition of an image to the bucket. This experiment highlighted the integration between AWS Lambda and S3.