

## **Experiment No. 12**

**Aim:** To construct a Lambda function which will log “An Image has been added” once you add an object to a specific bucket in S3. (LO1, LO6)

### **Theory:**

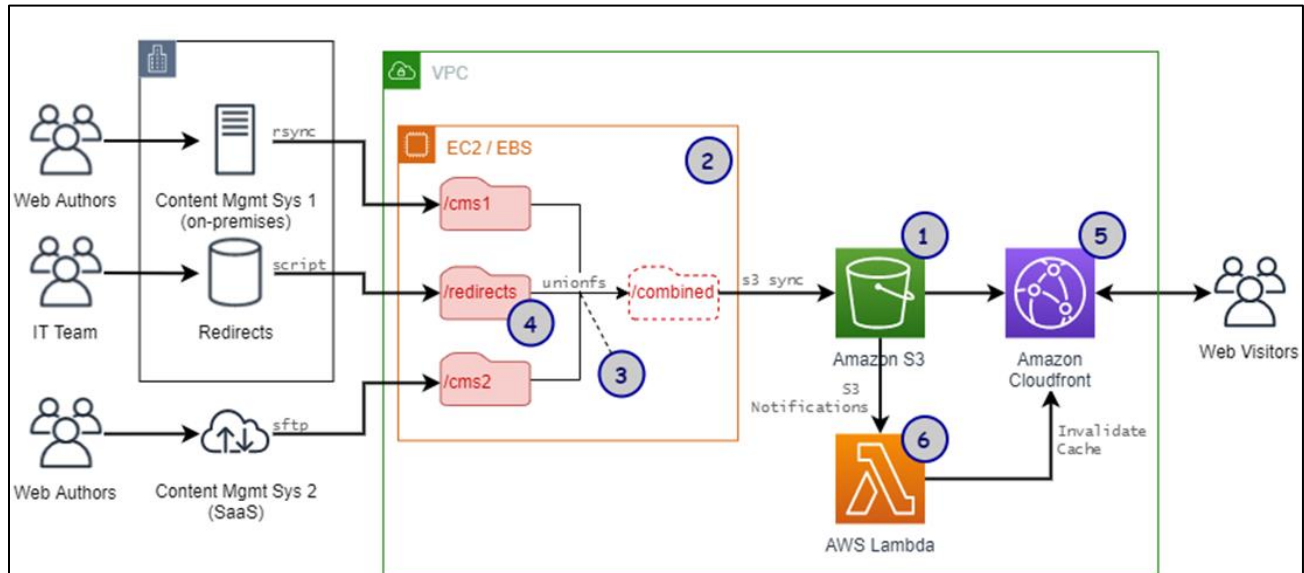
#### **What is Amazon S3?**

- Amazon Simple Storage Service (Amazon S3) is an object storage service that offers industry-leading scalability, data availability, security, and performance.
- Customers of all sizes and industries can use Amazon S3 to store and protect any amount of data for a range of use cases, such as data lakes, websites, mobile applications, backup and restore, archive, enterprise applications, IoT devices, and big data analytics.
- Amazon S3 provides management features so that you can optimize, organize, and configure access to your data to meet your specific business, organizational, and compliance requirements.
- To work with Amazon S3, you can configure Amazon S3 buckets to store, organize, and manage various data files using an easy-to-use online web interface because it duplicates or replicates data objects across multiple devices or servers in different S3 clusters on a regular basis,

#### **Features of Amazon S3:**

- **Storage management:** Amazon S3 has storage management features that you can use to manage costs, meet regulatory requirements, reduce latency, and save multiple distinct copies of your data for compliance requirements.
  1. **S3 Replication** – Replicate objects and their respective metadata and object tags to one or more destination buckets in the same or different AWS Regions for reduced latency, compliance, security, and other use cases.
  2. **S3 Batch Operations** – Manage billions of objects at scale with a single S3 API request or a few clicks in the Amazon S3 console. You can use Batch Operations to perform operations such as Copy, Invoke AWS Lambda function, and Restore on millions or billions of objects.
- **Access management:** Amazon S3 provides features for auditing and managing access to your buckets and objects. By default, S3 buckets and the objects in them are private. You have access only to the S3 resources that you create.
  1. **AWS Identity and Access Management (IAM)** – Create IAM users for your AWS account to manage access to your Amazon S3 resources. For example, you can use IAM with Amazon S3 to control the type of access a user or group of users has to an S3 bucket that your AWS account owns.

2. **Bucket policies** – Use IAM-based policy language to configure resource-based permissions for your S3 buckets and the objects in them.
3. **Amazon S3 access points** – Configure named network endpoints with dedicated access policies to manage data access at scale for shared datasets in Amazon S3.



### AWS S3 Benefits:

Some of the benefits of AWS S3 are:

- **Durability:** S3 provides 99.999999999 percent durability.
- **Low cost:** S3 lets you store data in a range of “storage classes.” These classes are based on the frequency and immediacy you require in accessing files.
- **Scalability:** S3 charges you only for what resources you actually use, and there are no hidden fees or overage charges. You can scale your storage resources to easily meet your organization’s ever-changing demands.
- **Availability:** S3 offers 99.99 percent availability of objects
- **Security:** S3 offers an impressive range of access management tools and encryption features that provide top-notch security.
- **Flexibility:** S3 is ideal for a wide range of uses like data storage, data backup, software delivery, data archiving, disaster recovery, website hosting, mobile applications, IoT devices, and much more.
- **Simple data transfer:** You don’t have to be an IT genius to execute data transfers on S3. The service revolves around simplicity and ease of use.

### **AWS Lambda:**

- AWS Lambda is a serverless compute service that runs your code in response to events and automatically manages the underlying compute resources for you. These events may include changes in state or an update, such as a user placing an item in a shopping cart on an ecommerce website.
- You can use AWS Lambda to extend other AWS services with custom logic, or create your own backend services that operate at AWS scale, performance, and security. AWS Lambda automatically runs code in response to multiple events, such as HTTP requests via Amazon API Gateway, modifications to objects in Amazon Simple Storage Service (Amazon S3) buckets, table updates in Amazon DynamoDB, and state transitions in AWS Step Functions.
- You can use AWS Lambda to extend other AWS services with custom logic, or create your own back end services that operate at AWS scale, performance, and security.
- AWS Lambda is a fully managed compute service that runs your code in response to events generated by custom code or from various AWS services such as Amazon S3, DynamoDB, Amazon Kinesis, Amazon SNS, and Amazon Cognito.
- Lambda runs your code on high availability compute infrastructure and performs all the administration of your compute resources. This includes server and operating system maintenance, capacity provisioning and automatic scaling, code and security patch deployment, and code monitoring and logging. All you need to do is supply the code.

### **AWS Lambda Features:**

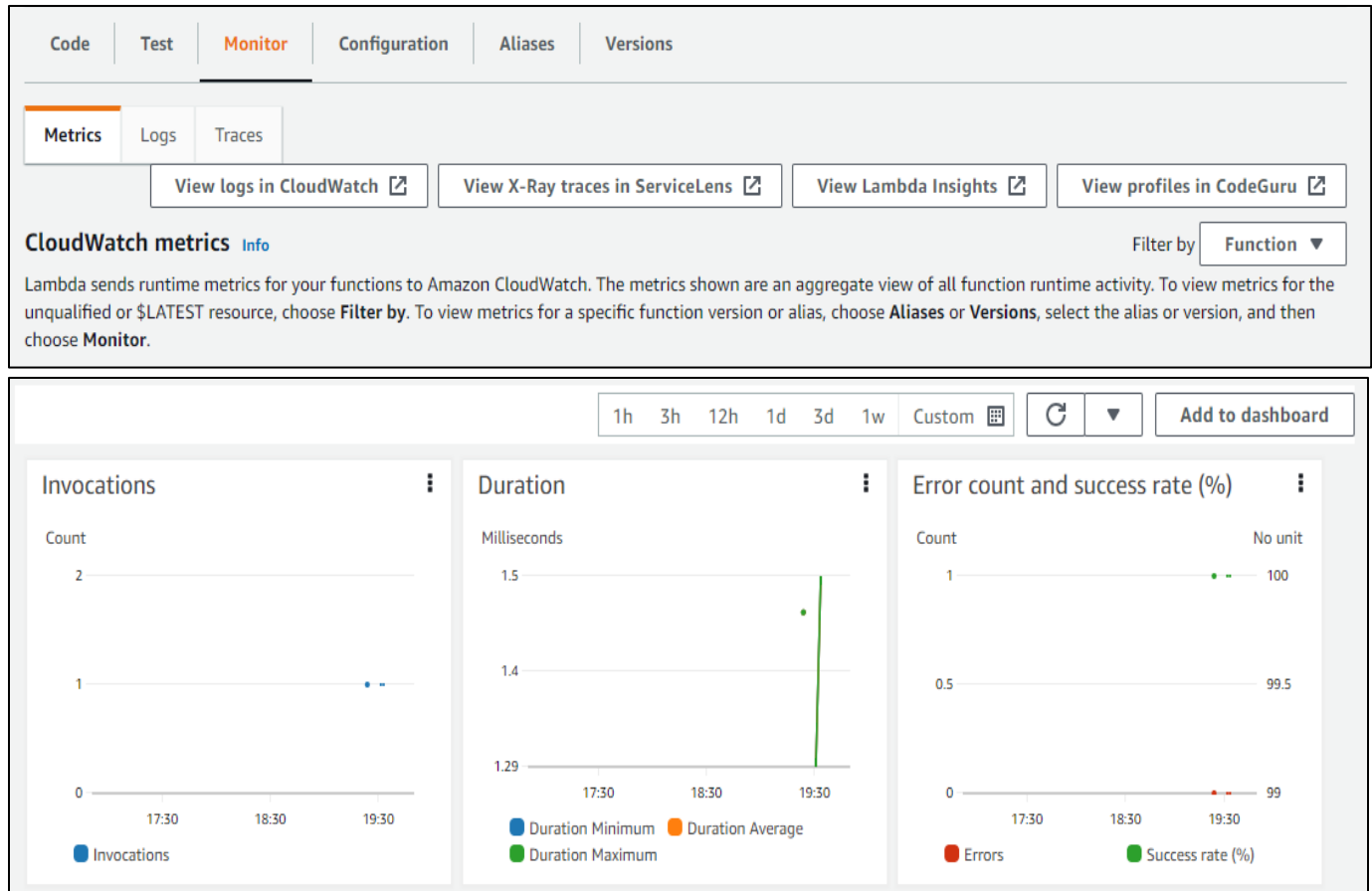
1. AWS lambda easily scales the infrastructure without any additional configuration. It reduces the operational work involved,
2. It offers multiple options like AWS S3, CloudWatch, DynamoDB, API Gateways, Kinesis, Code Commit, and many more to trigger an event.
3. You don't need to invest upfront. You can pay only for the memory used by the lambda function and minimal cost on the number of requests hence cost-efficient.
4. AWS Lambda is secure. It uses AWS IAM to define all the roles and security policies.
5. It offers fault tolerance for both services running the code and the function. You don't have to worry about the application being down.

Name: Harsh Dalvi  
Roll No: 13

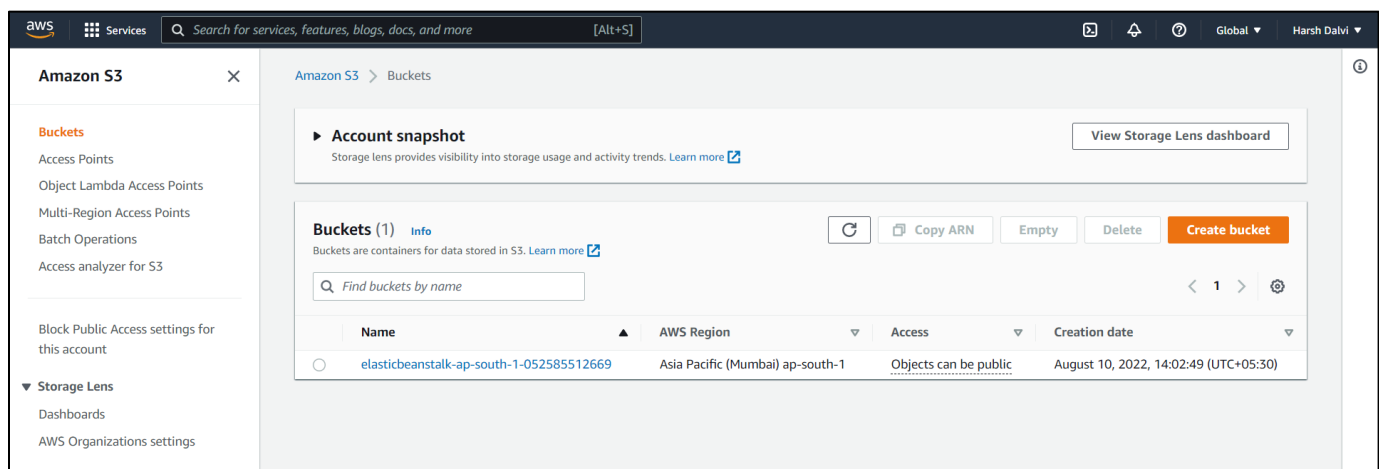
Subject: Advance DevOps , Sem: SEM V  
Class / Batch: TE-IT / Batch B

## Steps to perform the experiment:

**Step1:** Go to your Lambda function and Click on the Monitor section.



**Step 2:** Go to S3 Bucket and create a new bucket.



Name: Harsh Dalvi  
Roll No: 13

Subject: Advance DevOps , Sem: SEM V  
Class / Batch: TE-IT / Batch B

## Create bucket [Info](#)

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

### General configuration

Bucket name

Bucket name must be globally unique and must not contain spaces or uppercase letters. [See rules for bucket naming](#)

AWS Region

Asia Pacific (Mumbai) ap-south-1

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

Choose bucket

### 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 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.

Name: Harsh Dalvi  
Roll No: 13

Subject: Advance DevOps , Sem: SEM V  
Class / Batch: TE-IT / Batch B

Successfully created bucket "advexp12"  
To upload files and folders, or to configure additional bucket settings choose [View details](#).

Amazon S3 > Buckets

**Account snapshot**  
Storage lens provides visibility into storage usage and activity trends. [Learn more](#)

[View Storage Lens dashboard](#)

**Buckets (2)** [Info](#)  
Buckets are containers for data stored in S3. [Learn more](#)

[Refresh](#) [Copy ARN](#) [Empty](#) [Delete](#) [Create bucket](#)

	Name	AWS Region	Access	Creation date
<input type="radio"/>	advexp12	Asia Pacific (Mumbai) ap-south-1	Objects can be public	October 27, 2022, 14:11:12 (UTC+05:30)
<input type="radio"/>	elasticbeanstalk-ap-south-1-052585512669	Asia Pacific (Mumbai) ap-south-1	Objects can be public	August 10, 2022, 14:02:49 (UTC+05:30)

**Step 3:** Now, Under Lambda function, go to add trigger > trigger configuration.

**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.

☐ **Browse serverless app repository**  
Deploy a sample Lambda application from the AWS Serverless Application Repository.

**Basic information**

**Function name**  
Enter a name that describes the purpose of your function.  
  
Use only letters, numbers, hyphens, or underscores with no spaces.

**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

**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](#)

[Advanced settings](#)


[Cancel](#) [Create function](#)

Successfully created the function **adve12**. You can now change its code and configuration. To invoke your function with a test event, choose "Test".

Lambda > Functions > adve12

**adve12** [Throttle](#) [Copy ARN](#) [Actions](#)

**Function overview** [Info](#)


 **adve12**  
[Layers](#) (0)

[+ Add trigger](#)

[+ Add destination](#)

**Description**  
-

**Last modified**  
4 seconds ago

**Function ARN**  
 `arn:aws:lambda:ap-south-1:052585512669:function:adve12`


**Function URL** [Info](#)  
-

Name: Harsh Dalvi  
Roll No: 13

Subject: Advance DevOps , Sem: SEM V  
Class / Batch: TE-IT / Batch B

Select and add the newly created bucket here. This is your trigger bucket which can be seen in function overview.

### Trigger configuration

 **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.

× ↻

Bucket region: ap-south-1

**Event type**  
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.

**Prefix - optional**  
Enter a single optional prefix to limit the notifications to objects with keys that start with matching characters.

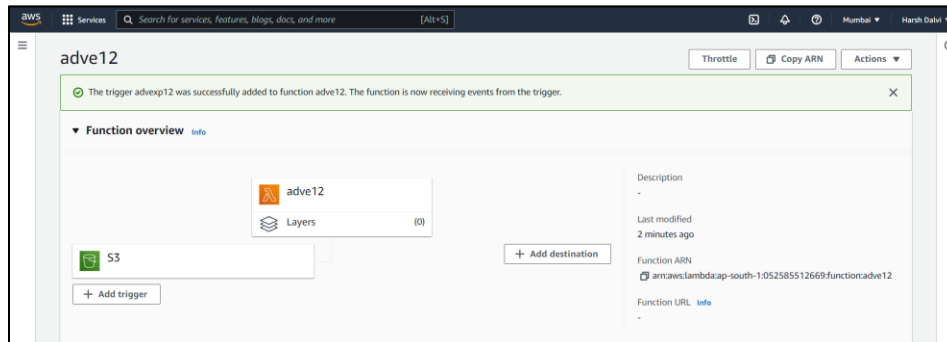
**Suffix - optional**  
Enter a single optional suffix to limit the notifications to objects with keys that end with matching characters.

**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](#)

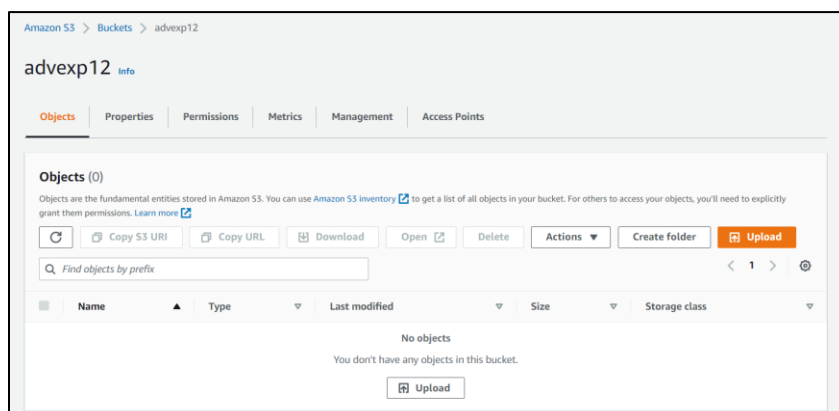
☒ I acknowledge that using the same S3 bucket for both input and output is not recommended and that this configuration can cause recursive invocations, increased Lambda usage, and increased costs.

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 4:** Now go to amazonS3, under the newly created bucket, go to upload > add files ,



Name: Harsh Dalvi  
Roll No: 13

Subject: Advance DevOps , Sem: SEM V  
Class / Batch: TE-IT / Batch B

Select a file and upload it.

Amazon S3 > Buckets > advexp12 > Upload

## Upload info

Add the files and folders you want to upload to S3. To upload a file larger than 160GB, use the AWS CLI, AWS SDK or Amazon S3 REST API. [Learn more](#)

Drag and drop files and folders you want to upload here, or choose **Add files**, or **Add folders**.

**Files and folders** (1 Total, 268.0 B)  
All files and folders in this table will be uploaded.

☐

Name	Folder	Type	Size
<input type="checkbox"/> HelloWorld.java	-	-	268.0 B

**Destination**  
Destination  
s3://advexp12  
**Destination details**  
Bucket settings that impact new objects stored in the specified destination.

**Permissions**  
Grant public access and access to other AWS accounts.

**Properties**  
Specify storage class, encryption settings, tags, and more.

Cancel Upload

**Upload succeeded**  
View details below.

## Upload: status Close

The information below will no longer be available after you navigate away from this page.

**Summary**  
Destination: s3://advexp12  
Succeeded: 1 File, 268.0 B (100.00%)  
Failed: 0 Files, 0 B (0%)

**Files and folders** (1 Total, 268.0 B)

Name	Folder	Type	Size	Status	Error
HelloWorld.java	-	-	268.0 B	Succeeded	-

**Step 5:** Come to the lambda function and in configuration check the triggers, select newly created bucket and check logs. Go to recent invocations and check if the log is present.

Code | Test | Monitor | **Configuration** | Aliases | Versions

**General configuration**  
**Triggers**  
Permissions  
Destinations  
Function URL  
Environment variables

**Triggers (1)**

☐

**Trigger**

☐ **S3: advexp12**  
arn:aws:s3:::advexp12  
**Details**

Fix errors Edit Delete Add trigger



Name: Harsh Dalvi  
Roll No: 13

Subject: Advance DevOps , Sem: SEM V  
Class / Batch: TE-IT / Batch B

**Step 6:** Go to cloudwatch>log groups> /aws/lambda/adve12 and check Log events.

The screenshot shows the AWS CloudWatch console interface for the log group `/aws/lambda/adve12`. The breadcrumb navigation at the top reads: `CloudWatch > Log groups > /aws/lambda/adve12`. The main header displays the log group name `/aws/lambda/adve12` along with buttons for `Actions`, `View in Logs Insights`, and `Search log group`. Below this, the `Log group details` section is expanded, showing a table with the following information:

Retention	Creation time	Subscription filters
Never expire	5 minutes ago	0
KMS key ID	Metric filters	Contributor Insights rules
-	0	-
Stored bytes	ARN	
-		<code>arn:aws:logs:ap-south-1:052585512669:log-group:/aws/lambda/adve12:*</code>

Below the details, there are tabs for `Log streams`, `Metric filters`, `Subscription filters`, `Contributor Insights`, and `Tags`. The `Log streams` tab is active, showing a section titled `Log streams (1)` with buttons for `Refresh`, `Delete`, `Create log stream`, and `Search all log streams`. A search bar is present with the placeholder `Filter log streams or try prefix search` and an `Exact match` checkbox. The log stream list has columns for `Log stream` and `Last event time`. One log stream is listed: `2022/10/27/[$LATEST]3e3a54f408b541d7a351acc2f725c3be` with a last event time of `2022-10-27 15:12:18 (UTC+05:30)`.

The screenshot shows the `Log events` page for the log stream `2022/10/27/[$LATEST]3e3a54f408b541d7a351acc2f725c3be`. The breadcrumb navigation is: `CloudWatch > Log groups > /aws/lambda/adve12 > 2022/10/27/[$LATEST]3e3a54f408b541d7a351acc2f725c3be`. The `Log events` section includes a filter bar with a search input, `View as text` checkbox, `Actions` button, and `Create metric filter` button. The filter bar also has a `Filter events` input and a time range selector with options: `Clear`, `1m`, `30m`, `1h`, `12h`, and `Custom`. The events table has columns for `Timestamp` and `Message`. The events shown are:

Timestamp	Message
	No older events at this moment. <a href="#">Retry</a>
2022-10-27T15:12:18.769+05:30	START RequestId: e4750bbc-e228-4448-9003-67169476ddf9 Version: \$LATEST
2022-10-27T15:12:18.776+05:30	END RequestId: e4750bbc-e228-4448-9003-67169476ddf9
2022-10-27T15:12:18.776+05:30	REPORT RequestId: e4750bbc-e228-4448-9003-67169476ddf9 Duration: 7.63 ms Billed Duration: 8 ms Memory Size: 128 MB Max Memory...
	No newer events at this moment. Auto retry paused. <a href="#">Resume</a>

**Conclusion:** From this experiment, it is concluded that we have understood the concepts of AWS S3 Buckets and AWS Lambda. In this experiment, we constructed a Lambda function which will log a message once an object is added to a specific bucket in S3. Hence, we have successfully achieved the Lab Outcome One and Lab Outcome Six (LO1 and LO6). Also, we have achieved PO1, PO2, PO3, PO4, PO5, PO9, PO10 and PO12 from this experiment.