

Experiment

Real-Time Log Processing

Problem Statement: Set up a Lambda function that triggers whenever a new log entry is added to a CloudWatch Log Group. The Lambda function should filter specific log events and store them in an S3 bucket.

Introduction:

Overview:

This case study explores the real-world implementation of monitoring, filtering, and storing AWS CloudWatch logs using an AWS Lambda function, with Amazon S3 providing persistent log storage. The project's main objective was to capture error logs generated by various AWS resources, filter them based on specific criteria, and store them in a centralized location—an Amazon S3 bucket—for further analysis.

Key Features and Applications:

Key Features:

1. **Automated Log Monitoring:** Real-time capture of CloudWatch logs without manual intervention.
2. **Error Log Filtering:** Focus on capturing logs that contain errors, filtering out unnecessary noise.
3. **S3 Storage for Error Logs:** Persistent storage of filtered error logs in S3 for easy retrieval and analysis.
4. **Scalable Serverless Solution:** Uses AWS Lambda for dynamic scaling, eliminating the need for infrastructure management.
5. **Real-time Error Detection:** Detects and processes errors as they happen.
6. **Cost-effective:** Low infrastructure costs due to the serverless architecture and S3 storage.
7. **JSON-structured Logs:** Logs are stored in a structured JSON format for easy analysis.

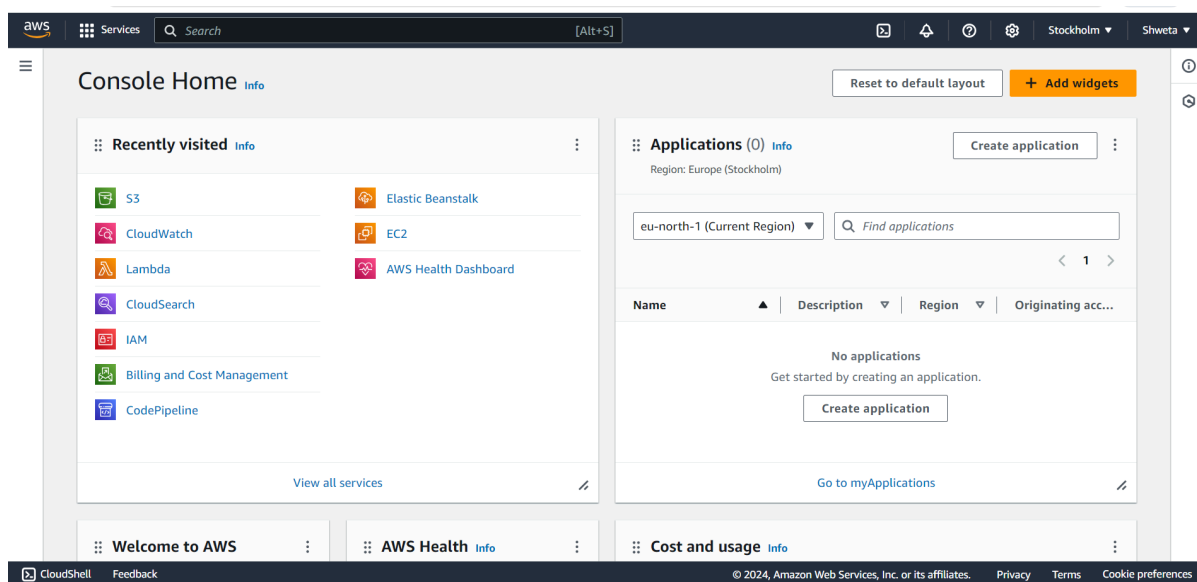
Applications:

1. **Application Monitoring:** Automatically captures and filters critical error logs in real-time, improving system performance tracking.
2. **System Diagnostics:** Provides a centralized, structured log storage solution for quick issue identification.
3. **Cloud Infrastructure Management:** Seamlessly scales with cloud services, reducing maintenance efforts and costs.

4. **Compliance Auditing:** Securely stores logs for long-term auditing and compliance checks.

Step-by-Step Execution:

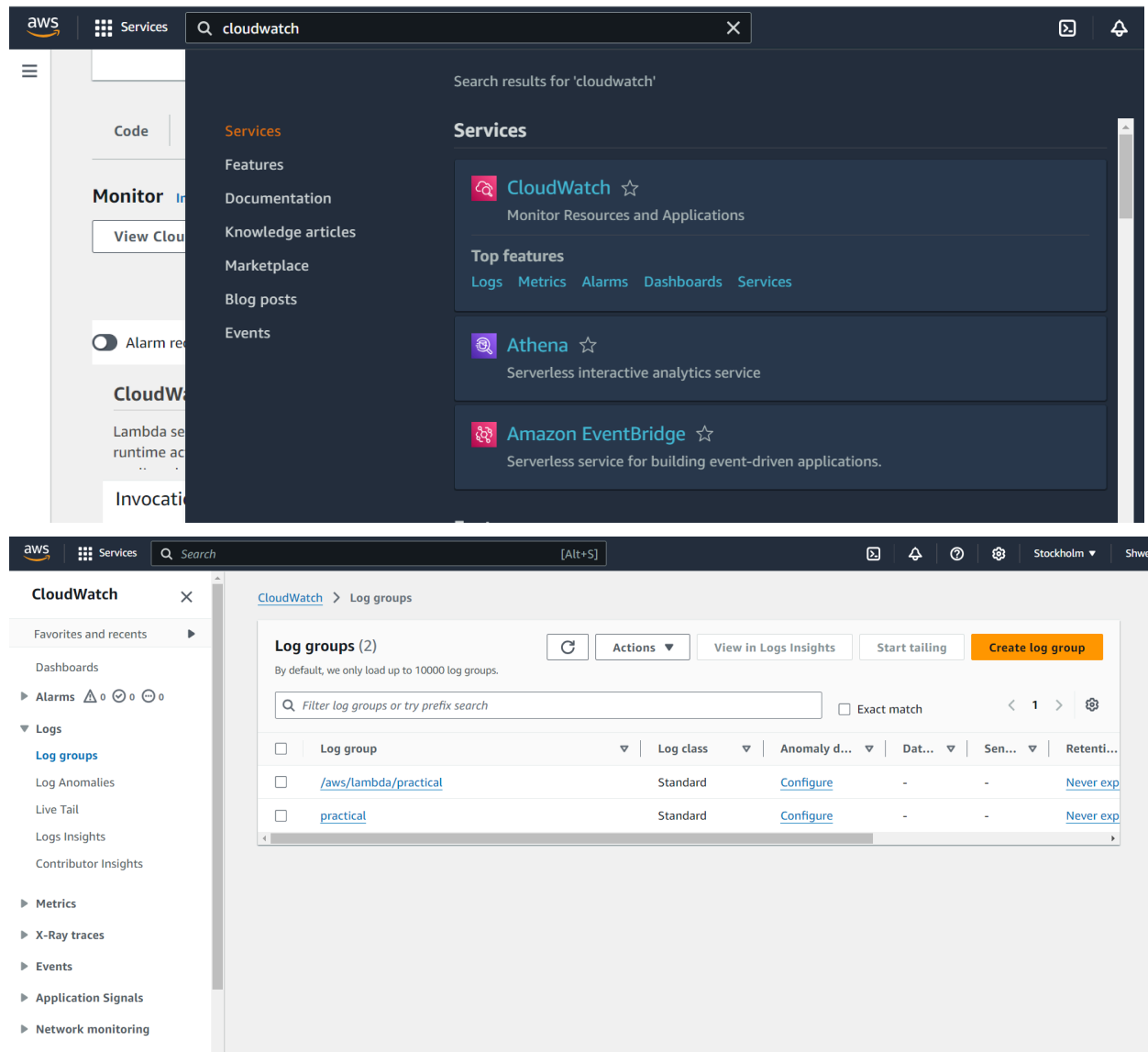
Open aws cloud service. Log into your AWS Console.



Create a CloudWatch Log Group

Open AWS CloudWatch

In the services search bar, type CloudWatch and select it from the search results.



Create a Log Group

In the CloudWatch dashboard, on the left-hand side, select Log Groups.

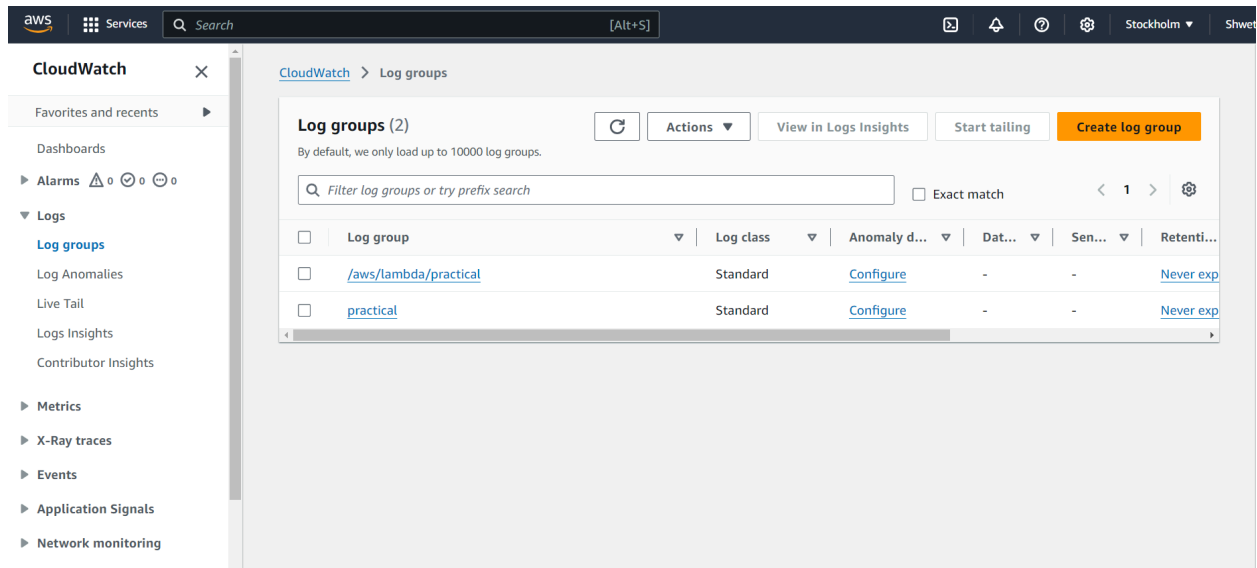
Click the Create log group button.

You will be prompted to name your log group. Choose a meaningful name that reflects the logs you intend to monitor.

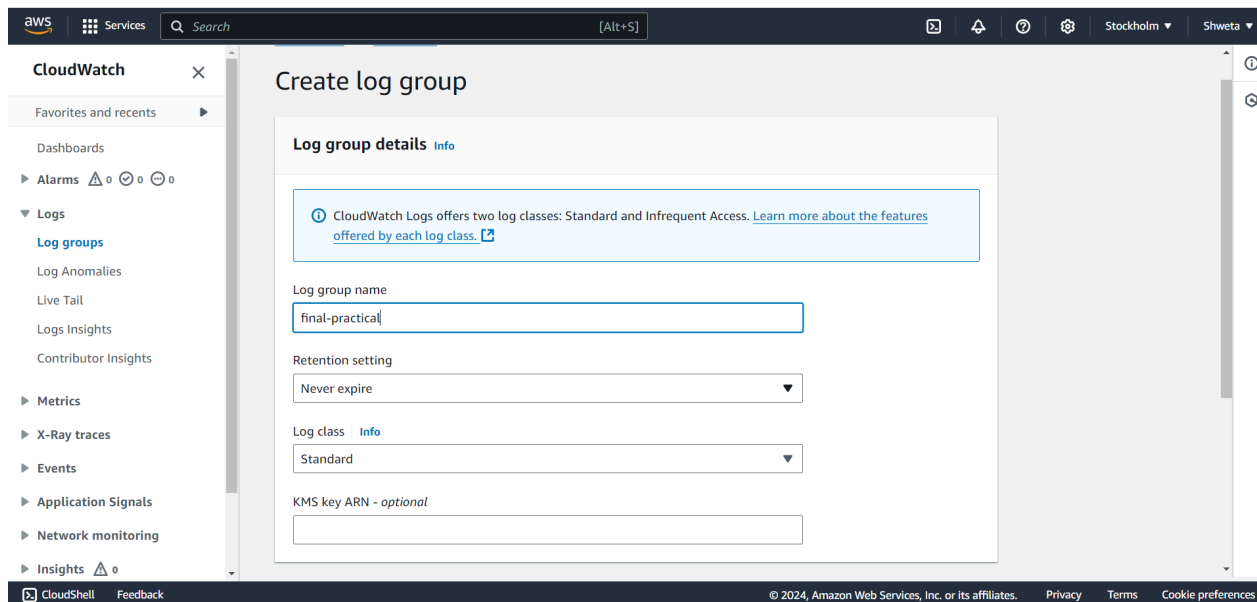
Finish Creating the Log Group

After naming the log group, click **Create**. Your log group is now ready to store logs.

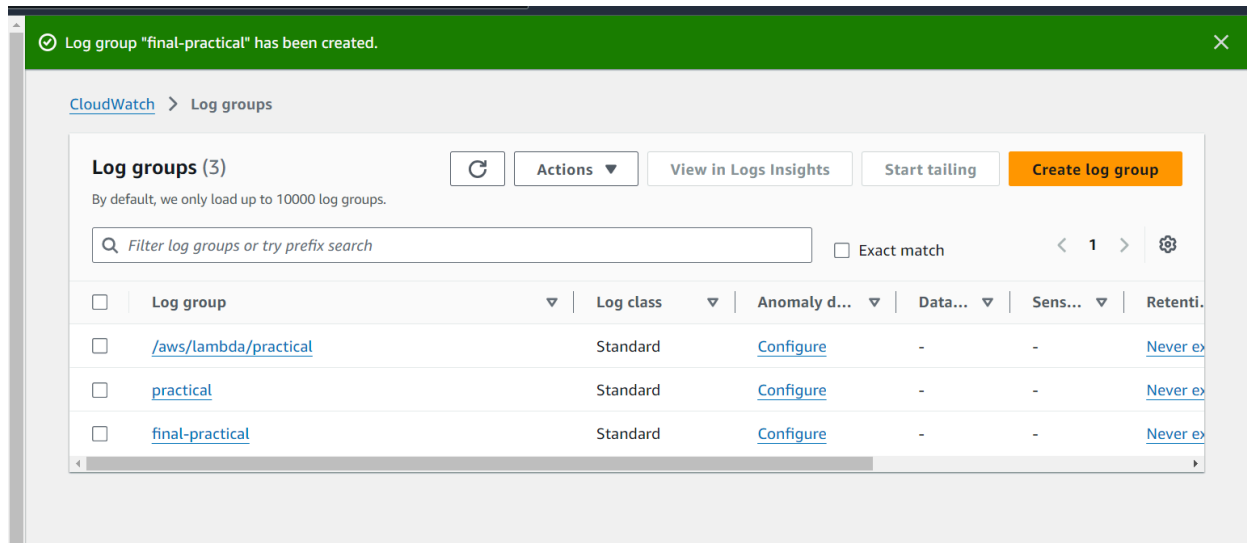
Create a CloudWatch Log Group



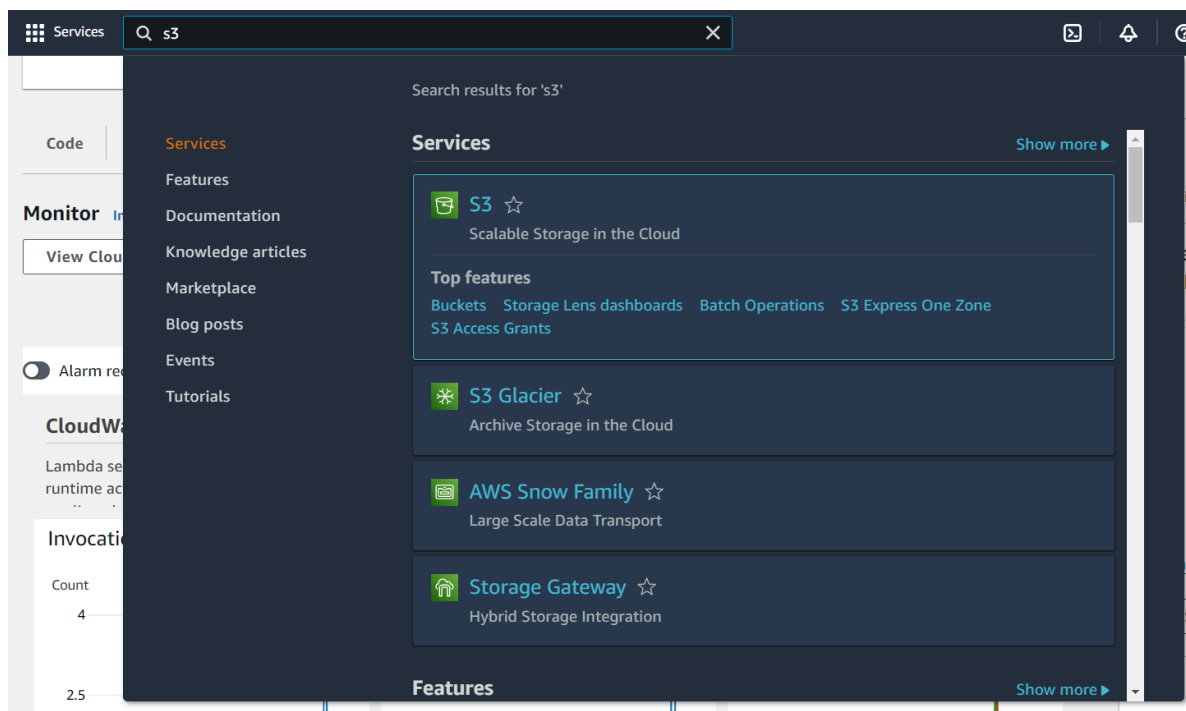
Initialize your log group by giving it a name.



A log group is created.



Search for s3 bucket.



Create a new s3 bucket.

[Amazon S3](#) > [Buckets](#) > Create bucket

Create bucket [Info](#)

Buckets are containers for data stored in S3.

General configuration

AWS Region
Europe (Stockholm) eu-north-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](#)

shweta-practical1

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](#)

CloudShell Feedback © 2024, Amazon

Give a name to your bucket. Then **untick the block all public access**, this will ensure that the bucket is public and has proper permissions for our Lambda function to write to it. Keeping the rest of the options default, we click on **Create**

aws Services Search [Alt+S] Stockholm Shweta

Block Public Access settings for this bucket

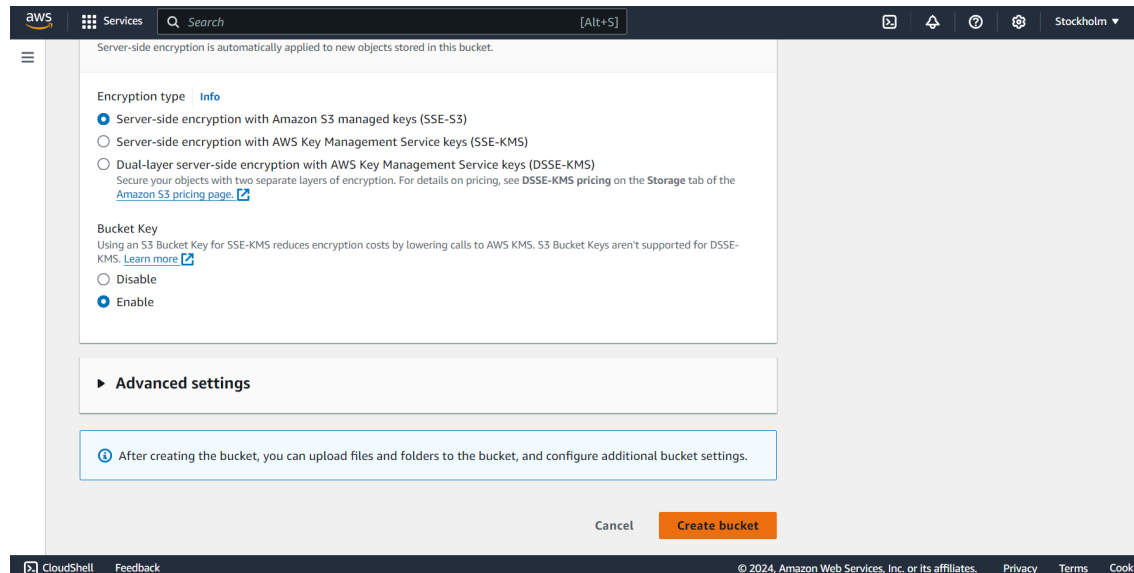
Public access is granted to buckets and objects through access control lists (ACLs), bucket policies, access point policies, or all. In order to ensure that public access to this bucket and its objects is blocked, turn on Block all public access. These settings apply only to this bucket and its access 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.

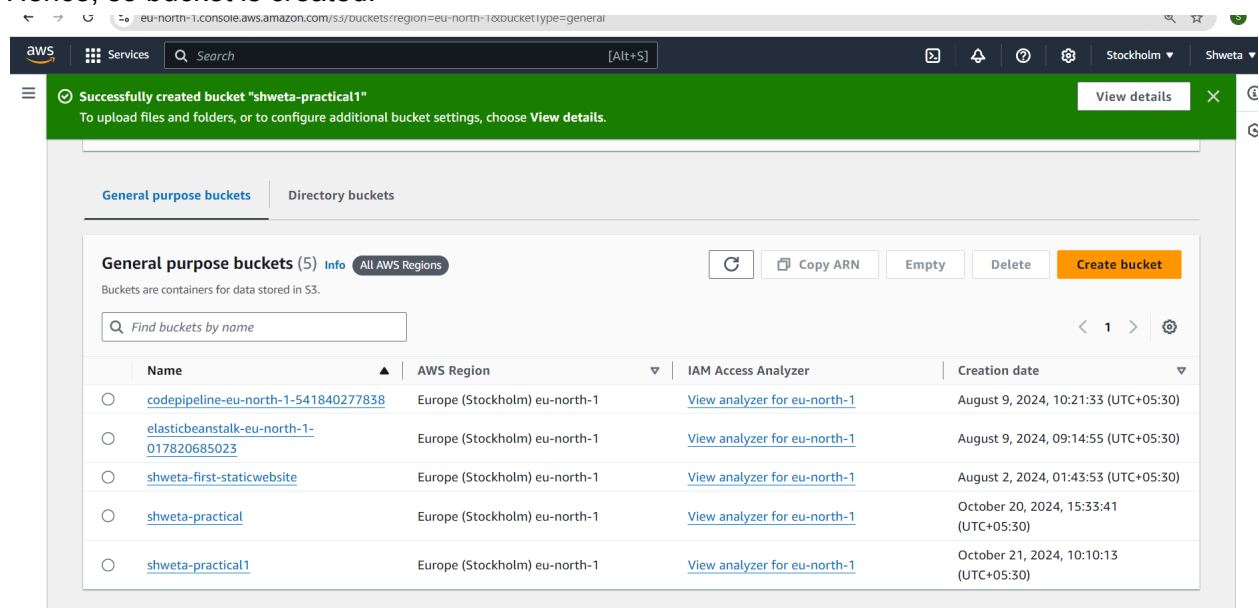
- ☐ **Block public access to buckets and objects granted through new access control lists (ACLs)**
S3 will block public access permissions applied to newly added buckets or objects, and prevent the creation of new public access ACLs for existing buckets and objects. This setting doesn't change any existing 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.

Warning 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.

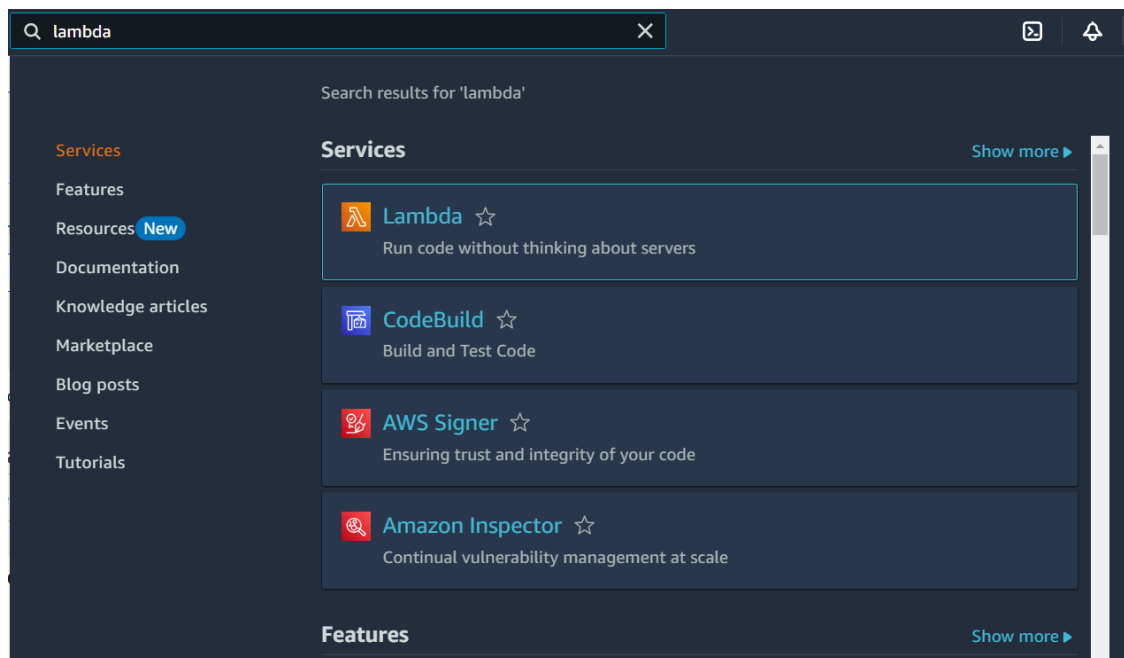
CloudShell Feedback © 2024, Amazon Web Services, Inc. or its affiliates. Privacy Terms Cookie preferences



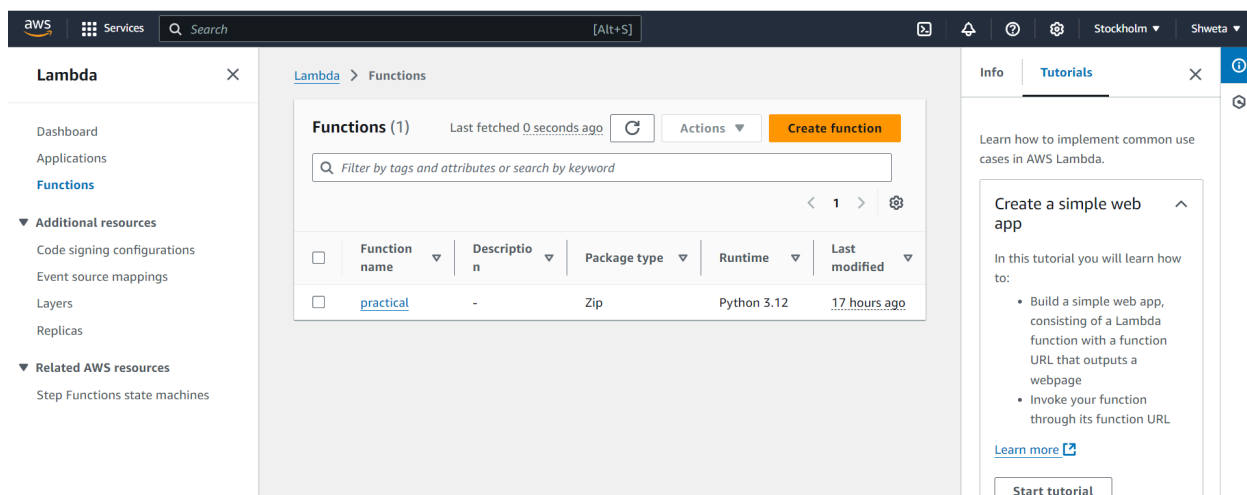
Hence, s3 bucket is created.



Now we need to create Lambda function and hence we search for Lambda.



Create a new Lambda Function.



We select the option '**Author from Scratch**'. Click **Create function**, choose **Author from scratch**, and give it a name like shweta-practical. Set **Runtime** to **Python 3.12**. We keep the rest of the settings default and create the function

Basic information

Function name
Enter a name that describes the purpose of your function.

shweta-practical

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.

Python 3.12

Architecture [Info](#)
Choose the instruction set architecture you want for your function code.

☒ x86_64
☐ arm64

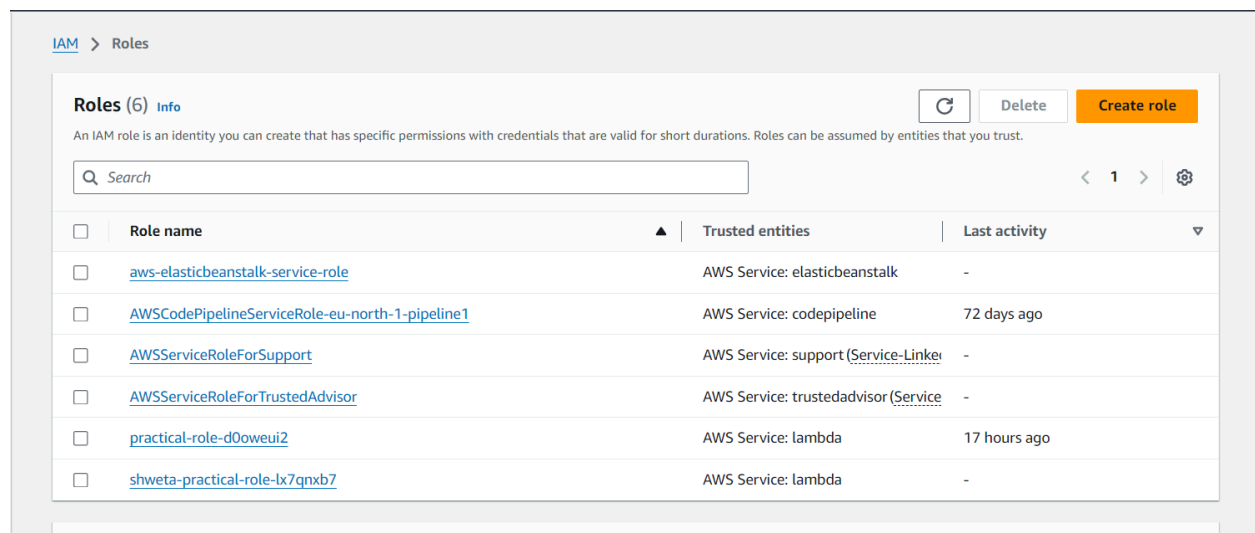
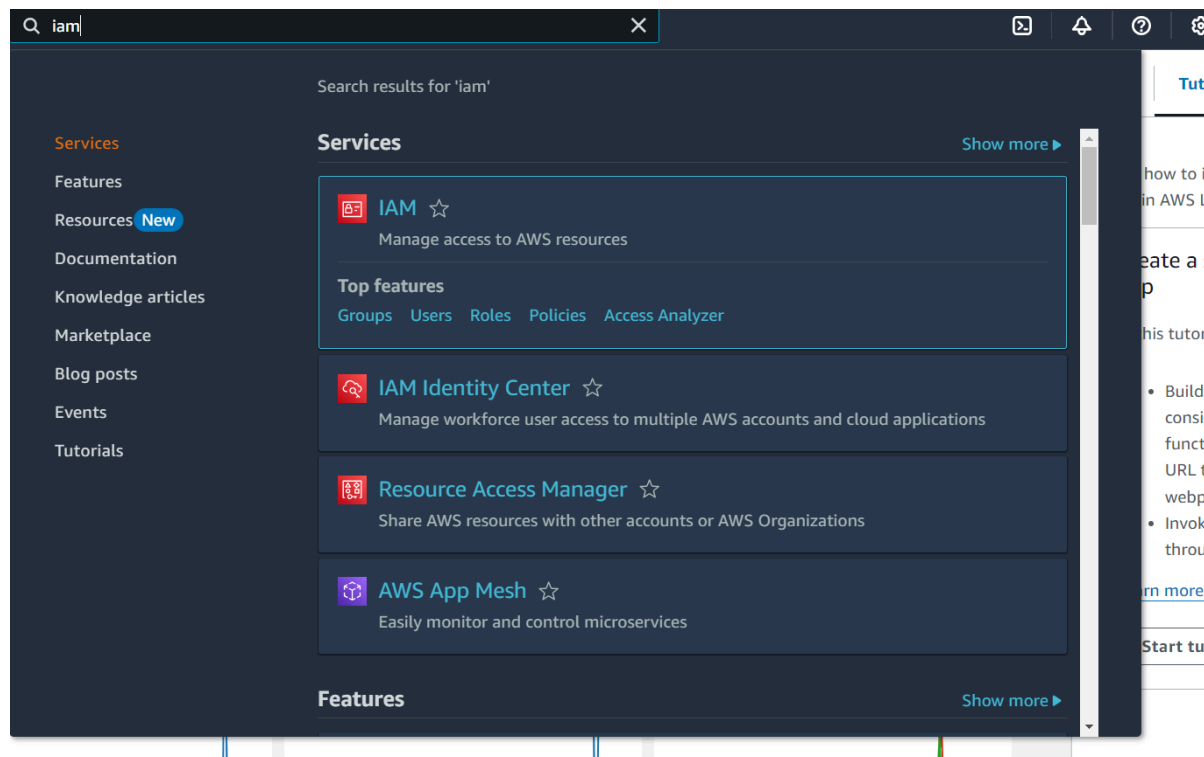
Permissions [Info](#)
By default, Lambda will create an execution role with permissions to upload logs to Amazon CloudWatch Logs. You can customize this default role later when adding triggers.

► Change default execution role

Lambda Function is now created.

The screenshot shows the AWS Lambda console interface. At the top, a green notification bar states: "Successfully created the function shweta-practical. You can now change its code and configuration. To invoke your function with a test event, choose 'Test'." The main content area is titled "shweta-practical" and includes buttons for "Throttle", "Copy ARN", and "Actions". Below this is the "Function overview" section, which has tabs for "Diagram" and "Template". The "Diagram" tab is active, showing a visual representation of the function with a box labeled "shweta-practical" and a "Layers" section with "(0)" layers. There are buttons for "+ Add trigger" and "+ Add destination". To the right of the diagram, a "Description" panel shows the function's details: "Last modified 11 seconds ago", "Function ARN arn:aws:lambda:eu-north-1:017820685023:function:shweta-practical", and "Function URL Info". On the far right, there is a "Tutorials" sidebar with a section titled "Create a simple web app" and a "Start tutorial" button.

Now, search for IAM roles in the console and then go to **Roles** from the options on the left sidebar.



A new role will be created automatically, you need not create a new role. you can see the role associated with your lambda function. It has the same name as your lambda function with role. Click on the name.

[IAM](#) > [Roles](#) > shweta-practical-role-lx7qnx7

shweta-practical-role-lx7qnx7 [Info](#)

[Delete](#)

Summary [Edit](#)

Creation date October 21, 2024, 10:11 (UTC+05:30)	ARN arn:aws:iam::017820685023:role/service-role/shweta-practical-role-lx7qnx7
Last activity -	Maximum session duration 1 hour

[Permissions](#) | [Trust relationships](#) | [Tags](#) | [Access Advisor](#) | [Revoke sessions](#)

Permissions policies (1) [Info](#)

[Refresh](#) [Simulate](#) [Remove](#) [Add permissions](#)

You can attach up to 10 managed policies.

Filter by Type
All types

< 1 > [Settings](#)

<input type="checkbox"/>	Policy name ?	Type	Attached entities
<input type="checkbox"/>	AWSLambdaBasicExecutionRole-3bdc3e...	Customer managed	1

Click onto the role created in that go to permissions

[Permissions](#) | [Trust relationships](#) | [Tags](#) | [Access Advisor](#) | [Revoke sessions](#)

Permissions policies (1) [Info](#)

[Refresh](#) [Simulate](#) [Remove](#) [Add permissions](#)

You can attach up to 10 managed policies.

Filter by Type
All types

< 1 > [Settings](#)

<input type="checkbox"/>	Policy name ?	Type	Attached entities
<input type="checkbox"/>	AWSLambdaBasicExecutionRole-3bdc3e...	Customer managed	1

[▶ Permissions boundary \(not set\)](#)

[▼ Generate policy based on CloudTrail events](#)

You can generate a new policy based on the access activity for this role, then customize, create, and attach it to this role. AWS uses your CloudTrail events to identify the services and actions used and generate a policy. [Learn more](#)

[Generate policy](#)

No requests to generate a policy in the past 7 days.

In Add permissions attach policies

Permissions policies (2) [Info](#)

You can attach up to 10 managed policies.

Filter by Type: All types

Policy name	Type	Attached entities
AmazonS3FullAccess	AWS managed	2
AWSLambdaBasicExecutionRole-3bdc3e...	Customer managed	1

Buttons: Add permissions, Attach policies, Create inline policy

Add the following permissions:

- Amazon S3 full access (to allow Lambda to write to S3).
- CloudWatchFullAccess (to allow Lambda to read logs from CloudWatch).

Policy was successfully attached to role.

Permissions policies (3) [Info](#)

You can attach up to 10 managed policies.

Filter by Type: All types

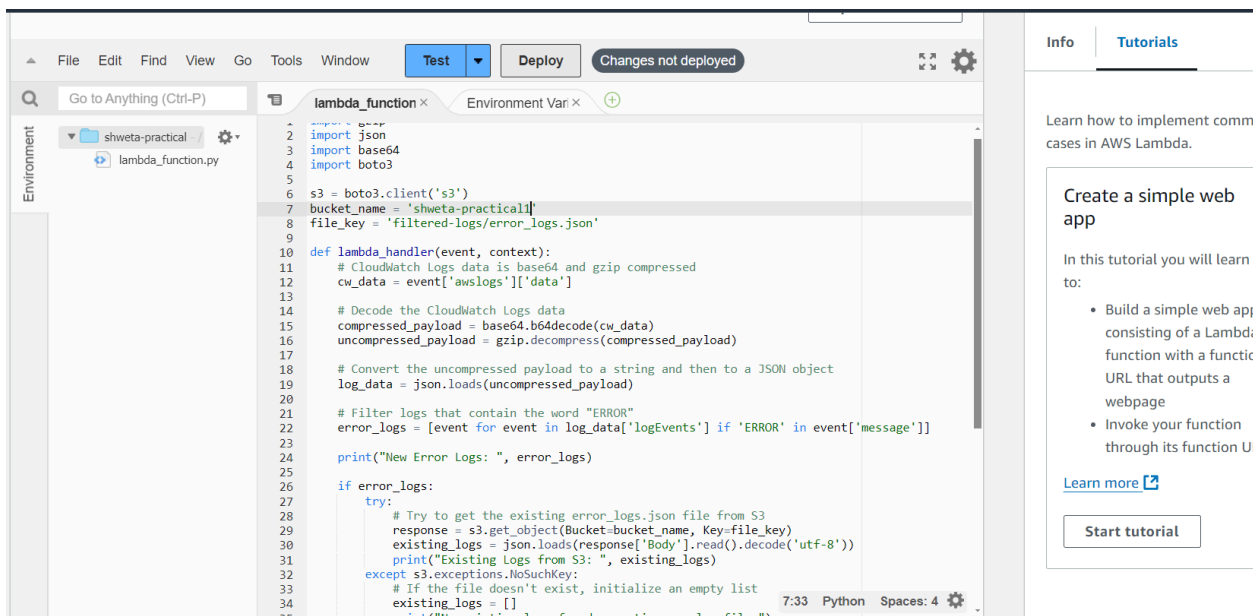
Policy name	Type	Attached entities
AmazonS3FullAccess	AWS managed	2
AWSLambdaBasicExecutionRole-3bdc3e...	Customer managed	1
CloudWatchLogsFullAccess	AWS managed	2

Buttons: Add permissions

► Permissions boundary (not set)

▼ Generate policy based on CloudTrail events

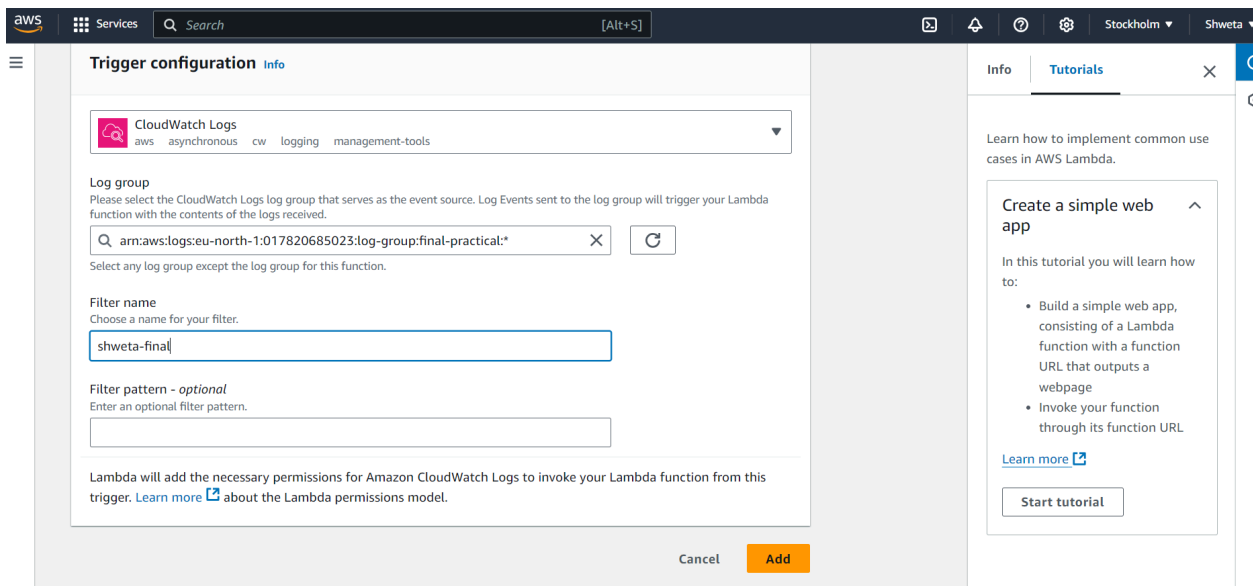
Navigate back to the lambda function, in the code section we add the following code. This Lambda function will filter logs containing the keyword "ERROR" and store them in an S3 bucket.



The screenshot shows a code editor with a file named `lambda_function.py`. The code is a Python Lambda function that processes CloudWatch Logs data. It imports `json`, `base64`, and `boto3`. It initializes an S3 client and defines bucket and file names. The `lambda_handler` function decodes the log data, filters for 'ERROR' messages, and prints them. It also attempts to retrieve existing logs from S3 and append the new logs.

```
1 import json
2 import base64
3 import boto3
4
5
6 s3 = boto3.client('s3')
7 bucket_name = 'shweta-practical1'
8 file_key = 'filtered-logs/error_logs.json'
9
10 def lambda_handler(event, context):
11     # CloudWatch Logs data is base64 and gzip compressed
12     cw_data = event['awslogs']['data']
13
14     # Decode the CloudWatch Logs data
15     compressed_payload = base64.b64decode(cw_data)
16     uncompressed_payload = gzip.decompress(compressed_payload)
17
18     # Convert the uncompressed payload to a string and then to a JSON object
19     log_data = json.loads(uncompressed_payload)
20
21     # Filter logs that contain the word "ERROR"
22     error_logs = [event for event in log_data['logEvents'] if 'ERROR' in event['message']]
23
24     print("New Error Logs: ", error_logs)
25
26     if error_logs:
27         try:
28             # Try to get the existing error_logs.json file from S3
29             response = s3.get_object(Bucket=bucket_name, Key=file_key)
30             existing_logs = json.loads(response['Body'].read().decode('utf-8'))
31             print("Existing Logs from S3: ", existing_logs)
32         except s3.exceptions.NoSuchKey:
33             # If the file doesn't exist, initialize an empty list
34             existing_logs = []
```

In Lambda function, click on **Add Trigger**. Select **CloudWatch Logs** from the dropdown. Choose the log group you created earlier (shweta-final). Choose a filter name and click on **Add**.



The screenshot shows the AWS Lambda console's 'Trigger configuration' dialog. The 'Log group' is set to 'arn:aws:logs:eu-north-1:017820685023:log-group:final-practical:*'. The 'Filter name' is 'shweta-final'. The 'Filter pattern' is optional and empty. The dialog includes a 'Cancel' button and an 'Add' button.

Trigger configuration Info

CloudWatch Logs
aws asynchronous cw logging management-tools

Log group
Please select the CloudWatch Logs log group that serves as the event source. Log Events sent to the log group will trigger your Lambda function with the contents of the logs received.

arn:aws:logs:eu-north-1:017820685023:log-group:final-practical:*

Select any log group except the log group for this function.

Filter name
Choose a name for your filter.

shweta-final

Filter pattern - optional
Enter an optional filter pattern.

Lambda will add the necessary permissions for Amazon CloudWatch Logs to invoke your Lambda function from this trigger. [Learn more](#) about the Lambda permissions model.

Cancel Add

The screenshot displays the AWS Lambda console interface. The top section shows the 'shweta-practical' function with a success message: 'The trigger final-practical was successfully added to function shweta-practical. The function is now receiving events from the trigger.' Below this, the 'Function overview' tab is active, showing a diagram of the function with its layers and destinations (CloudWatch Logs). The function's description, last modified time (7 minutes ago), and function ARN are also visible. On the right, a 'Tutorials' sidebar offers a guide on 'Create a simple web app'.

The bottom section shows the 'shweta-practical-role-lx7qnx7' IAM role. The 'Summary' tab is active, displaying the role's creation date (October 21, 2024, 10:11 (UTC+05:30)) and last activity. A green notification bubble indicates 'ARN copied', showing the role's ARN: `arn:aws:iam::017820685023:role/service-role/shweta-practical-role-lx7qnx7`. The role's maximum session duration is set to 1 hour.

Go to the **S3 Console** and select the bucket you created. Navigate to the **Permissions** tab. Scroll down to **Bucket Policy** and click **Edit**. In the bucket policy, make sure you add the correct **bucket name** in the resource and the correct **role ARN** for your lambda role. This policy gives your Lambda function permission to upload objects to your bucket.

The bucket policy, written in JSON, provides access to the objects stored in the bucket. Bucket policies don't apply to objects owned by other accounts. [Learn more](#)

Bucket ARN
arn:aws:s3:::shweta-practical1

Policy

```
1 {  
2   "Version": "2012-10-17",  
3   "Statement": [  
4     {  
5       "Effect": "Allow",  
6       "Principal": {  
7         "AWS": "arn:aws:iam::017820685023:role/service-role/shweta-practical-role-lx",  
8       },  
9       "Action": "s3:PutObject",  
10      "Resource": "arn:aws:s3:::shweta-practical1/*"  
11    }  
12  ]  
13 }
```

Edit statement Revert

Add actions

Choose a service

Filter services

Included

S3

Available

AMP

API Gateway

Successfully edited bucket policy.

[Amazon S3](#) > [Buckets](#) > shweta-practical1

shweta-practical1 [Info](#)

Objects | Properties | **Permissions** | Metrics | Management | Access Points

Permissions overview

Access finding

Access findings are provided by IAM external access analyzers. Learn more about [How IAM analyzer findings work](#)

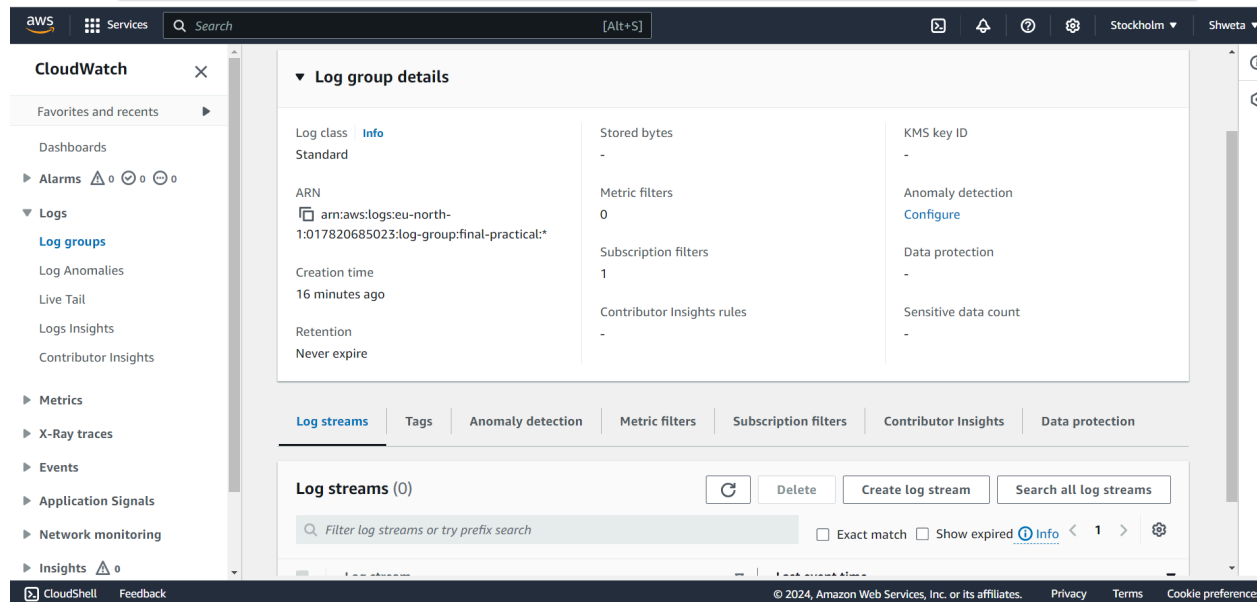
[View analyzer for eu-north-1](#)

Block public access (bucket settings)

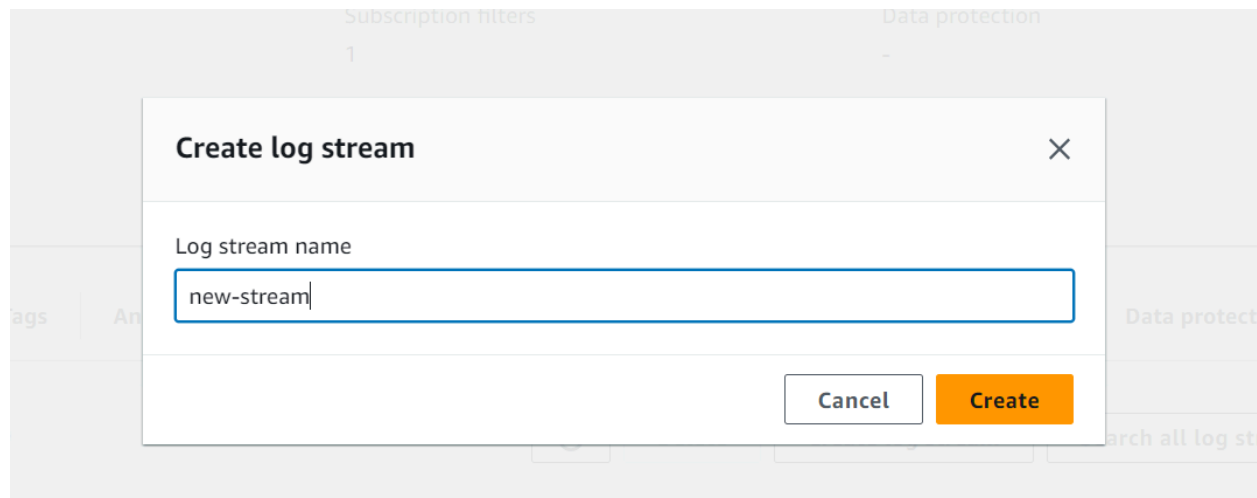
Public access is granted to buckets and objects through access control lists (ACLs), bucket policies, access point policies, or all. In order to ensure that public access to all your S3 buckets and objects is blocked, turn on Block all public access. These settings apply only to this bucket and its access 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 your buckets or objects within, you can customize the individual settings below to suit your specific storage use cases. [Learn more](#)

[Edit](#)

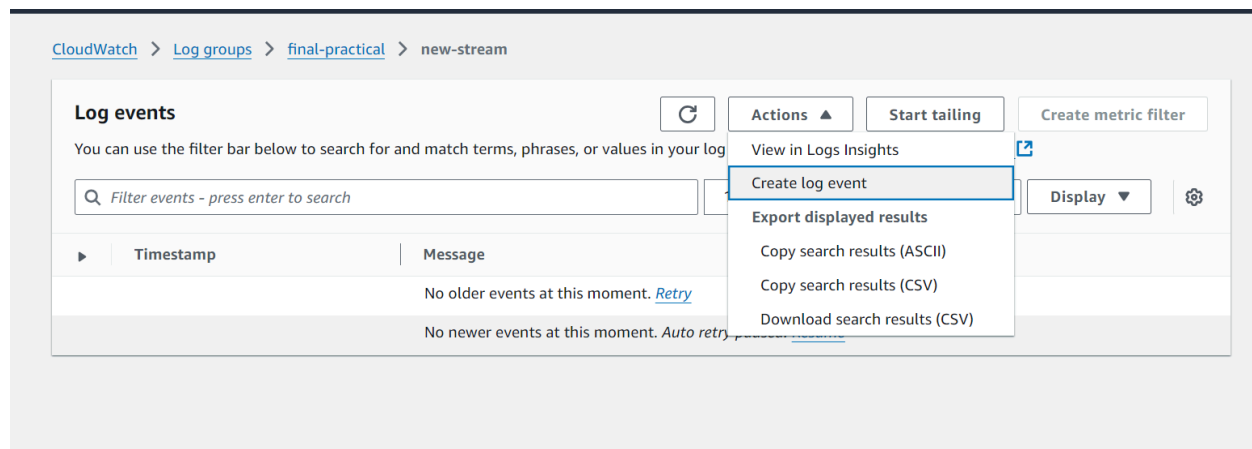
On the CloudWatch dashboard, on the left sidebar, click on **Logs** → **Log Groups**. This will display all the Log Groups associated with your AWS account. Select the one associated with your lambda function.



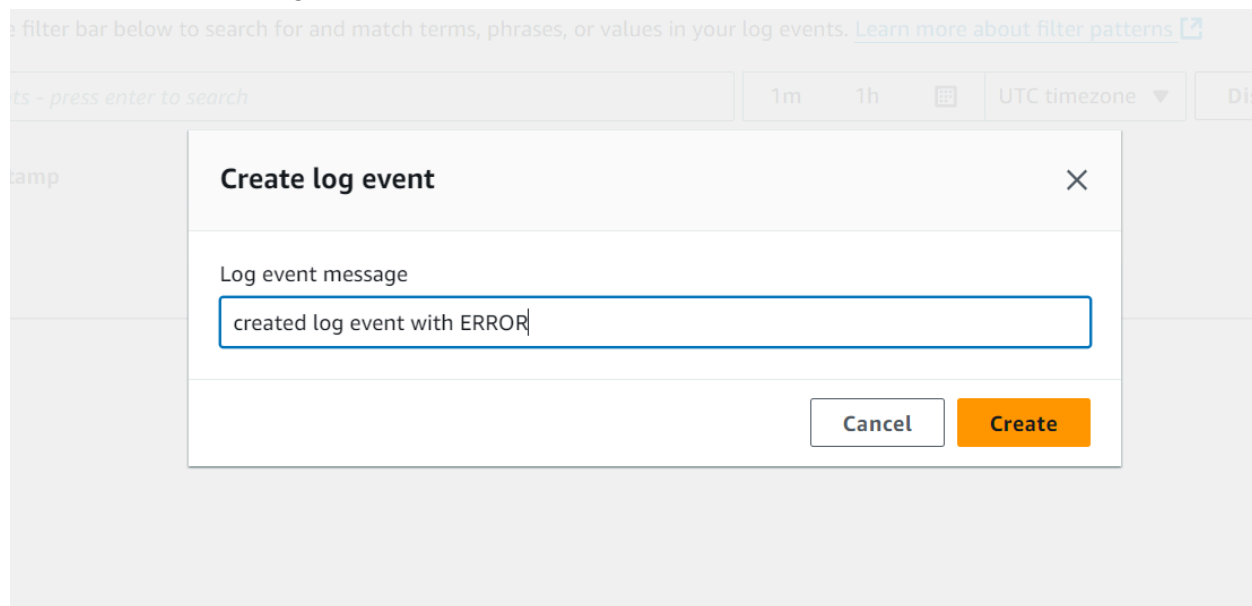
Inside the Log Group, you'll see **Log Streams**. A log stream is a sequence of log events for a specific resource that writes to CloudWatch. Click the **Create log stream** button, name the stream (e.g., new-stream), and proceed.



Click on your **Log Stream** (e.g., new-stream). You will now see an option to add log events manually. Click **Actions** → **Create log event**.



Enter the log message you want to trigger your Lambda function. Since we're filtering for the keyword ERROR, enter something like:



And some without the ERROR keyword:

press enter to search

1m 1h UTC timezone ▼ Dis

Create log event

Log event message

only log event

Cancel Create

No newer events at this moment. Auto retry paused. [Resume](#)

- press enter to search

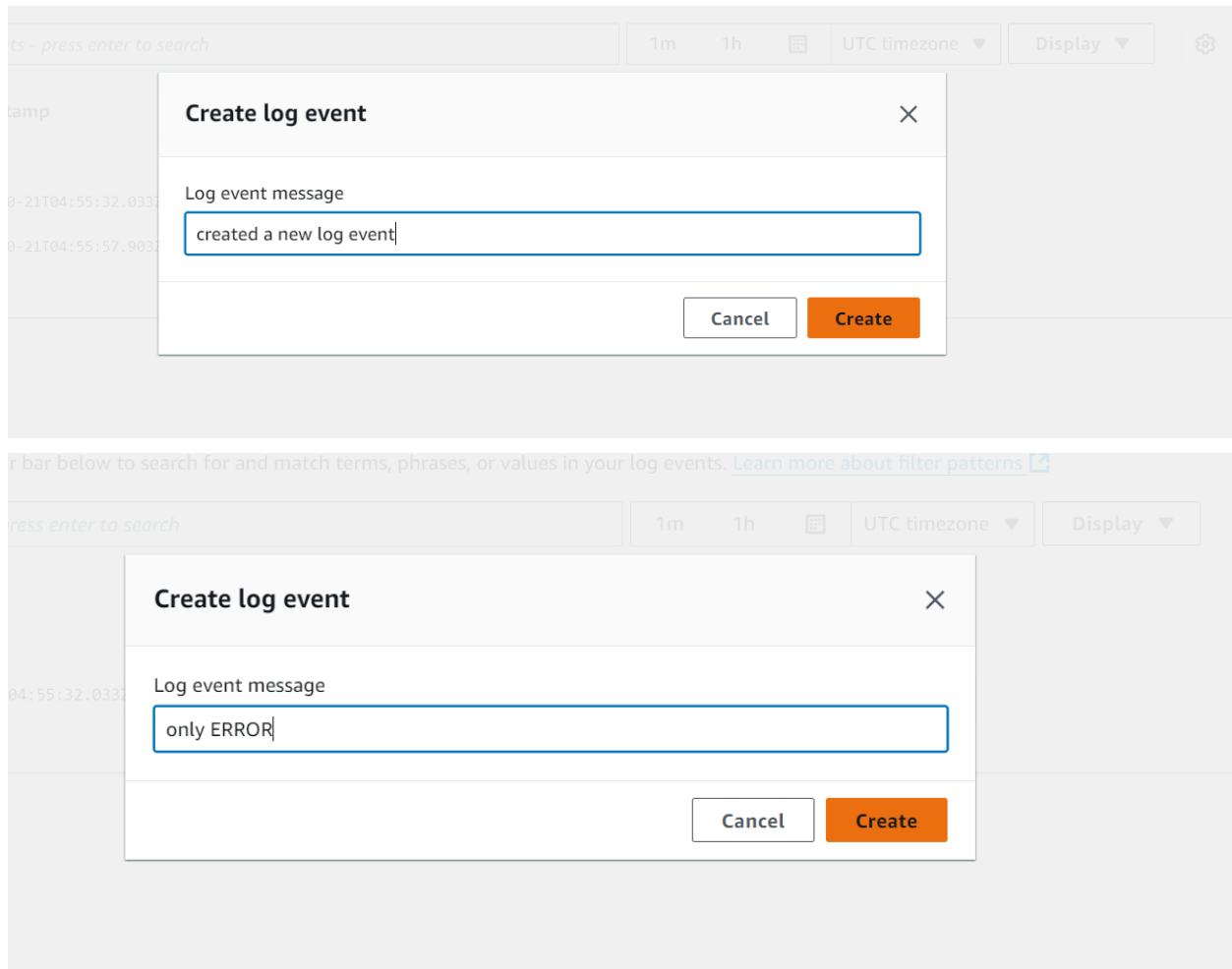
1m 1h UTC timezone ▼

Create log event

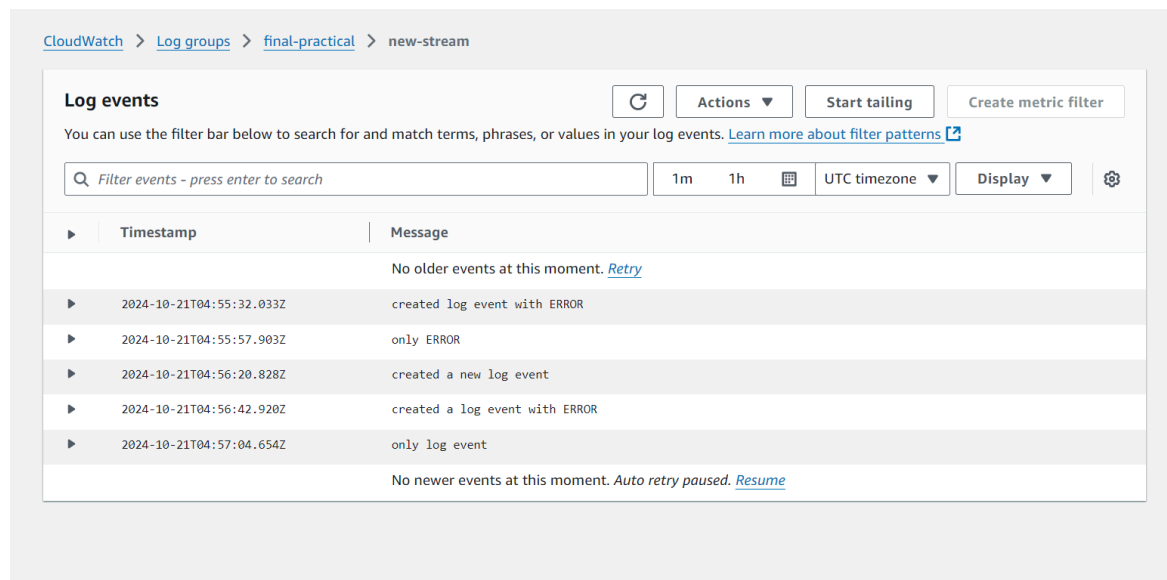
Log event message

created a log event with ERROR

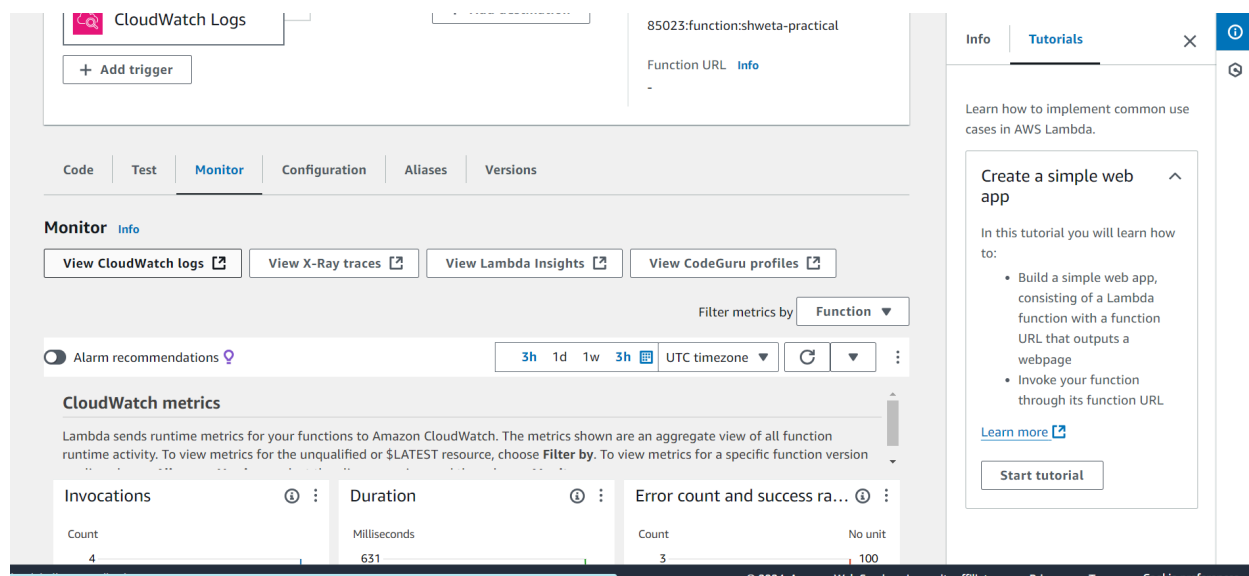
Cancel Create



Create more log events like so with and without the keyword ERROR.



Find your Lambda function (the one you set up to filter logs and write to S3), and click on it. In the Lambda function dashboard, click on the **Monitoring** tab. In the Monitoring tab, click on **View logs in CloudWatch**.



Then it will lead you to a generated log group for this function. Select the latest log stream. This will open CloudWatch and show you logs generated by your Lambda function.

The screenshot displays the AWS CloudWatch console interface. At the top, the breadcrumb navigation shows 'CloudWatch > Log groups > /aws/lambda/shweta-practical'. The main header for the log group is '/aws/lambda/shweta-practical', followed by buttons for 'Actions', 'View in Logs Insights', 'Start tailing', and a 'Search log group' button.

The 'Log group details' section is expanded, showing a table with three columns: Log class, Stored bytes, and KMS key ID. The 'Log class' column contains 'Standard' and 'ARN' (arn:aws:logs:eu-north-1:017820685023:log-group:/aws/lambda/shweta-practical:*). The 'Stored bytes' column shows '0'. The 'KMS key ID' column shows '-'. Other details include 'Creation time' (3 minutes ago), 'Retention' (Never expire), 'Metric filters' (0), 'Subscription filters' (0), 'Contributor Insights rules' (-), 'Anomaly detection' (Configure), 'Data protection' (-), and 'Sensitive data count' (-).

Below the details section is a horizontal tab bar with 'Log streams' selected. The 'Log streams' section shows a table with one log stream. The table has columns for 'Log stream' and 'Last event time'. The log stream is named '2024/10/21/[\$LATEST]c7a4253d714d437889af653389c14b54' and its last event time is '2024-10-21 04:55:34 (UTC)'.

Log class	Stored bytes	KMS key ID
Standard	-	-
ARN arn:aws:logs:eu-north-1:017820685023:log-group:/aws/lambda/shweta-practical:*	Metric filters 0	Anomaly detection Configure
Creation time 3 minutes ago	Subscription filters 0	Data protection -
Retention Never expire	Contributor Insights rules -	Sensitive data count -

Log stream	Last event time
2024/10/21/[\$LATEST]c7a4253d714d437889af653389c14b54	2024-10-21 04:55:34 (UTC)

Look for recent log entries to see if the Lambda function was triggered. Check for any errors or information logs indicating that the function processed the log event and uploaded data to S3.

▶	2024-10-21T05:03:38.346Z	END RequestId: d4b57e7f-76f6-4b7b-8bec-4b255d03541d
▶	2024-10-21T05:03:38.346Z	REPORT RequestId: d4b57e7f-76f6-4b7b-8bec-4b255d03541d Duration: 518.54 ms Billed Duration: 519 ms Memo...
▶	2024-10-21T05:03:57.616Z	START RequestId: fe58b43c-a8f8-4396-b84e-eea22b02f135 Version: \$LATEST
▶	2024-10-21T05:03:57.616Z	New Error Logs: []
▶	2024-10-21T05:03:57.617Z	END RequestId: fe58b43c-a8f8-4396-b84e-eea22b02f135
▶	2024-10-21T05:03:57.617Z	REPORT RequestId: fe58b43c-a8f8-4396-b84e-eea22b02f135 Duration: 1.40 ms Billed Duration: 2 ms Memory S...
▶	2024-10-21T05:04:27.711Z	START RequestId: 710ebe0d-9a4b-4345-a789-0c65762cf852 Version: \$LATEST
▶	2024-10-21T05:04:27.712Z	New Error Logs: [{'id': '38568850392252823442488811680852097048507749676624379904', 'timestamp': 172948...
▶	2024-10-21T05:04:28.144Z	Existing Logs from S3: [{'id': '38567386639823580397073265589402655548577489296348086272', 'timestamp': ...
▶	2024-10-21T05:04:28.144Z	Combined Logs (Existing + New): [{'id': '38567386639823580397073265589402655548577489296348086272', 'ti...
▶	2024-10-21T05:04:28.205Z	END RequestId: 710ebe0d-9a4b-4345-a789-0c65762cf852
▶	2024-10-21T05:04:28.205Z	REPORT RequestId: 710ebe0d-9a4b-4345-a789-0c65762cf852 Duration: 493.35 ms Billed Duration: 494 ms Memo...
▶	2024-10-21T05:04:44.235Z	START RequestId: bd580351-ef24-46c3-a490-a6ff4b58ed01 Version: \$LATEST
▶	2024-10-21T05:04:44.235Z	New Error Logs: []
▶	2024-10-21T05:04:44.236Z	END RequestId: bd580351-ef24-46c3-a490-a6ff4b58ed01
▶	2024-10-21T05:04:44.236Z	REPORT RequestId: bd580351-ef24-46c3-a490-a6ff4b58ed01 Duration: 1.54 ms Billed Duration: 2 ms Memory S...
No newer events at this moment. <i>Auto retry paused.</i> Resume		

[Back to top](#)

aws

Services

Search

[Alt+S]

CloudWatch

Log groups

/aws/lambda/shweta-practical

2024/10/21/[LATEST]eb3f7ef4abe746c39cae4ea5d2c064d8

Log events

You can use the filter bar below to search for and match terms, phrases, or values in your log events. [Learn more about filter patterns](#)

Filter events - press enter to search

1m 1h

UTC timezone

Display

Timestamp	Message
No older events at this moment. Retry	
▶ 2024-10-21T05:03:11.882Z	INIT_START Runtime Version: python:3.12.v36 Runtime Version ARN: arn:aws:lambda:eu-north-1::runtime:188...
▶ 2024-10-21T05:03:12.313Z	START RequestId: a9345ff1-65e4-487c-a3b0-a2bb8d584b36 Version: \$LATEST
▶ 2024-10-21T05:03:12.314Z	New Error Logs: [{'id': '38568848705156847683250109685548612906128122591324798976', 'timestamp': 172948...
▶ 2024-10-21T05:03:12.846Z	Existing Logs from S3: [{'id': '38567386639823580397073265589402655548577489296348086272', 'timestamp': ...
▶ 2024-10-21T05:03:12.846Z	Combined Logs (Existing + New): [{'id': '38567386639823580397073265589402655548577489296348086272', 'ti...
▶ 2024-10-21T05:03:12.925Z	END RequestId: a9345ff1-65e4-487c-a3b0-a2bb8d584b36
▶ 2024-10-21T05:03:12.925Z	REPORT RequestId: a9345ff1-65e4-487c-a3b0-a2bb8d584b36 Duration: 611.58 ms Billed Duration: 612 ms Memo...
▶ 2024-10-21T05:03:37.827Z	START RequestId: d4b57e7f-76f6-4b7b-8bec-4b255d03541d Version: \$LATEST
▶ 2024-10-21T05:03:37.827Z	New Error Logs: [{'id': '38568849277572375439134144514025535692791253433999425536', 'timestamp': 172948...

In the list of buckets, find and click on the bucket you configured in your Lambda function. Inside the S3 bucket, look for newly uploaded files. They should contain logs that match the filter pattern.

The image shows two screenshots of the Amazon S3 console. The top screenshot displays the 'shweta-practical1' bucket with a list of objects. The bottom screenshot shows the 'filtered-logs/' folder containing a single object named 'error_logs.json'.

Top Screenshot: Bucket Overview

Navigation: Amazon S3 > Buckets > shweta-practical1

shweta-practical1 [Info](#)

Objects | Properties | Permissions | Metrics | Management | Access Points

Objects (1) [Info](#)

Actions: Refresh, Copy S3 URI, Copy URL, Download, Open, Delete, Actions, Create folder

Upload

Objects are the fundamental entities stored in Amazon S3. You can use [Amazon S3 inventory](#) to get a list of all objects in your bucket. For others to access your objects, you'll need to explicitly grant them permissions. [Learn more](#)

Find objects by prefix

<input type="checkbox"/>	Name	Type	Last modified	Size	Storage class
<input type="checkbox"/>	filtered-logs/	Folder	-	-	-

Bottom Screenshot: Folder Overview

Navigation: Amazon S3 > Buckets > shweta-practical1 > filtered-logs/

filtered-logs/ [Copy S3 URI](#)

Objects | Properties

Objects (1) [Info](#)

Actions: Refresh, Copy S3 URI, Copy URL, Download, Open, Delete, Actions, Create folder

Upload

Objects are the fundamental entities stored in Amazon S3. You can use [Amazon S3 inventory](#) to get a list of all objects in your bucket. For others to access your objects, you'll need to explicitly grant them permissions. [Learn more](#)

Find objects by prefix

<input type="checkbox"/>	Name	Type	Last modified	Size	Storage class
<input type="checkbox"/>	error_logs.json	json	October 21, 2024, 10:41:06 (UTC+05:30)	505.0 B	Standard

Like we configured in the code part. The logs are stored in the error_logs.json file. Download or Open the file.

The screenshot shows the AWS S3 console interface. The breadcrumb navigation is: Amazon S3 > Buckets > shweta-practical1 > filtered-logs/ > error_logs.json. The object name 'error_logs.json' is displayed with an 'Info' link. Action buttons include 'Copy S3 URI', 'Download', 'Open', and 'Object actions'. Below the navigation tabs (Properties, Permissions, Versions), the 'Object overview' section displays the following details:

Owner a73a302f40e2004301d8663053076b0c789017bbd8870828daa5acdd9f193feb	S3 URI s3://shweta-practical1/filtered-logs/error_logs.json
AWS Region Europe (Stockholm) eu-north-1	Amazon Resource Name (ARN) arn:aws:s3:::shweta-practical1/filtered-logs/error_logs.json
Last modified October 21, 2024, 10:41:06 (UTC+05:30)	Entity tag (Etag) 372cafb1750e09769aed43be5707db2
Size 505.0 B	Object URL https://shweta-practical1.s3.eu-north-1.amazonaws.com/filtered-l ogs/error_logs.json
Type	

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

In the file, we can see the records of all the logs with the keyword ERROR.

The screenshot shows a code editor with the file 'error_logs (7).json' open. The file path is 'C:\> Users > excel > Downloads > error_logs (7).json > ...'. The JSON content is as follows:

```
1 [
2   {
3     "id": "38568857452178136902916424964166734763694543777333379072",
4     "timestamp": 1729487382992,
5     "message": "created log event with ERROR"
6   },
7   {
8     "id": "38568858527230160688482174806642852030532022189133791232",
9     "timestamp": 1729487431199,
10    "message": "only ERROR"
11  },
12  {
13    "id": "38568859260991579955735268071601963095410208889384402944",
14    "timestamp": 1729487464102,
15    "message": "created a log event with ERROR"
16  }
17 ]
```


Additional Guidelines for Troubleshooting

1. **CloudWatch Logs:** Review CloudWatch logs for any errors or warnings during Lambda function execution.
2. **Check IAM Permissions:** Ensure the Lambda function has the necessary permissions to read from CloudWatch and write to S3.
3. **Validate S3 Configuration:** Double-check that the bucket name in your Lambda code matches the actual S3 bucket name.
4. **Examine Lambda Logs:** Confirm that logs are being correctly filtered and stored. Adjust the code if logs are being overwritten instead of appended.
5. **Event Source Mapping:** Make sure Lambda is correctly triggered by CloudWatch Logs or other event sources.

Conclusion

This case study shows how AWS services like Lambda, CloudWatch Logs, and S3 can be combined to automate error-log monitoring. By using CloudWatch to capture logs, Lambda to filter for errors, and S3 for long-term storage, this solution enables real-time log processing without requiring complex infrastructure. This approach is highly scalable, cost-effective, and suitable for large-scale systems, providing a simple and efficient way to manage error logs and improve system monitoring.