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.

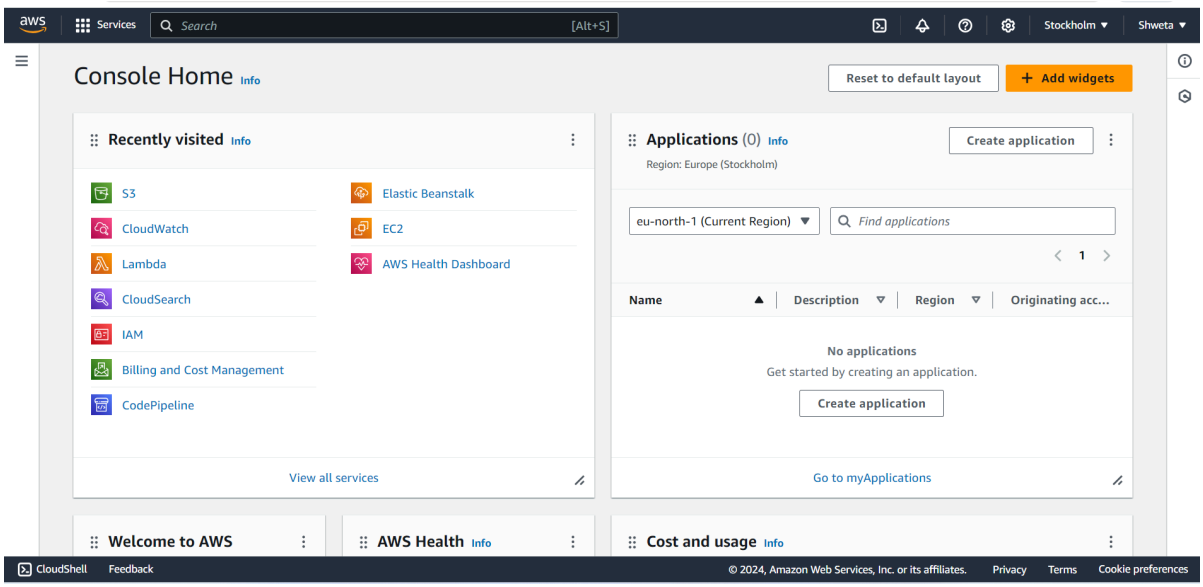
Applications:

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:

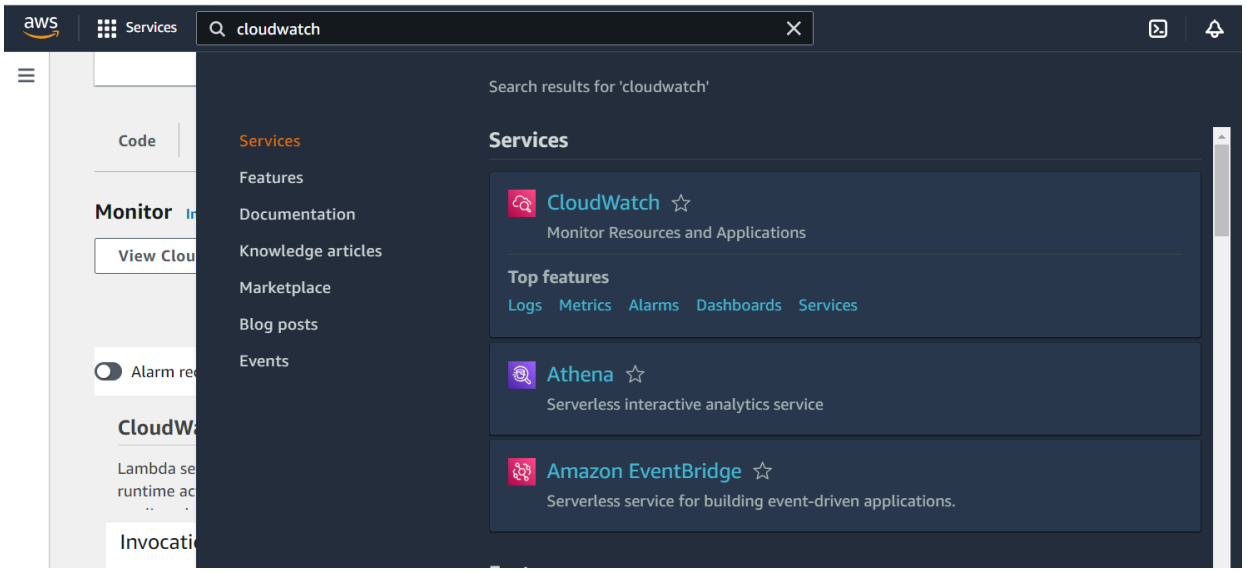
Open aws cloud service.Log into your AWS Console.

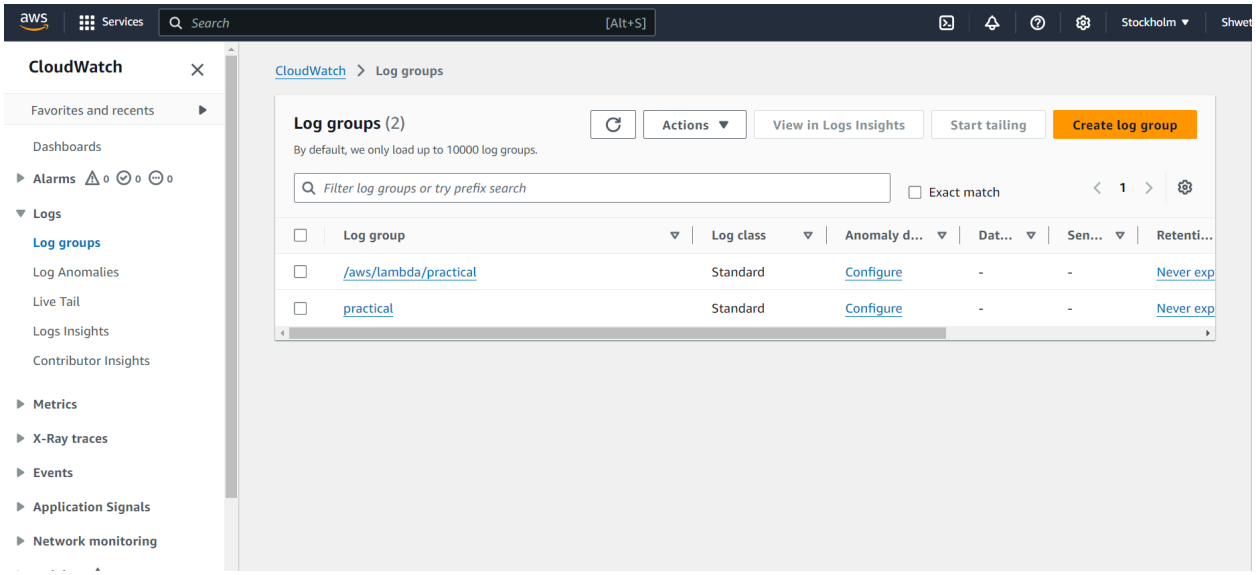


Create a CloudWatch Log Group

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.

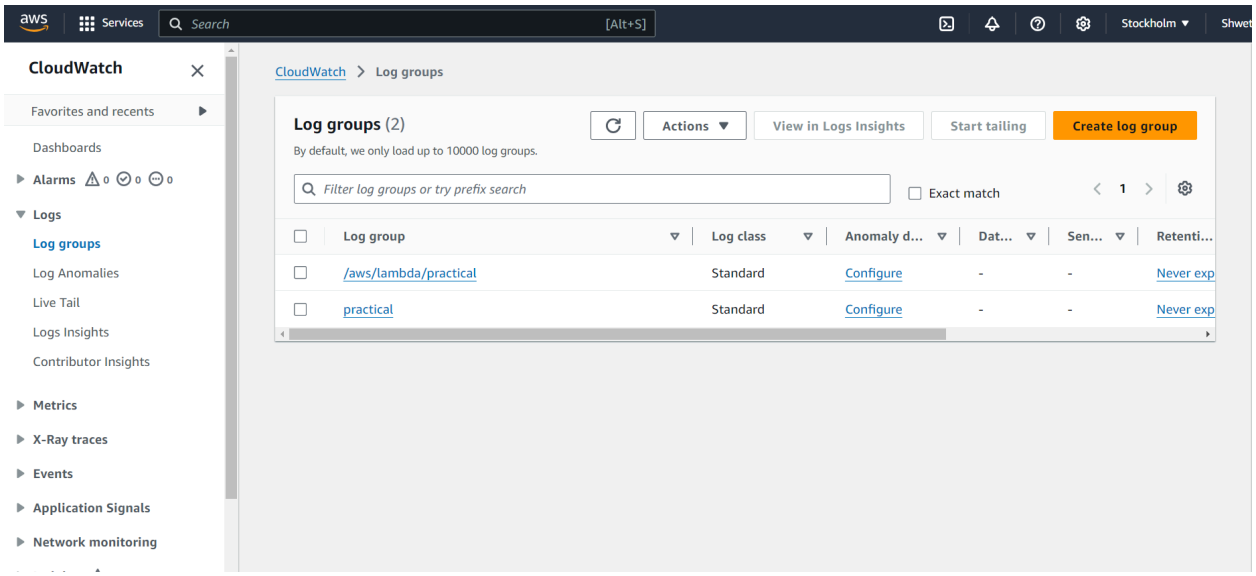
Click the Create log group button.

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

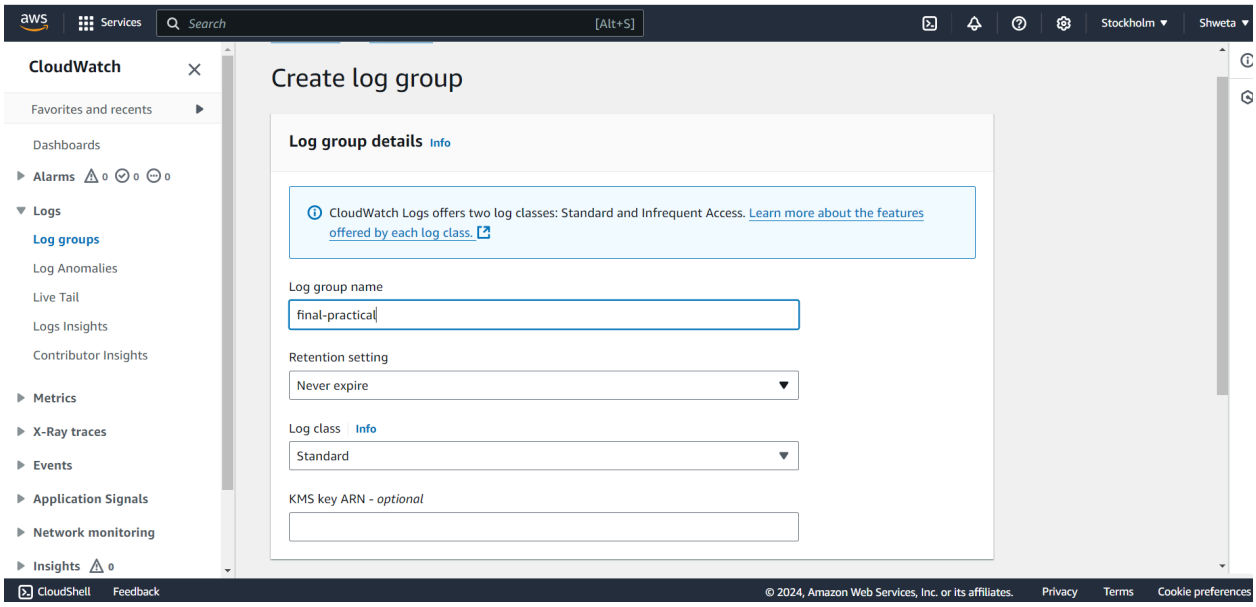
Finish Creating the Log Group

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

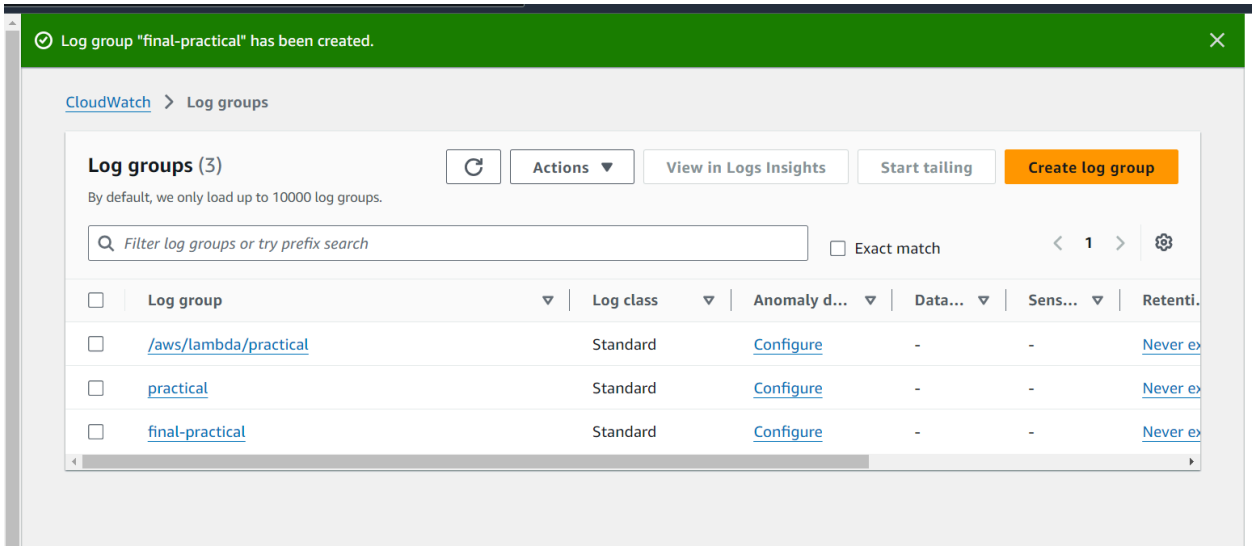
Create a CloudWatch Log Group



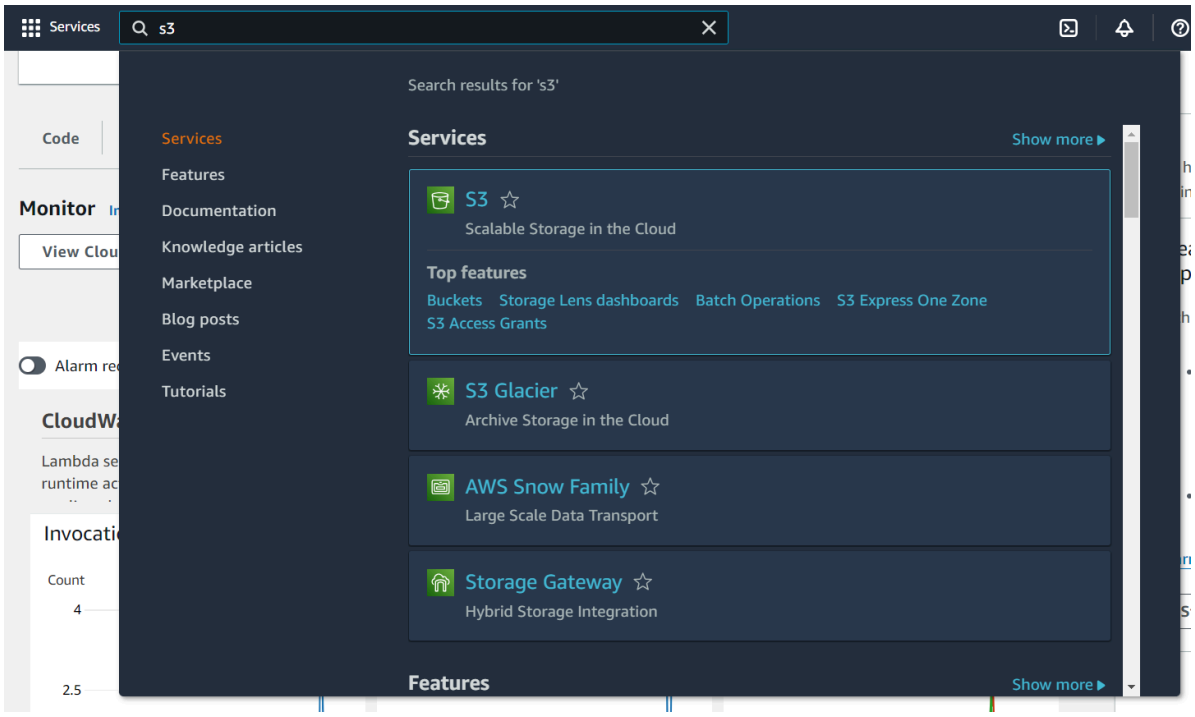
Initialize your log group by giving it a name.



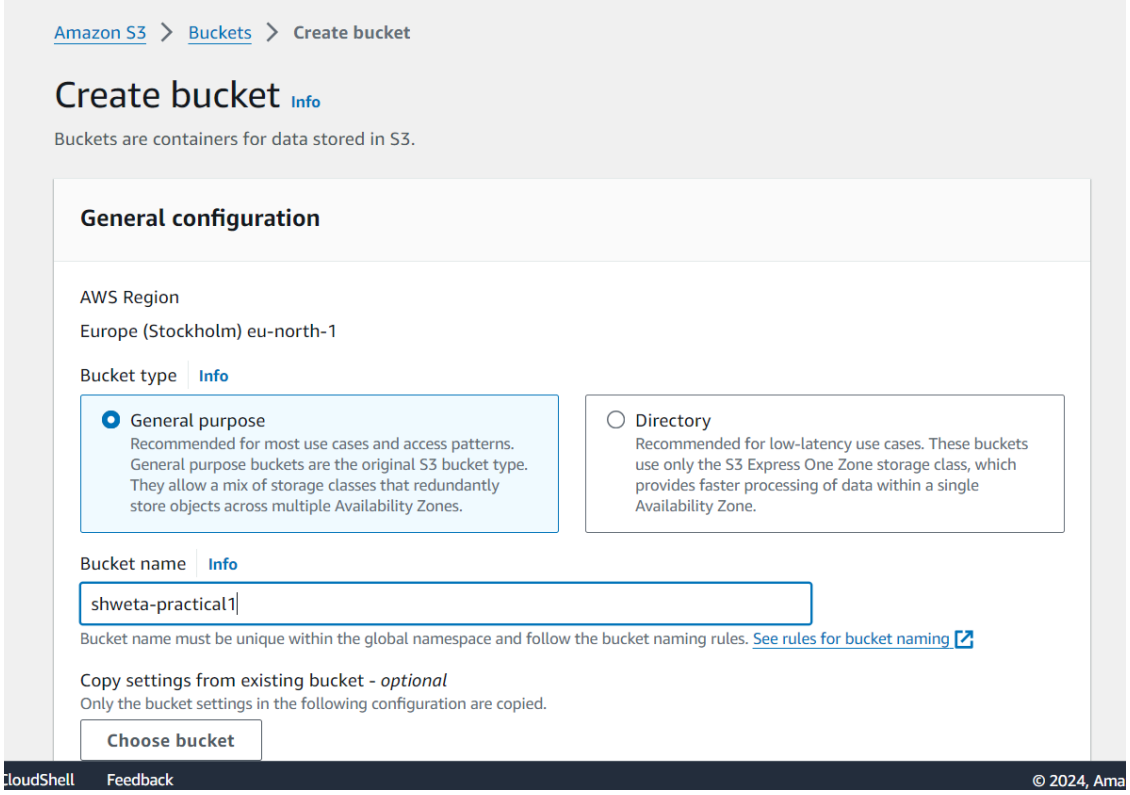
A log group is created.



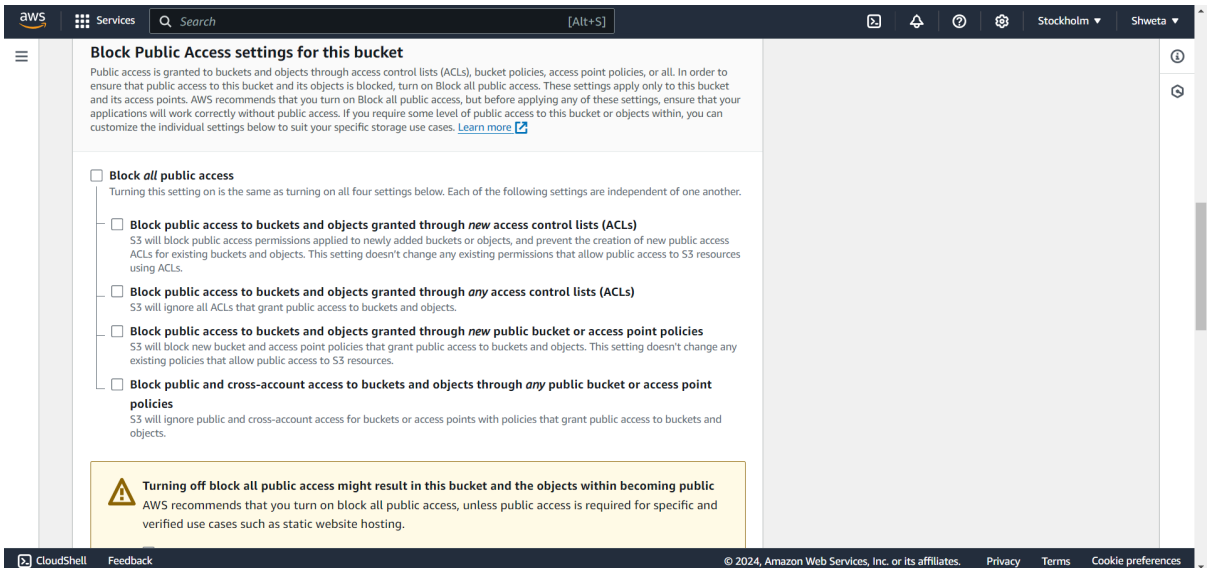
Search for s3 bucket.

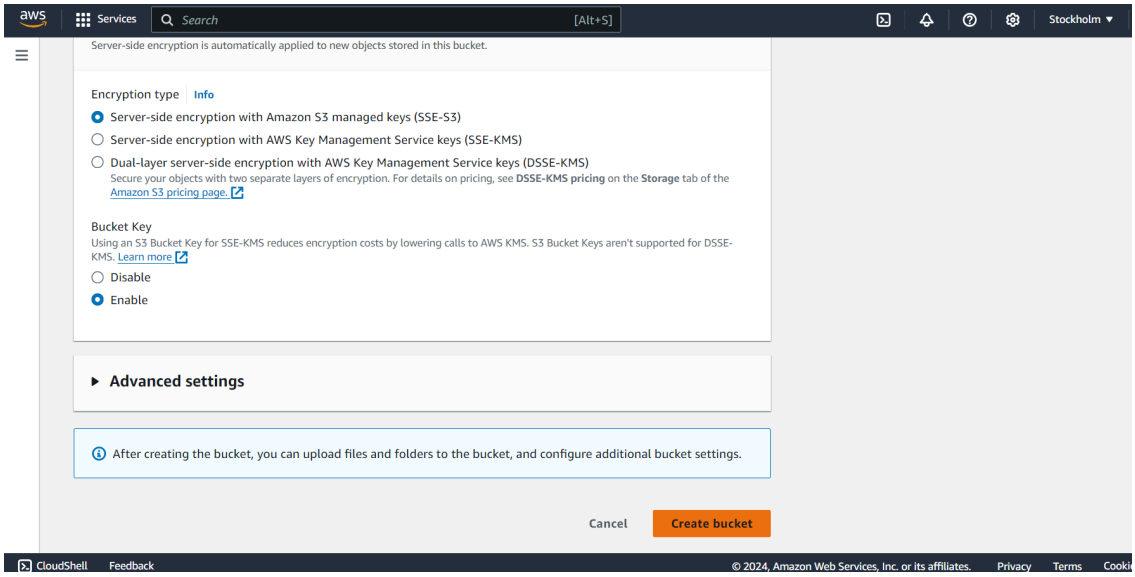


Create a new s3 bucket.

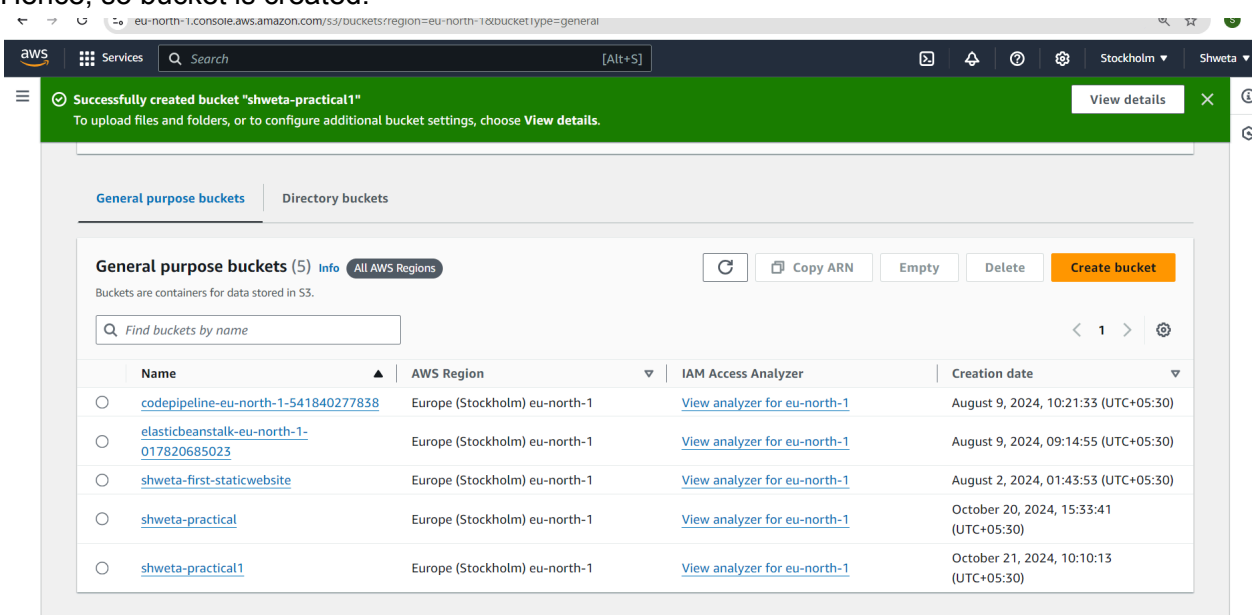


Give a name to your bucket. Then **untick the block all public access**, this will ensure that the bucket is public and has proper permissions for our Lambda function to write to it. Keeping the rest of the options default, we click on **Create**

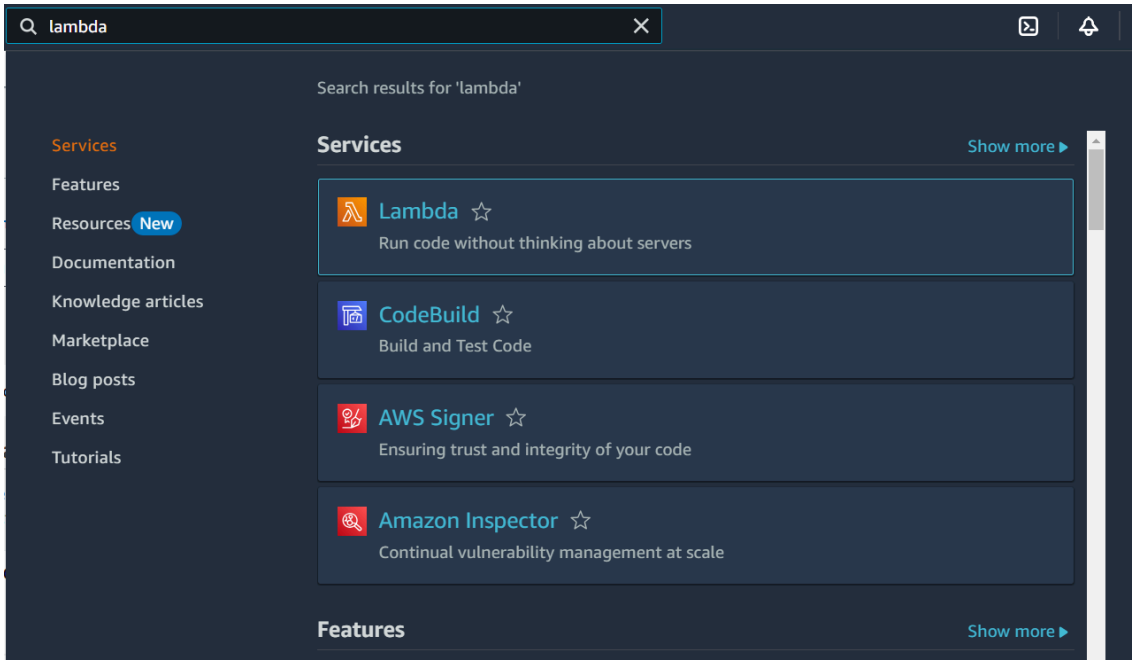




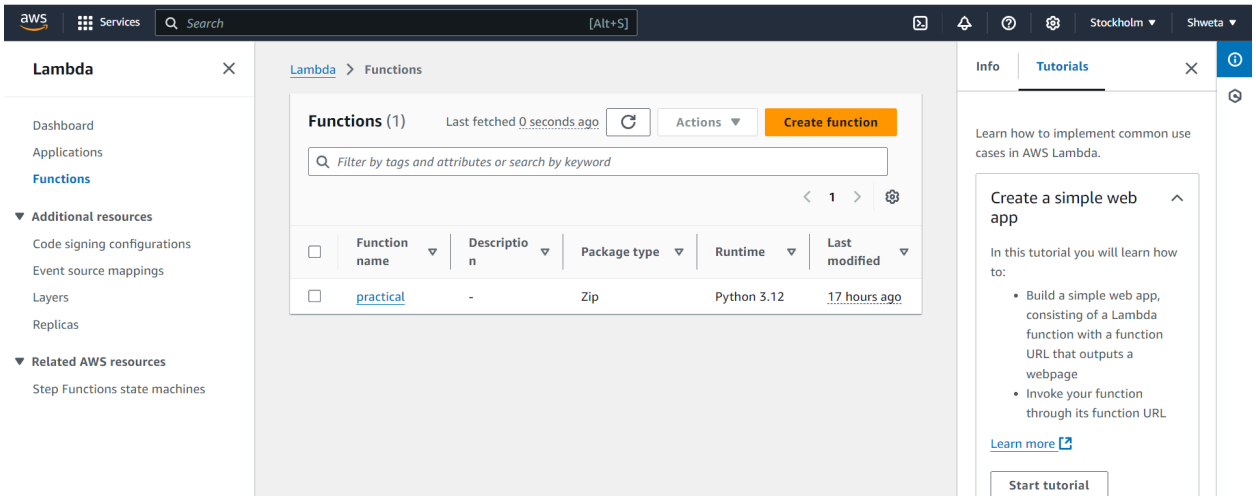
Hence, s3 bucket is created.



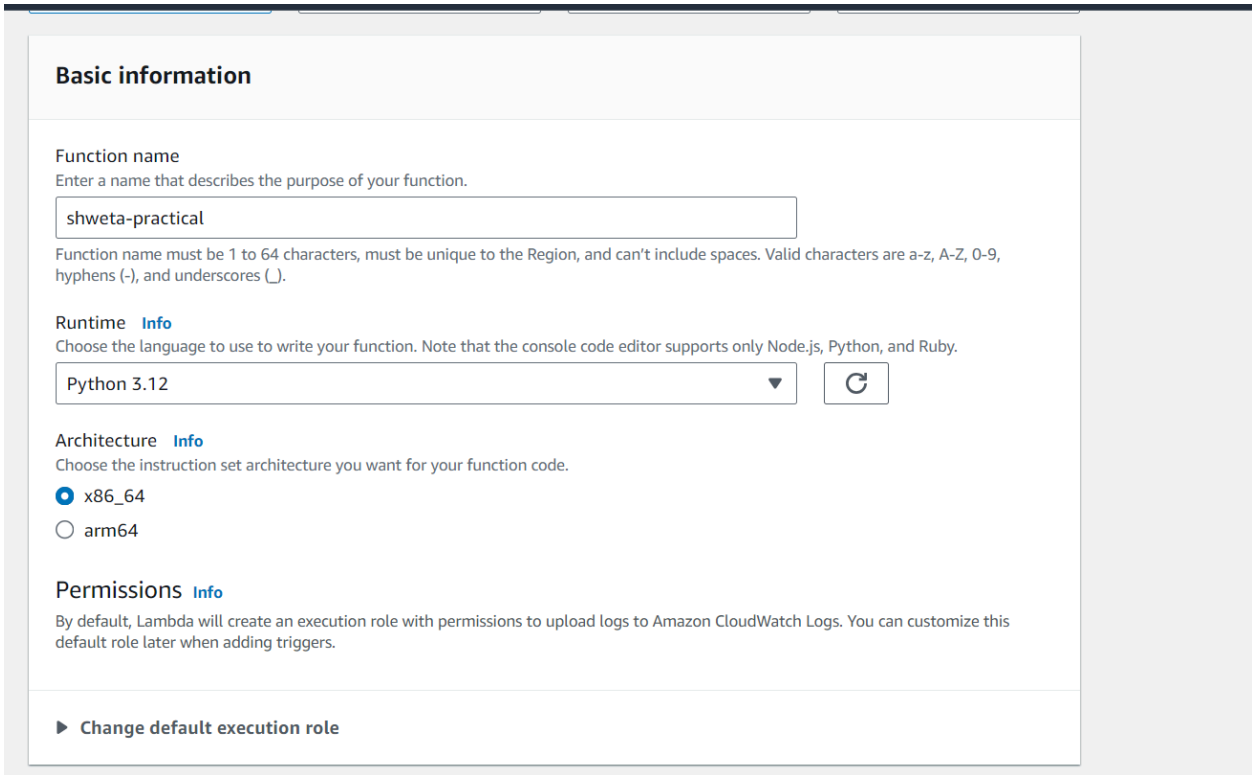
Now we need to create Lambda function and hence we search for Lambda.



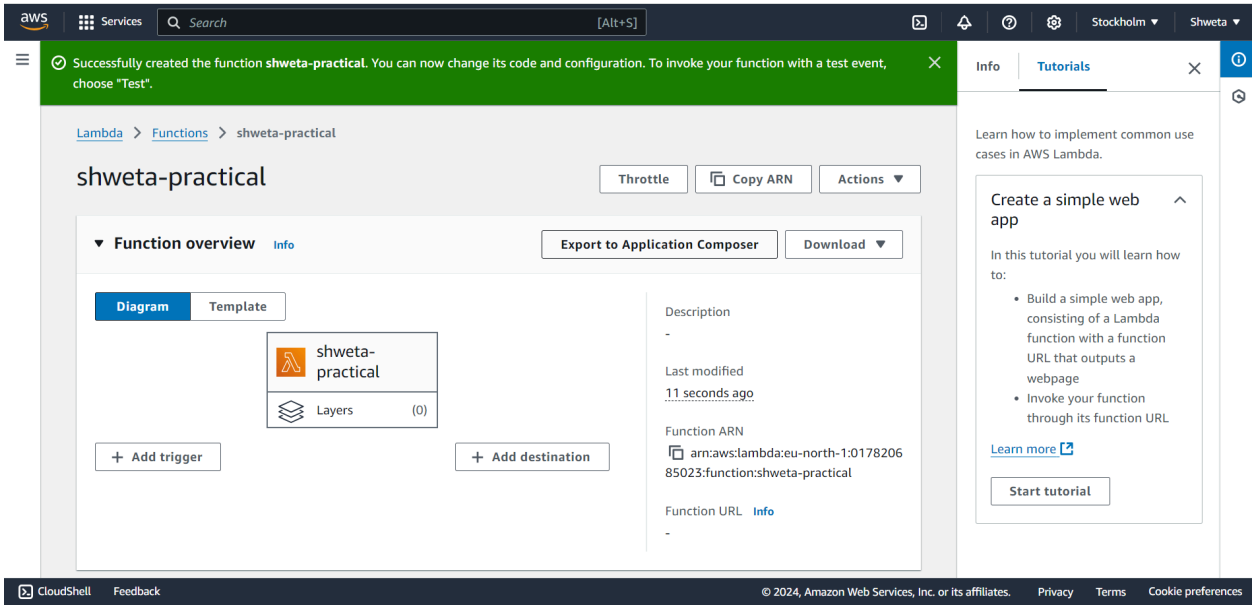
Create a new Lambda Function.



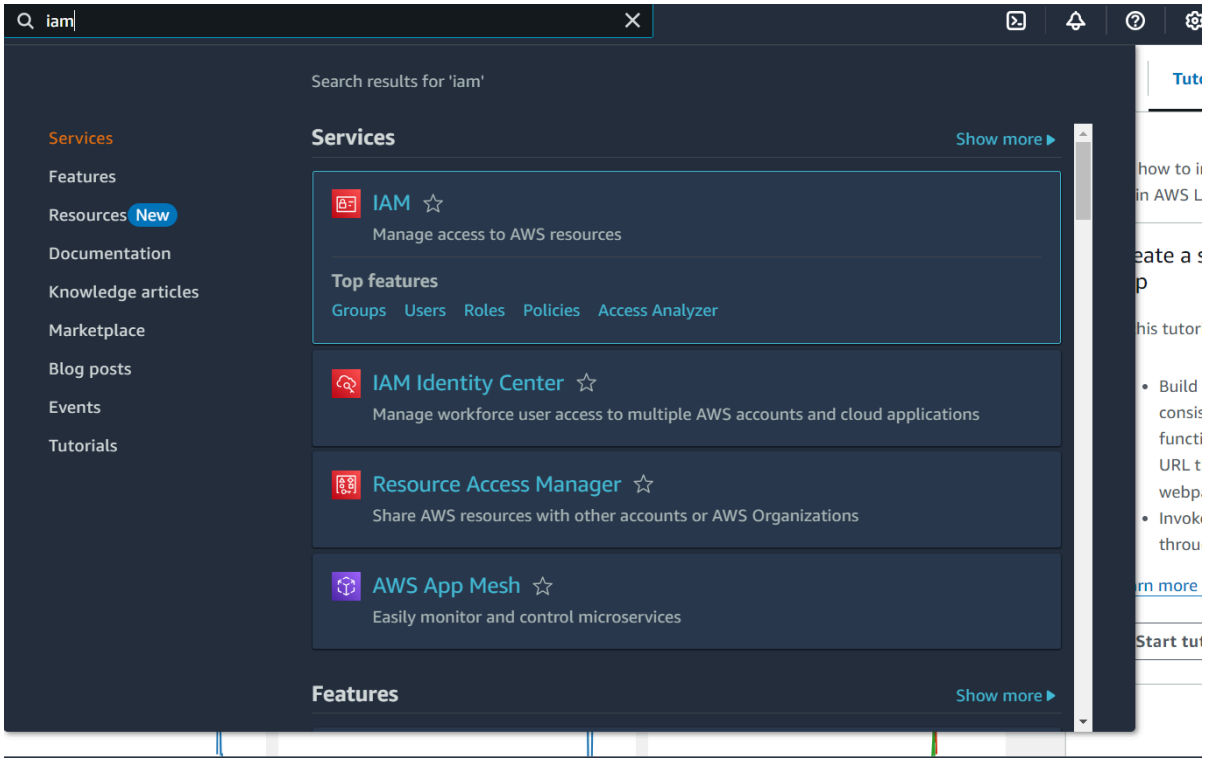
We select the option '**Author from Scratch**'. Click **Create function**, choose **Author from scratch**, and give it a name like shweta-practical. Set **Runtime** to **Python 3.12**. We keep the rest of the settings default and create the function

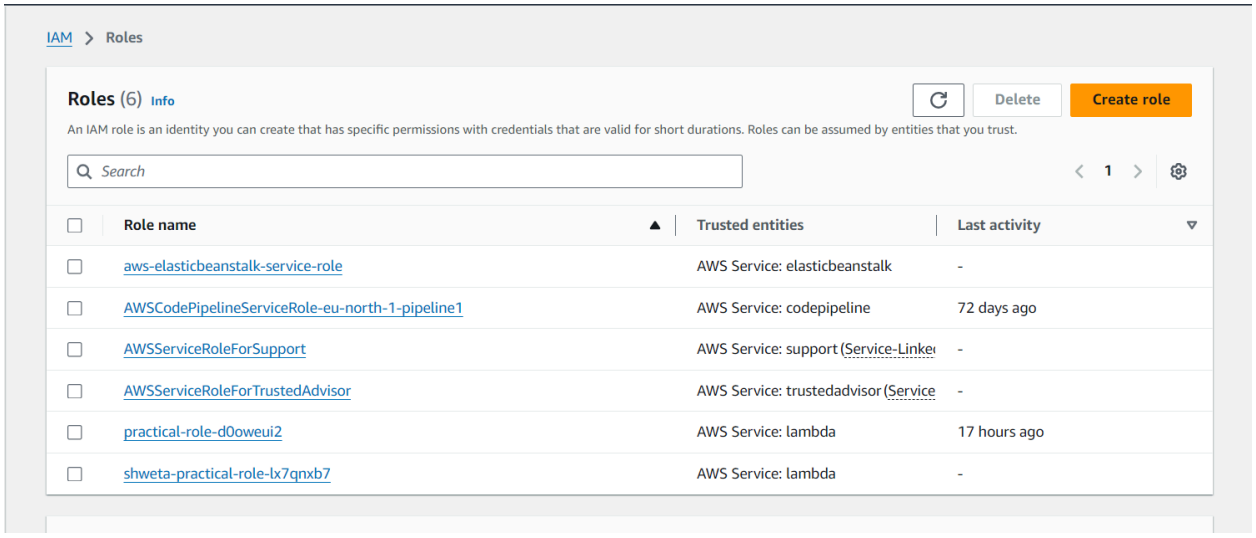


Lambda Function is now created.

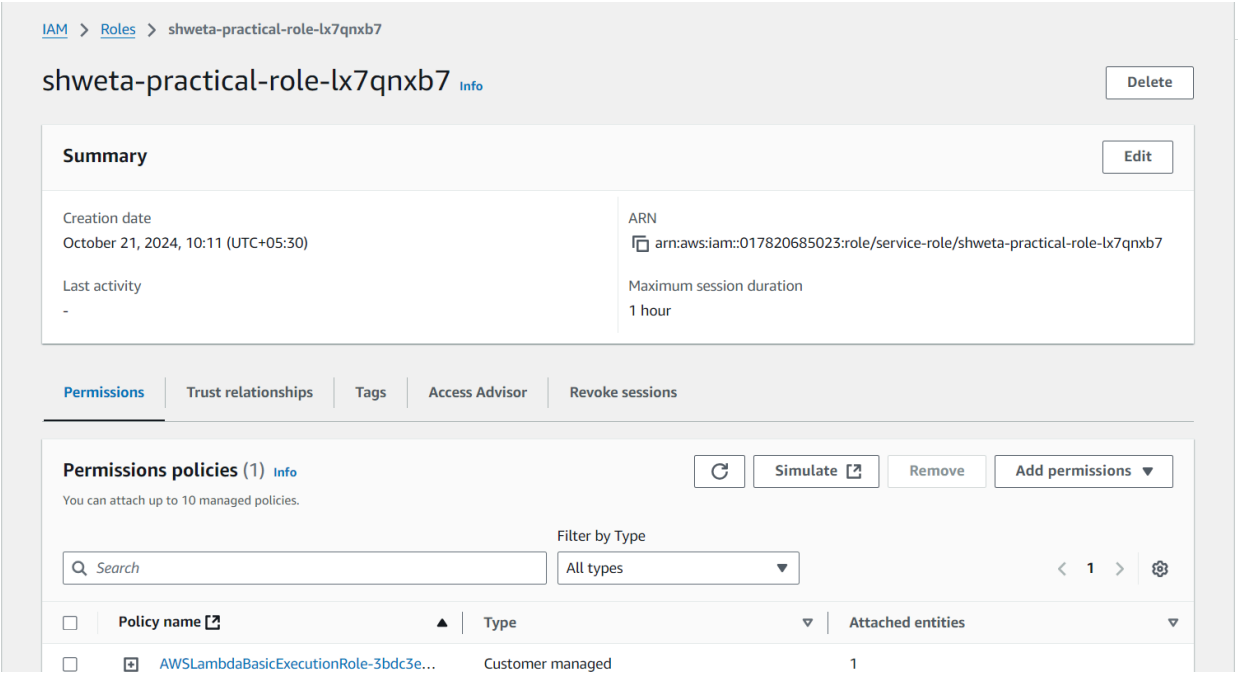


Now, search for IAM roles in the console and then go to **Roles** from the options on the left sidebar.

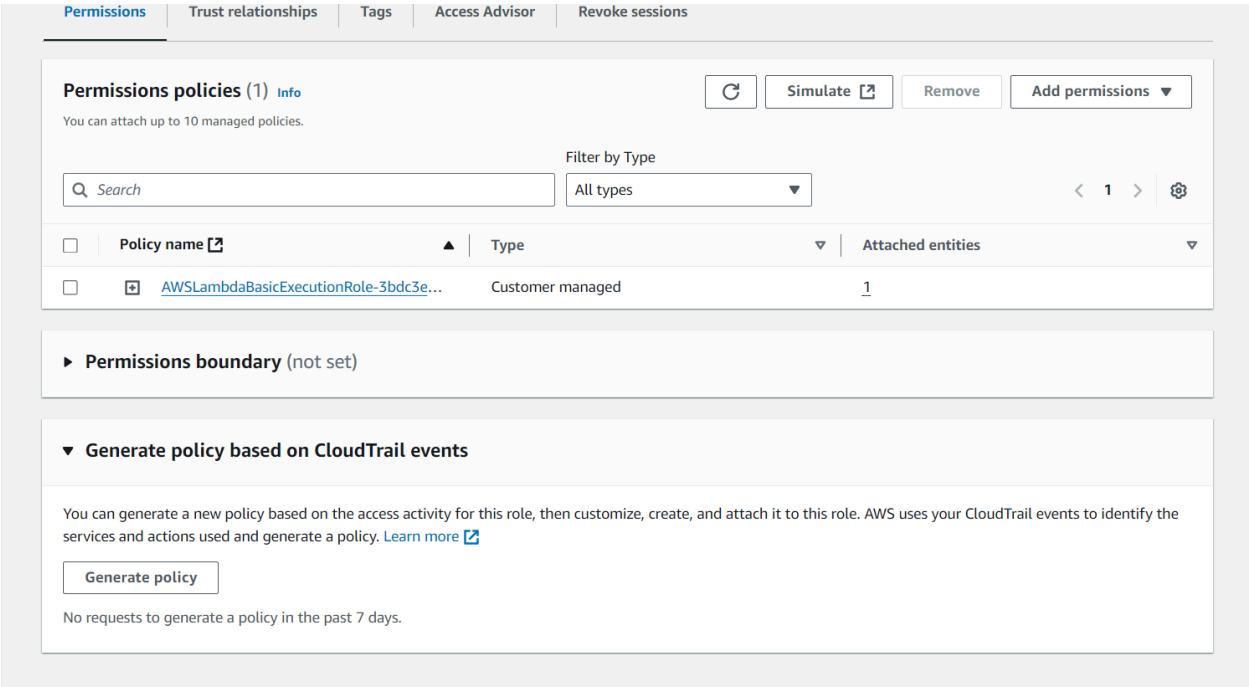




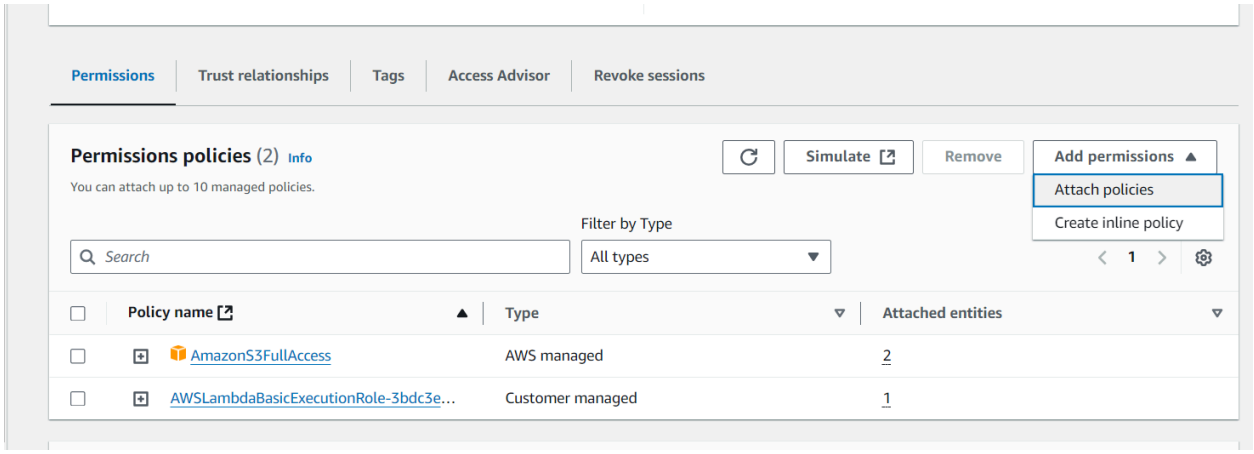
A new role will be created automatically , you need not create a new role. you can see the role associated with your lambda function. It has the same name as your lambda function with role. Click on the name.



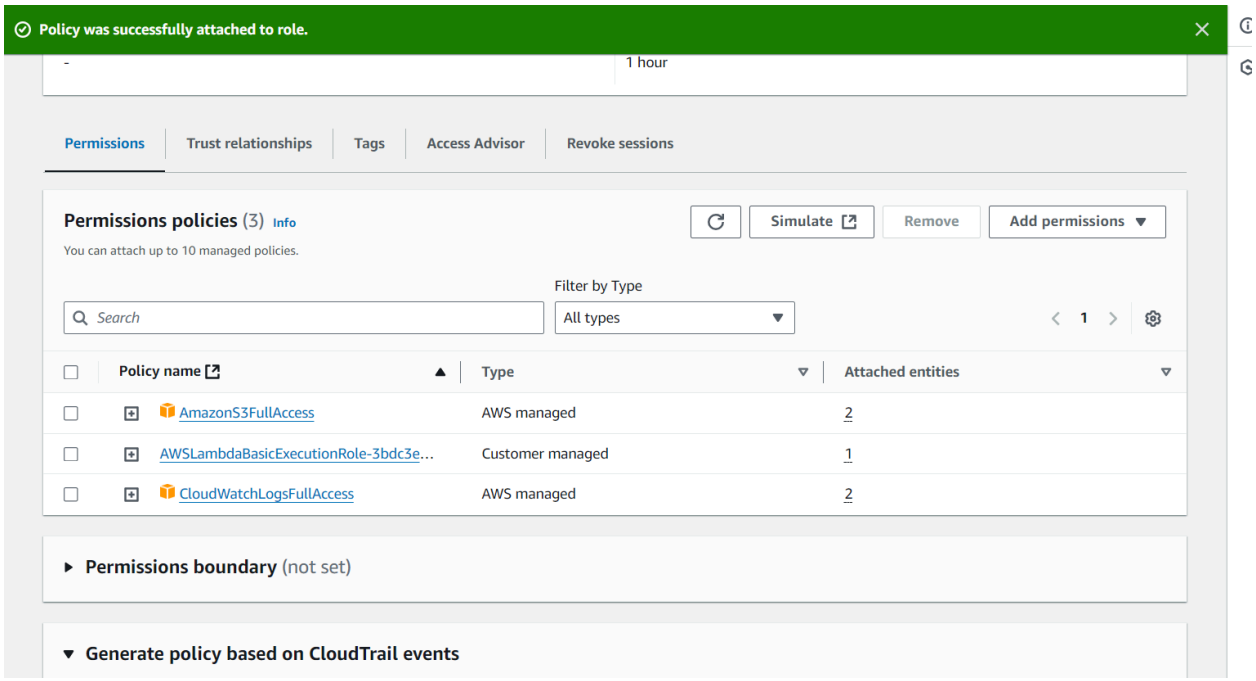
Click onto the role created in that go to permissions



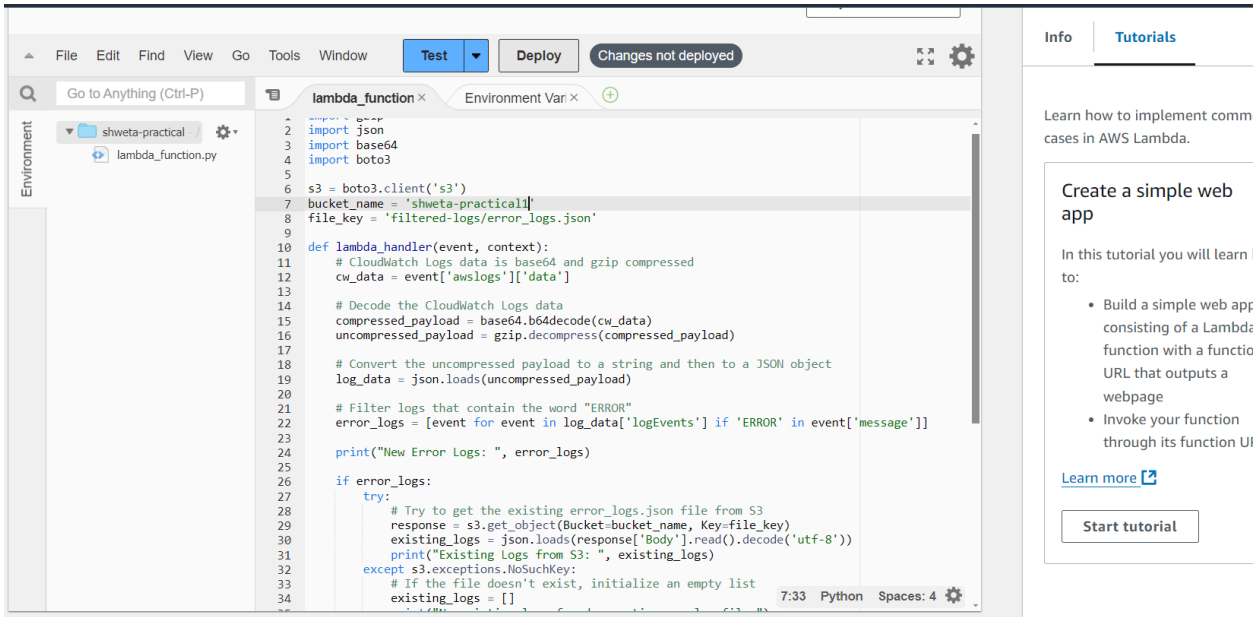
In Add permissions attach policies



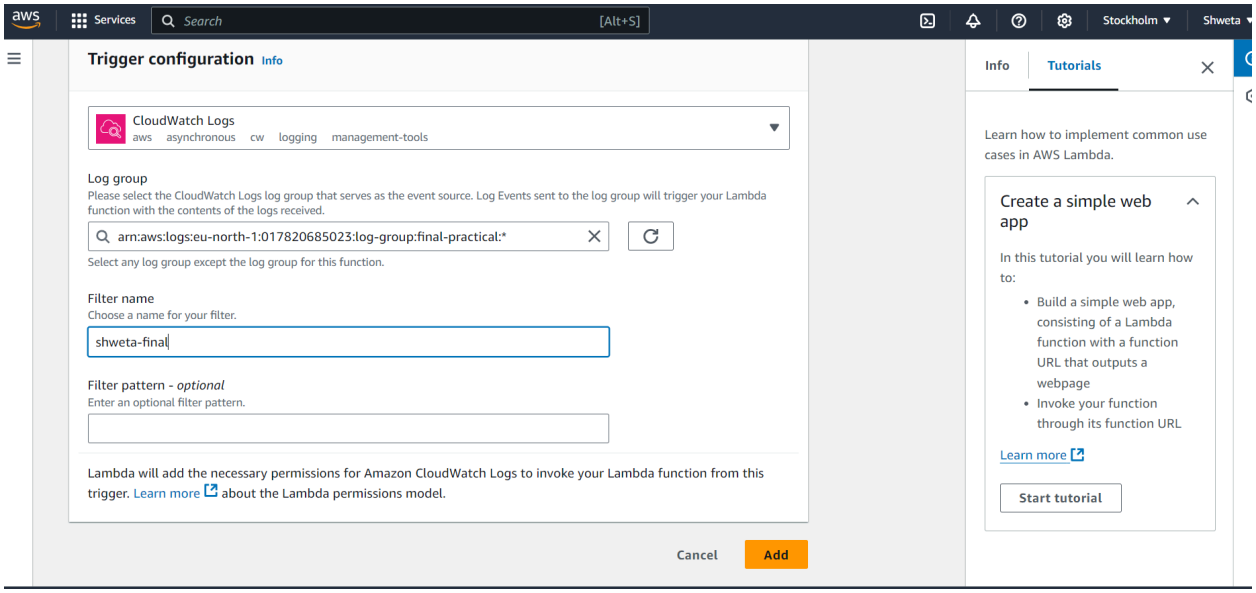
Add the following permissions:
– Amazon S3 full access (to allow Lambda to write to S3).
–CloudWatchFullAccess (to allow Lambda to read logs from CloudWatch).

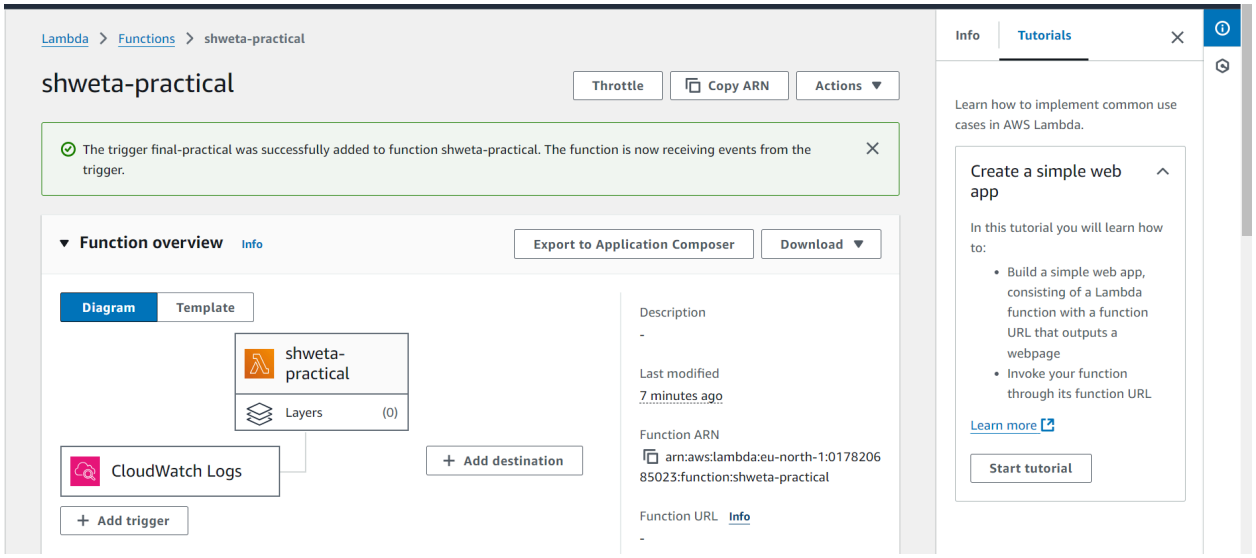


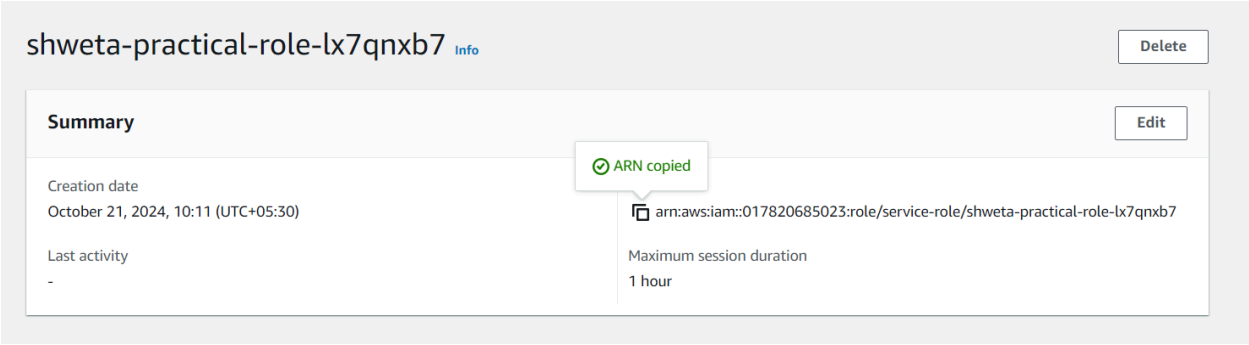
Navigate back to the lambda function, in the code section we add the following code. This Lambda function will filter logs containing the keyword "ERROR" and store them in an S3 bucket.



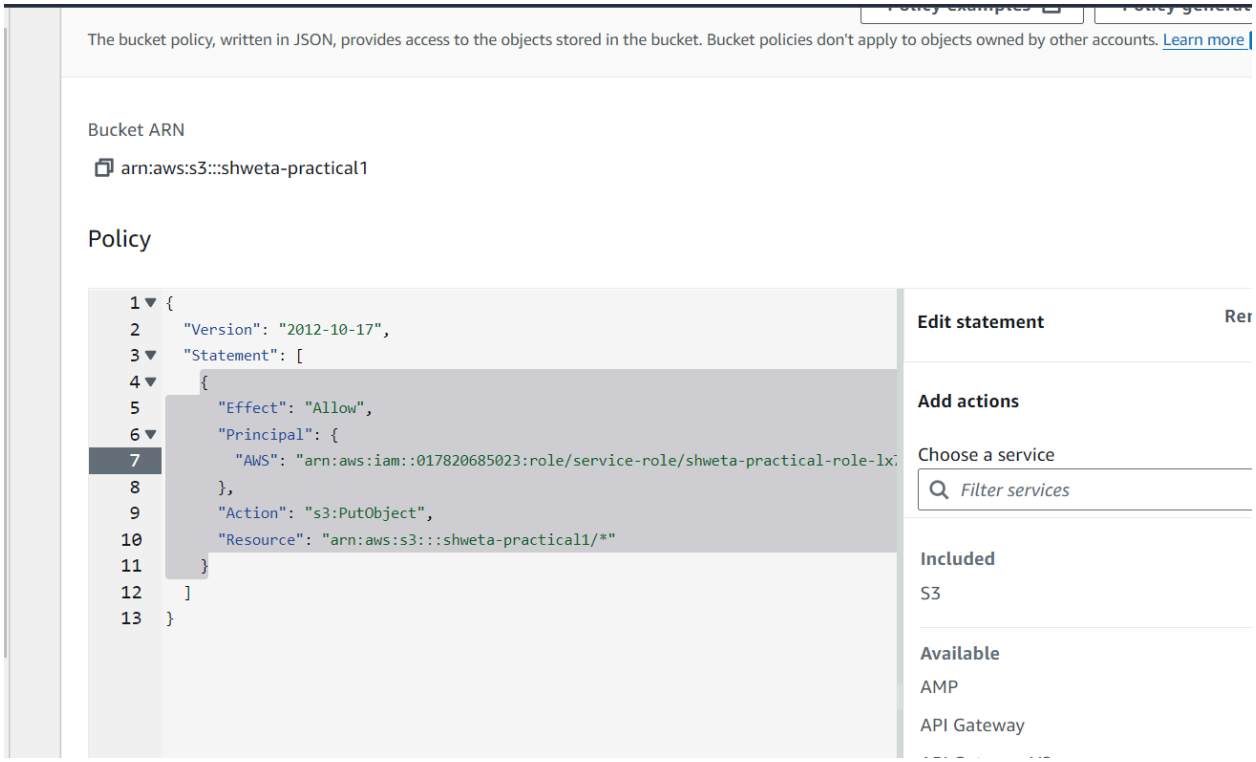
In Lambda function, click on **Add Trigger**.Select **CloudWatch Logs** from the dropdown. Choose the log group you created earlier (shweta-final). Choose a filter name and click on **Add.**

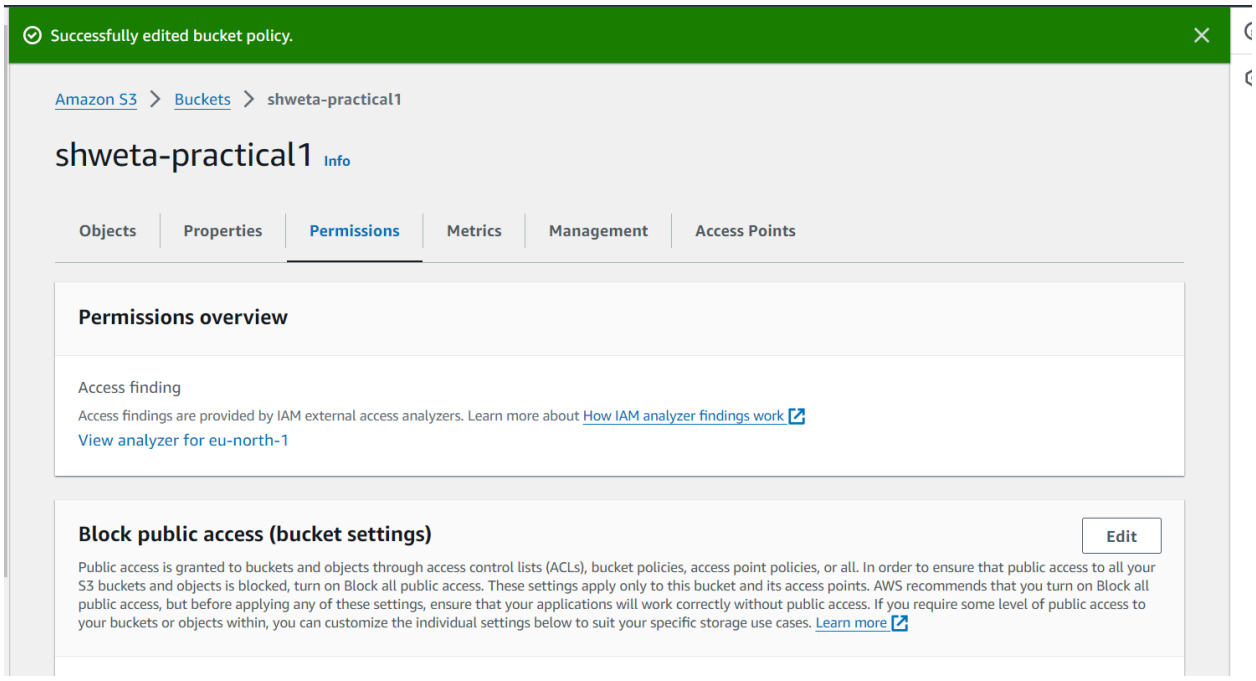




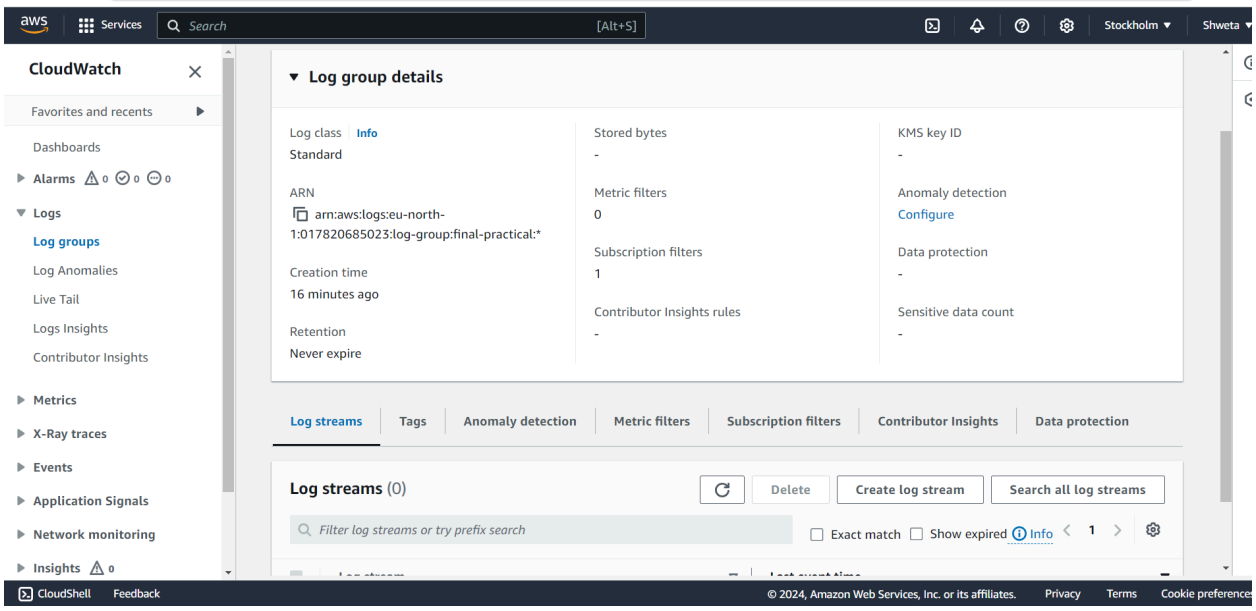


Go to the **S3 Console** and select the bucket you created. Navigate to the **Permissions** tab. Scroll down to **Bucket Policy** and click **Edit**.In the bucket policy, make sure you add the correct **bucket name** in the resource and the correct **role ARN** for your lambda role. This policy gives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

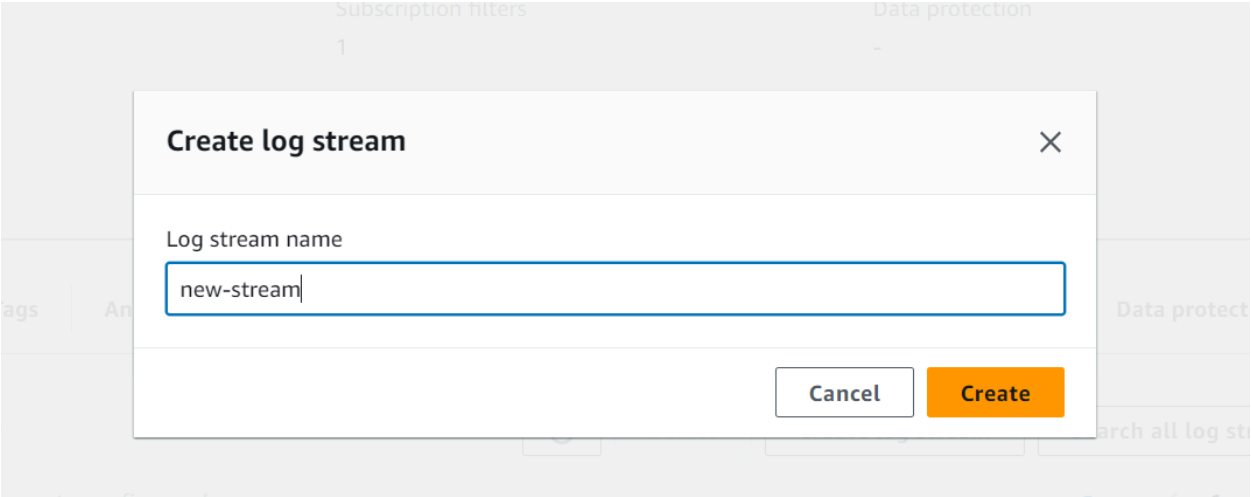




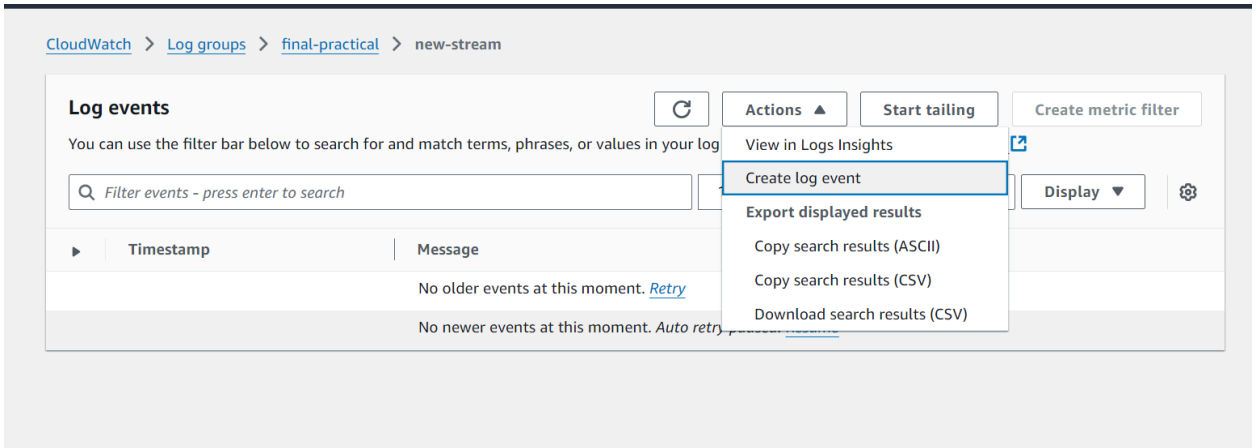
On the CloudWatch dashboard, on the left sidebar, click on **Logs** → **Log Groups**. This will display all the Log Groups associated with your AWS account. Select the one associated with your lambda function.



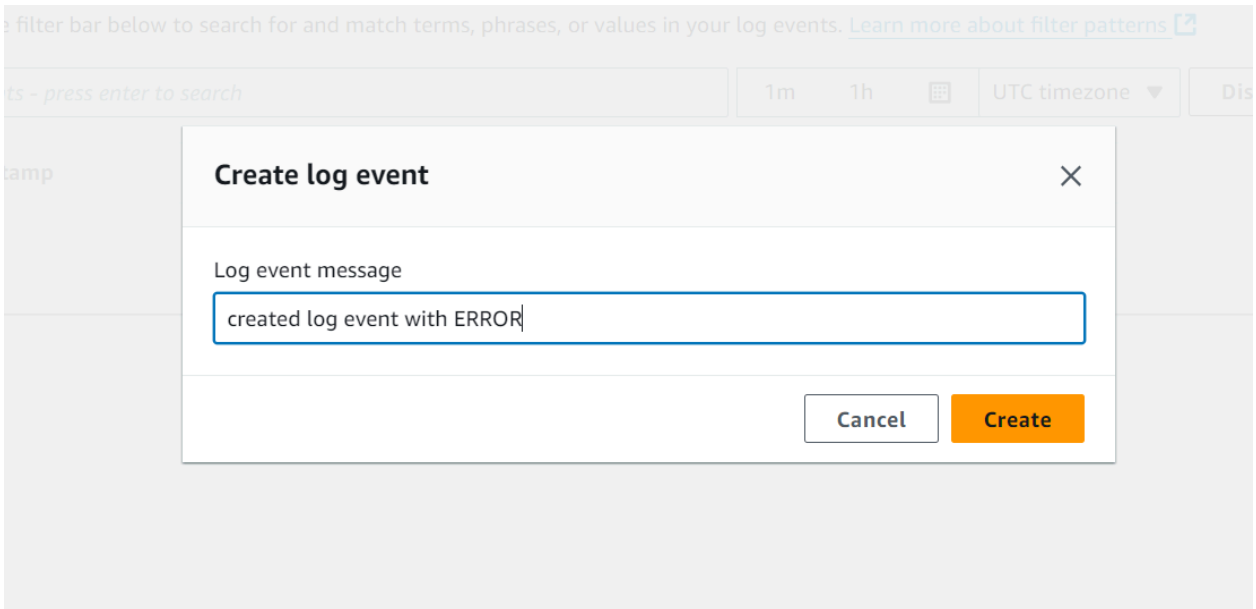
Inside the Log Group, you'll see **Log Streams**. A log stream is a sequence of log events for a specific resource that writes to CloudWatch. Click the **Create log stream** button, name the stream (e.g., new-stream), and proceed.



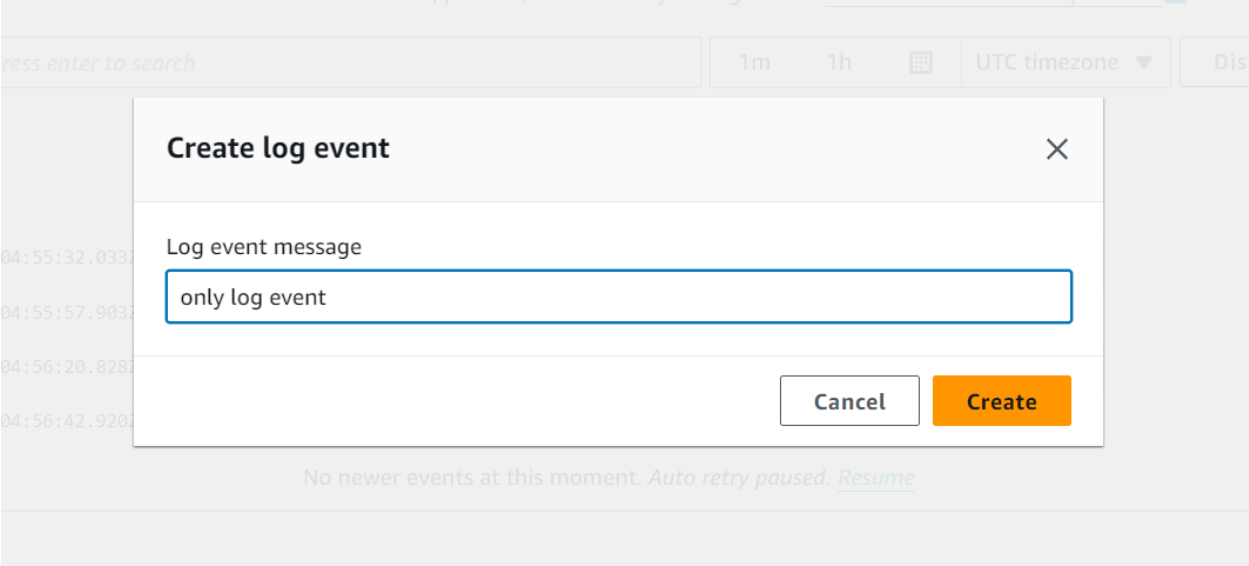
Click on your **Log Stream** (e.g., new-stream). You will now see an option to add log events manually. Click **Actions** → **Create log event**.

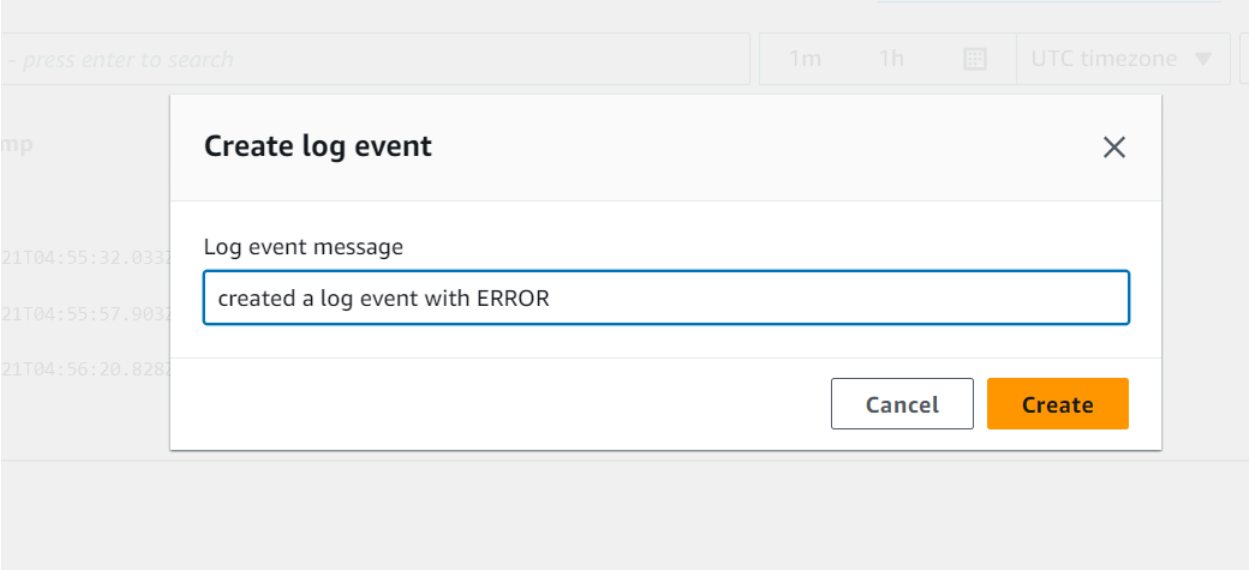


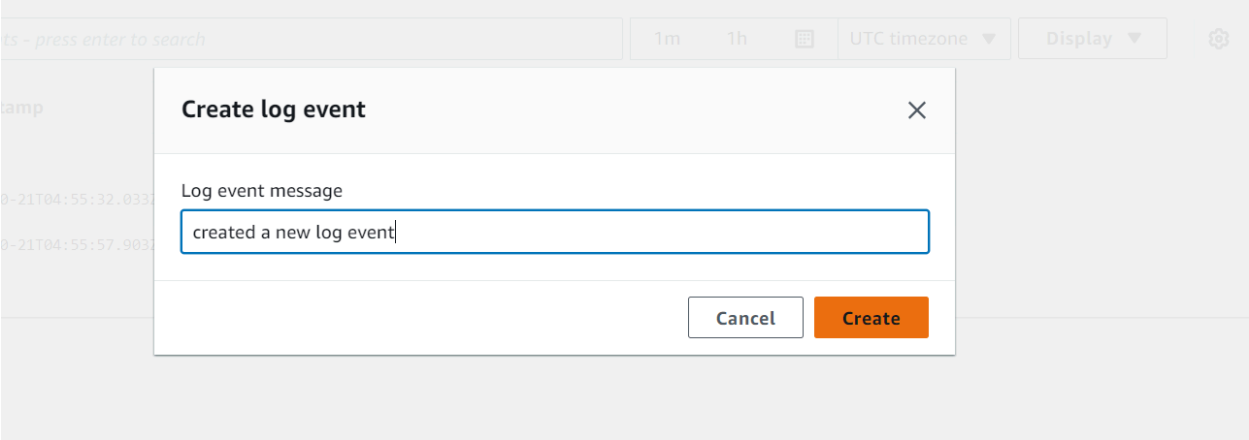
Enter the log message you want to trigger your Lambda function. Since we're filtering for the keyword ERROR, enter something like:

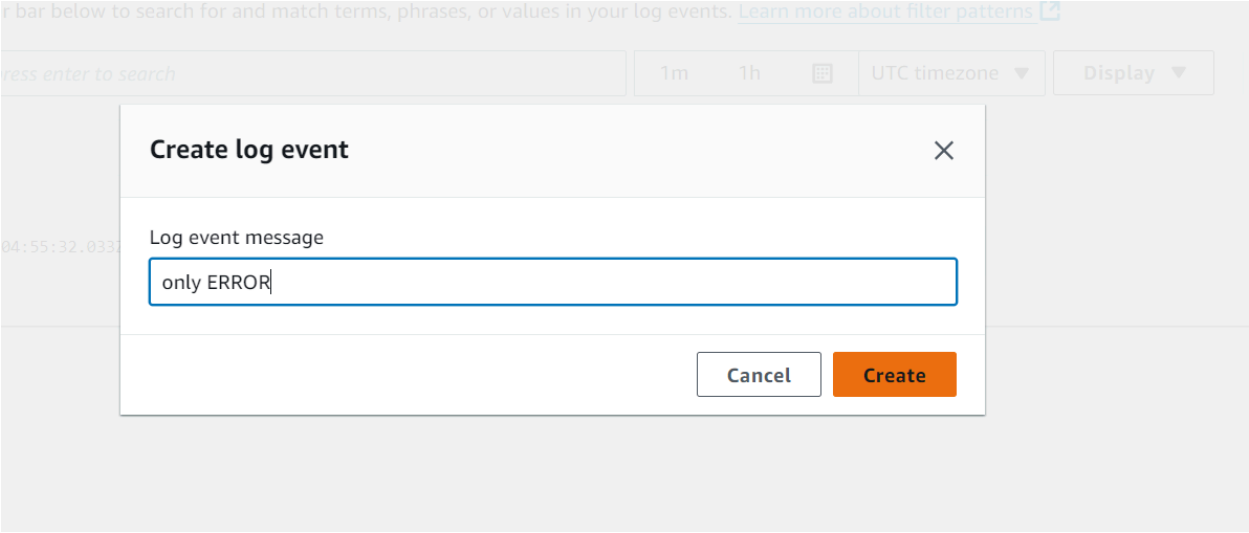


And some without the ERROR keyword:

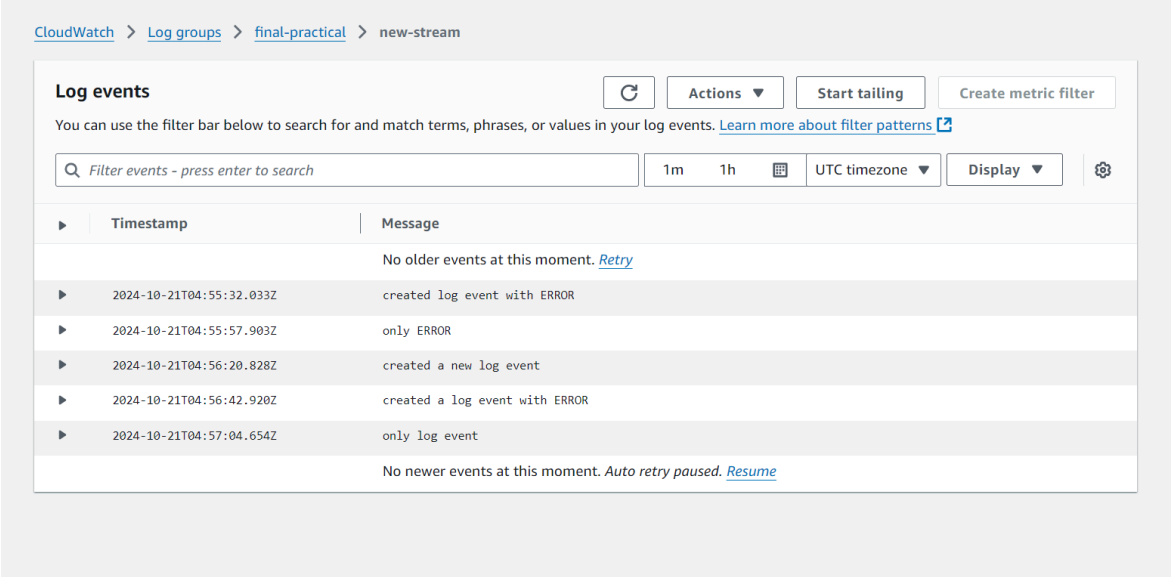

press enter to search
1m
1h
UTC timezone
Create log event
Log event message
only log event
Cancel
Create
No newer events at this moment. Auto retry paused. Resume


press enter to search
1m
1h
UTC timezone
Create log event
Log event message
created a log event with ERROR
Cancel
Create

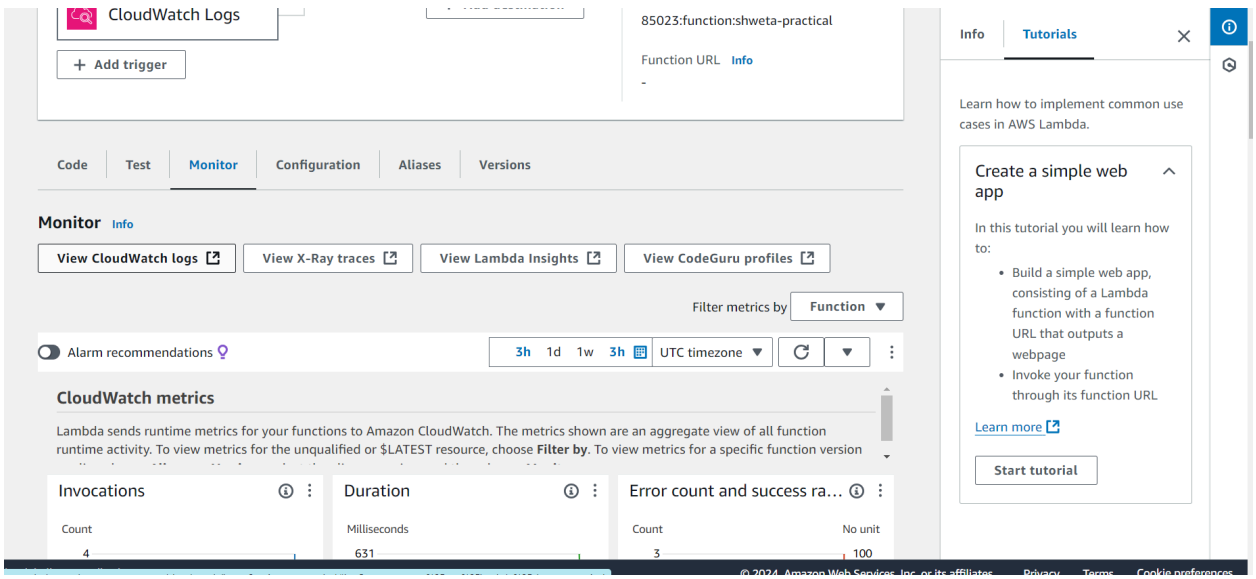




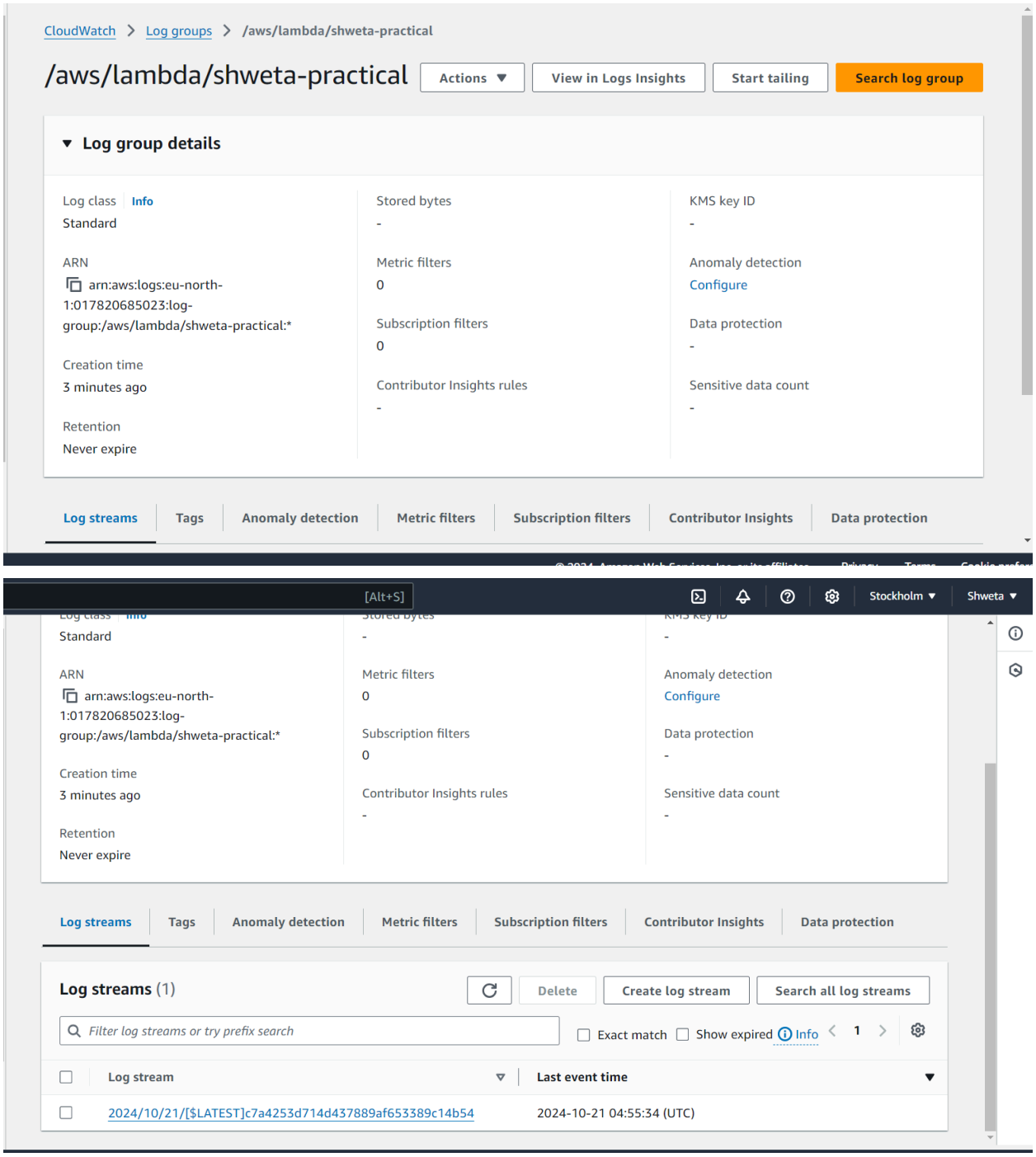
Create more log events like so with and without the keyword ERROR.



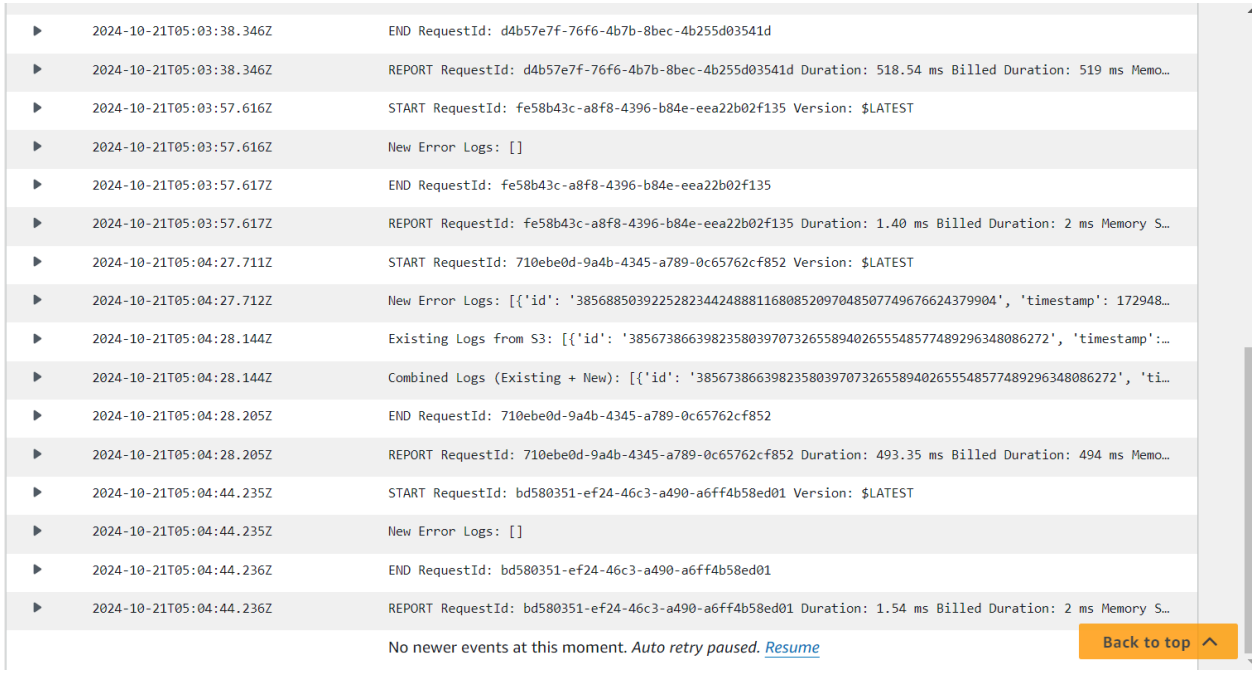
Find your Lambda function (the one you set up to filter logs and write to S3), and click on it. In the Lambda function dashboard, click on the **Monitoring** tab. In the Monitoring tab, click on **View logs in CloudWatch**.

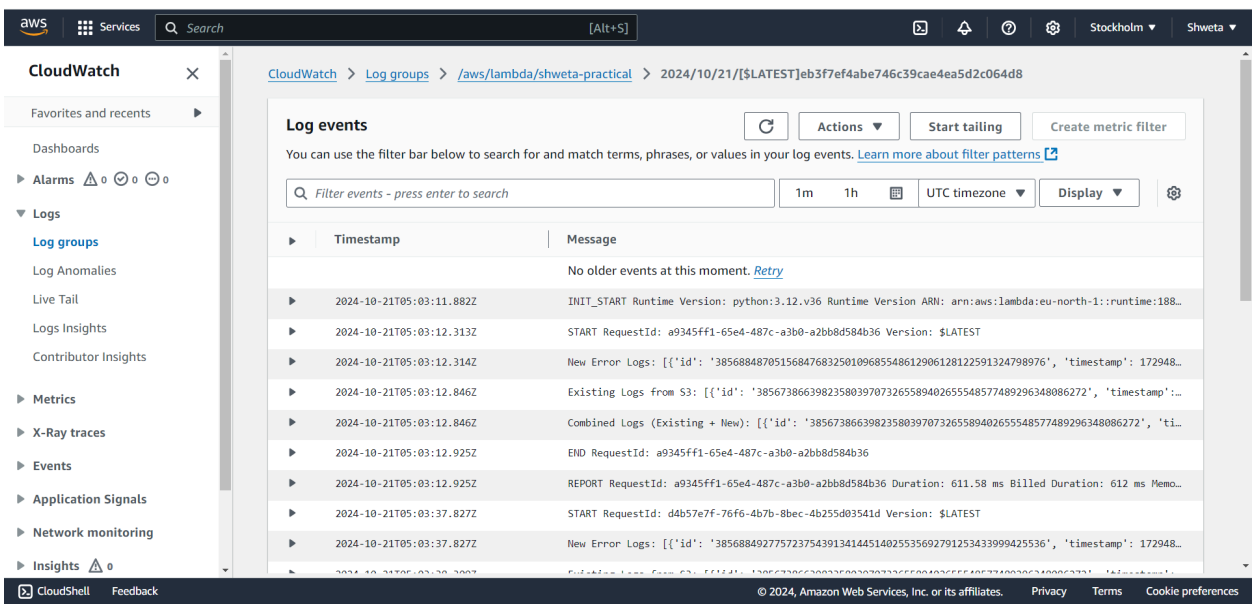


Then it will lead you to a generated log group for this function. Select the latest log stream. This will open CloudWatch and show you logs generated by your Lambda function.

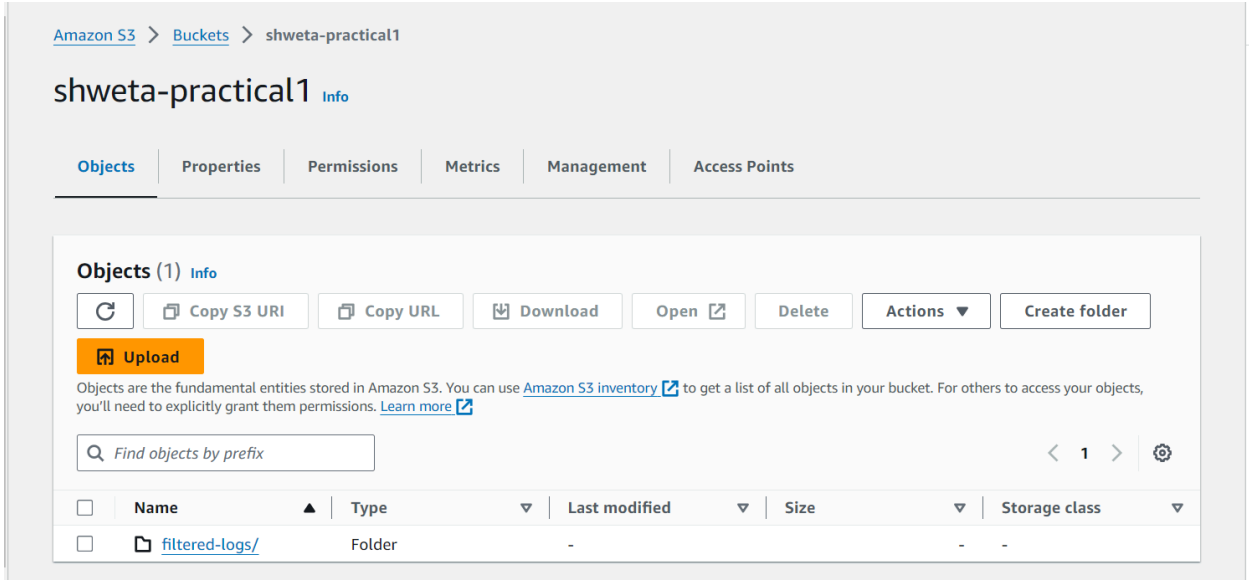


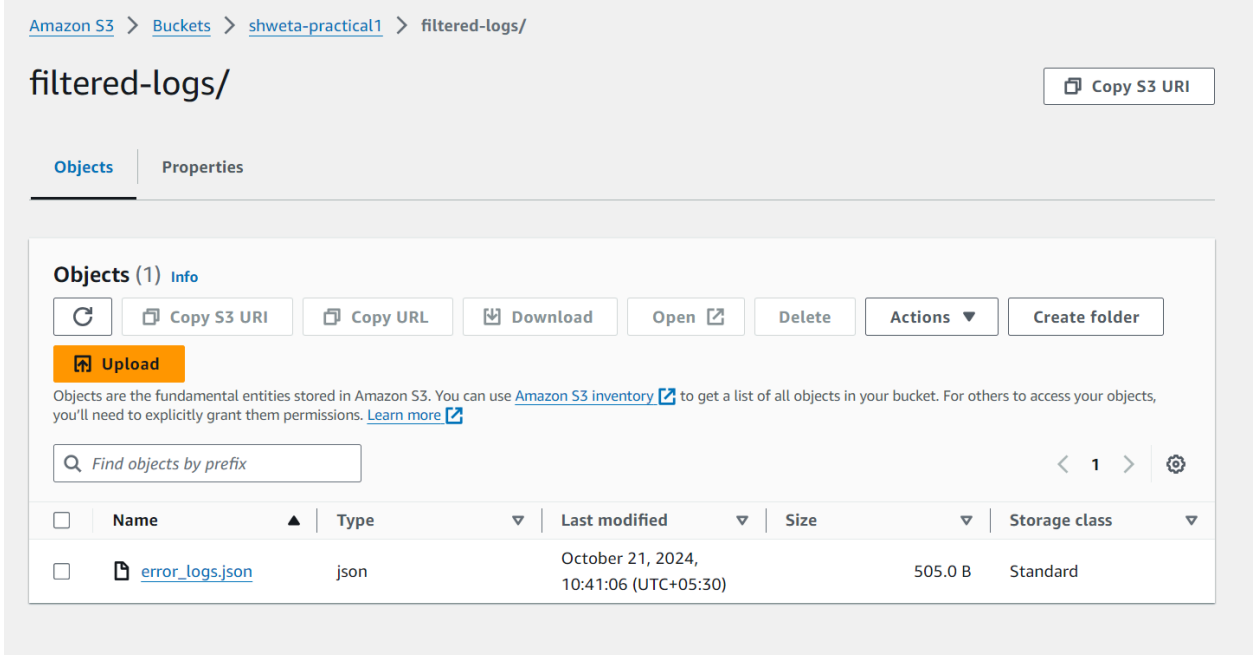
Look for recent log entries to see if the Lambda function was triggered. Check for any errors or information logs indicating that the function processed the log event and uploaded data to S3.



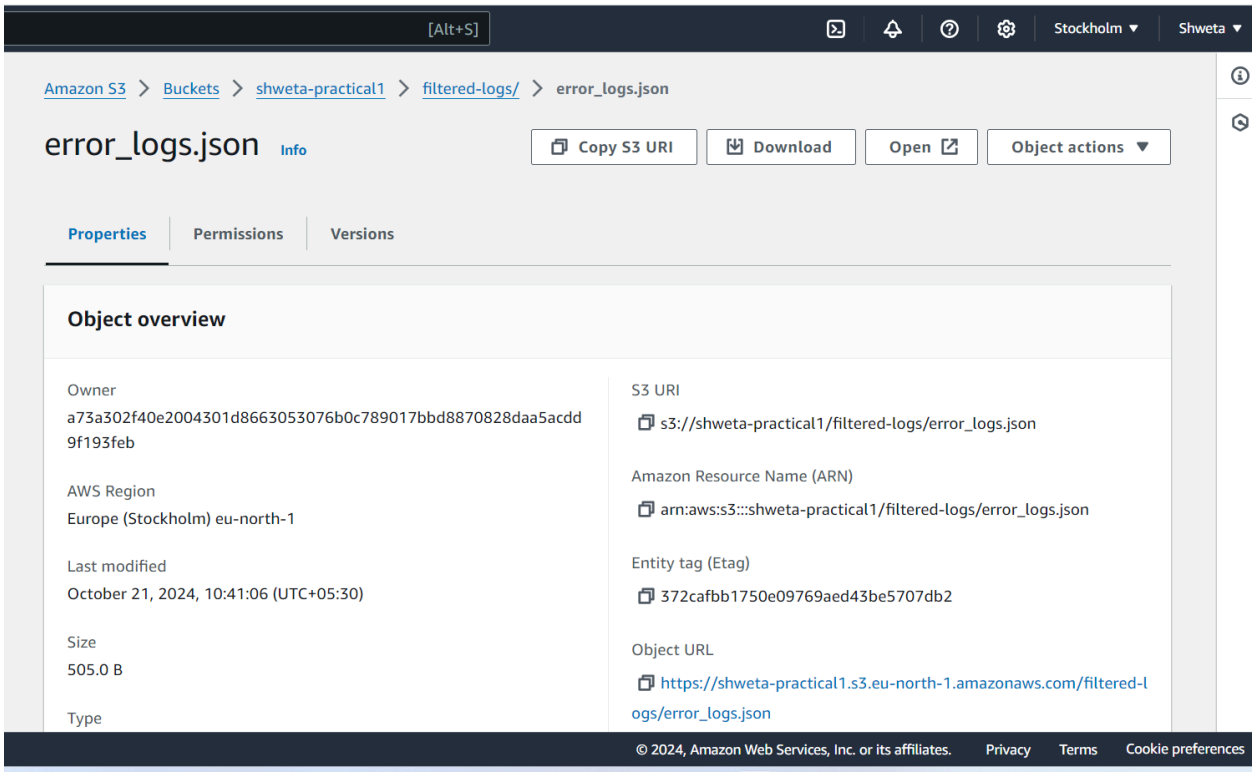


In the list of buckets, find and click on the bucket you configured in your Lambda function. Inside the S3 bucket, look for newly uploaded files. They should contain logs that match the filter pattern.

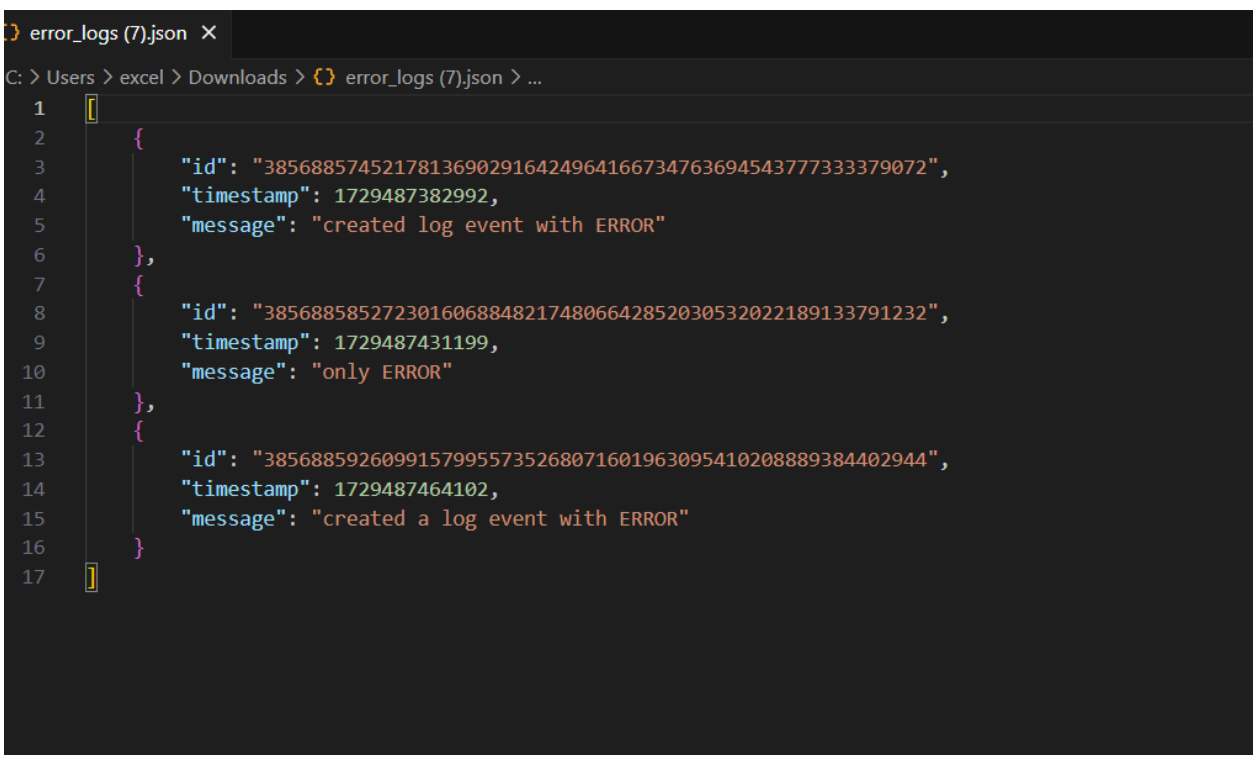




Like we configured in the code part. The logs are stored in the error_logs.json file. Download or Open the file.



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



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.