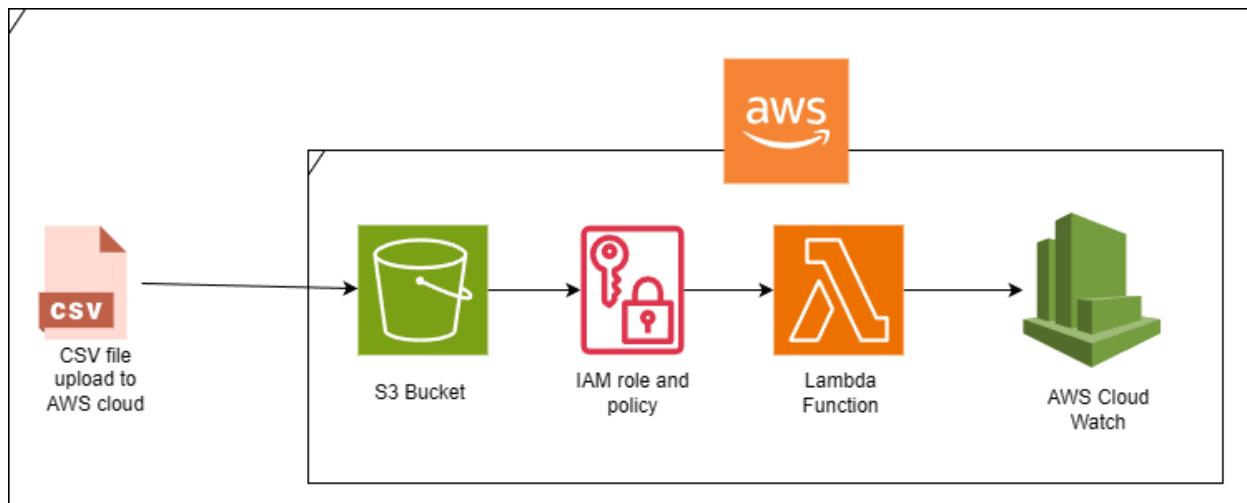




AWS S3 + Lambda + Cloud Watch Project



Step by Step how to Create that Project

- ❖ **Steps 1:-** Create a S3 bucket and upload CSV file in to the bucket,
- In the AWS S3 console, create a new bucket with a unique name, specify the AWS Region, adjust the "Block all public access" setting as required, and proceed to create the bucket.

The screenshot shows the AWS S3 console with a green success message: "Successfully created bucket 'service-costs-bucket'. To upload files and folders, or to configure additional bucket settings, choose View details." Below this, the "General purpose buckets (1)" section displays the newly created bucket "service-costs-bucket" with its details: Name (service-costs-bucket), AWS Region (Europe (Stockholm) eu-north-1), and Creation date (December 22, 2025, 19:48:11 (UTC+05:30)). On the right, there are "Account snapshot" and "External access summary" cards. The bottom of the screen shows the standard AWS navigation bar with links like CloudShell, Feedback, Console Mobile App, and various status indicators.

- Upload the CSV file into your S3 bucket

The screenshot shows the AWS S3 console interface. At the top, there's a green success message: "Upload succeeded" with a link to "Files and folders table". Below this, the "Summary" section shows a "Destination" of "S3://service-costs-bucket". It lists "Succeeded" with "1 file, 556.0 B (100.00%)" and "Failed" with "0 files, 0 B (0%)". Under "Files and folders", there's a table with one row: "service_costs.csv" (text/csv, 556.0 B, Status: Succeeded). The bottom of the screen shows standard AWS navigation links like CloudShell, Feedback, and Console Mobile App, along with copyright information and a cookie preferences link.

❖ Steps 2 : - Create IAM policy for Lambda and S3

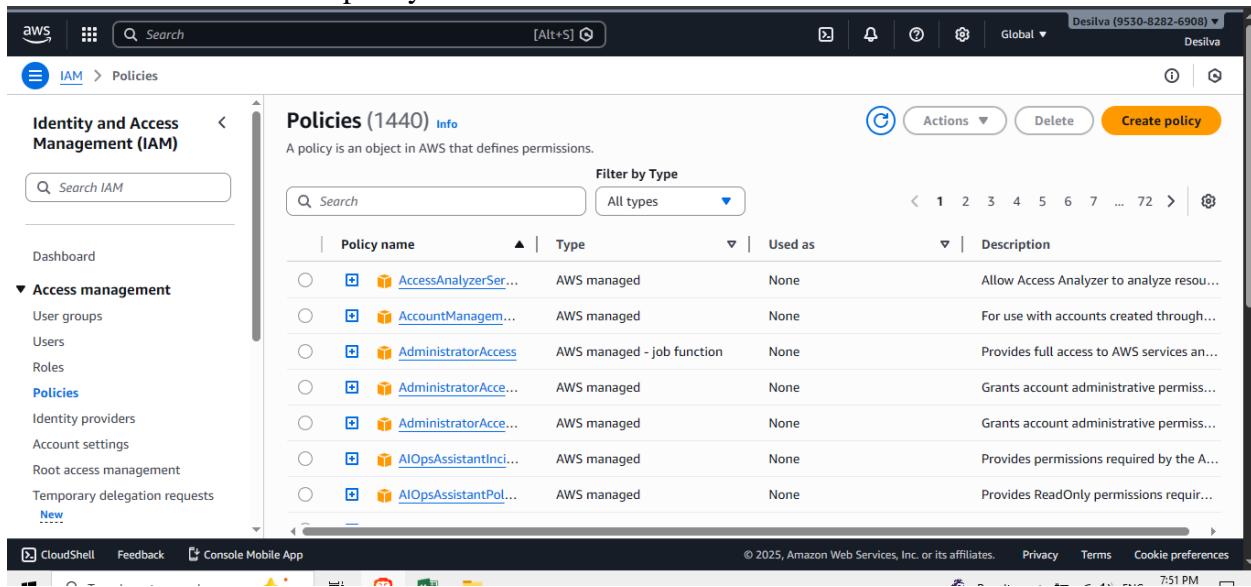
- Go to the IAM Console, click Policies, then Create policy. Click the JSON tab, paste the code below, and name it LambdaS3ReadPolicy

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Effect": "Allow",
      "Action": [
        "s3:GetObject"
      ],
      "Resource": "arn:aws:s3:::service-costs-bucket/*"
    },
    {
      "Effect": "Allow",
      "Action": [
        "logs:CreateLogGroup",
        "logs:CreateLogStream",
        "logs:PutLogEvents"
      ],
      "Resource": "*"
    }
  ]
}
```

- Make sure, put the S3 bucket name under the Resource.

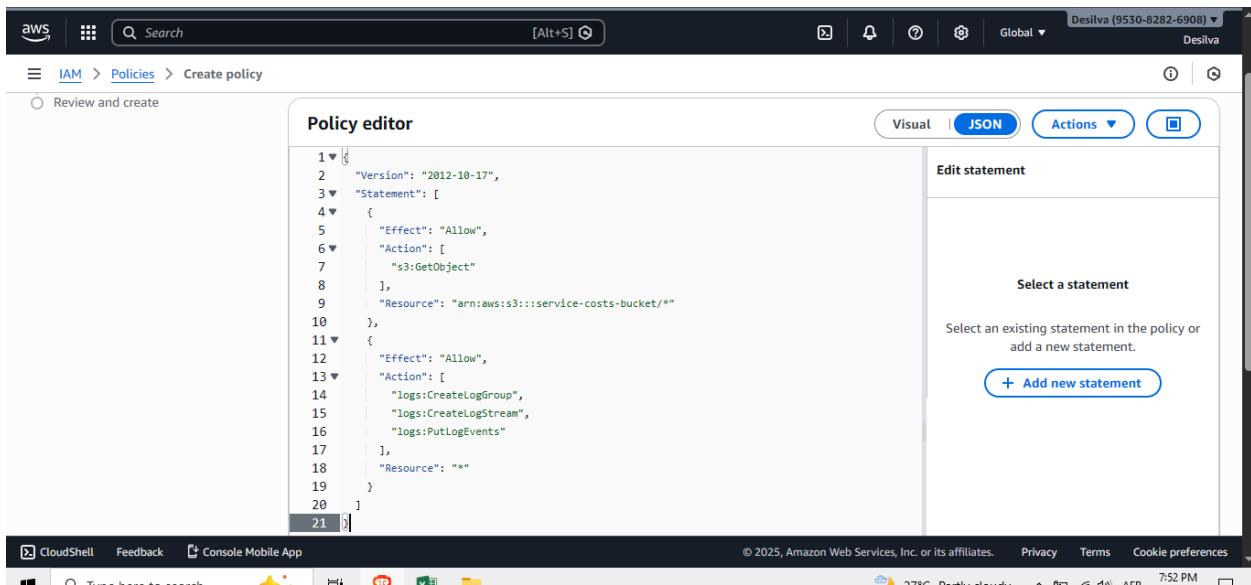
< "Resource": "arn:aws:s3:::service-costs-bucket/*" >

➤ Create IAM policy



The screenshot shows the AWS IAM Policies page. The left sidebar includes sections for Identity and Access Management (IAM), Dashboard, Access management (User groups, Users, Roles), Policies (selected), Identity providers, Account settings, Root access management, and Temporary delegation requests. A 'New' link is also present. The main area displays a table titled 'Policies (1440)'. The table has columns for Policy name, Type, Used as, and Description. Several AWS-managed policies are listed, such as AccessAnalyzerServiceRole, AccountManagement, AdministratorAccess, and others. The table includes a search bar, a filter by type dropdown, and navigation links (1-72). The bottom of the page features standard AWS footer links: CloudShell, Feedback, Console Mobile App, Privacy, Terms, and Cookie preferences.

➤ Attached S3 bucket policy json type code here,



The screenshot shows the 'Create policy' page in the AWS IAM console. The left sidebar shows the path: IAM > Policies > Create policy. The main area is titled 'Policy editor' and contains a JSON code editor. The JSON code defines a policy with two statements. The first statement allows 's3:GetObject' on any resource in the 'service-costs-bucket'. The second statement allows 'logs:CreateLogGroup', 'logs:CreateLogStream', and 'logs:PutLogEvents' on any resource. The right side of the screen shows an 'Edit statement' panel with a 'Select a statement' section and a '+ Add new statement' button. The bottom of the page includes standard AWS footer links: CloudShell, Feedback, Console Mobile App, Privacy, Terms, and Cookie preferences.

```
1 "Version": "2012-10-17",
2 "Statement": [
3     {
4         "Effect": "Allow",
5         "Action": [
6             "s3:GetObject"
7         ],
8         "Resource": "arn:aws:s3:::service-costs-bucket/*"
9     },
10    {
11        "Effect": "Allow",
12        "Action": [
13            "logs:CreateLogGroup",
14            "logs:CreateLogStream",
15            "logs:PutLogEvents"
16        ],
17        "Resource": "*"
18    }
19 ],
20 ]
21 ]
```

- After completing the steps above, create the S3 policy.

- Provide S3 policy name,

Review and create info

Review the permissions, specify details, and tags.

Policy details

Policy name
Enter a meaningful name to identify this policy.
LambdaS3ReadPolicy

Description - optional
Add a short explanation for this policy.

Permissions defined in this policy Info

Service	Access level	Resource	Request condition
CloudWatch Logs	Limited: Write	All resources	None
S3	Limited: Read	BucketName string like service-costs-bucket, ObjectPath string like All	None

Add tags - optional info

Tags are key-value pairs that you can add to AWS resources to help identify, organize, or search for resources.

No tags associated with the resource.

Add new tag

You can add up to 50 more tags.

Cancel **Previous** **Create policy**

- Navigate to the IAM Console, click Roles, then click Create role. Select AWS service as the trusted entity, choose Lambda as the use case, and click Next. Attach the policy named **LambdaS3ReadPolicy**. Proceed to the next step, provide the role name **LambdaS3TriggerRole**, and finally click Create role.

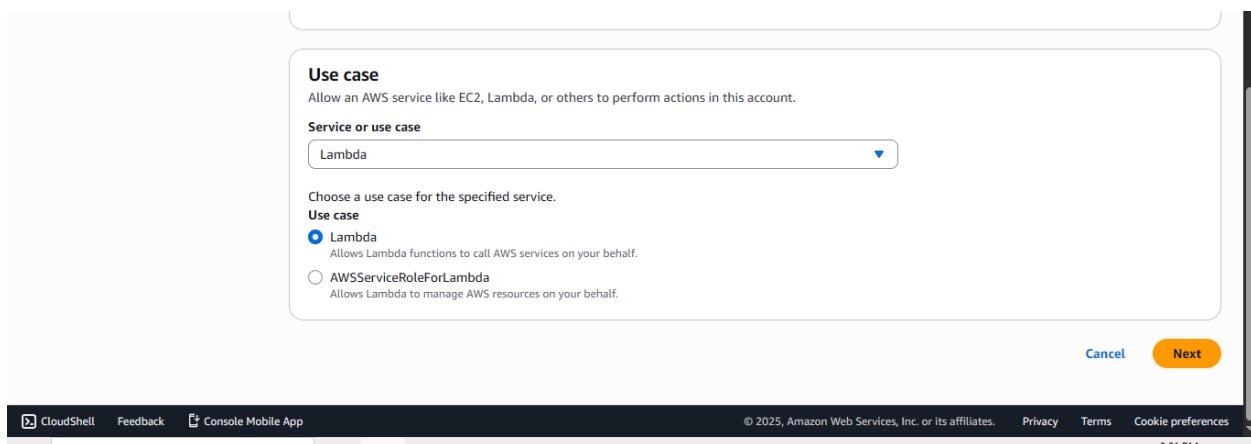
➤ Create role for Lambda

The screenshot shows the AWS IAM Roles page. The left sidebar has 'Identity and Access Management (IAM)' selected. Under 'Access management', 'Roles' is also selected. A search bar at the top says 'Search IAM'. The main area shows a table titled 'Roles (3) Info' with columns for 'Role name', 'Trusted entities', and 'Last activity'. Three roles are listed: 'AWSServiceRoleForResourceExplorer' (AWS Service: resource-explorer-2 Service, 14 minutes ago), 'AWSServiceRoleForSupport' (AWS Service: support (Service-Linker) -), and 'AWSServiceRoleForTrustedAdvisor' (AWS Service: trustedadvisor (Service) -). Below the table is a section titled 'Roles Anywhere' with icons for 'Access AWS from your non AWS workloads', 'X.509 Standard', and 'Temporary credentials', and a 'Manage' button.

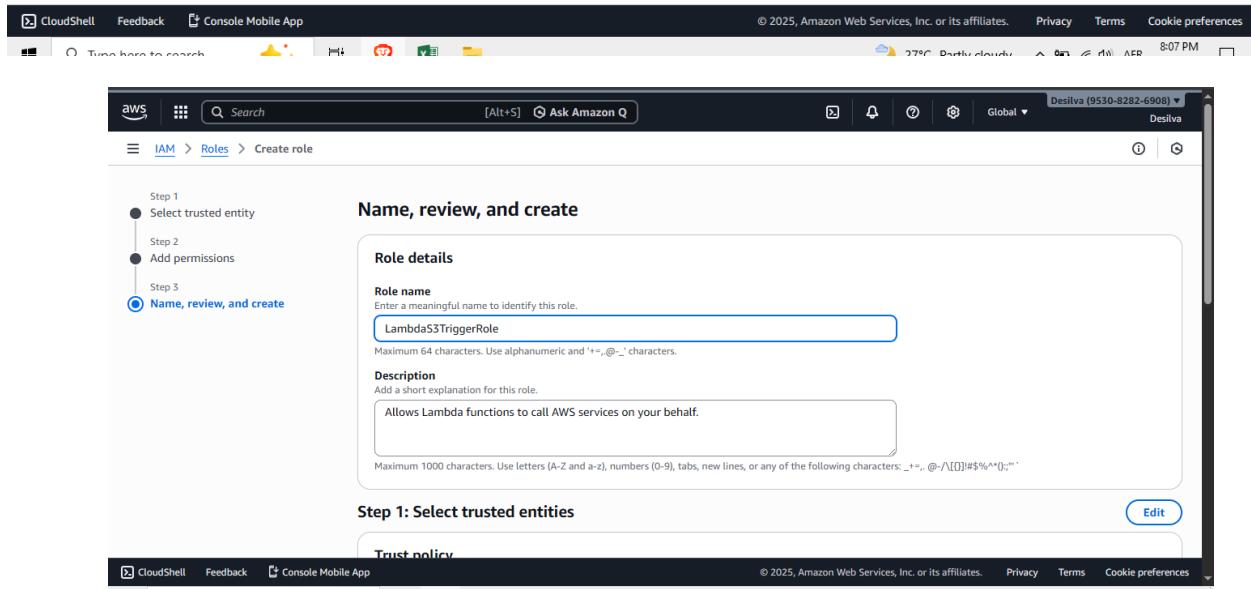
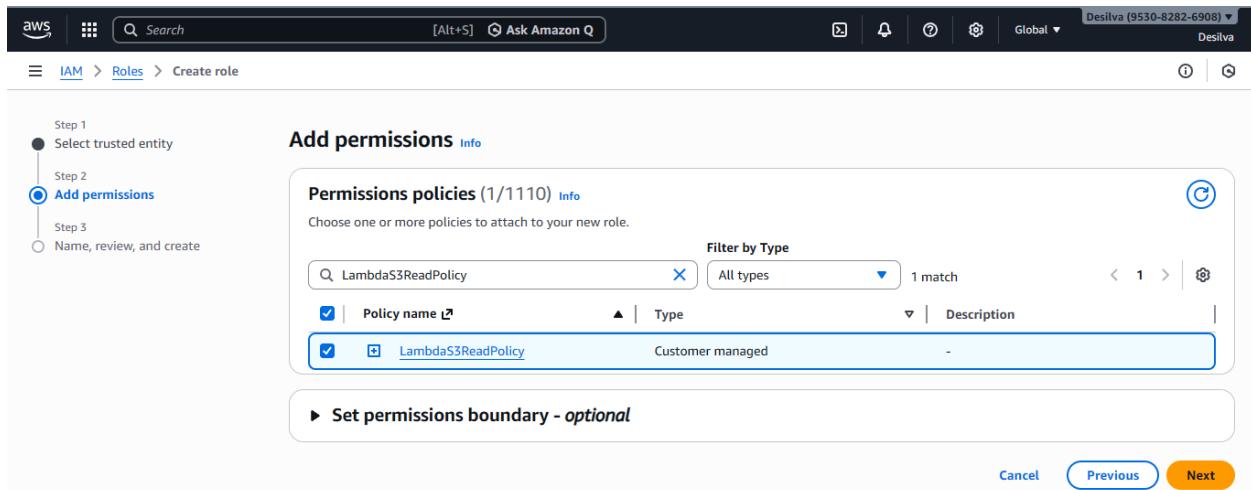
➤ Navigate AWS service

The screenshot shows the 'Create role' wizard, Step 1: Select trusted entity. The left sidebar shows 'Step 1' with 'Select trusted entity' selected, and 'Step 2' (Add permissions), 'Step 3' (Name, review, and create) are shown below. The main area is titled 'Select trusted entity' and contains a 'Trusted entity type' section. It lists four options: 'AWS service' (selected, highlighted in blue), 'AWS account', 'Web identity', and 'SAML 2.0 federation'. Each option has a brief description. Below this is a 'Use case' section with three cards: 'AWS Lambda', 'Amazon API Gateway', and 'Amazon Kinesis'. The bottom navigation bar includes 'CloudShell', 'Feedback', 'Console Mobile App', and standard footer links.

Select Lambda and click next



- Attached policy for Lambda (**LambdaS3ReadPolicy**), role name Provide **LambdaS3TriggerRole**



➤ Created Lambda Role,

The screenshot shows the AWS IAM Roles page. On the left, there's a sidebar with 'Identity and Access Management (IAM)' and a search bar. The main area has a heading 'Roles (1/4) Info' with a note about IAM roles. It lists four roles: 'AWSServiceRoleForResourceExplorer', 'AWSServiceRoleForSupport', 'AWSServiceRoleForTrustedAdvisor', and 'LambdaS3TriggerRole', which is highlighted with a blue border. There are columns for 'Role name', 'Trusted entities', and 'Last activity'. Below this is a section titled 'Roles Anywhere' with a note about authenticating non-AWS workloads.

❖ Steps 3 :- Create Lambda Function

- Go to the Lambda Console, create a function from scratch named "ProcessServiceCosts" with a Python 3.9+ runtime, and under Permissions choose "Use existing role" to select the LambdaS3TriggerRole before clicking Create function.

The screenshot shows the AWS Lambda 'Create function' page. At the top, it says 'Create function Info' and 'Choose one of the following options to create your function.' There are three options: 'Author from scratch' (selected), 'Use a blueprint', and 'Container image'. The 'Basic information' section includes fields for 'Function name' (set to 'ProcessServiceCosts'), 'Runtime' (set to 'Node.js 24.x'), and 'Durable execution - new' (disabled). To the right, there's a 'Tutorials' tab with a 'Create a simple web app' section and a 'Start tutorial' button.

➤ Choose Runtime as a python 3.9

The screenshot shows the 'Runtime' configuration page for a Lambda function. It includes sections for 'Runtime' (set to Python 3.9), 'Info' (describing supported languages), 'Durable execution - new' (enabling checkpointing), and a 'how to:' section with two bullet points: 'Build a simple web app, consisting of a Lambda function with a function URL that outputs a webpage' and 'Invoke your function'.

➤ Choose LambdaS3TriggerRole as execution role

The screenshot shows the 'Permissions' configuration page for a Lambda function. It includes sections for 'Change default execution role' (with 'Use an existing role' selected and 'LambdaS3TriggerRole' chosen), 'Execution role' (with 'Create a new role with basic Lambda permissions' and 'Create a new role from AWS policy templates' options), and 'Existing role' (with a link to view the role in the IAM console).

❖ Step 4 :- In the Code source tab, replace the default code

```
import json
import boto3
import csv
from datetime import datetime

s3 = boto3.client('s3')

def lambda_handler(event, context):
    # Get bucket and file name from event
    bucket = event['Records'][0]['s3']['bucket']['name']
    key = event['Records'][0]['s3']['object']['key']

    # Get CSV file from S3
    response = s3.get_object(Bucket=bucket, Key=key)
    lines = response['Body'].read().decode('utf-8').splitlines()

    # Parse CSV
    reader = csv.DictReader(lines)
    total_cost = 0

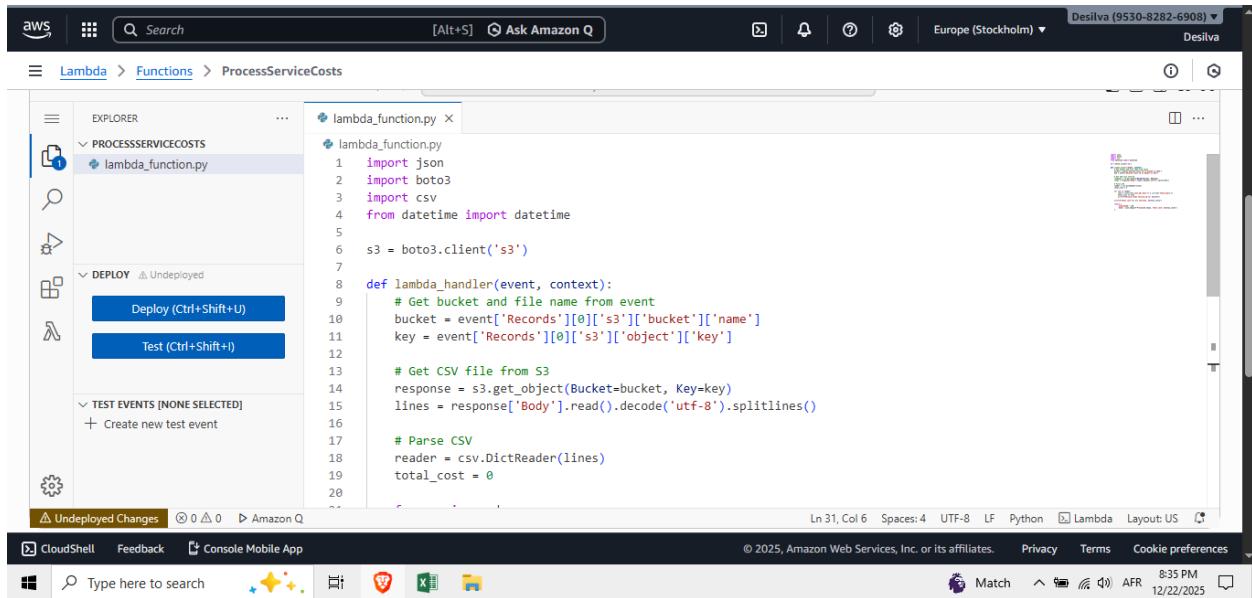
    for row in reader:
        cost = float(row['cost_per_hour']) * int(row['total_hours'])
        total_cost += cost
```

```
print(f"Service {row['service_id']}: ${cost}")

print(f"Total cost for all services: ${total_cost}")

return {
    'statusCode': 200,
    'body': json.dumps(f'Processed {key}. Total cost: ${total_cost}')
}
```

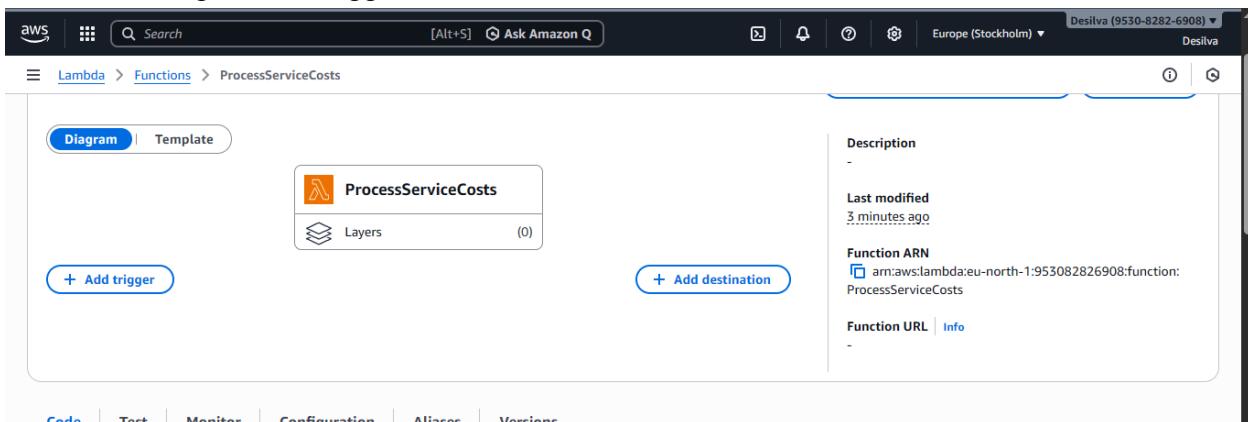
- Replace the default code instead existing sample code



❖ **Steps 5 :-** S3 Bucket trigger using Lambda

- On the Lambda function's Configuration page, add an S3 trigger by selecting your "service-costs-bucket" for all object create events, then click Add.

- Navigate Add trigger



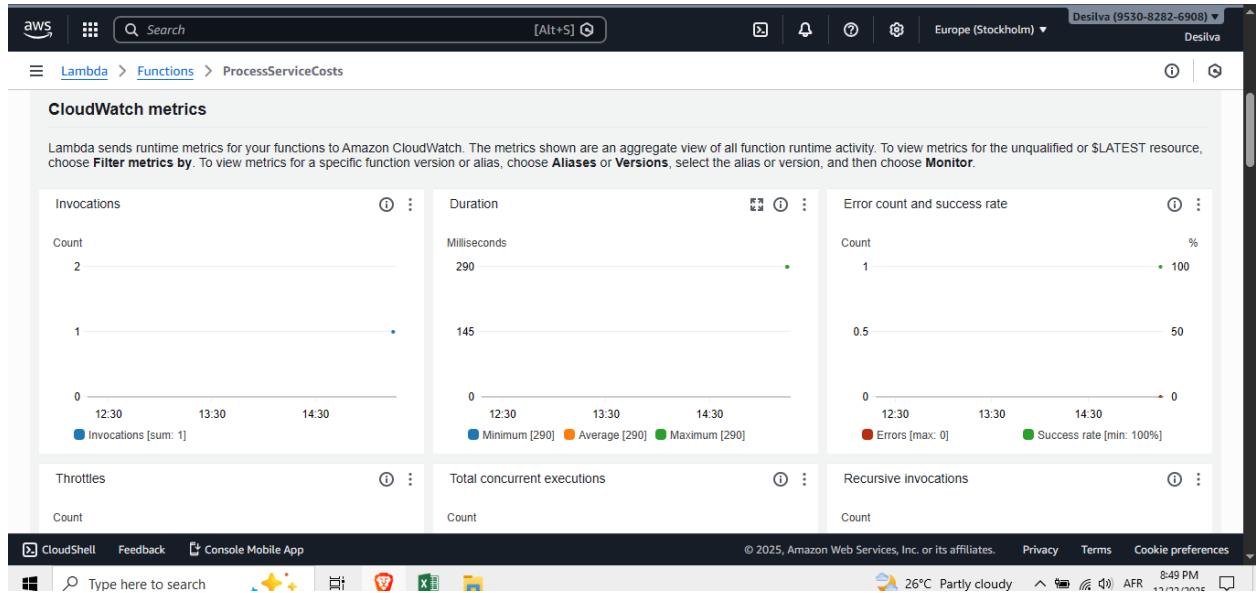
➤ Trigger the configuration

The screenshot shows the 'Add trigger' configuration page for a Lambda function. Under 'Trigger configuration', 'S3' is selected as the event source. A dropdown menu shows 'aws asynchronous storage'. The 'Bucket' field contains 's3/service-costs-bucket' with a region of 'eu-north-1'. Under 'Event types', 'All object create events' is selected. A note about 'Prefix - optional' is present. The bottom navigation bar includes CloudShell, Feedback, Console Mobile App, and links to Privacy, Terms, and Cookie preferences.

❖ Steps 6 : - Test the Set up

- To test the function, upload the service_costs.csv file again to the S3 bucket and then view its output by navigating to the Lambda's Monitor section and checking the CloudWatch logs.

➤ Monitor CloudWatch using Metrics



- Navigate to the Log management under the cloud watch

The screenshot shows the AWS CloudWatch Log Management interface. The left sidebar is collapsed, and the main area displays the 'Log group details' for the log group '/aws/lambda/ProcessServiceCosts'. The details include:

- Log class:** Info
- Standard**
- ARN:** arn:aws:logs:eu-north-1:953082826908:log-group:/aws/lambda/ProcessServiceCosts:*
- Creation time:** 8 minutes ago
- Retention:** Never expire
- Stored bytes:** -
- Metric filters:** 0
- Subscription filters:** 0
- Contributor Insights rules:** -
- KMS key ID:** -
- Deletion protection:** Off
- Data protection:** -
- Sensitive data count:** -
- Custom field indexes:** [Configure](#)
- Transformer:** [Configure](#)
- Anomaly detection:** [Configure](#)

Below the details, there are tabs for Log streams, Tags, Anomaly detection, Metric filters, Subscription filters, Contributor Insights, Data, and a navigation bar with CloudShell, Feedback, Console Mobile App, and links to © 2025, Amazon Web Services, Inc. or its affiliates., Privacy, Terms, and Cookie preferences.

- View the CSV file as log in AWS cloud watch

The screenshot shows the AWS CloudWatch Log Events interface. The left sidebar is collapsed, and the main area displays the 'Log events' for the log group '/aws/lambda/ProcessServiceCosts'. The log entries are listed in a table:

Timestamp	Message
2025-12-22T15:18:12.875Z	START RequestId: ce7a31c5-fd17-4d75-a142-8f609719cec6 Version: \$LATEST
2025-12-22T15:18:13.131Z	Service SVR101: \$300.0
2025-12-22T15:18:13.131Z	Service SVR102: \$3800.0
2025-12-22T15:18:13.131Z	Service DB205: \$2400.0
2025-12-22T15:18:13.131Z	Service DB310: \$600.0
2025-12-22T15:18:13.131Z	Service NET501: \$600.0
2025-12-22T15:18:13.131Z	Service SVR220: \$2400.0
2025-12-22T15:18:13.131Z	Service DB450: \$800.0

At the top right, there are buttons for Actions, Start tailing, Create metric filter, and a filter bar with a search input, time range (1m, 1h), and UTC timezone dropdown. At the bottom right, there is a Back to top button.