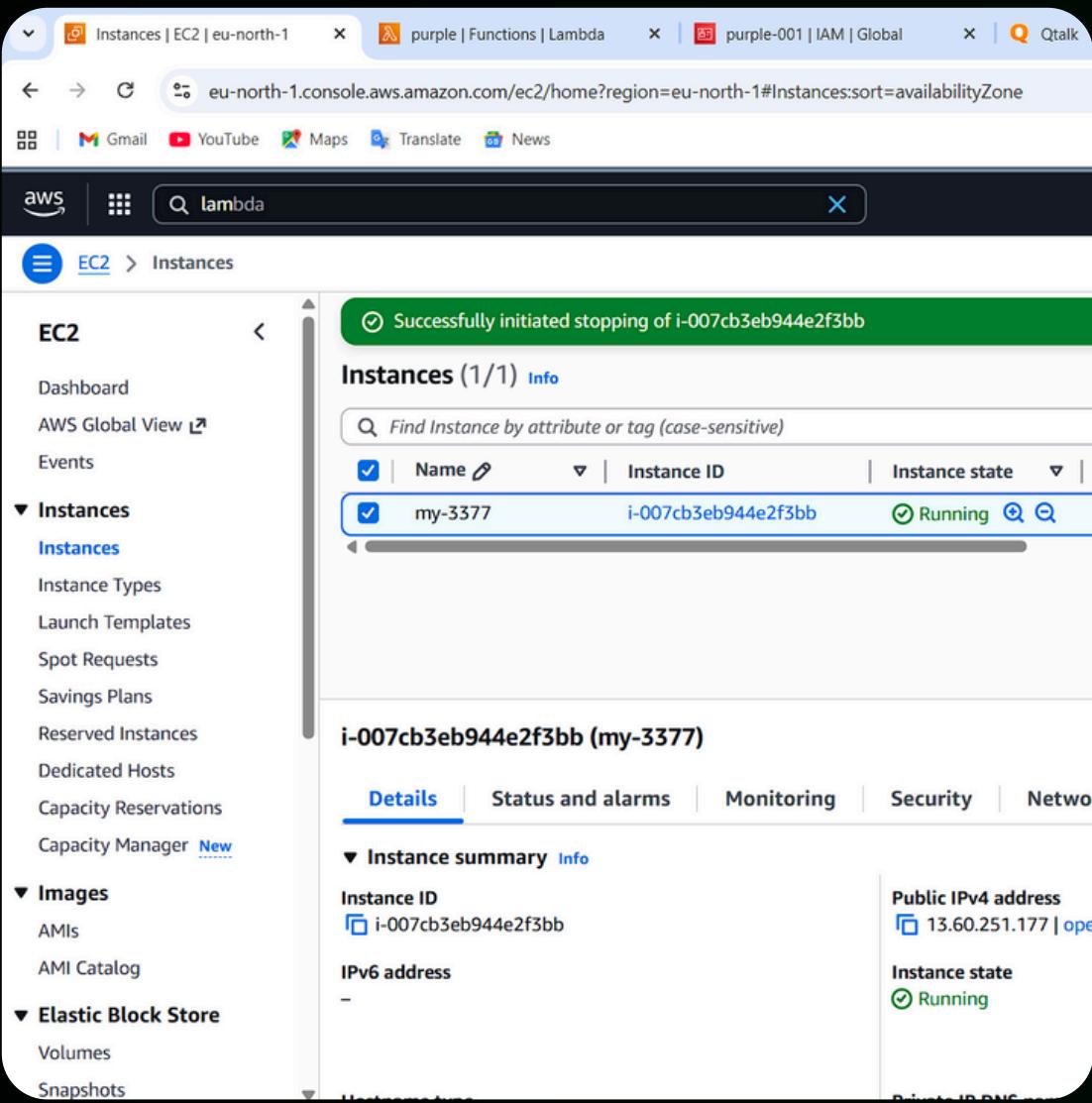


AWS Lambda Function Task

HI IM FORM M-5 AWS BATCH



This section contains the tasks performed by the AWS Lambda function, including its operations, workflow, and execution steps



EC2 Instance

During the creation of the EC2 instance, ensure that it is properly configured to connect with the AWS Lambda function by setting up a synchronous event-source connection

synchronous

The screenshot shows the AWS Identity and Access Management (IAM) service interface. The left sidebar has a 'Search IAM' bar and a navigation menu with 'Identity and Access Management (IAM)' at the top, followed by 'Dashboard', 'Access management' (with 'User groups', 'Users', and 'Roles' listed), and 'Policies'. The main content area is titled 'Roles (1/5)' with a 'Info' link. It contains a search bar and a table with the following data:

Role name	Trusted entities	Last activity
AWSServiceRoleForResourceExplorer	AWS Service: resource-explorer-2 (Service-Linked)	1 hour ago
AWSServiceRoleForSupport	AWS Service: support (Service-Linked)	-
AWSServiceRoleForTrustedAdvisor	AWS Service: trustedadvisor (Service-Linked)	-
purple-001	AWS Service: lambda	-
purple-role-tyzc07io	AWS Service: lambda	-

At the top right of the main content area are three buttons: a blue 'Edit' button, a grey 'Delete' button, and an orange 'Create role' button.

IAM - Roles

During the creation of the IAM role, assign the required permissions that allow the role to access the AWS Lambda function and interact with its configured destinations of start & stop.

Instances | EC2 | eu-north-1 purple | Functions | Lambda Roles | IAM | Global Qtalk

eu-north-1.console.aws.amazon.com/lambda/home?region=eu-north-1#/functions/purple?newFunction=true

Gmail YouTube Maps Translate News

aws | X

Lambda > Functions > purple

Code Test Monitor Configuration Aliases Versions

General configuration

Triggers

Permissions

Destinations

Function URL

Environment variables

Tags

VPC

RDS databases

Monitoring and operations tools

Concurrency and recursion detection

Asynchronous invocation

Execution role

Role name
purple-001 ↗

Resource summary

To view the resources and actions that your function has permission to access, choose a resource or action in the list below.

Action	Resource
Amazon EC2	2 actions, 1 resource

By action By resource

Resource	Actions
All resources	Allow: ec2:StartInstances Allow: ec2:StopInstances

Information Lambda obtained this information from the following policy statements:

- Inline policy **carrot-001**, statement **VisualEditor0**

Lambda

Function

While creating the Lambda function, configure the required permissions and enable access to its destination so the function can connect and operate properly.

lambda-code

Within the Lambda function code, the Instance ID and Region can be updated according to your requirements. Simply replace the existing values in the code to reflect the correct instance details.

Script - 1

```
import boto3
region = 'eu-north-1'
instances = ['i-007cb3eb944e2f3bb']
ec2=boto3.client('ec2',region_name=region)

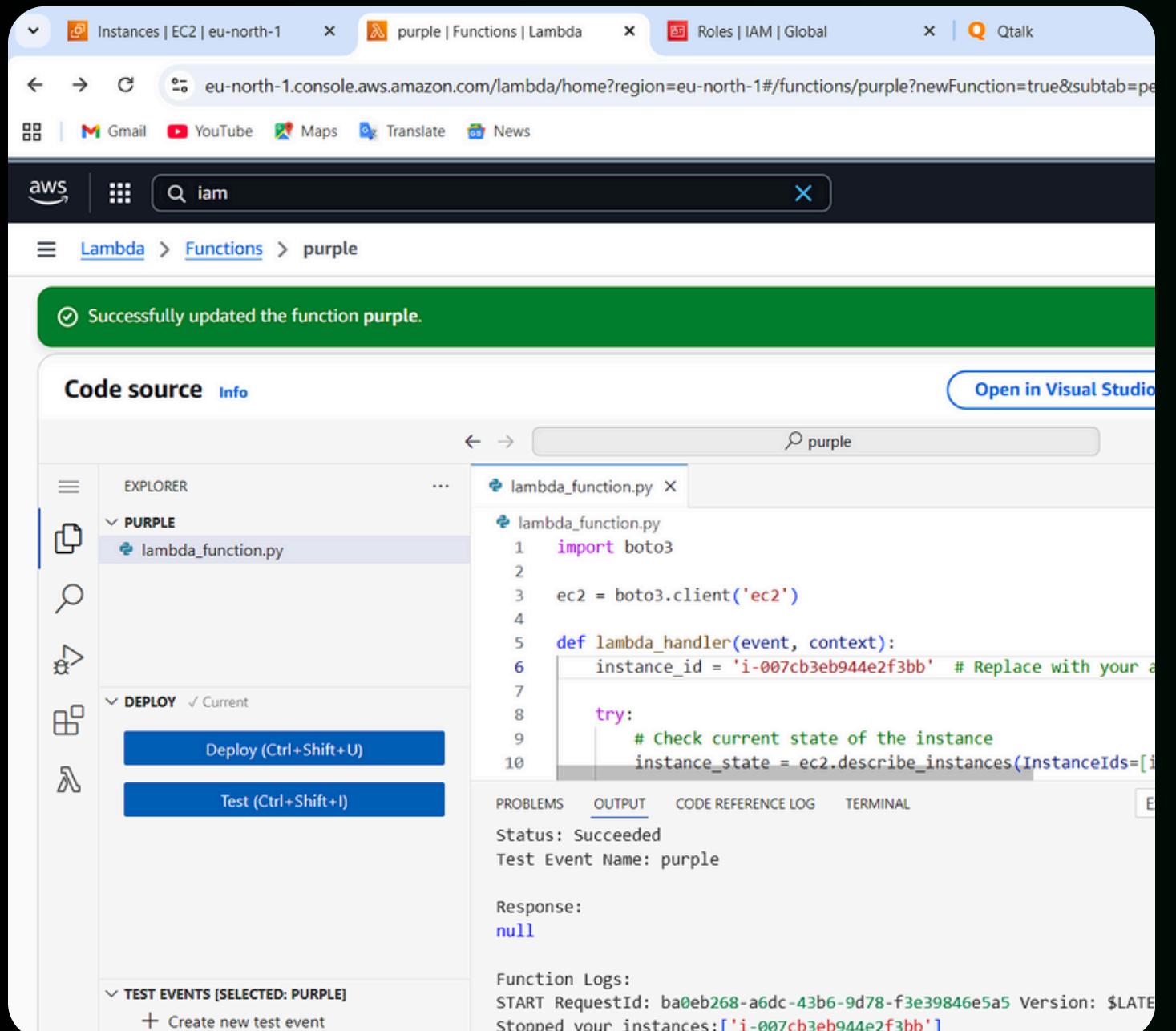
def lambda_handler(event,context):
    ec2.stop_instances(InstanceIds=instances)
    print('Started your instances:' +str(instances))

Status: Succeeded
Test Event Name: purple

Response:
null

Function Logs:
START RequestId: 8234796d-e988-4dba-945f-86a717c99911 Version: $LATEST
Started your instances:[ 'i-007cb3eb944e2f3bb' ]
```

lambda - code!



Script - 2

Within the Lambda function code, the Instance ID and Region can be updated according to your requirements. Simply replace the existing values in the code to reflect the correct instance details.

lambda - code!

Within the Lambda function code, the Instance ID and Region can be updated according to your requirements. Simply replace the existing values in the code to reflect the correct instance details.

The image shows four separate code editor windows, each titled with a script number (script_1.t, script_2.t, script_3.t, script_4.t). Each window contains a snippet of Python code for an AWS Lambda function. The code uses the boto3 library to interact with the EC2 service. The snippets differ in the AWS region they target (e.g., 'us-east-1' vs 'eu-north-1') and the specific actions performed (e.g., starting or stopping instances). The code is presented in a monospaced font within a dark-themed editor.

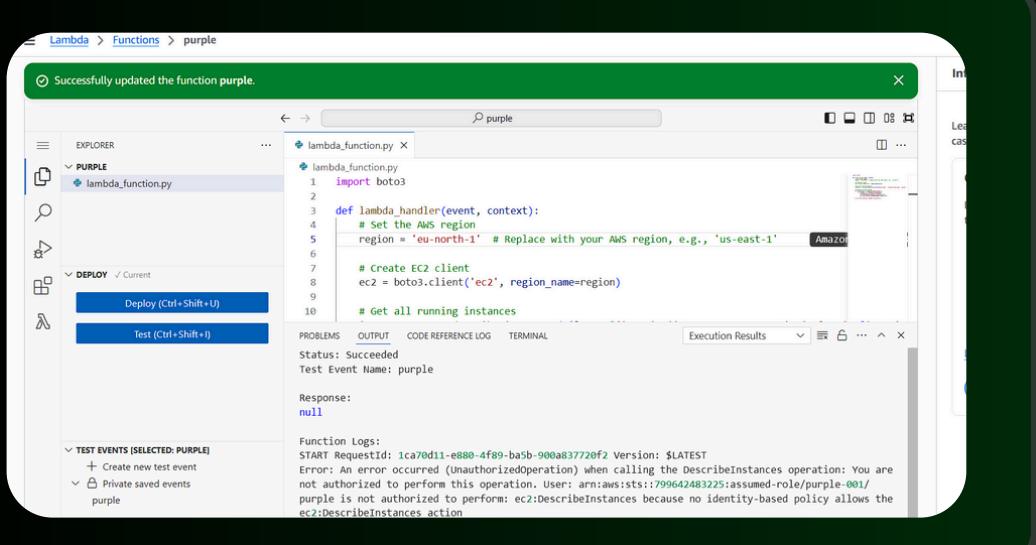
```
script_1.t:
import boto3
region = 'region'
instances =['instance id' ]
ec2
=boto3.client('ec2',region_name=region)
def lambda_handler(event,context):
    ec2.start_instances(InstanceIds=instances)
    print('Started your
instances:'+str(instances))

script_2.t:
import boto3
region = 'region'
instances =['instance id' ]
ec2
=boto3.client('ec2',region_name=region)
def lambda_handler(event,context):
    instance_id = 'instance ID' # Replace with your actual EC2 instance ID
    try:
        instance_state =
ec2.describe_instances(InstanceIds=[instance_id])['Reservations'][0]
        ['Instances'][0]['State']['Name']
    except:
        pass
    if instance_state == 'stopped':
        ec2.start_instances(InstanceIds=[instance_id])
        print(f"Started EC2 instance {instance_id}")
    elif instance_state ==
        'running':
        ec2.stop_instances(InstanceIds=[instance_id])

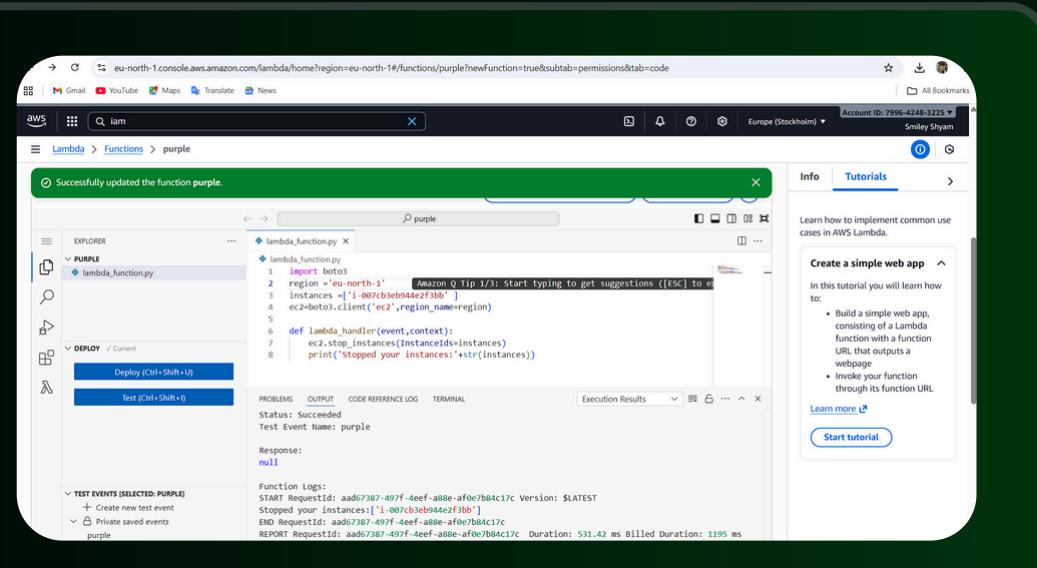
script_3.t:
import boto3
ec2 = boto3.client('ec2')
def lambda_handler(event, context):
    instance_id = 'instance ID' # Replace with your actual EC2 instance ID
    instances =
ec2.describe_instances(Filters=[{'Name': 'instance-state-name', 'Values': ['running']}])
    for reservation in instances['Reservations']:
        for instance in reservation['Instances']:
            if instance['State']['Name'] ==
                'running':
                ec2.stop_instances(InstanceIds=[instance['InstanceId']])
                print(f"Stopping instance: {instance['InstanceId']}")

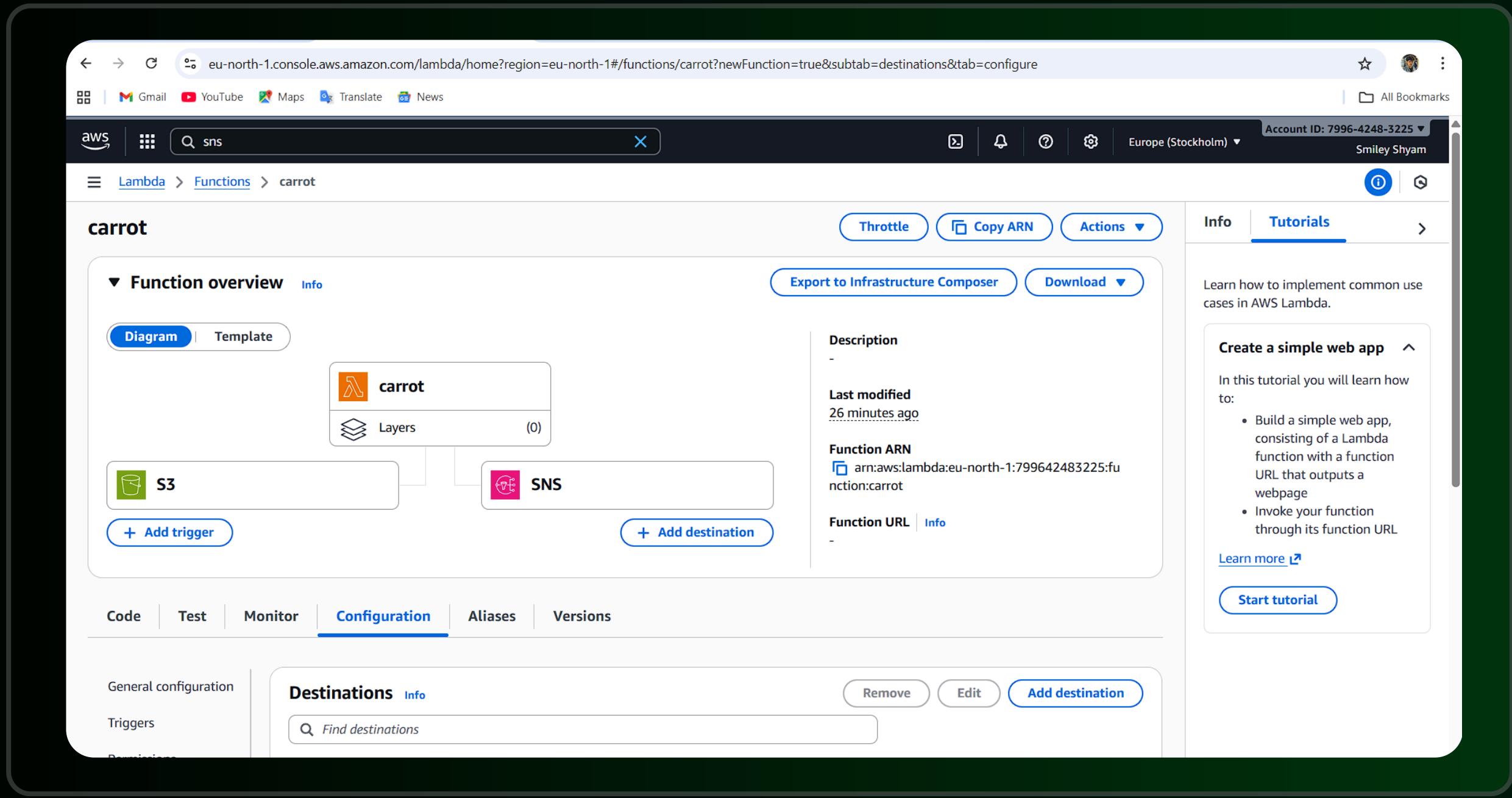
script_4.t:
def lambda_handler(event, context):
    # Set the AWS region
    region = 'region_name' # Replace with your AWS region, e.g., 'us-east-1'
    # Create EC2 client
    ec2 = boto3.client('ec2',
    region_name=region)
    # Get all running instances
    instances =
ec2.describe_instances(Filters=[{'Name': 'instance-state-name', 'Values': ['running']}])
    # Stop each running instance
    for reservation in instances['Reservations']:
        for instance in reservation['Instances']:
            if instance['State']['Name'] ==
                'running':
                ec2.stop_instances(InstanceIds=[instance['InstanceId']])
                print(f"Stopping instance: {instance['InstanceId']}")
```

Script - 3



Script - 4





Lambda - Function

Create a Lambda function that connects to an S3 event source and delivers its output to an SNS destination, enabling asynchronous processing through the Lambda service.

Asynchronous

Simple Notification Service - (SNS)

Create an SNS destination to integrate with the Lambda function using the asynchronous invocation method. This setup allows the process to send notifications either by email or SMS to the selected recipient.

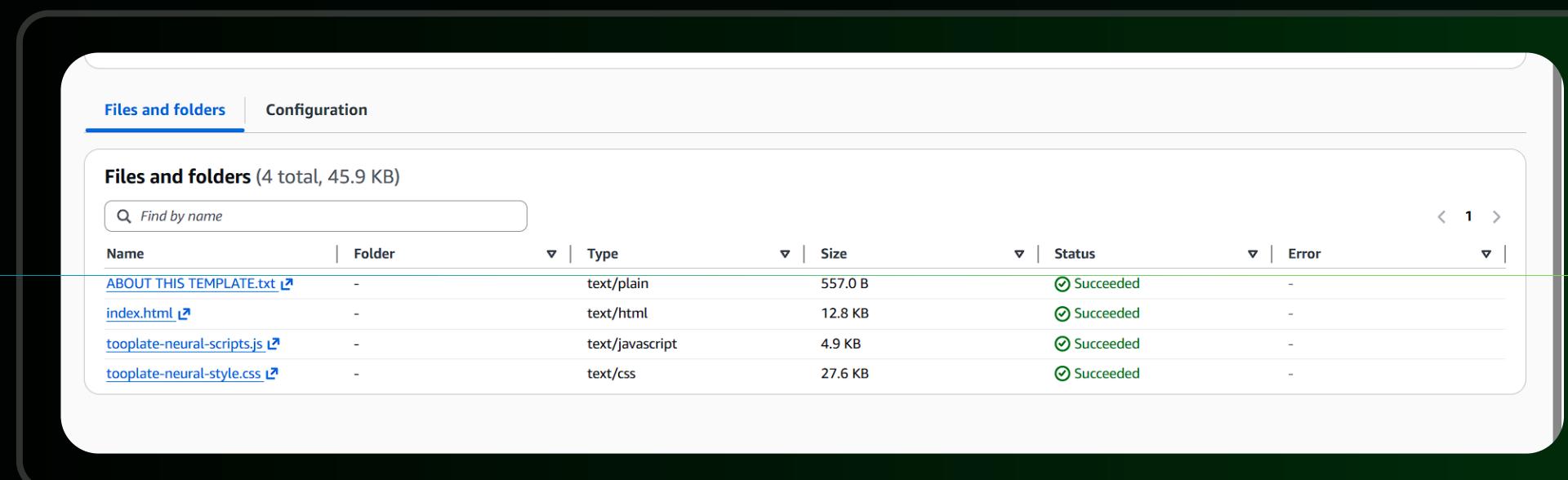
The screenshot shows the Amazon SNS Topics page. On the left, there's a navigation sidebar with 'Amazon SNS' at the top, followed by 'Dashboard', 'Topics' (which is highlighted in blue), 'Subscriptions', and a 'Mobile' section containing 'Push notifications' and 'Text messaging (SMS)'. The main area has a title 'Des' with 'Edit', 'Delete', and 'Publish message' buttons. Below this is a 'Details' section with fields: 'Name' (Des), 'Display name' (carrot-Dec), 'ARN' (arn:aws:sns:eu-north-1:799642483225:Des), and 'Topic owner' (799642483225). Underneath is a 'Type' field set to 'Standard'. At the bottom of this section is a navigation bar with tabs: 'Subscriptions' (which is blue and underlined), 'Access policy', 'Data protection policy', 'Delivery policy (HTTP/S)', 'Delivery status logging', 'Encryption', and 'Tags'. The 'Subscriptions' tab is active, showing a table with one row. The table columns are 'ID', 'Endpoint', 'Status', and 'Protocol'. The single row shows an ID of '23aa919b-e547-4d66-8e78-f8eb6...', an endpoint of 'shyamsundar05285518@gmail.com', a status of 'Confirmed' (with a green checkmark icon), and a protocol of 'EMAIL'.

IAM - Roles

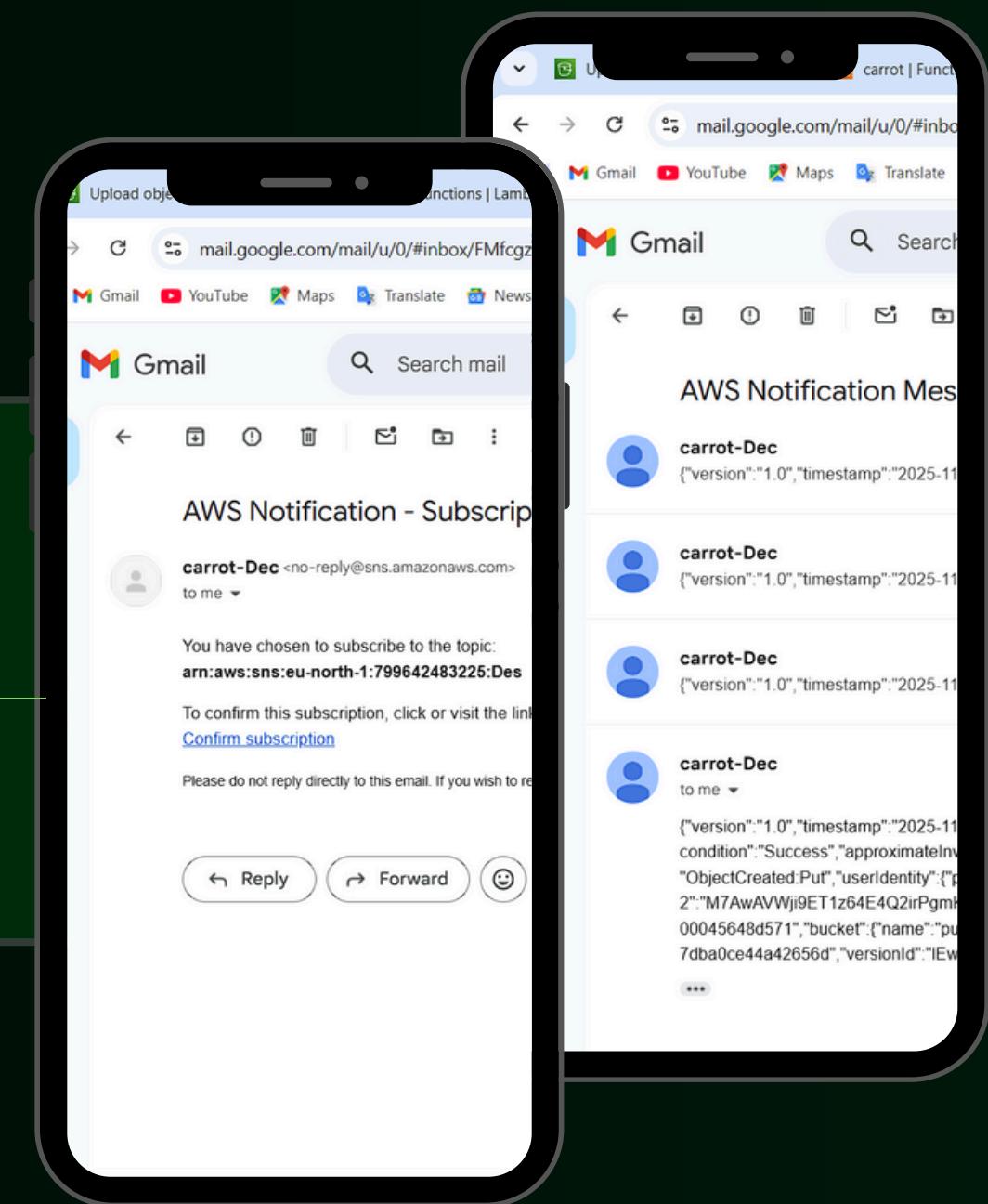
Configure appropriate IAM roles that enable S3 to connect with SNS via the Lambda function. Assign the required permissions—such as s3:PutObject and s3:DeleteObject—to ensure secure and seamless processing

The screenshot shows the AWS IAM Roles Permissions page for a user named 'shyam'. The top navigation bar includes links for IAM, Roles, and shyam. The main menu on the left lists Identity and Access Management (IAM), Dashboard, Access management (User groups, Users, Roles, Policies, Identity providers, Account settings), and a search bar for 'Search IAM'. The 'Permissions' tab is selected in the top navigation bar. Below it, tabs for Trust relationships, Tags, Last Accessed, and Revoke sessions are visible. A section titled 'Permissions policies (1)' contains a link to 'carrot-001'. A note says 'You can attach up to 10 managed policies.' Below this is a 'Filter by Type' section with a search bar, a dropdown for 'All types', and buttons for 'Simulate', 'Remove', and 'Add permissions'. A table lists the policy 'carrot-001' as a 'Customer inline' policy with 0 attached entities. At the bottom, a section titled 'Permissions boundary (not set)' is shown.

Upload-s3 files



After uploading files to the S3 bucket, the configured asynchronous workflow will send the file information to your registered email. Make sure the email ID has the necessary access permissions in advance; otherwise, the notifications will not appear as expected.



mail Messages

THANK YOU

F O R Y O U R T A S K S

Both Lambda tasks—synchronous and asynchronous—are successfully completed, including the EC2 instance setup and the S3-to-SNS workflow. Thank you for guiding me through the process.