

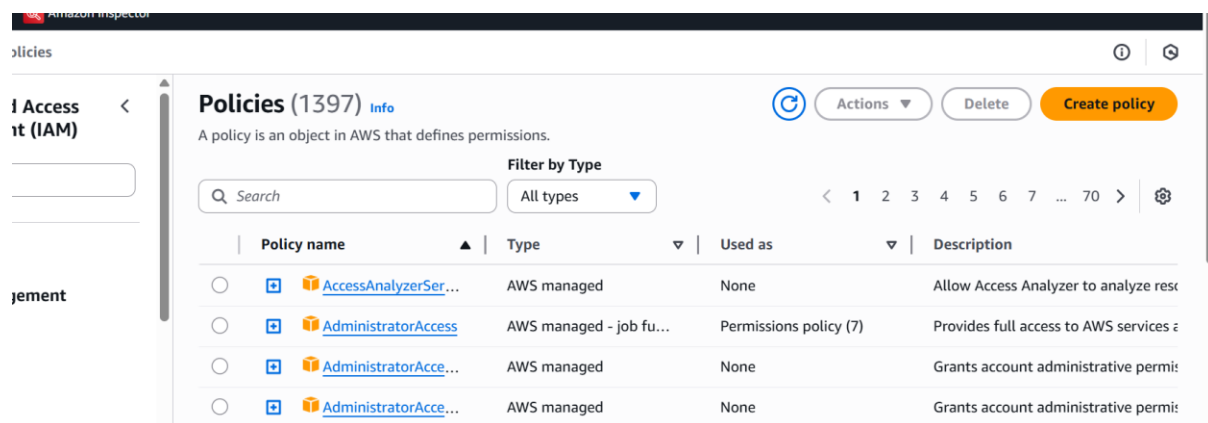
SNS and SQS

SNS and SQS with AWS Lambda

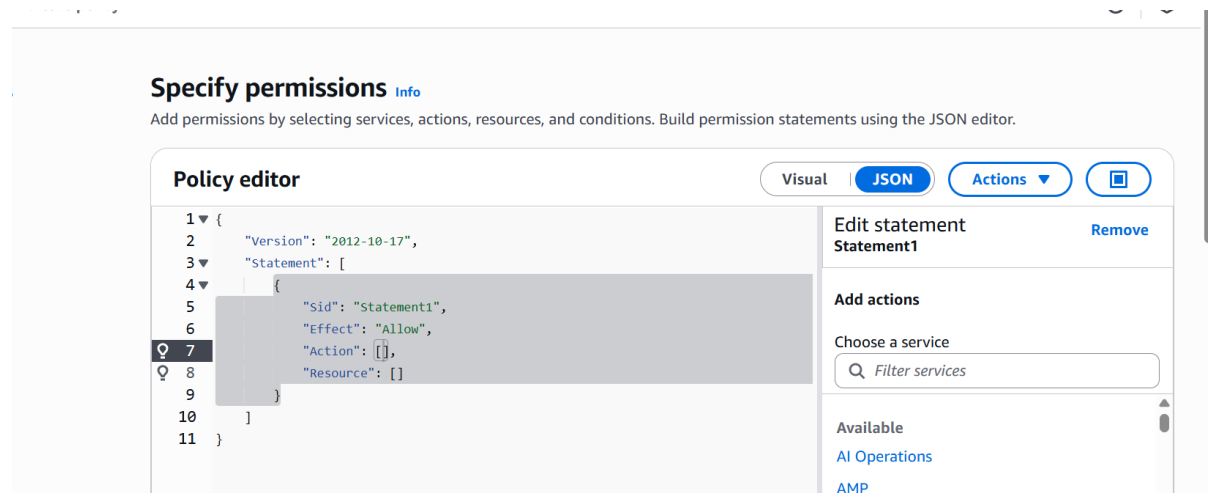
1. IAM Role Update

Ensure the Lambda role (lambda-ec2-stop-role) has the following permissions:

Iam → policy → Create Policy



Copy and paste below json



```

{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Sid": "VisualEditor0",
      "Effect": "Allow",
      "Action": [
        "ec2:DescribeInstances",
        "ec2:DescribeRegions",
        "ec2:StopInstances",
        "rds:DescribeDBInstances",
        "rds:StopDBInstance",
        "sns:Publish",
        "sqs:SendMessage"
      ],
      "Resource": "*"
    }
  ]
}

```

Policy details

Policy name

Enter a meaningful name to identify this policy.

Maximum 128 characters. Use alphanumeric and '+=, @-_' characters.

Description - *optional*

Add a short explanation for this policy.

Maximum 1,000 characters. Use alphanumeric and '+=, @-_' characters.

Create Role

Select trusted entity [Info](#)

Trusted entity type



AWS service

Allow AWS services like EC2, Lambda, or others to perform actions in this account.



AWS account

Allow entities in other AWS accounts belonging to you or a 3rd party to perform actions in this account.



Web identity

Allows users federated by the specified external web identity provider to assume this role to perform actions in this account.



SAML 2.0 federation

Allow users federated with SAML 2.0 from a corporate directory to perform actions in this account.



Custom trust policy

Create a custom trust policy to enable others to perform actions in this account.

Use case

Allow an AWS service like EC2, Lambda, or others to perform actions in this account.

Service or use case

Lambda

Choose a use case for the specified service.

Use case



Lambda

Allows Lambda functions to call AWS services on your behalf.




[Cancel](#)

[Next](#)

Select the policy

Permissions policies (1/1089) [Info](#)

Choose one or more policies to attach to your new role.

Filter by Type			
<input type="text" value="ec2s"/>	<input type="button" value="X"/>	<input type="button" value="All types"/>	3 matches
<input type="checkbox"/>	Policy name ↗	Type	Description
<input type="checkbox"/>	 AmazonEC2SpotFleetAut...	AWS managed	Policy to enable Autoscaling for Amaz...
<input type="checkbox"/>	 AmazonEC2SpotFleetTag...	AWS managed	Allows EC2 Spot Fleet to request, term...
<input checked="" type="checkbox"/>	 ec2snssqs	Customer managed	-

► Set permissions boundary - optional

[Cancel](#)

[Previous](#)

[Next](#)

Role details

Role name
Enter a meaningful name to identify this role.

lambdasqs

Maximum 64 characters. Use alphanumeric and '+=, @- _ ' characters.

Description
Add a short explanation for this role.

Allows Lambda functions to call AWS services on your behalf.

Maximum 1000 characters. Use letters (A-Z and a-z), numbers (0-9), tabs, new lines, or any of the following characters: _+=, @-/[{}!#\$%^&*(){};:'"``

Step 1: Select trusted entities [Edit](#)

Trust policy

Create SNS

sns

Services

Features

Resources **New**

Documentation


Knowledge articles


Marketplace


Blog posts

Events

Tutorials

 **Simple Notification Service**
SNS managed message topics for Pub/Sub

 **Route 53 Resolver**
Resolve DNS queries in your Amazon VPC and on-premises network.

 **Route 53**
Scalable DNS and Domain Name Registration

Features

Show more

- Go to SNS → Topics → Create topic

Application Integration

Amazon Simple Notification Service

Pub/sub messaging for microservices and serverless applications.

Amazon SNS is a highly available, durable, secure, fully managed pub/sub messaging service that enables you to decouple microservices, distributed systems, and event-driven serverless applications. Amazon SNS provides topics for high-throughput, push-based, many-to-many

Create topic

Topic name
A topic is a message channel. When you publish a message to a topic, it fans out the message to all subscribed endpoints.

Next step

[Start with an overview](#)

Create topic

Topic name

A topic is a message channel. When you publish a message to a topic, it fans out the message to all subscribed endpoints.

Next step

[Start with an overview](#)

- Choose **Standard** queue

FSx VPC Amazon Inspector

Amazon SNS > Topics > Create topic

Details

Type | [Info](#)
Topic type cannot be modified after topic is created

☐ **FIFO (first-in, first-out)**

- Strictly-preserved message ordering
- Exactly-once message delivery
- Subscription protocols: SQS

☒ **Standard**

- Best-effort message ordering
- At-least once message delivery
- Subscription protocols: SQS, Lambda, Data Firehose, HTTP, SMS, email, mobile application endpoints

Name

Maximum 256 characters. Can include alphanumeric characters, hyphens (-) and underscores (_).

Display name - optional | [Info](#)
To use this topic with SMS subscriptions, enter a display name. Only the first 10 characters are displayed in an SMS message.

Create topic

that you can assign to an Amazon SNS topic. Each tag consists of a key and an optional value. You can use tags to search and filter your topics and track

Optional Info

Clicking for this topic to view its traces and service map in Amazon CloudWatch. Additional costs apply.

Cancel

Create topic

- Use Standard, name it, and copy the ARN.

Details

Name
lambdaec2

Display name

-

ARN
arn:aws:sns:us-east-1:883308508227:lambdaec2

Topic owner
883308508227

Type
Standard

- Add a subscription (e.g., Email).

FSxVPCAmazon Inspector

Amazon SNS > Subscriptions > Create subscription

Create subscription

Details

Topic ARN

arn:aws:sns:us-east-1:883308508227:lambdaec2

Protocol

The type of endpoint to subscribe

Email

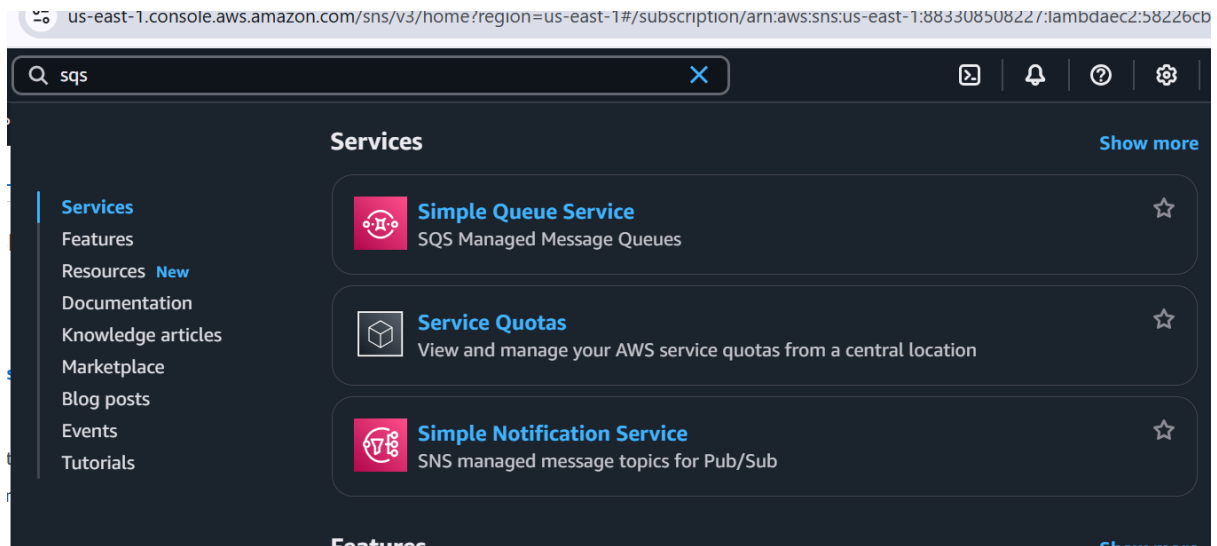
Endpoint

An email address that can receive notifications from Amazon SNS.

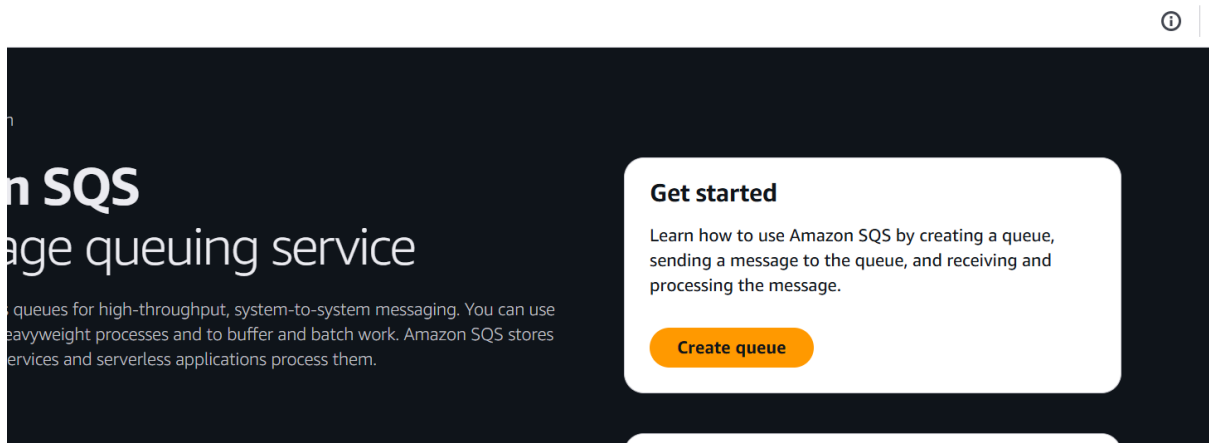
amit@openwriteup.com

SQS Setup

- Go to SQS → Create Queue



- Choose Standard queue



FSx VPC Amazon Inspector

[Amazon SQS](#) > [Queues](#) > Create queue

Details

Type
Choose the queue type for your application or cloud infrastructure.

☒ **Standard** [Info](#)
At-least-once delivery, message ordering isn't preserved

- At-least once delivery
- Best-effort ordering

☐ **FIFO** [Info](#)
First-in-first-out delivery, message ordering is preserved

- First-in-first-out delivery
- Exactly-once processing

You can't change the queue type after you create a queue.

Name
mylambda

A queue name is case-sensitive and can have up to 80 characters. You can use alphanumeric characters, hyphens (-), and underscores (_).

Configuration

[Info](#)

- Copy the **Queue URL** for use in the code.

Amazon SQS > Queues > mylambda

Queue mylambda created successfully
You can now send and receive messages.

mylambda

[Edit](#)[Delete](#)[Purge](#)[Send and receive messages](#)[Start D](#)

Details [Info](#)

Name

mylambda

Type

Standard

ARN

arn:aws:sqs:us-east-1:883308508227:mylar

Encryption

Amazon SQS key (SSE-SQS)

URL

https://sqs.us-east-1.amazonaws.com/883308508227/mylambda

Dead-letter queue

-

[More](#)

Lambda

Lambda > Functions > Create function

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 fu

Basic information

Function name

Enter a name that describes the purpose of your function.

lambdasns

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 3.13

**Architecture** [Info](#)

Choose the instruction set architecture you want for your function code.

Execution roleChoose a role that defines the permissions of your function. To create a custom role, go to the [IAM console](#).

Create a new role with basic Lambda permissions



Use an existing role



Create a new role from AWS policy templates

Existing role

Choose an existing role that you've created to be used with this Lambda function. The role must have permission to upload logs to Amazon CloudWatch Logs.

lambdasqs

[View the lambdasqs role](#) on the IAM console.

Additional configurations

Use additional configurations to set up code signing, function URL, tags, and Amazon VPC access for your function.

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

Modify below code

- Provide SNS arn
- SQS url

```

lambda_function.py
3  def lambda_handler(event, context):
4      }
5
6
7      # SNS Notification
8      sns_client.publish(
9          TopicArn=arn:aws:sns:us-east-1:883308508227:lambdaec2,
10         Subject='AWS Shutdown Automation Report',
11         Message=json.dumps(message, indent=2)
12     )
13
14     # SQS Message
15     sqs_client.send_message(
16         QueueUrl=https://sqs.us-east-1.amazonaws.com/883308508227/mylambda,
17         MessageBody=json.dumps(message)
18     )
19
20     return message
21
22 Amazon Q Tip 1/3: Start typing to get suggestions ([ESC] to exit)

```

```

import boto3

import json

def lambda_handler(event, context):

    stopped_ec2 = []

    stopped_rds = []

    # Clients for default region (used for SNS, SQS, and describing all regions)
    ec2_client = boto3.client('ec2')
    sns_client = boto3.client('sns')
    sqs_client = boto3.client('sqs')

    # Replace with your actual SNS Topic ARN and SQS Queue URL
    sns_topic_arn = 'arn:aws:sns:us-east-1:883308508227:lambdaec2'
    sqs_queue_url = 'https://sqs.us-east-1.amazonaws.com/883308508227/mylambda'

    # Get list of all regions

```

```

try:
    regions = [r['RegionName'] for r in ec2_client.describe_regions()['Regions']]
except Exception as e:
    print(f"Error fetching regions: {e}")
    return {"error": "Unable to fetch AWS regions"}

for region in regions:
    print(f"Processing region: {region}")
    # Stop EC2 instances
    try:
        ec2 = boto3.resource('ec2', region_name=region)
        running_instances = ec2.instances.filter(
            Filters=[{'Name': 'instance-state-name', 'Values': ['running']}]
        )

        for instance in running_instances:
            instance.stop()
            stopped_ec2.append({'Region': region, 'InstanceId': instance.id})
            print(f"Stopped EC2: {instance.id} in {region}")
    except Exception as e:
        print(f"Error processing EC2 in {region}: {str(e)}")

    # Stop RDS instances
    try:
        rds = boto3.client('rds', region_name=region)
        db_instances = rds.describe_db_instances()['DBInstances']

        for db in db_instances:
            if db['DBInstanceStatus'] == 'available':

```

```

        db_id = db['DBInstanceIdentifier']

        rds.stop_db_instance(DBInstanceIdentifier=db_id)

        stopped_rds.append({'Region': region, 'DBInstanceIdentifier': db_id})

        print(f"Stopped RDS: {db_id} in {region}")

    except Exception as e:

        print(f"Error processing RDS in {region}: {str(e)}")

# Final message
message = {
    'StoppedEC2': stopped_ec2,
    'StoppedRDS': stopped_rds
}

# Publish to SNS
try:
    sns_client.publish(
        TopicArn=sns_topic_arn,
        Subject='AWS Shutdown Automation Report',
        Message=json.dumps(message, indent=2)
    )

    print("SNS notification sent.")
except Exception as e:
    print(f"Error sending SNS notification: {str(e)}")

# Send message to SQS
try:
    sqs_client.send_message(
        QueueUrl=sqs_queue_url,
        MessageBody=json.dumps(message)
    )

```

```
)  
  
print("SQS message sent.")  
  
except Exception as e:  
    print(f"Error sending SQS message: {str(e)}")  
  
return message
```

Test & Deploy

- Paste this code into Lambda's inline editor.
- Create Ec2 vm
- Click **Deploy**, then **Test** using {}.

☰ Lambda > Functions > lambdasns

☒ Private
This event is only available in the Lambda console and to the event creator. You can configure a total of 10. [Learn more](#)

☐ Shareable
This event is available to IAM users within the same account who have permissions to access and use shareable events. [Learn more](#)

Template - *optional*

Hello World ▼

Event JSON

1	{ }
---	-----

- Launch a EC2 instance
- Check CloudWatch logs, SNS email, and SQS queue for messages.

AWS Notifications <no-reply@sns.amazonaws.com>

20/06/20

To: amit@openwriteup.com

```
{  
  "StoppedEC2": [  
    {  
      "Region": "us-east-1",  
      "InstanceId": "i-09d1c89c509fa4711"  
    }  
  ],  
  "StoppedRDS": []  
}
```

--

If you wish to stop receiving notifications from this topic, please click or visit the link below to unsubscribe:

<https://sns.us-east-1.amazonaws.com/unsubscribe.html?SubscriptionArn=arn:aws:sns:us-east-1:883308508227:lambdaec2:580c8a-4f05-b0fb-978a4e5d38f5&Endpoint=amit@openwriteup.com>

In SQS

Scroll down to the “Messages” section

You'll see sections like:

- “Send and receive messages”
- “Message retention”
- “Monitoring”
- **Click on “Send and receive messages”**

In the **“Receive messages”** section:

- Click **“Poll for messages”**
- You will see the messages currently available in the queue (in your case, 2).
- You can **click to view the message body** and metadata (like message attributes).