

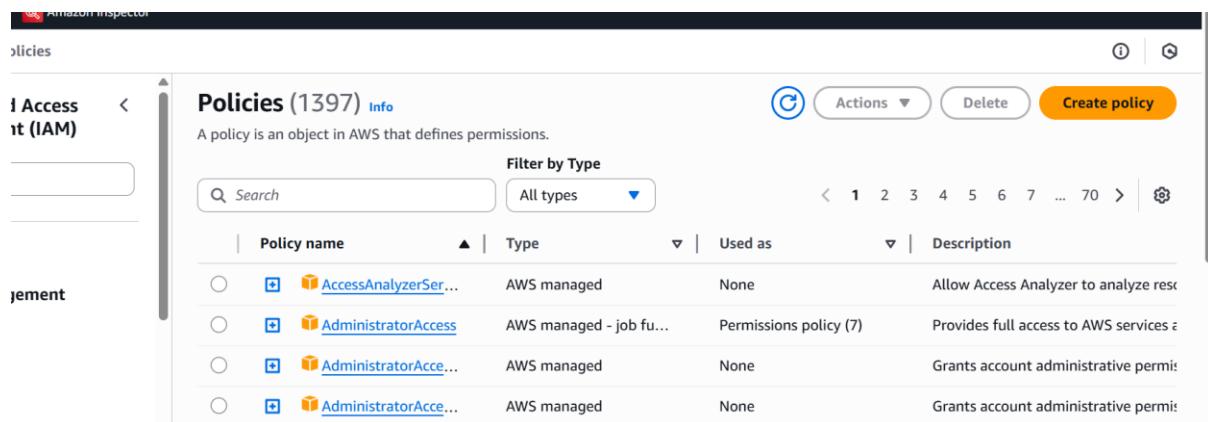
SNS and SQS

SNS and SQS with AWS Lambda

1. IAM Role Update

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

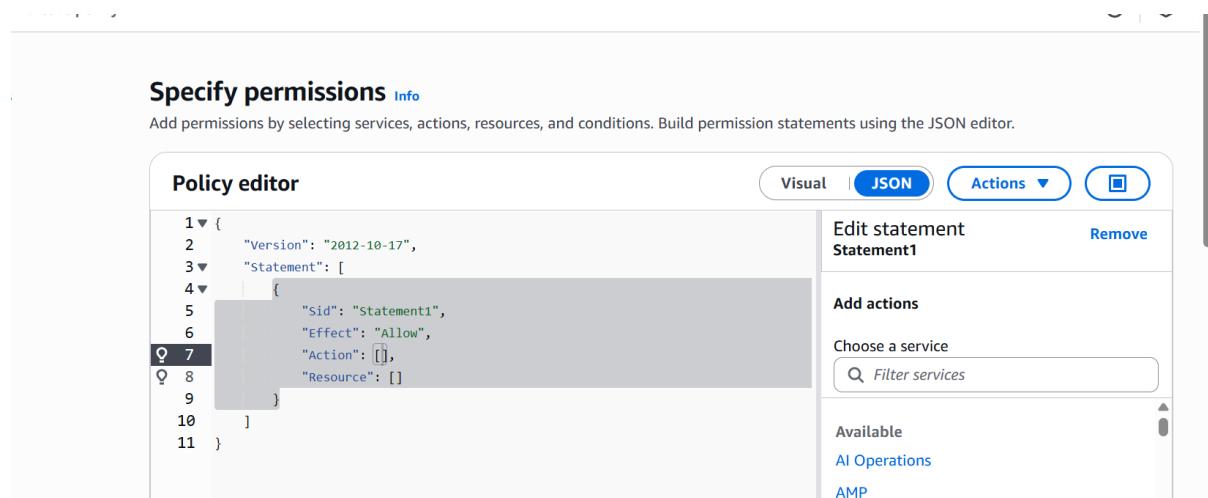
Iam → policy → Create Policy



The screenshot shows the AWS IAM Policies page. The left sidebar has 'Access & Identity (IAM)' selected. The main area title is 'Policies (1397)'. A sub-header says 'A policy is an object in AWS that defines permissions.' Below is a search bar and a 'Filter by Type' dropdown set to 'All types'. A table lists four AWS managed policies:

Policy name	Type	Used as	Description
AccessAnalyzerServiceRole	AWS managed	None	Allow Access Analyzer to analyze resources
AdministratorAccess	AWS managed - job function	Permissions policy (7)	Provides full access to AWS services
AdministratorAccess	AWS managed	None	Grants account administrative permissions
AdministratorAccess	AWS managed	None	Grants account administrative permissions

Copy and paste below json



The screenshot shows the AWS Policy Editor. The top navigation bar includes 'Visual', 'JSON' (which is selected), 'Actions', and a copy icon. The main area is titled 'Specify permissions' and contains JSON code for a policy statement:

```
1 Version: "2012-10-17",
2 Statement: [
3   {
4     Sid: "Statement1",
5     Effect: "Allow",
6     Action: [],
7     Resource: []
8   }
9 ]
10 ]
11 }
```

To the right, there's an 'Edit statement' section for 'Statement1' with a 'Remove' button. Below it is an 'Add actions' section with a 'Choose a service' dropdown and a 'Filter services' search bar. Underneath are sections for 'Available', 'AI Operations', and 'AMP'.

```
{  
    "Version": "2012-10-17",  
    "Statement": [  
        {  
            "Sid": "VisualEditor0",  
            "Effect": "Allow",  
            "Action": [  
                "ec2:DescribeInstances",  
                "ec2:DescribeRegions",  
                "ec2:StopInstances",  
                "rds:DescribeDBInstances",  
                "rds:StopDBInstance",  
                "sns:Publish",  
                "sns: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

ec2s



All types

3 matches

< 1 >

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"/>		ec2snsqs	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

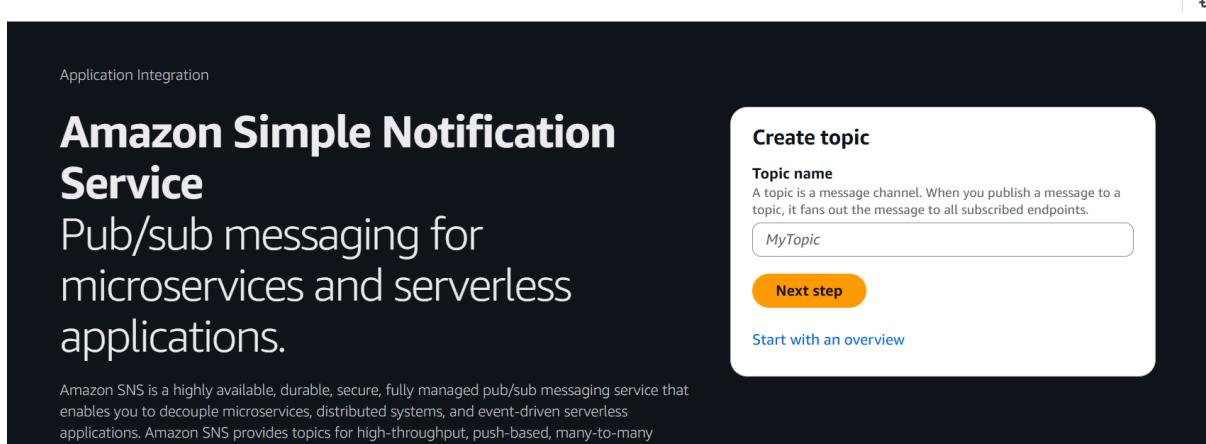
Edit

Trust policy

Create SNS

The screenshot shows the AWS Services page with the search bar set to "sns". On the left, there's a sidebar with links like Services, Features, Resources, Documentation, Knowledge articles, Marketplace, Blog posts, Events, and Tutorials. The main area is titled "Services" and lists three services: "Simple Notification Service" (selected), "Route 53 Resolver", and "Route 53". Each service has a brief description and a star icon. At the bottom, there's a "Features" section with a "Show more" link.

- Go to SNS → Topics → Create topic



The screenshot shows the Amazon Simple Notification Service (SNS) landing page. At the top left, it says "Application Integration". The main title is "Amazon Simple Notification Service" with the subtitle "Pub/sub messaging for microservices and serverless applications." Below the title, a brief description states: "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". To the right, there's a "Create topic" section with a "Topic name" input field containing "MyTopic", a "Next step" button, and a link "Start with an overview".

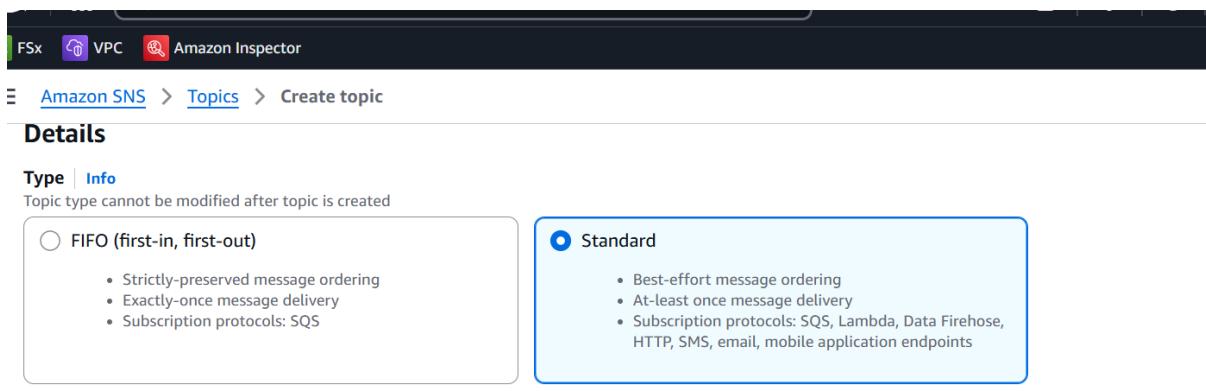
Create topic

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



The screenshot shows the "Create topic" configuration screen. At the top, there are navigation links: FSx, VPC, Amazon Inspector, and a breadcrumb trail: Amazon SNS > Topics > Create topic. Below that is a "Details" section with a "Type" dropdown set to "Standard". Two options are shown: "FIFO (first-in, first-out)" and "Standard". The "Standard" option is selected and includes a description: "Best-effort message ordering", "At-least once message delivery", and "Subscription protocols: SQS, Lambda, Data Firehose, HTTP, SMS, email, mobile application endpoints". Further down, there's a "Name" input field containing "lambdaec2", a note about character limits, and a "Display name - optional" input field containing "My Topic".

Name

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

Display name - optional

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 them.

Optional Info

ing 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).

Amazon 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

The screenshot shows the AWS SQS home page. At the top, there's a search bar with 'sq' and a navigation bar with icons for file, settings, and help. Below the search bar, a sidebar on the left lists 'Services', 'Features', 'Resources New', 'Documentation', 'Knowledge articles', 'Marketplace', 'Blog posts', 'Events', and 'Tutorials'. The main content area has sections for 'Services' and 'Features'. Under 'Services', there are cards for 'Simple Queue Service' (SQS Managed Message Queues), 'Service Quotas' (View and manage your AWS service quotas from a central location), and 'Simple Notification Service' (SNS managed message topics for Pub/Sub). Each card has a star icon for favoriting.

- Choose Standard queue

The screenshot shows the Amazon SQS landing page. It features a large hero section with the text 'Amazon SQS' and 'Simple message queuing service'. Below this, a paragraph explains that SQS is for high-throughput, system-to-system messaging, suitable for both heavyweight processes and batch work. To the right, a 'Get started' box contains instructions on how to use SQS by creating a queue, sending a message, and processing it. A prominent orange 'Create queue' button is at the bottom of this box. The top navigation bar includes links for FSx, VPC, and Amazon Inspector.

The screenshot shows the 'Create queue' wizard in the AWS Management Console. The current step is 'Details'. It starts with a 'Type' section where users choose the queue type. Two options are shown: 'Standard Info' (selected) and 'FIFO Info'. 'Standard Info' describes at-least-once delivery and non-preserved message ordering, with sub-points for best-effort ordering. 'FIFO Info' describes first-in-first-out delivery and preserved message ordering, with sub-points for exactly-once processing. Below this is a note: 'You can't change the queue type after you create a queue.' The next section is 'Name', where the user has typed 'mylambda'. A note below says names must be case-sensitive and up to 80 characters. The final section is 'Configuration'.

- Copy the Queue URL for use in the code.

☰ Amazon SQS > Queues > mylambda

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

mymambda

Edit Delete Purge Send and receive messages Start D

Details Info

Name mymambda	Type Standard	ARN arn:aws:sqs:us-east-1:883308508227:mylam
Encryption Amazon SQS key (SSE-SQS)	URL https://sqs.us-east-1.amazonaws.com/883308508227/mymambda	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 function.

Basic information

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

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.
 ▼ (C)

Architecture | Info
Choose the instruction set architecture you want for your function code.
—

Execution role
Choose 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.
 ▼ (C)
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,
0         Subject='AWS Shutdown Automation Report',
1         Message=json.dumps(message, indent=2)
2     )
3
4
5     # SQS Message
6     sqs_client.send_message(
7         QueueUrl=https://sqs.us-east-1.amazonaws.com/883308508227/mylambda,
8         MessageBody=json.dumps(message)
9     )
0
1
return message
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.")

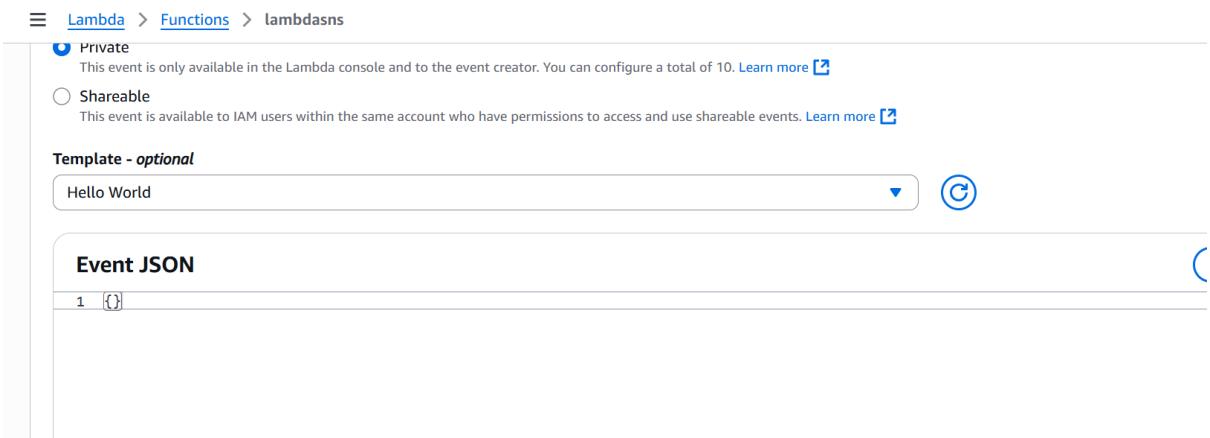
except Exception as e:
    print(f"Error sending SQS message: {str(e)}")

return message

```

Test & Deploy

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



- 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:lambdaec2:58:0c8a-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).