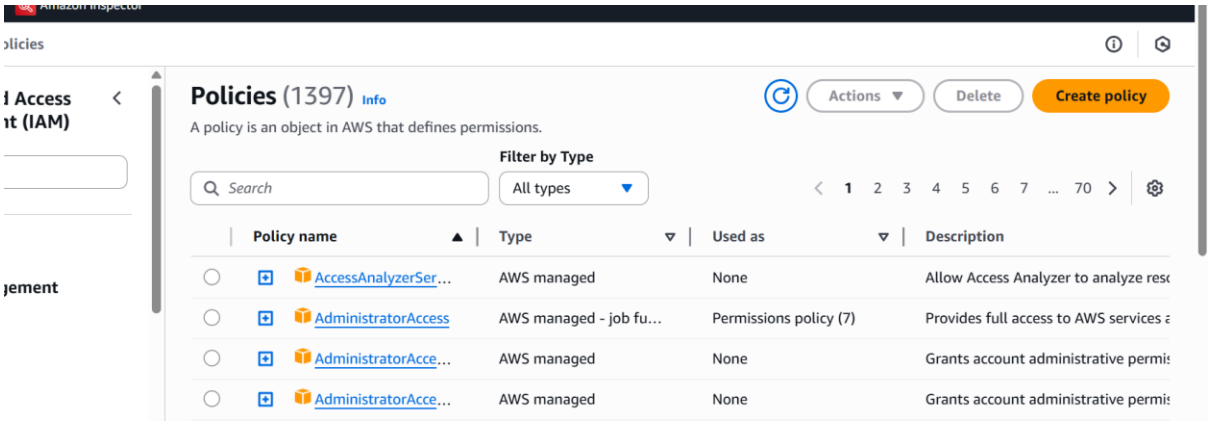
SNS and SQS

SNS and SQS with AWS Lambda

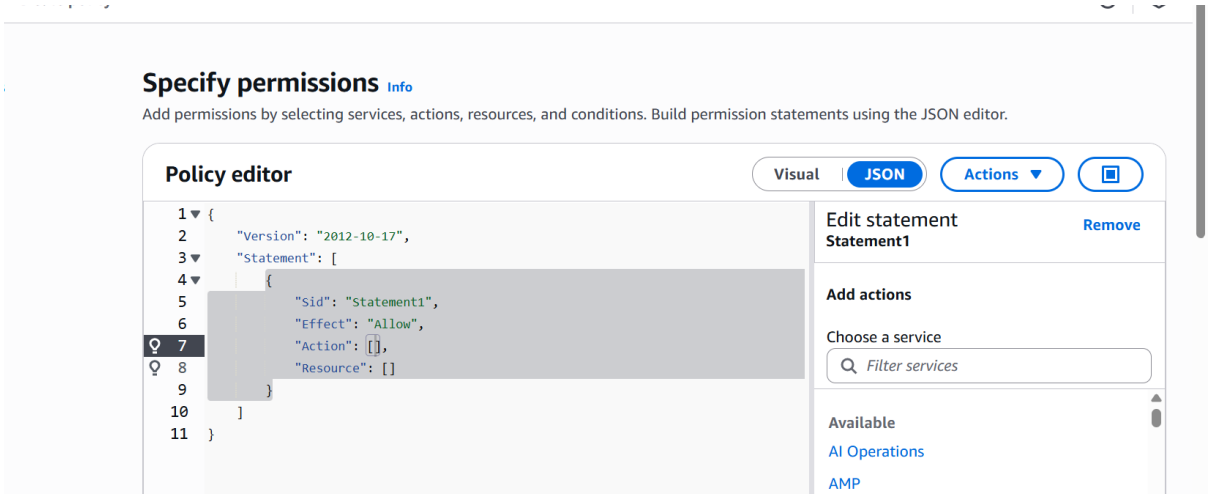
1. IAM Role Update

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

Iam →policy→Create Policy

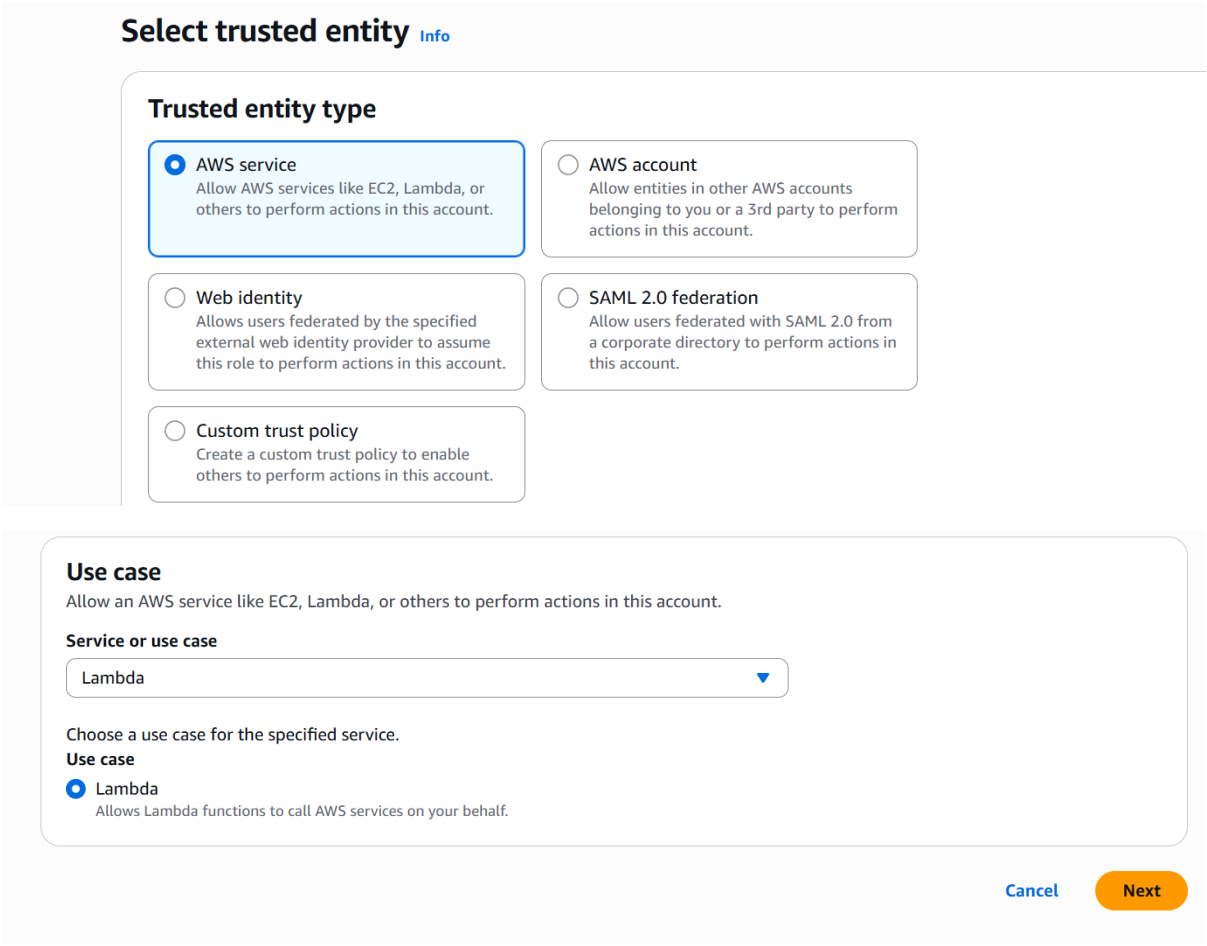


Copy and paste below json

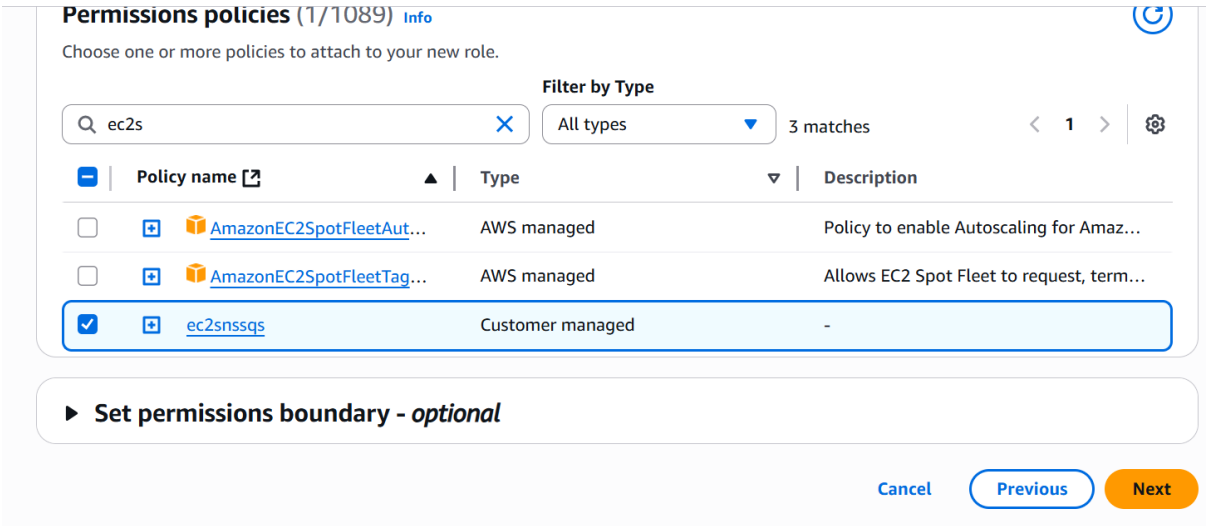


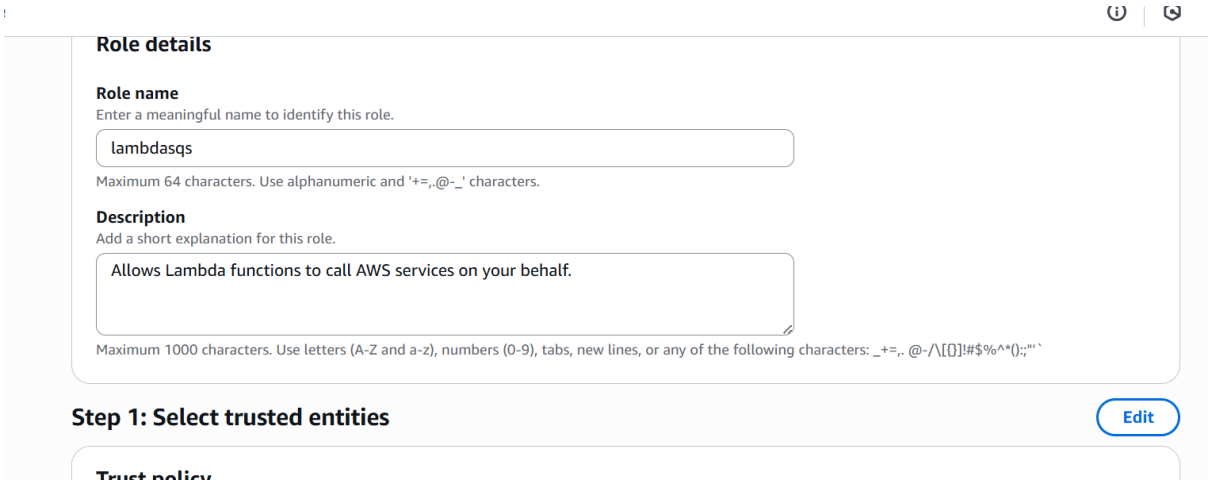
```
{
	"Version": "2012-10-17",
	"Statement": [
		{
			"Sid": "VisualEditor0",
			"Effect": "Allow",
			"Action": [
				"ec2:DescribeInstances",
				"ec2:DescribeRegions",
				"ec2:StopInstances",
				"rds:DescribeDBInstances",
				"rds:StopDBInstance",
				"sns:Publish",
				"sqs:SendMessage"
			],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

Create Role

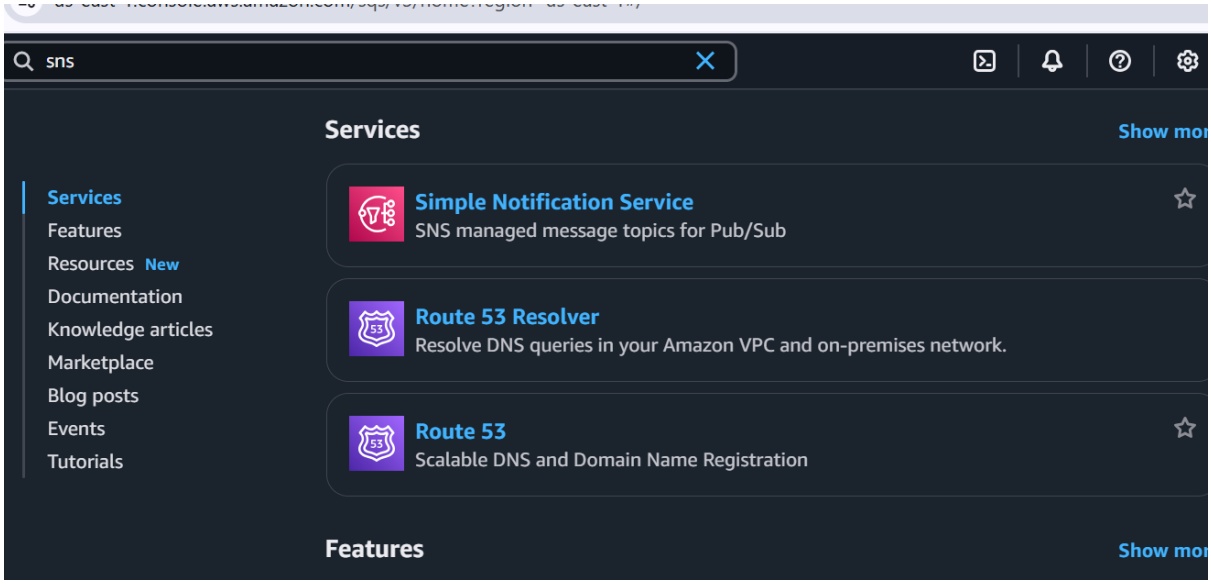


Select the policy

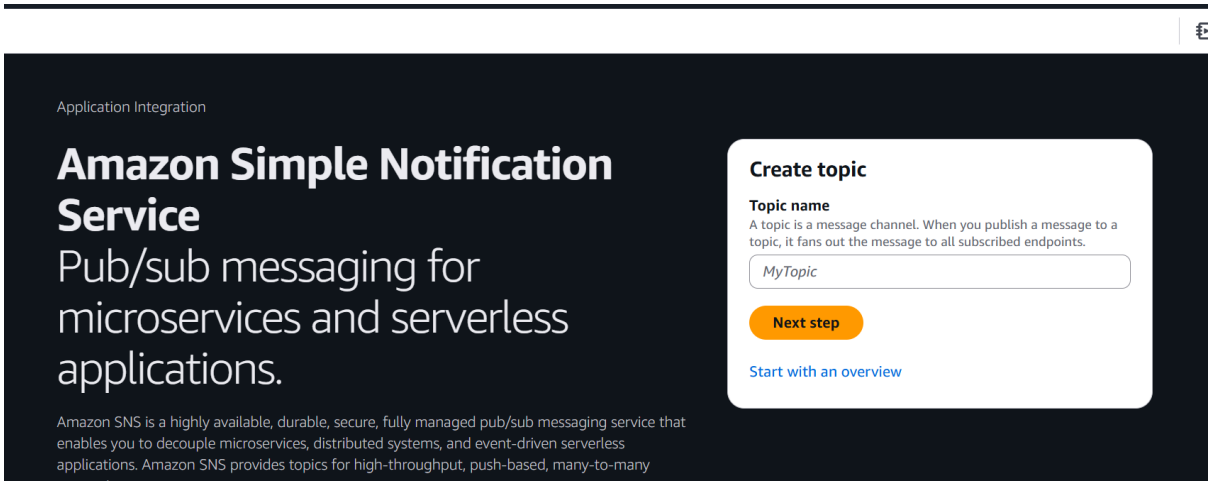


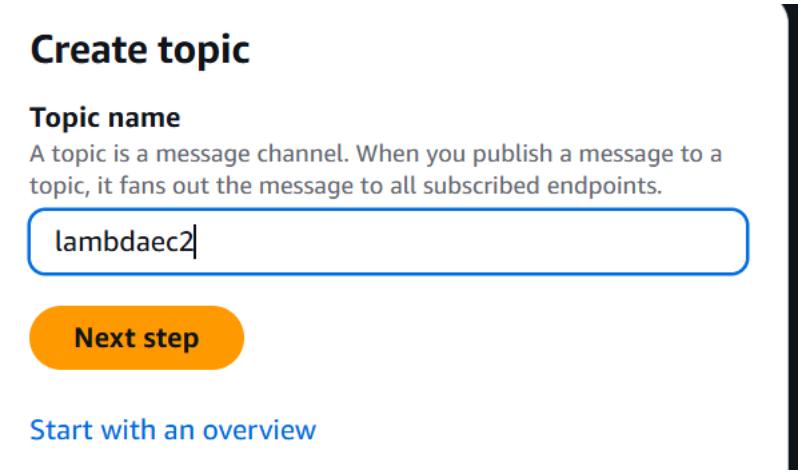


Create SNS

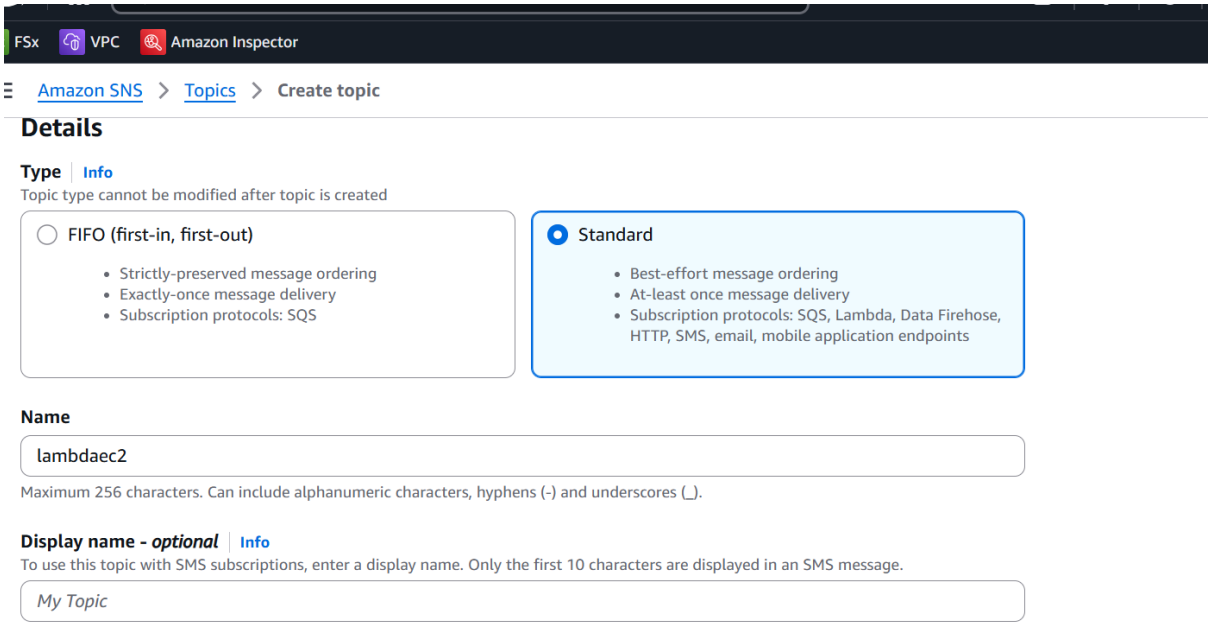


- Go to SNS → Topics → Create topic

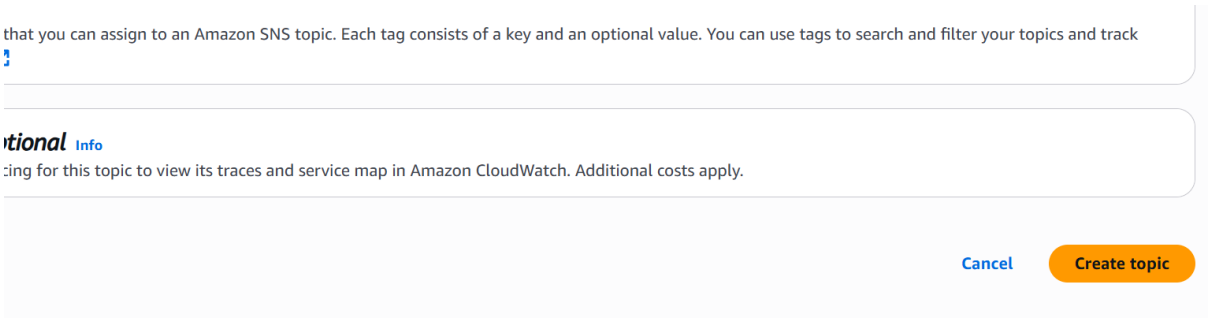




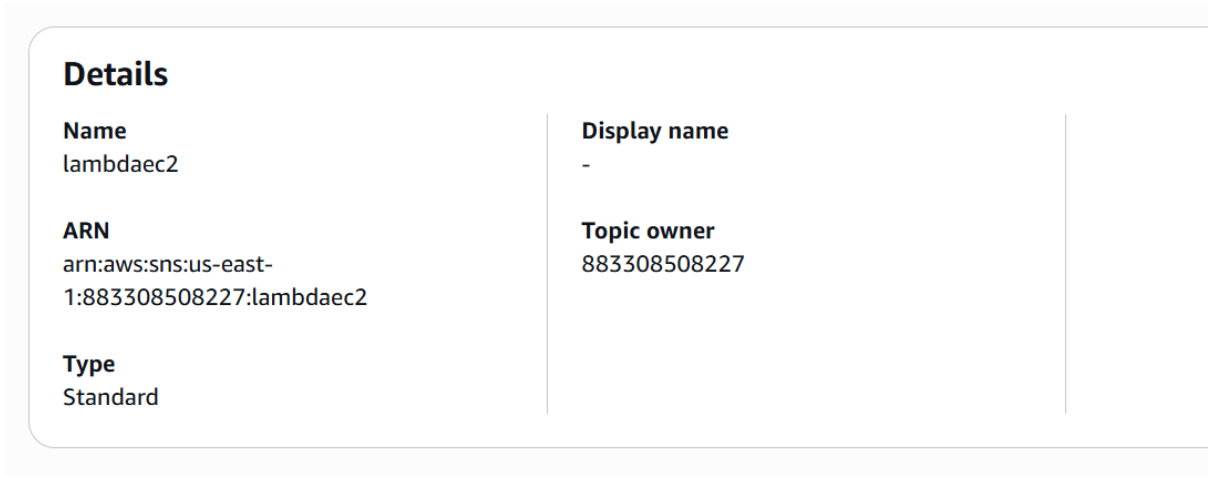
- Choose **Standard queue**



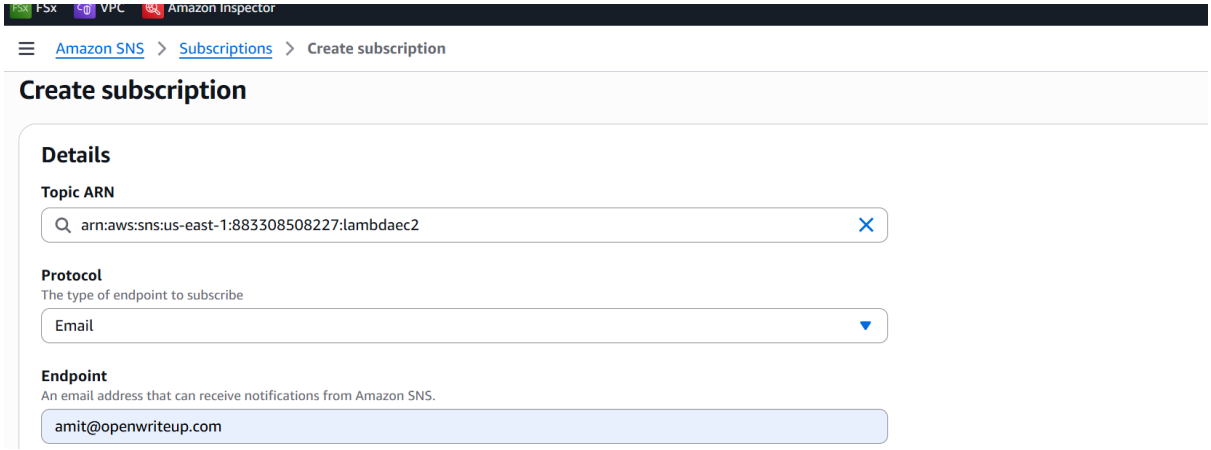
Create topic



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

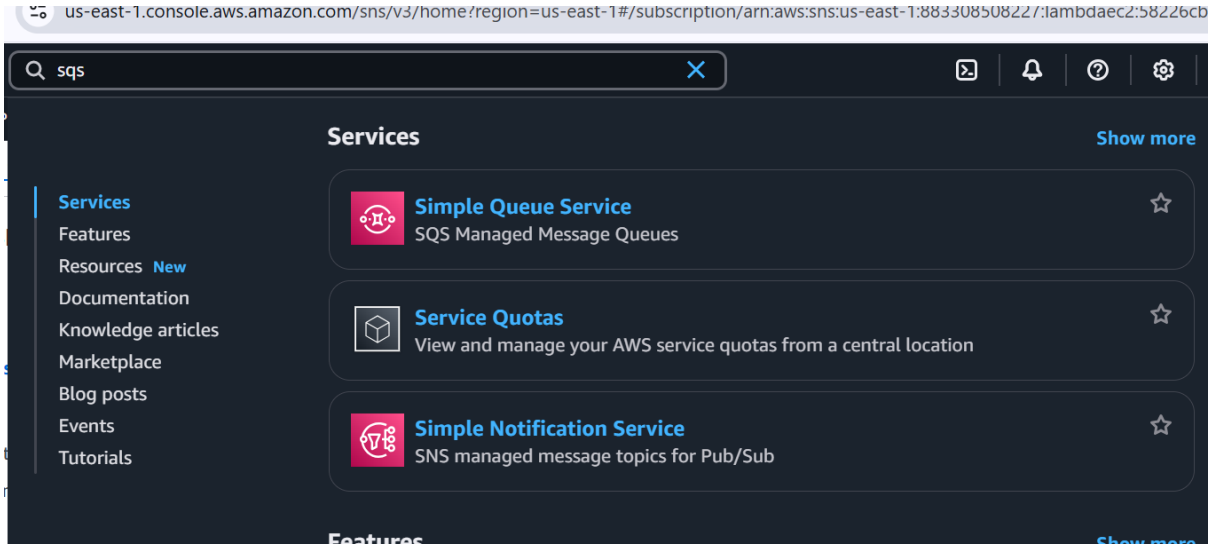


- Add a subscription (e.g., Email).

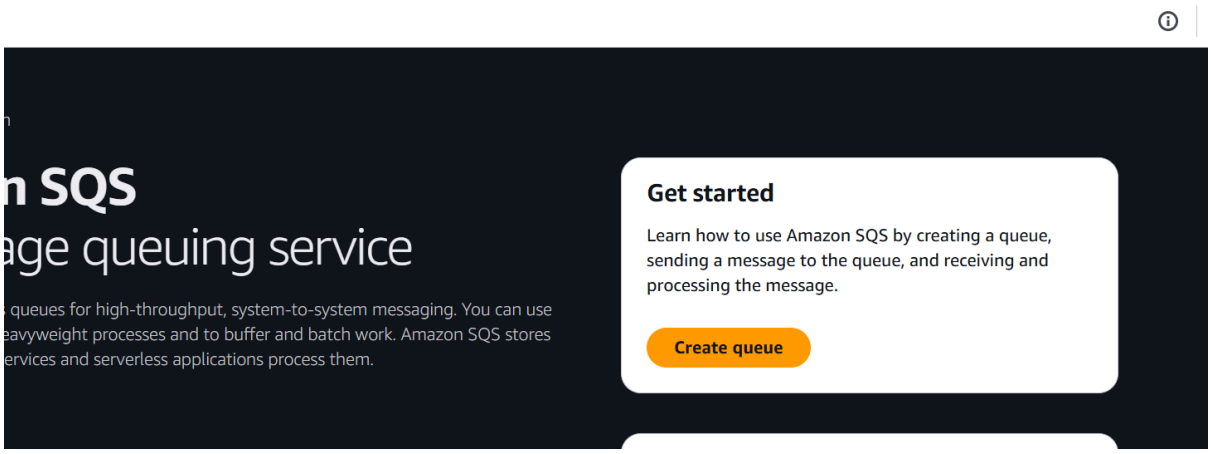


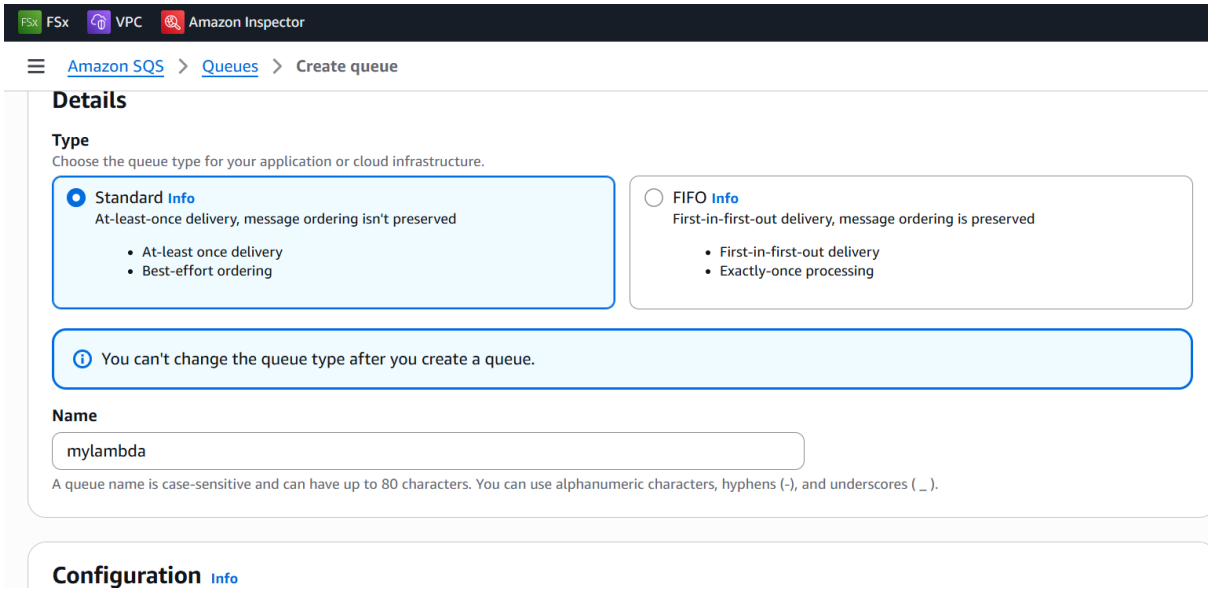
SQS Setup

- Go to SQS → Create Queue

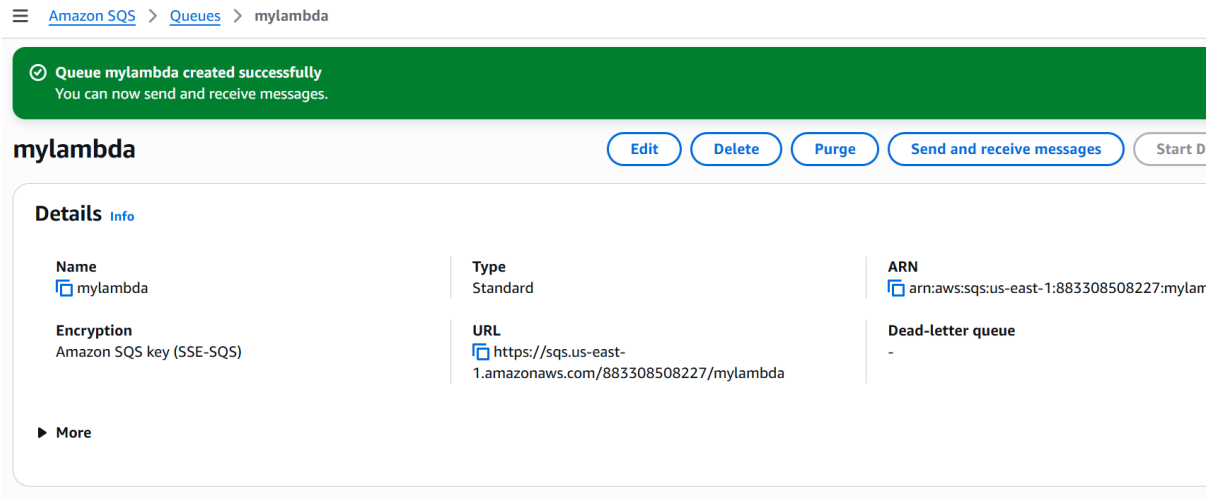


- Choose **Standard queue**

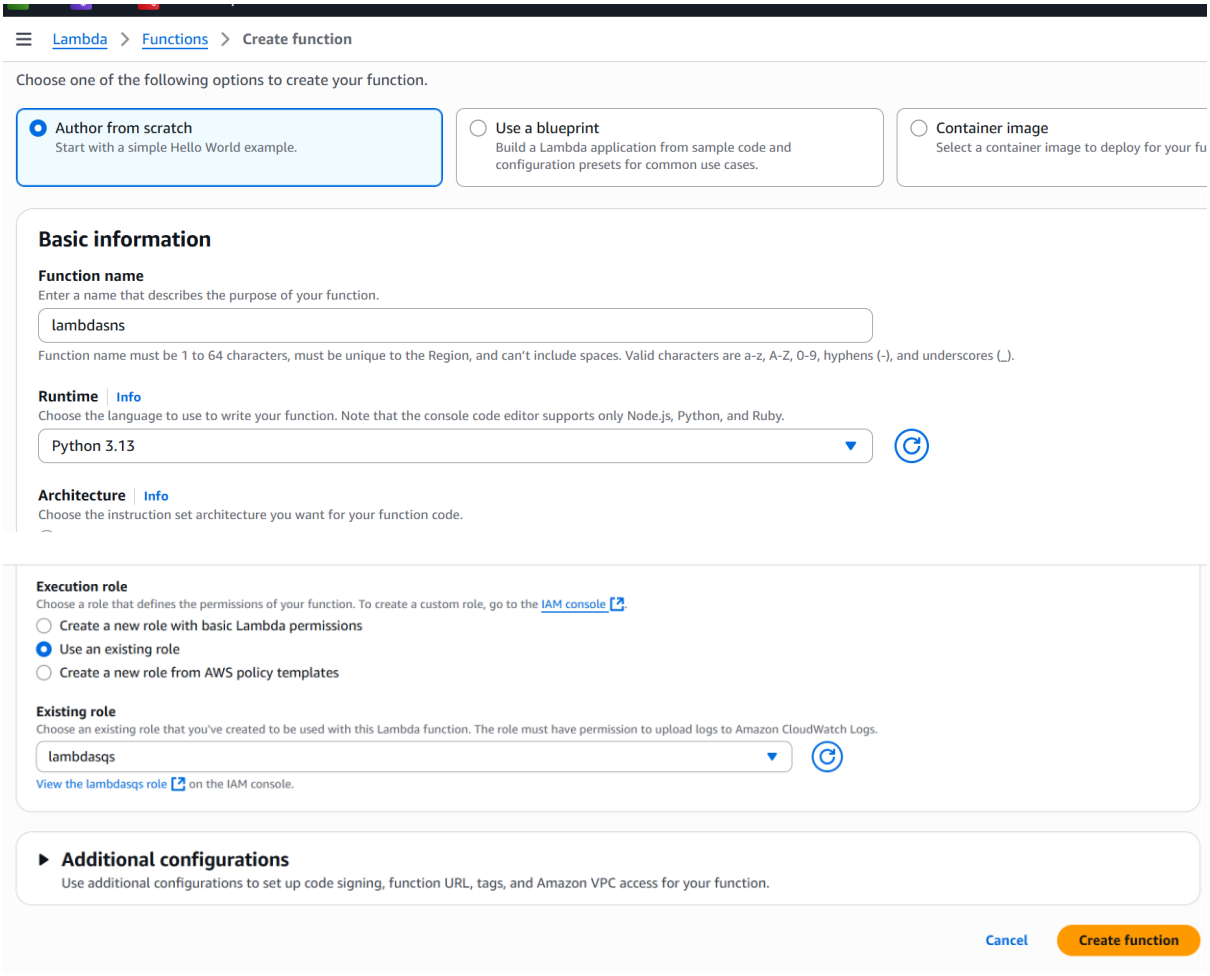




- Copy the **Queue URL** for use in the code.



Lambda



Modify below code

- Provide SNS arn
- SQS url

lambda_function.py

```
def lambda_handler(event, context):
    }

    # SNS Notification
    sns_client.publish(
        TopicArn=arn:aws:sns:us-east-1:883308508227:lambdaec2,
        Subject='AWS Shutdown Automation Report',
        Message=json.dumps(message, indent=2)
    )

    # SQS Message
    sqs_client.send_message(
        QueueUrl=https://sqs.us-east-1.amazonaws.com/883308508227/mylambda,
        MessageBody=json.dumps(message)
    )

    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
```

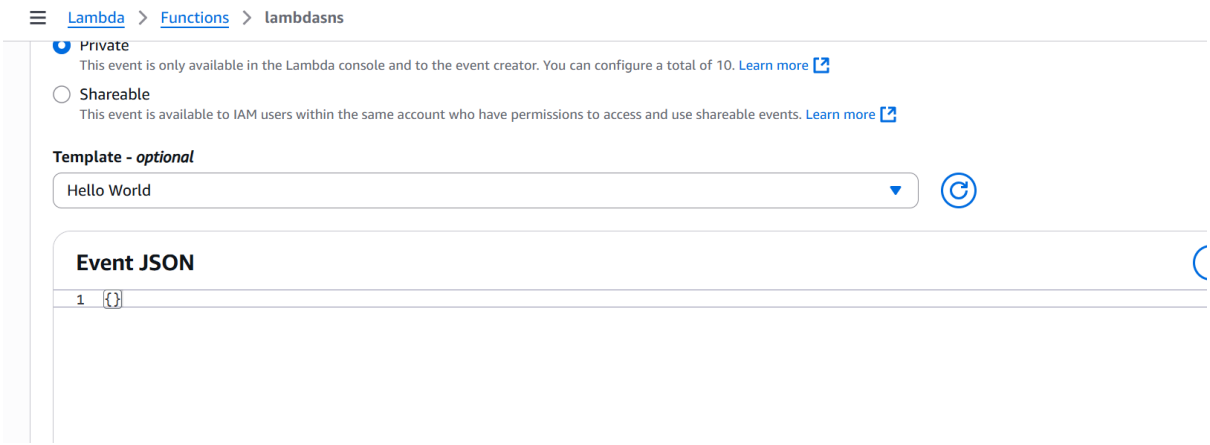
```
try:
    regions = [r['RegionName'] for r in ec2_client.describe_regions()['Regions']]
except Exception as e:
    print(f"Error fetching regions: {e}")
    return {"error": "Unable to fetch AWS regions"}

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)
```

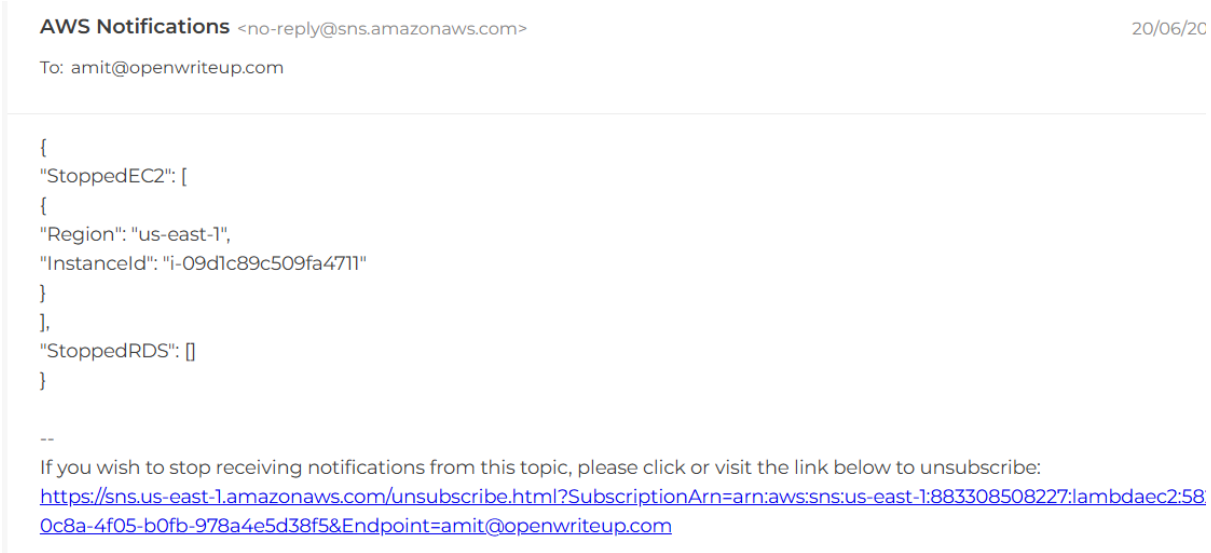
```
    )
    print("SQS message sent.")
except Exception as e:
    print(f"Error sending SQS message: {str(e)}")


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.



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).