Contents

IAM > Roles > salesAnalysisReportRole > Edit policy

Modify permissions in AmazonSNSFullAccess

```
{
	"Version": "2012-10-17",
	"Statement": [
		{
			"Action": [
				"sns:*"
			],
			"Resource": "*",
			"Effect": "Allow"
		}
	]
}
```

Modify permissions in AmazonSSMReadOnlyAccess

```
{
	"Version": "2012-10-17",
```

```
	"Statement": [
		{
			"Action": [
				"ssm:Describe*",
				"ssm:Get*",
				"ssm:List*"
			],
			"Resource": "*",
			"Effect": "Allow"
		}
	]
}
```

Modify permissions in AWSLambdaBasicRunRole

```
{
	"Version": "2012-10-17",
	"Statement": [
		{
			"Action": [
				"logs:CreateLogGroup",
				"logs:CreateLogStream",
				"logs:PutLogEvents"
			],
			"Resource": "*",
			"Effect": "Allow"
		}
	]
}
```

Modify permissions in AWSLambdaRole

```
{
	"Version": "2012-10-17",
	"Statement": [
		{
			"Action": [
				"lambda:InvokeFunction"
```

```
                ],
                "Resource": [
                        "*"
                ],
                "Effect": "Allow"
        }
    ]
}
```

IAM > Roles > salesAnalysisReportDERole > Edit policy

Modify permissions in AWSLambdaBasicRunRole

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Action": [
                "logs:CreateLogGroup",
                "logs:CreateLogStream",
                "logs:PutLogEvents"
            ],
            "Resource": "*",
            "Effect": "Allow"
        }
    ]
}
```

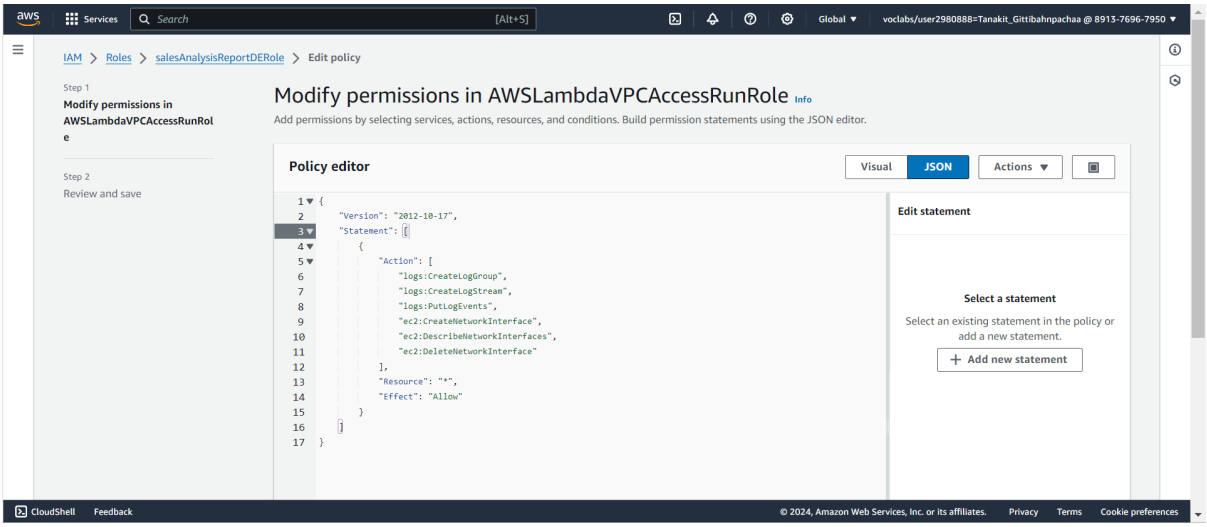
Modify permissions in AWSLambdaVPCAccessRunRole

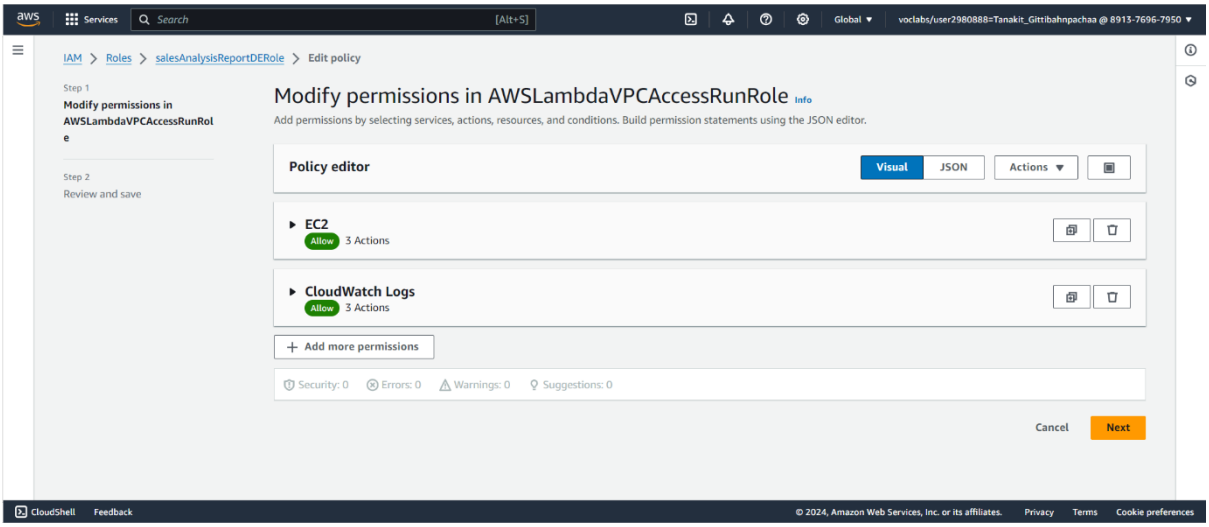
```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Action": [
                "logs:CreateLogGroup",
                "logs:CreateLogStream",
                "logs:PutLogEvents",
```

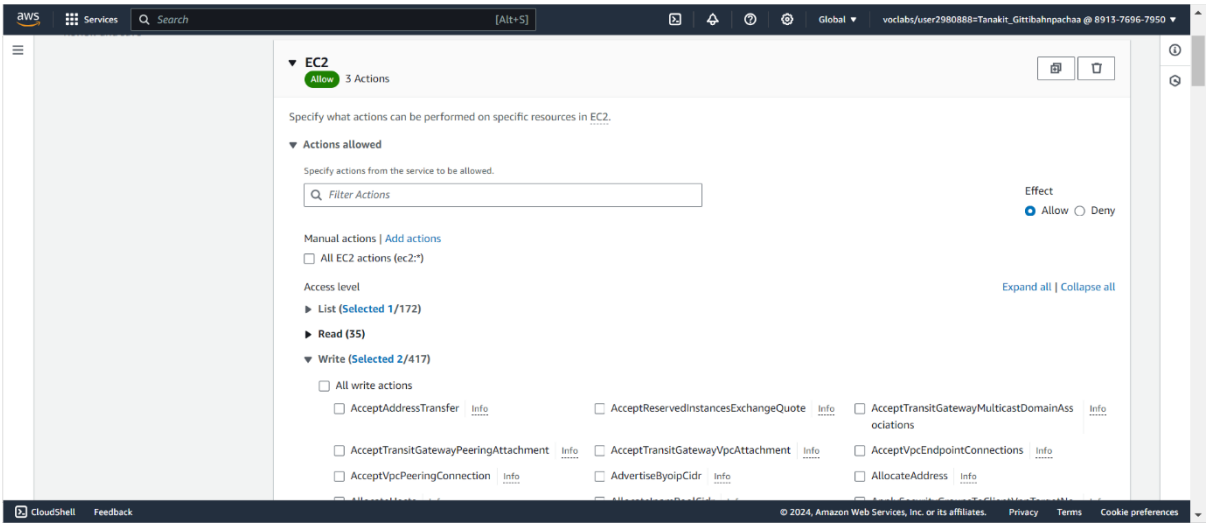
```
                "ec2:CreateNetworkInterface",
                "ec2:DescribeNetworkInterfaces",
                "ec2:DeleteNetworkInterface"
            ],
            "Resource": "*",
            "Effect": "Allow"
        }
    ]
}
```

Config policy in role

สามารถตั้งค่า Action ที่จะกระทำต่อ Resources ได้โดยการเขียนโค้ด (JSON) หรือ config (Visual) ก็ได้ เนื่องจาก IAM Policies เป็นประเภท Customer inline







AWS access levels

AWS ได้กำหนดระดับการเข้าถึงสำหรับการดำเนินการในบริการต่างๆ ดังนี้:

List: สิทธิในการดูรายการทรัพยากรภายในบริการเพื่อตรวจสอบว่าออบเจกต์นั้นมีอยู่หรือไม่ การดำเนินการในระดับนี้สามารถดูรายการออบเจกต์ได้ แต่จะไม่สามารถดูเนื้อหาของทรัพยากรนั้น เช่น คำสั่ง ListBucket ใน Amazon S3 มีระดับการเข้าถึง คือ List

Read: สิทธิในการอ่านแต่ไม่สามารถแก้ไขเนื้อหาและแอททริบิวต์ของทรัพยากรในบริการนั้นได้ เช่น คำสั่ง GetObject และ GetBucketLocation ใน Amazon S3 มีระดับการเข้าถึง คือ Read

Tagging: สิทธิในการดำเนินการที่เปลี่ยนแปลงสถานะของแท็กทรัพยากรเท่านั้น เช่น คำสั่ง TagRole และ UntagRole ใน IAM มีระดับการเข้าถึง คือ Tagging เนื่องจากอนุญาตให้ทำแท็กหรือถอดแท็กบทบาทได้

เท่านั้น อย่างไรก็ตาม คำสั่ง CreateRole อนุญาตให้ทำแท็กบทบาทได้ในขณะสร้างบทบาทนั้น เนื่องจากคำสั่งนี้ไม่ได้เพียงแค่เพิ่มแท็ก จึงมีระดับการเข้าถึง คือ Write

Write: สิทธิในการสร้าง ลบ หรือแก้ไขทรัพยากรในบริการนั้น เช่น คำสั่ง CreateBucket, DeleteBucket และ PutObject ใน Amazon S3 มีระดับการเข้าถึง คือ Write คำสั่งระดับ Write อาจอนุญาตให้แก้ไขแท็กทรัพยากรได้ด้วย อย่างไรก็ตาม คำสั่งที่อนุญาตให้เปลี่ยนแปลงเฉพาะแท็กเท่านั้นจะมีระดับการเข้าถึง คือ Tagging

Permissions management: สิทธิในการให้หรือแก้ไขสิทธิ์การเข้าถึงทรัพยากรในบริการนั้น เช่น ส่วนใหญ่ของคำสั่ง IAM และ AWS Organizations รวมถึงคำสั่งอย่าง PutBucketPolicy และ DeleteBucketPolicy ใน Amazon S3 ล้วนมีระดับการเข้าถึง คือ Permissions management

Code source salesAnalysisReportDataExtractor.py

```
import boto3
import pymysql
import sys

def lambda_handler(event, context):

    # Retrieve the database connection information from the event input parameter.

    dbUrl = event['dbUrl']
    dbName = event['dbName']
    dbUser = event['dbUser']
    dbPassword = event['dbPassword']

    # Establish a connection to the Cafe database, and set the cursor to return
results as a Python dictionary.

    try:
        conn = pymysql.connect(host=dbUrl, user=dbUser, passwd=dbPassword,
db=dbName, cursorclass=pymysql.cursors.DictCursor)

    except pymysql.Error as e:
```

```
        print('ERROR: Failed to connect to the Cafe database.')
        print('Error Details: %d %s' % (e.args[0], e.args[1]))
        sys.exit()

    # Execute the query to generate the daily sales analysis result set.

    with conn.cursor() as cur:
        cur.execute("SELECT  c.product_group_number, c.product_group_name,
a.product_id, b.product_name, CAST(sum(a.quantity) AS int) as quantity FROM
order_item a, product b, product_group c WHERE b.id = a.product_id AND
c.product_group_number = b.product_group GROUP BY c.product_group_number,
a.product_id")
        result = cur.fetchall()

    # Close the connection.

    conn.close()

    # Return the result set.

    return {'statusCode': 200, 'body': result}
```

Source code: salesAnalysisReport.py

```
import boto3
import os
import json
import io
import datetime

def setTabsFor(productName):

    # Determine the required number of tabs between Item Name and Quantity based on
the item name's length.

    nameLength = len(productName)
```

```
    if nameLength < 20:
        tabs='\t\t\t'
    elif 20 <= nameLength <= 37:
        tabs = '\t\t'
    else:
        tabs = '\t'

    return tabs

def lambda_handler(event, context):

    # Retrieve the topic ARN and the region where the lambda function is running
from the environment variables.

    TOPIC_ARN = os.environ['topicARN'] ## Line 26 ##
    FUNCTION_REGION = os.environ['AWS_REGION']

    # Extract the topic region from the topic ARN.

    arnParts = TOPIC_ARN.split(':')
    TOPIC_REGION = arnParts[3]

    # Get the database connection information from the Systems Manager Parameter
Store.

    # Create an SSM client.

    ssmClient = boto3.client('ssm', region_name=FUNCTION_REGION)

    # Retrieve the database URL and credentials.

    parm = ssmClient.get_parameter(Name='/cafe/dbUrl')
    dbUrl = parm['Parameter']['Value']

    parm = ssmClient.get_parameter(Name='/cafe/dbName')
    dbName = parm['Parameter']['Value']
```

```
    parm = ssmClient.get_parameter(Name='/cafe/dbUser')

    dbUser = parm['Parameter']['Value']


    parm = ssmClient.get_parameter(Name='/cafe/dbPassword')

    dbPassword = parm['Parameter']['Value']


    # Create a lambda client and invoke the lambda function to extract the daily
sales analysis report data from the database.


    lambdaClient = boto3.client('lambda', regi
```