**OVERVIEW:**

* Whenever the associated account publishes a SNS message then lambda will get triggered and read the message. It will find the important information like email, phone, address and store it in MySQL database.
* I have created a SNS topic and gave access to publish message from mentioned account number (875938798788) in the assignment.
* I wrote a python script (lambda.py) which will trigger the lambda function upon publishing message to SNS.
* I have created MySQL RDS from AWS console where we can store sensitive information.
* I have created two tables in MySQL DB one is for storing sensitive information like email, phone, address and another to save the execution status and ingestion time in audit MYSQL table on assignment database over RDS MySQL.  
    
  Q1) An SNS message is posted from one of our AWS accounts (AWS account ID **875938798788**). The SNS topic while remaining private should be configured to accept messages from our account.

A) I have created a SNS topic with name “assignment” and gave access to **875938798788** with below access policy: -  
{

"Version": "2008-10-17",

"Id": "\_\_default\_policy\_ID",

"Statement": [

{

"Sid": "\_\_default\_statement\_ID",

"Effect": "Allow",

"Principal": {

"AWS": "\*"

},

"Action": [

"SNS:Publish",

"SNS:RemovePermission",

"SNS:SetTopicAttributes",

"SNS:DeleteTopic",

"SNS:ListSubscriptionsByTopic",

"SNS:GetTopicAttributes",

"SNS:AddPermission",

"SNS:Subscribe"

],

"Resource": "arn:aws:sns:us-east-1:306374858684:assignment",

"Condition": {

"StringEquals": {

"AWS:SourceOwner": "306374858684"

}

}

},

{

"Sid": "\_\_console\_pub\_0",

"Effect": "Allow",

"Principal": {

"AWS": "arn:aws:iam::875938798788:root"

},

"Action": "SNS:Publish",

"Resource": "arn:aws:sns:us-east-1:306374858684:assignment"

},

{

"Sid": "\_\_console\_sub\_0",

"Effect": "Allow",

"Principal": {

"AWS": "\*"

},

"Action": "SNS:Subscribe",

"Resource": "arn:aws:sns:us-east-1:306374858684:assignment"

}  
Screenshot of created SNS topic:-  
Graphical user interface, application

Description automatically generated

Q2) The message content will be delimited by white space and could contain garbage text up to the message limits. In amongst the garbage, you **may** find

* 1. an email address
  2. a phone number
  3. a postal code
  4. all of the above

A) I have created below python script to be used by Lambda function upon arriving any SNS message on “assignment” topic. It will find the required information like phone,email,address and I have also created MySQL database from aws console where we can store the important information and one record for each message.  
  
import json

import re  
import sys  
message = event['Records'][0]['Sns']['Message']  
message\_str = json.dumps(message)  
email\_id = re.findall(r"[A-Za-z0-9.\_%+-]+"

r"@[A-Za-z0-9.-]+"

r"\.[A-Za-z]{2,4}"

r"\.[A-Za-z]{2,4}", message\_str)

phone\_number = re.findall(r'\b\d{10}\b', message\_str, flags=0)

postal\_code = re.findall(r'\b\d{5}\b', message\_str, flags=0)

# printing the list output

if email\_id:

* 1. print(str(email\_id) + " " + str(phone\_number) + " " + str(postal\_code))

**Connection from lambda to RDS:**  
I have connected lambda function with RDS by using custom pymsql level (ARN: arn:aws:lambda:us-east-1:770693421928:layer:Klayers-python38-PyMySQL:4) at lambda and then I have imported pymysql library in lambda function.  
  
**Meanwhile in production environment:**we can create a custom lambda layer by running, packaging and uploading the pip installer module in zip format to s3.   
  
**Connection to RDS MySQL**  
  
Below is code snippet written to make connection with RDS MySQL, create table and store the important information in it.  
  
import pymysql  
import sys  
import logging  
#rds settings

rds\_host = "database-1.cfvieomujhur.us-east-1.rds.amazonaws.com"

name = "admin"

password = "admin2021"

db\_name = "assignment"

port = 3306

logger = logging.getLogger()

logger.setLevel(logging.INFO)

try:

conn = pymysql.connect(host=rds\_host, user=name, passwd=password, db=db\_name, connect\_timeout=5)

except pymysql.MySQLError as e:

logger.error("ERROR: Unexpected error: Could not connect to MySQL instance.")

logger.error(e)

sys.exit()

logger.info("SUCCESS: Connection to RDS MySQL instance succeeded")  
 with conn.cursor() as cur:

try:

cur.execute("create table if not exists details ( phone int NOT NULL, email varchar(255) NOT NULL, postal int NOT NULL, created\_ts datetime)")

cur.execute('insert into details (phone, email, postal, created\_ts) values(%s, %s, %s, %s)', (phone\_number, email, postal, datetime.now()))

conn.commit()

print("Record inserted successfully into details table")

QUERY\_STATUS = "Successful"

except mysql.connector.Error as error:

print("Failed to insert into MySQL table {}".format(error))

QUERY\_STATUS = "Failed"

Q3) A record of the ingested message, when it was ingested, and the execution status is kept somewhere in the system as an audit trail  
  
A) Below code helps to save the execution status and ingestion time in audit MYSQL table on assignment database over RDS MySQL.  
  
cur.execute("SELECT MAX(created\_ts) FROM details2")

for QUERY\_EXECUTION\_TIMESTAMP in cur:

print(QUERY\_EXECUTION\_TIMESTAMP)

cur.execute("create table if not exists audit ( created\_ts datetime NOT NULL, query\_status varchar(255) NOT NULL)")

cur.execute('insert into audit (created\_ts, query\_status) values(%s, %s)', (QUERY\_EXECUTION\_TIMESTAMP, QUERY\_STATUS))

conn.commit()