Objective

We want to understand your experience in developing, deploying, operating, and maintaining software/infrastructure capabilities within AWS, which will best represent how you normally operate in a working environment.  To demonstrate this, we would like you to showcase a fully functional system residing within AWS that only uses free tier services (and limits) of your choice.  The system shall allow a human to trigger a dataflow as follows:

1. 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.

-🡪 I have created a SNS topic with name “assignment” 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

1. 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

🡪 I have created below python script to be used by Lambda function upon arriving any SNS message on “assignment” topic.  
  
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:

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

The important information is then stored in a database one record for each message.   
-🡪 Firstly I have connected lambda function with RDS by putting 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.  
In production environment we create a custom lambda layer by running, packaging and uploading the pip installer module in zip format to s3.   
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"

1. A record of the ingested message, when it was ingested, and the execution status is kept somewhere in the system as an audit trail  
   -🡪 I have written below code 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()

1. (BONUS) Put together metrics, monitoring, and alarm solution to detect an operational anomaly. Define, present, and prepare to demonstrate the anomaly use case

Other technical requirements are as follows

1. (BONUS) Showcasing how your assignment is be defined “in code” using your IaC tool of choice (e.g., Cloudformation, Terraform, etc)
2. (BONUS) Showcasing how your assignment goes through a CICD process

Be prepared to demonstrate the above functionality for 100s messages posted at the same time.

Please be ready to discuss how this solution may be different in the enterprise environment where you are not limited to the AWS free tier services and the service is subject to production workload.  Be prepared to answer cost-related questions.

Logistics:

* Please have all your code, tests, scripts, docs (anything you feel is required to support follow on questions) available in a git-based repository for your submission
* Please have this exercise in a working state such that it can be demo-ed
* Please timebox your effort to a maximum of 4 hours for this exercise

Additional Information:

The following are not mandatory requirements for this take home assignment, but it’s mainly for your information to give you a better understanding of our technology stack.

* Our Software Engineering practices full automation and 99%+ of all software/infrastructure/config are all in code with appropriate test coverage
* Our Infrastructure as Code (IaC) is powered by Cloudformation and the tool Sceptre
* Our systems are fully cloud native and is fully in AWS
* Our system leverages a mix of serverless technology backed by AWS lambda and container/server technology backed by AWS ECS
* Our database is powered by Postgres/PostGIS and DynamoDB
* Our development/production environment are powered by docker running within a linux environment
* Our CICD environment are powered by Jenkins and GitlabCI
* Our programming languages is primarily python with pinch of go