Assignment-6

The objective of this assignment is to create a real-time data processing pipeline using Kinesis data streams, Lambda, DynamoDB, and S3 using the boto3 library in Python.
Input
AWS account credentials
A stream of data records in JSON format, generated by a mock data generator

Output
● A Kinesis data stream created in the specified AWS region
● A Lambda function triggered by the Kinesis data stream that processes the data records in real-time
● The processed data stored in a DynamoDB table and backed up in an S3 bucket

Creating resources in cloudformation-

```
#kinesis stream
Resources:
  MyStream:
    Type: AWS::Kinesis::Stream
    Properties:
        Name: assign-6-kinesis-2-2-34
        RetentionPeriodHours: 48
        ShardCount: 1

 #Dynamo DB table to store the data streaming data
  kinesisDB:
    Type: AWS::DynamoDB::Table
    Properties:
      AttributeDefinitions:
          -
            AttributeName: "ArtistId"
            AttributeType: "S"
      KeySchema:
        -
          AttributeName: "ArtistId"
```

```
          KeyType: "HASH"
      ProvisionedThroughput:
        ReadCapacityUnits: 5
        WriteCapacityUnits: 5
      TableName: kinesisDB27



#lambda function to generate random data to stream
  datagen:
    Type: AWS::Lambda::Function
    Properties:
      Description: generates random data to feed the kinesis stream
      FunctionName: datagenerator2_1
      MemorySize: 128
      Handler: datagen.lambda_handler
      Role: !GetAtt LambdaExecutionRole.Arn
      Runtime:  python3.9
      Code:
        S3Bucket: fkinesistest
        S3Key: datagen.zip
      Environment:
        Variables:
          kinesistream: !Ref MyStream



# lambda function to fetch the data and store it in the dynamoDB table
  feeder:
    Type: AWS::Lambda::Function
    Properties:
      Description: inserts data into dynamodb
      FunctionName: feeder2_1
      MemorySize: 128
      Handler: feeder.lambda_handler
      Role: !GetAtt LambdaExecutionRole.Arn
      Runtime: python3.9
      Code:
        S3Bucket: fkinesistest
        S3Key: feeder.zip
```

```
    Environment:
      Variables:
        dynamotable: !Ref kinesisDB
        s3Bucket: !Ref s3Bucket

      #   S3ObjectVersion:
      #   ZipFile:
# to store backup of the data in s3 bucket
 s3Bucket:
   Type: AWS::S3::Bucket
   Properties:
     AccessControl: Private
     BucketName: kinesissfinal909

  # lambda role with permissions to access the services
 LambdaExecutionRole:
     Type: AWS::IAM::Role
     Properties:
       AssumeRolePolicyDocument:
         Version: '2012-10-17'
         Statement:
           - Effect: Allow
             Principal:
               Service: lambda.amazonaws.com
             Action:
               - sts:AssumeRole
       Path: "/"
       Policies:
         - PolicyName: "dynamodb-full-access"
           PolicyDocument:
             Version: '2012-10-17'
             Statement:
               - Effect: Allow
                 Action:
                   - dynamodb:*
                 Resource: "*"
         - PolicyName: "kinesis-full-access"
           PolicyDocument:
             Version: '2012-10-17'
             Statement:
```

```
              - Effect: Allow
                Action:
                  - kinesis:*
                Resource: "*"
          - PolicyName: "s3-full-access"
            PolicyDocument:
              Version: '2012-10-17'
              Statement:
                - Effect: Allow
                  Action:
                    - s3:*
                  Resource: "*"
          - PolicyName: "LambdaExecutionPolicy"
            PolicyDocument:
              Version: "2012-10-17"
              Statement:
                - Effect: "Allow"
                  Action:
                    - "logs:CreateLogGroup"
                    - "logs:CreateLogStream"
                    - "logs:PutLogEvents"
                  Resource: "*"



# to add the trigger , whenever there is data stream to kinesis the feeder lambda function has to get triggered and store data in DynamoDB and create back up in s3
  MyKinesisTrigger:
    Type: "AWS::Lambda::EventSourceMapping"
    Properties:
      BatchSize: 100
      EventSourceArn: !GetAtt MyStream.Arn
      FunctionName: !Ref feeder
      StartingPosition: "TRIM_HORIZON"
```

Datagenerator-

```
import json
import random
import boto3
import os

kinesis=boto3.client('kinesis')

adjectives = ['Electric', 'Mystical', 'Funky', 'Cosmic', 'Soulful',
'Gritty', 'Hypnotic', 'Dreamy', 'Jazzy']
nouns = ['Groove', 'Vibes', 'Harmony', 'Rhythm', 'Melody', 'Beat', 'Tune',
'Jam', 'Chord']

StreamName=os.environ['kinesistream']

def lambda_handler(event, context):
    def generate_artist_name():
            adjective = random.choice(adjectives)
            noun = random.choice(nouns)
            return f"{adjective} {noun}"
    for i in range(5):
            send= generate_artist_name()
            print(send)
            response = kinesis.put_record(
            StreamName=StreamName,
            Data=json.dumps(send),
            PartitionKey="1")
```

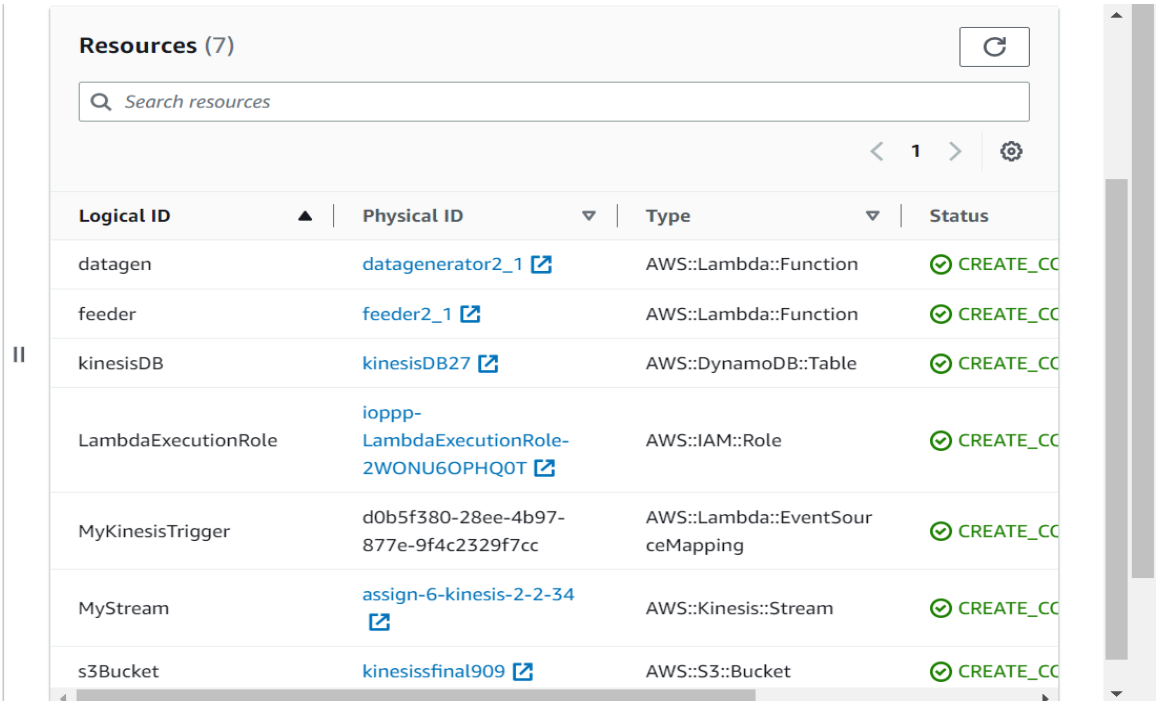
Feeder lambda function-

```
import boto3
import json
import base64
import os
import datetime

dynamodb = boto3.resource('dynamodb')

s3 = boto3.client('s3')
kinesis = boto3.client('kinesis')
data=[]

kinesisDB=os.environ['dynamotable']
s3Bucket=os.environ['s3Bucket']
table = dynamodb.Table(kinesisDB)
fors3=[]

print(kinesisDB,s3Bucket)

def lambda_handler(event, context):
    print(event)

    for i in event['Records']:

json1=json.loads(base64.b64decode(i['kinesis']['data']).decode('utf-8'))

fors3.append(json.loads(base64.b64decode(i['kinesis']['data']).decode('utf
-8')))
        # print(json1)
        table.put_item(
            Item={
                'ArtistId':json1
            })



    data=json.dumps(fors3)
    print(type(data))
```

```
key=datetime.datetime.now().strftime("%Y-%m-%d-%H-%M-%S")+".json"
s3.put_object(Bucket=s3Bucket, Key=key, Body=data)
```

- Here the data from kinesis is decoded and its fed to dynamodb table using put item , similarly data is appended to empty array to be put to the s3 bucket for backup
-

-Resources that got created after uploading stack



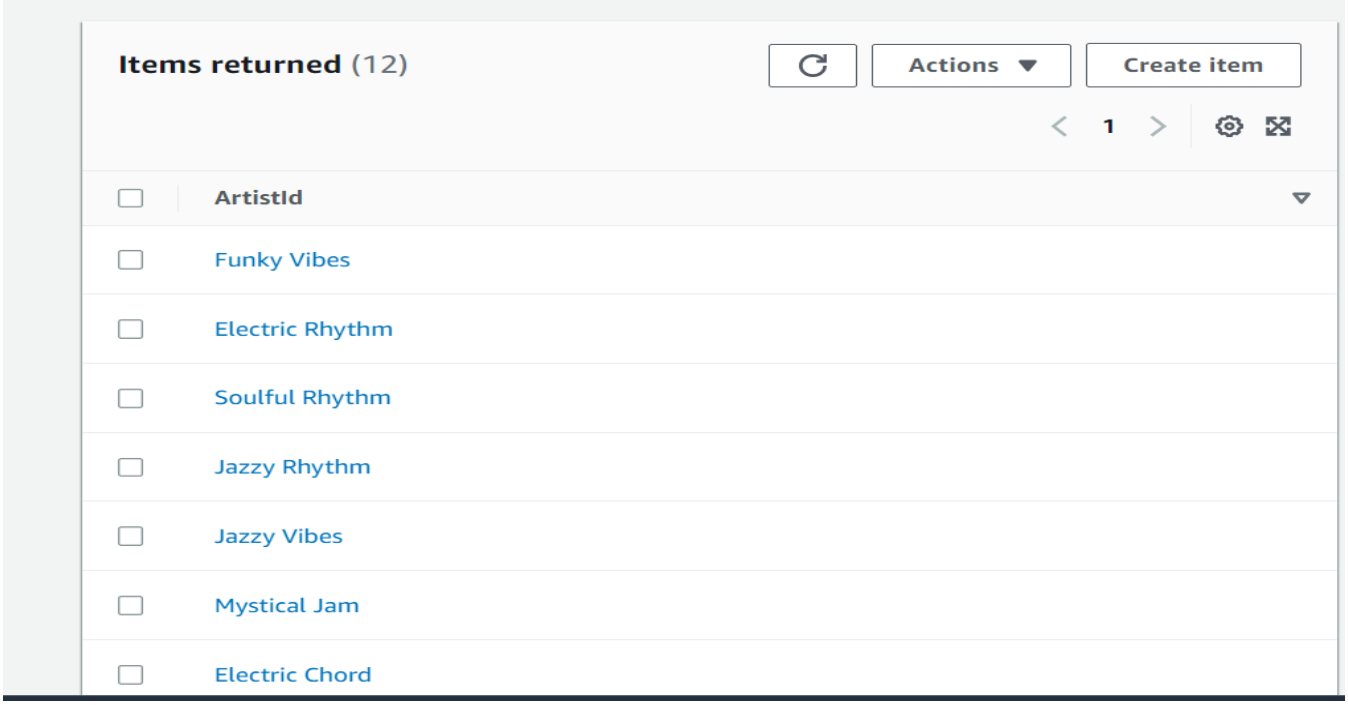
Resources (7)

Search resources

1

Logical ID	Physical ID	Type	Status
datagen	datagenerator2_1	AWS::Lambda::Function	CREATE_CC
feeder	feeder2_1	AWS::Lambda::Function	CREATE_CC
kinesisDB	kinesisDB27	AWS::DynamoDB::Table	CREATE_CC
LambdaExecutionRole	ioppp-LambdaExecutionRole-2WONU6OPHQ0T	AWS::IAM::Role	CREATE_CC
MyKinesisTrigger	d0b5f380-28ee-4b97-877e-9f4c2329f7cc	AWS::Lambda::EventSourceMapping	CREATE_CC
MyStream	assign-6-kinesis-2-2-34	AWS::Kinesis::Stream	CREATE_CC
s3Bucket	kinesissfinal909	AWS::S3::Bucket	CREATE_CC

-DynamoDB table



S3 with back up files