



## **Software Engineer for Cloud Project 1**

### **Instructions**

Every screenshot requested in this workbook is compulsory and carries 5 marks. Updated Python Script carries 20 marks. Lambda Function Code In Python carries 25 marks.

Your AWS account ID must be clearly visible in every screenshot using the AWS console; missing id or using someone else's id is not permitted. Such cases will be considered as plagiarism and severe penalty will be imposed.

All screenshots must be in the order mentioned under "Expected Screenshots" for every step

DO NOT WAIT UNTIL THE LAST MINUTE. The program office will not extend the project submission deadline under any circumstances.

The file should be renamed in the format BATCH\_FIRSTNAME\_LASTNAME\_PROJECT1.

For example: ACSEOCT20\_VIJAY\_DWIVEDI\_PROJECT1.docx

### **Resource Clean Up**

Cloud is always pay per use model and all resources/services that we consume are chargeable.

Cleaning up when you've completed your lab or project is always necessary. This is true whether you're doing a lab or implementing a project at your workplace.

After completing the lab, make sure to delete each resource created in reverse chronological order.

### **Submission Files -**

You need to submit the following -

1. Project workbook with added screenshots.
2. Updated Python script (StockPriceIngestion.py)
3. Lambda Function Code In Python



## Problem Statement :

We want to build a system that streams stock pricing information for various stocks at different times and then notifies the stakeholders when the values cross specific points of interest (POIs).

We'll use the Yahoo Finance APIs to query the running price of stocks and general information like 52-week high/low values.

**Yahoo Finance API** - It provides functions to download historical market data from Yahoo! finance. While this functionality in production would generally run on a paid api that provides real-time stock price data, we'll mimic it by using historical data for an older time period and streaming it over kinesis.

**Link** - <https://pypi.org/project/yfinance/>

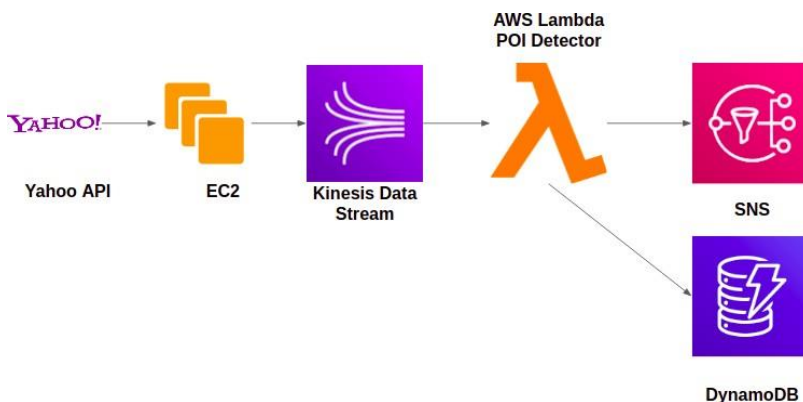
Please go through this link to understand how to access different stocks data and information.

You need to pull the data for the following 10 stocks -

MSFT, MVIS, GOOG, SPOT, INO, OCGN, ABML, RLLCF, JNJ, PSFE

You can check the details of these stocks here - <https://finance.yahoo.com/lookup/>

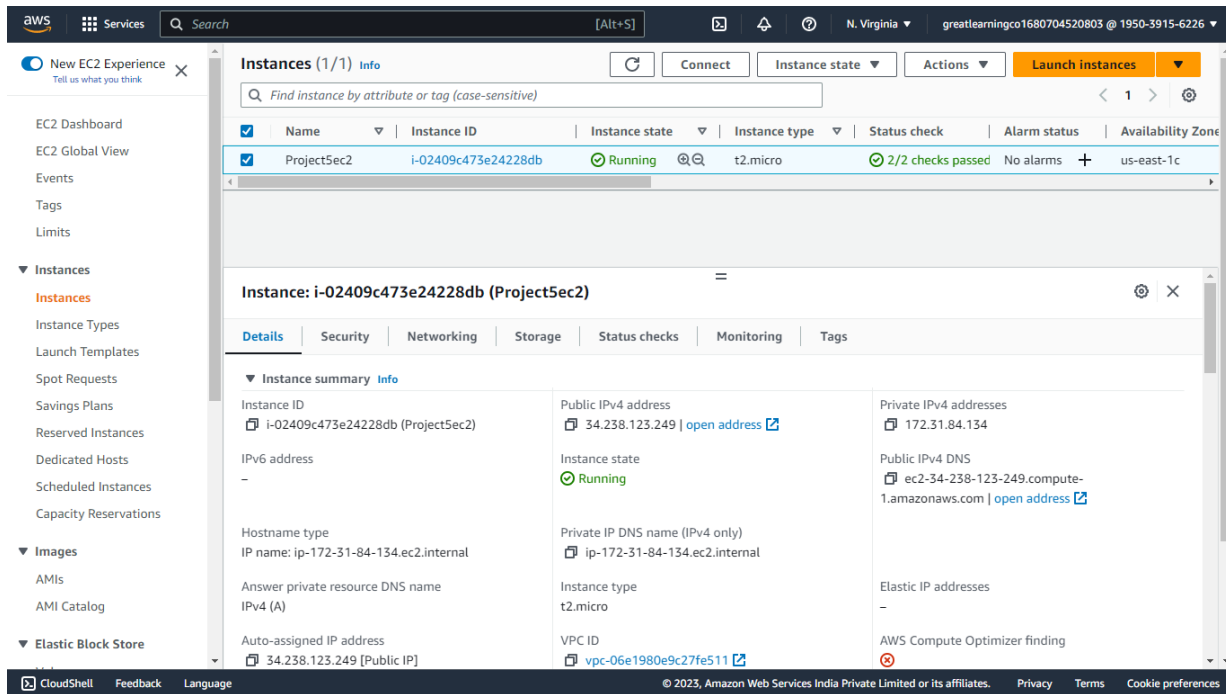
## Architecture diagram



Architecture Implementation	
1	Create EC2 Instance
2	Run Python Script to Pull data from Yahoo Stock API
3	Use boto3 and kinesis client to Push data from EC2 to Kinesis Stream
4	Configure SNS to publish the notification through mail
5	Write a Lambda function to detect the POIs and to Push the Notification to SNS
6	Use the same Lambda Function to push data to DynamoDB

## Step 1: Create EC2 Instance

Step number	a
Step name	Create EC2 Instance
Instructions	<ol style="list-style-type: none"> <li>1) Navigate to EC2 Services from AWS management console page</li> <li>2) Choose `instances` available in the left pane.</li> <li>3) Click on `launch instances`</li> <li>4) Choose the free tier for `Amazon Linux`</li> <li>5) Choose instance type as t2.micro</li> <li>6) Configure instance details keep all the option as be default</li> <li>7) Keep default storage option and provide optional tag details</li> <li>8) Create a new security group for the instance.(Configure SSH option) Make it more secure by choosing source as MyIP.</li> <li>9) Click on the launch button, It will prompt you to create a new key pair. Provide the name and download the key for login purposes.</li> </ol>
Expected screenshots	<ol style="list-style-type: none"> <li>1) Created EC2 instance running on the instances page.</li> </ol>



Step number	b
Step name	Use SSH to login into EC2 instance
Instructions	<ol style="list-style-type: none"> <li>1) Depending on the operating system you are using do the SSH login on the EC2 instance</li> <li>2) For windows system use puttygen to convert the pem file into ppk and then do the login using putty</li> <li>3) For linux based system perform the normal SSH guideline available on AWS instance→ connect page</li> </ol>
Expected screenshots	<ol style="list-style-type: none"> <li>1) Logged in screenshot of the newly created instance</li> </ol>

<Insert Screenshot b(1) here>

```

ubuntu@ip-172-31-84-134: ~
Swap usage: 0%

* Ubuntu Pro delivers the most comprehensive open source security and
  compliance features.

  https://ubuntu.com/aws/pro

* Introducing Expanded Security Maintenance for Applications.
  Receive updates to over 25,000 software packages with your
  Ubuntu Pro subscription. Free for personal use.

  https://ubuntu.com/aws/pro

Expanded Security Maintenance for Applications is not enabled.

0 updates can be applied immediately.

Enable ESM Apps to receive additional future security updates.
See https://ubuntu.com/esm or run: sudo pro status

*** System restart required ***
Last login: Wed Apr  5 15:54:02 2023 from 49.205.104.61
ubuntu@ip-172-31-84-134:~$
  
```

## Step 2 : Run Python Script to Pull data from Public API

Step number	a
Step name	
Instructions	<ol style="list-style-type: none"> <li>1) Your goal is to get per-hour stock price data for a time range for the ten stocks specified in the doc. Please take the closing price.</li> <li>2) Further, you should call the static info api for the stocks to get their current 52WeekHigh and 52WeekLow values.</li> <li>3) Update the given python code (StockPriceIngestion.py) accordingly.</li> <li>4) Print the data on console and run the script.</li> </ol>
Expected screenshots	<ol style="list-style-type: none"> <li>1) EC2 terminal after successful execution of the edited script. It doesn't have to show all of the data.</li> </ol>

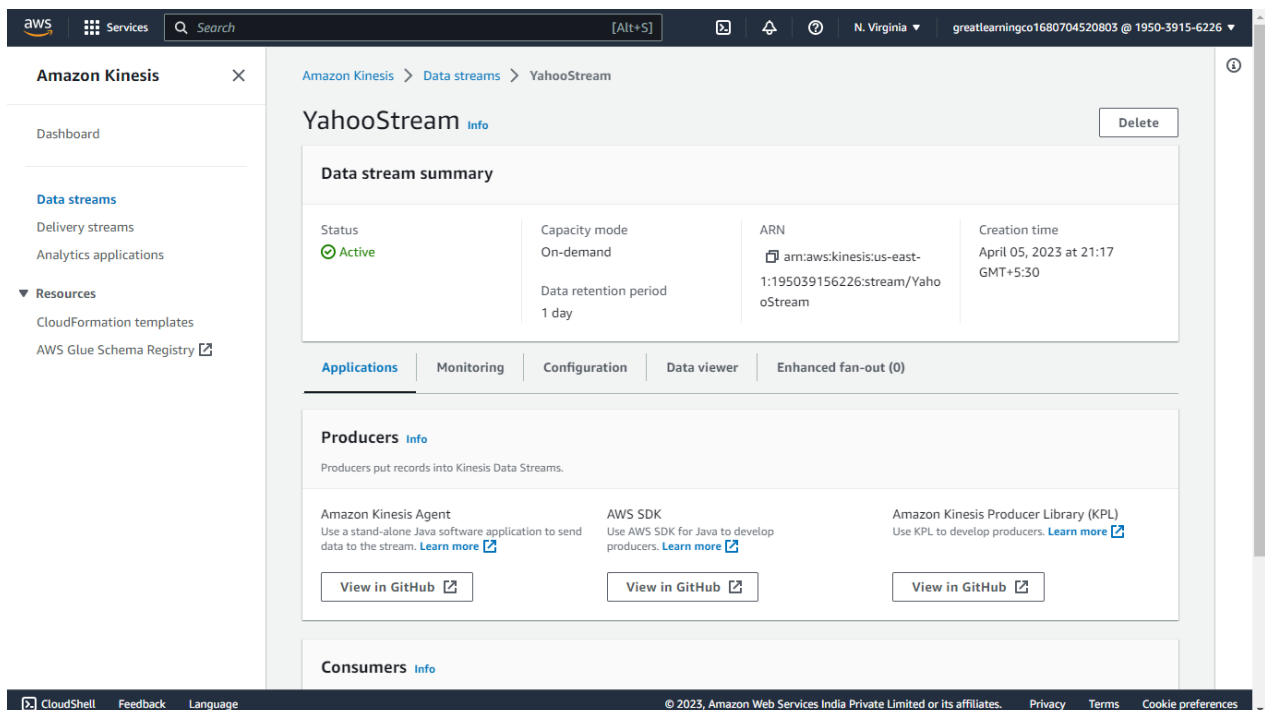
<Insert Screenshot a(1) here >

```
ubuntu@ip-172-31-04-134:~$ python3 StockPriceIngestion.py
({'Ticker': 'MSFT', 'Datetime': '2023-03-30 09:30:00-04:00', 'Close': '283.53', '52WeekLow': '213.43', '52WeekHigh': '307.0'})
({'Ticker': 'MSFT', 'Datetime': '2023-03-30 10:30:00-04:00', 'Close': '282.71', '52WeekLow': '213.43', '52WeekHigh': '307.0'})
({'Ticker': 'MSFT', 'Datetime': '2023-03-30 11:30:00-04:00', 'Close': '282.89', '52WeekLow': '213.43', '52WeekHigh': '307.0'})
({'Ticker': 'MSFT', 'Datetime': '2023-03-30 12:30:00-04:00', 'Close': '281.77', '52WeekLow': '213.43', '52WeekHigh': '307.0'})
({'Ticker': 'MSFT', 'Datetime': '2023-03-30 13:30:00-04:00', 'Close': '283.03', '52WeekLow': '213.43', '52WeekHigh': '307.0'})
({'Ticker': 'MSFT', 'Datetime': '2023-03-31 13:30:00-04:00', 'Close': '283.61', '52WeekLow': '213.43', '52WeekHigh': '307.0'})
({'Ticker': 'MSFT', 'Datetime': '2023-03-30 15:30:00-04:00', 'Close': '284.09', '52WeekLow': '213.43', '52WeekHigh': '307.0'})
({'Ticker': 'MSFT', 'Datetime': '2023-03-31 09:30:00-04:00', 'Close': '283.35', '52WeekLow': '213.43', '52WeekHigh': '307.0'})
({'Ticker': 'MSFT', 'Datetime': '2023-03-31 10:30:00-04:00', 'Close': '284.13', '52WeekLow': '213.43', '52WeekHigh': '307.0'})
({'Ticker': 'MSFT', 'Datetime': '2023-03-31 11:30:00-04:00', 'Close': '285.78', '52WeekLow': '213.43', '52WeekHigh': '307.0'})
({'Ticker': 'MSFT', 'Datetime': '2023-03-31 12:30:00-04:00', 'Close': '286.34', '52WeekLow': '213.43', '52WeekHigh': '307.0'})
({'Ticker': 'MSFT', 'Datetime': '2023-03-31 13:30:00-04:00', 'Close': '286.69', '52WeekLow': '213.43', '52WeekHigh': '307.0'})
({'Ticker': 'MSFT', 'Datetime': '2023-03-31 14:30:00-04:00', 'Close': '288.66', '52WeekLow': '213.43', '52WeekHigh': '307.0'})
({'Ticker': 'MSFT', 'Datetime': '2023-03-31 15:30:00-04:00', 'Close': '288.23', '52WeekLow': '213.43', '52WeekHigh': '307.0'})
({'Ticker': 'MSFT', 'Datetime': '2023-04-03 09:30:00-04:00', 'Close': '287.92', '52WeekLow': '213.43', '52WeekHigh': '307.0'})
({'Ticker': 'MSFT', 'Datetime': '2023-04-03 10:30:00-04:00', 'Close': '285.82', '52WeekLow': '213.43', '52WeekHigh': '307.0'})
({'Ticker': 'MSFT', 'Datetime': '2023-04-03 11:30:00-04:00', 'Close': '284.67', '52WeekLow': '213.43', '52WeekHigh': '307.0'})
({'Ticker': 'MSFT', 'Datetime': '2023-04-03 12:30:00-04:00', 'Close': '284.17', '52WeekLow': '213.43', '52WeekHigh': '307.0'})
({'Ticker': 'MSFT', 'Datetime': '2023-04-03 13:30:00-04:00', 'Close': '284.88', '52WeekLow': '213.43', '52WeekHigh': '307.0'})
({'Ticker': 'MSFT', 'Datetime': '2023-04-03 14:30:00-04:00', 'Close': '286.79', '52WeekLow': '213.43', '52WeekHigh': '307.0'})
({'Ticker': 'MSFT', 'Datetime': '2023-04-03 15:30:00-04:00', 'Close': '287.13', '52WeekLow': '213.43', '52WeekHigh': '307.0'})
({'Ticker': 'MSFT', 'Datetime': '2023-04-04 09:30:00-04:00', 'Close': '289.13', '52WeekLow': '213.43', '52WeekHigh': '307.0'})
({'Ticker': 'MSFT', 'Datetime': '2023-04-04 10:30:00-04:00', 'Close': '288.34', '52WeekLow': '213.43', '52WeekHigh': '307.0'})
({'Ticker': 'MSFT', 'Datetime': '2023-04-04 11:30:00-04:00', 'Close': '288.56', '52WeekLow': '213.43', '52WeekHigh': '307.0'})
({'Ticker': 'MSFT', 'Datetime': '2023-04-04 12:30:00-04:00', 'Close': '288.08', '52WeekLow': '213.43', '52WeekHigh': '307.0'})
({'Ticker': 'MSFT', 'Datetime': '2023-04-04 13:30:00-04:00', 'Close': '286.83', '52WeekLow': '213.43', '52WeekHigh': '307.0'})
({'Ticker': 'MSFT', 'Datetime': '2023-04-04 14:30:00-04:00', 'Close': '285.74', '52WeekLow': '213.43', '52WeekHigh': '307.0'})
({'Ticker': 'MSFT', 'Datetime': '2023-04-04 15:30:00-04:00', 'Close': '287.25', '52WeekLow': '213.43', '52WeekHigh': '307.0'})
({'Ticker': 'MSFT', 'Datetime': '2023-04-05 09:30:00-04:00', 'Close': '284.77', '52WeekLow': '213.43', '52WeekHigh': '307.0'})
({'Ticker': 'MSFT', 'Datetime': '2023-04-05 10:30:00-04:00', 'Close': '283.71', '52WeekLow': '213.43', '52WeekHigh': '307.0'})
({'Ticker': 'MSFT', 'Datetime': '2023-04-05 11:30:00-04:00', 'Close': '284.15', '52WeekLow': '213.43', '52WeekHigh': '307.0'})
({'Ticker': 'MSFT', 'Datetime': '2023-04-05 12:03:38-04:00', 'Close': '284.14', '52WeekLow': '213.43', '52WeekHigh': '307.0'})
({'Ticker': 'MWIS', 'Datetime': '2023-03-30 09:30:00-04:00', 'Close': '2.57', '52WeekLow': '2.04', '52WeekHigh': '5.96'})
({'Ticker': 'MWIS', 'Datetime': '2023-03-30 10:30:00-04:00', 'Close': '2.62', '52WeekLow': '2.04', '52WeekHigh': '5.96'})
({'Ticker': 'MWIS', 'Datetime': '2023-03-30 11:30:00-04:00', 'Close': '2.58', '52WeekLow': '2.04', '52WeekHigh': '5.96'})
({'Ticker': 'MWIS', 'Datetime': '2023-03-30 12:30:00-04:00', 'Close': '2.56', '52WeekLow': '2.04', '52WeekHigh': '5.96'})
({'Ticker': 'MWIS', 'Datetime': '2023-03-30 13:30:00-04:00', 'Close': '2.61', '52WeekLow': '2.04', '52WeekHigh': '5.96'})
({'Ticker': 'MWIS', 'Datetime': '2023-03-30 14:30:00-04:00', 'Close': '2.59', '52WeekLow': '2.04', '52WeekHigh': '5.96'})
({'Ticker': 'MWIS', 'Datetime': '2023-03-30 15:30:00-04:00', 'Close': '2.59', '52WeekLow': '2.04', '52WeekHigh': '5.96'})
({'Ticker': 'MWIS', 'Datetime': '2023-03-31 09:30:00-04:00', 'Close': '2.64', '52WeekLow': '2.04', '52WeekHigh': '5.96'})
({'Ticker': 'MWIS', 'Datetime': '2023-03-31 10:30:00-04:00', 'Close': '2.67', '52WeekLow': '2.04', '52WeekHigh': '5.96'})
({'Ticker': 'MWIS', 'Datetime': '2023-03-31 11:30:00-04:00', 'Close': '2.71', '52WeekLow': '2.04', '52WeekHigh': '5.96'})
({'Ticker': 'MWIS', 'Datetime': '2023-03-31 12:30:00-04:00', 'Close': '2.69', '52WeekLow': '2.04', '52WeekHigh': '5.96'})
```

## Step 3 : Use boto3 and kinesis client to Push data from EC2 to Kinesis Stream

Step number	a
Step name	Data in kinesis stream
Instructions	<ol style="list-style-type: none"><li>1) Go to the Kinesis stream services.</li><li>2) Configure the data stream to accept the data pushed by the script.</li><li>3) Create and attach appropriate policy and the IAM role to push the data to Kinesis stream from EC2. (helpful link - <a href="https://docs.amazonaws.cn/en_us/streams/latest/dev/tutorial-stock-data-kplkcl-iam.html">https://docs.amazonaws.cn/en_us/streams/latest/dev/tutorial-stock-data-kplkcl-iam.html</a>)</li><li>4) Edit the name of the created input kinesis stream inside the python script as required.</li><li>5) You should craft individual data records with information about the stockid, price, price timestamp, 52WeekHigh and 52WeekLow values and push them individually on the Kinesis stream.</li></ol>
Expected screenshots	<ol style="list-style-type: none"><li>1) Screenshot of created kinesis data stream</li><li>2) Screenshot of the minimum 10 data points pushed on kinesis data stream (Screenshot of the EC2 terminal)</li></ol>

<Insert Screenshot a(1) here>



<Insert Screenshot a(2) here>



ubuntu@ip-172-31-84-134: ~

```
{
  "Ticker": "600G",
  "Datetime": "2023-04-05 11:54:39-04:00",
  "Close": "104.58",
  "52WeekLow": "83.45",
  "52WeekHigh": "139.85"
},
{
  "ShardId": "shardId-000000000003",
  "SequenceNumber": "4963955455702755758956788",
  "ResponseMetadata": {
    "RequestId": "ef199d02-ec",
    "HostId": "c56a-b3be-01037ce7ba9e",
    "HTTPStatusCode": 200,
    "HTTPHeaders": {
      "x-amzn-requestid": "ef199d02-ec",
      "x-amzn-id-2": "3nsEh1JwxdxbjU2Sfx",
      "x-amzn-id-1": "3nsEh1JwxdxbjU2Sfx"
    },
    "date": "Wed, 05 Apr 2023 15:54:41 GMT",
    "content-type": "application/x-amz-jso",
    "content-length": "110",
    "RetryAttempts": 0
  },
  "Ticker": "SPOT",
  "Datetime": "2023-03-30 09:30:00-04:00",
  "Close": "131.04",
  "52WeekLow": "69.28",
  "52WeekHigh": "148.47"
},
{
  "ShardId": "shardId-000000000001",
  "SequenceNumber": "4963955455702795609917082",
  "ResponseMetadata": {
    "RequestId": "f6e610a5-f2",
    "HostId": "2c-d93e-aad1-8ca46221a6ca",
    "HTTPStatusCode": 200,
    "HTTPHeaders": {
      "x-amzn-requestid": "f6e610a5-f2",
      "x-amzn-id-2": "JXK1OFfagQ062hC02X",
      "x-amzn-id-1": "JXK1OFfagQ062hC02X"
    },
    "date": "Wed, 05 Apr 2023 15:54:41 GMT",
    "content-type": "application/x-amz-jso",
    "content-length": "110",
    "RetryAttempts": 0
  },
  "Ticker": "SPOT",
  "Datetime": "2023-03-30 10:30:00-04:00",
  "Close": "130.4",
  "52WeekLow": "69.28",
  "52WeekHigh": "148.47"
},
{
  "ShardId": "shardId-000000000001",
  "SequenceNumber": "4963955455702795609917082",
  "ResponseMetadata": {
    "RequestId": "eee09e72-70",
    "HostId": "1e-e223-b247-0273e0139dd7",
    "HTTPStatusCode": 200,
    "HTTPHeaders": {
      "x-amzn-requestid": "eee09e72-70",
      "x-amzn-id-2": "UoJJCPsLH4UkHkE+",
      "x-amzn-id-1": "UoJJCPsLH4UkHkE+"
    },
    "date": "Wed, 05 Apr 2023 15:54:41 GMT",
    "content-type": "application/x-amz-jso",
    "content-length": "110",
    "RetryAttempts": 0
  },
  "Ticker": "SPOT",
  "Datetime": "2023-03-30 11:30:00-04:00",
  "Close": "130.8",
  "52WeekLow": "69.28",
  "52WeekHigh": "148.47"
},
{
  "ShardId": "shardId-000000000001",
  "SequenceNumber": "4963955455702795609917082",
  "ResponseMetadata": {
    "RequestId": "ca8a3610-50",
    "HostId": "6d-df84-962d-aallc060a070",
    "HTTPStatusCode": 200,
    "HTTPHeaders": {
      "x-amzn-requestid": "ca8a3610-50",
      "x-amzn-id-2": "ELR1qu0auCBjfoAMP",
      "x-amzn-id-1": "ELR1qu0auCBjfoAMP"
    },
    "date": "Wed, 05 Apr 2023 15:54:41 GMT",
    "content-type": "application/x-amz-jso",
    "content-length": "110",
    "RetryAttempts": 0
  },
  "Ticker": "SPOT",
  "Datetime": "2023-03-30 12:30:00-04:00",
  "Close": "130.84",
  "52WeekLow": "69.28",
  "52WeekHigh": "148.47"
},
{
  "ShardId": "shardId-000000000001",
  "SequenceNumber": "4963955455702795609917082",
  "ResponseMetadata": {
    "RequestId": "dae715e3-a6",
    "HostId": "al-ca09-8640-89e236acb5fd",
    "HTTPStatusCode": 200,
    "HTTPHeaders": {
      "x-amzn-requestid": "dae715e3-a6",
      "x-amzn-id-2": "gFhnh2fPC0huQay/jZ",
      "x-amzn-id-1": "gFhnh2fPC0huQay/jZ"
    },
    "date": "Wed, 05 Apr 2023 15:54:41 GMT",
    "content-type": "application/x-amz-jso",
    "content-length": "110",
    "RetryAttempts": 0
  },
  "Ticker": "SPOT",
  "Datetime": "2023-03-30 13:30:00-04:00",
  "Close": "130.84",
  "52WeekLow": "69.28",
  "52WeekHigh": "148.47"
}
```

Amazon Kinesis

Dashboard

Data streams

Delivery streams

Analytics applications

Resources

CloudFormation templates

AWS Glue Schema Registry

Status

Active

Capacity mode

On-demand

Data retention period

1 day

ARN

arn:aws:kinesis:us-east-1:195039156226:stream/YahooStream

Creation time

April 05, 2023 at 21:17 GMT+5:30

Applications

Monitoring

Configuration

Data viewer

Enhanced fan-out (0)

Shard

shardId-000000000001

Starting position

Trim horizon

Get records

Records (50)

Next records

Shard: shardId-000000000001

Starting position: Trim horizon

Find records

< 1 > ⚙

Partition key	Data	Approximate arrival timestamp	Sequence number
SPOT	{"Ticker": "SPOT", "Datetime": "20...	April 05, 2023 at 21:24:41 GMT+...	4963955455702795609917082...
SPOT	{"Ticker": "SPOT", "Datetime": "20...	April 05, 2023 at 21:24:41 GMT+...	4963955455702795609917082...
SPOT	{"Ticker": "SPOT", "Datetime": "20...	April 05, 2023 at 21:24:41 GMT+...	4963955455702795609917082...
SPOT	{"Ticker": "SPOT", "Datetime": "20...	April 05, 2023 at 21:24:41 GMT+...	4963955455702795609917082...
SPOT	{"Ticker": "SPOT", "Datetime": "20...	April 05, 2023 at 21:24:41 GMT+...	4963955455702795609917082...
SPOT	{"Ticker": "SPOT", "Datetime": "20...	April 05, 2023 at 21:24:41 GMT+...	4963955455702795609917082...

CloudShell

Feedback

Language

© 2023, Amazon Web Services India Private Limited or its affiliates. Privacy Terms Cookie preferences

This file is meant for personal use by santhoshkumar.d89@gmail.com only.  
Sharing or publishing the contents in part or full is liable for legal action.



## Step 4 : Write a Lambda function to detect POIs and to push the Notification to SNS and add to DynamoDB

Step number	a
Step name	Write the lambda function
Instructions	<ol style="list-style-type: none"><li>1) Choose lambda services and create a lambda handler. Set it up to act as a consumer to your Kinesis data stream (<a href="https://docs.aws.amazon.com/lambda/latest/dg/with-kinesis.html">https://docs.aws.amazon.com/lambda/latest/dg/with-kinesis.html</a>)</li><li>2) A particular price is a POI (point of interest) if it's either <math>\geq 80\%</math> of 52WeekHigh or <math>\leq 120\%</math> of 52WeekLow. If this event happens for a stock, then that data record should be notified via SNS and stored in a DynamoDB alert table as well.</li><li>3) <b>Please note</b> that you may not find any alerts created due to the price values of the stock on that particular day, compared to the 52WeekHigh/Low values. You are free to change the percentages in point 2 to accomplish an alert trigger.</li><li>4) For each stock, if there is already an alert raised on a <b>particular day</b>, then any further alerts on that day should be skipped. Please think about the dynamodb table structure accordingly, to make this easier to query and accomplish.</li></ol>
Expected screenshots	<ol style="list-style-type: none"><li>1) Created Lambda handler</li></ol>

<Insert Screenshot a(1) here>

aws

Services

Search

[Alt+S]

N. Virginia

greatlearningco1680704520803 @ 1950-3915-6226

Lambda > Functions > YahooLambda

YahooLambda

Throttle Copy ARN Actions

Function overview Info

YahooLambda

Layers (0)

Kinesis

+ Add trigger

Amazon SNS

+ Add destination

Description

-

Last modified

19 minutes ago

Function ARN

arn:aws:lambda:us-east-1:195039156226:function:YahooLambda

Function URL

Info

Code Test Monitor Configuration Aliases Versions

General configuration

Triggers

Destinations Info

Remove Edit Add destination

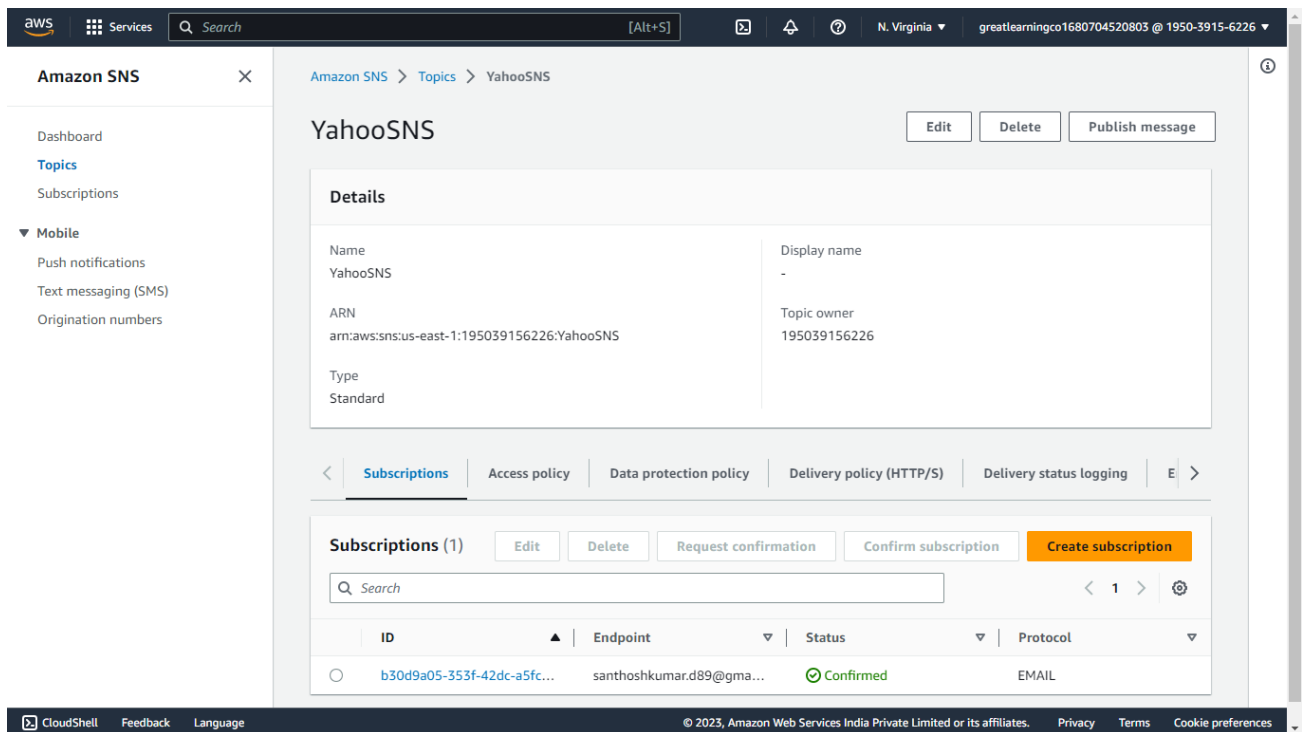
Find destinations

CloudShell Feedback Language

© 2023, Amazon Web Services India Private Limited or its affiliates. Privacy Terms Cookie preferences

Step number	b
Step name	SNS topic and subscription creation
Instructions	<ol style="list-style-type: none"><li>1) Goto AWS services look for SNS service</li><li>2) Create a standard topic</li><li>3) Create a subscription and subscribe to the topic using your email</li></ol>
Expected screenshots	<ol style="list-style-type: none"><li>1) SNS ARN for the created topic</li><li>2) Confirmation email for the created subscription</li><li>3) Point of interest data received through, once lambda handler is deployed</li></ol>

<Insert Screenshot for b(1) here >



The screenshot displays the Amazon SNS console interface. The top navigation bar includes the AWS logo, 'Services' menu, a search bar, and account information for 'greatlearningco1680704520803 @ 1950-3915-6226' in the 'N. Virginia' region. The left sidebar shows the 'Amazon SNS' dashboard with links to 'Topics', 'Subscriptions', and 'Mobile' services. The main content area is titled 'YahooSNS' and includes buttons for 'Edit', 'Delete', and 'Publish message'. Below this, the 'Details' section shows the topic's Name ('YahooSNS'), ARN ('arn:aws:sns:us-east-1:195039156226:YahooSNS'), Type ('Standard'), Display name ('-'), and Topic owner ('195039156226'). The 'Subscriptions' tab is active, showing a table with one subscription. The subscription has ID 'b30d9a05-353f-42dc-a5fc...', Endpoint 'santhoshkumar.d89@gmail.com', Status 'Confirmed', and Protocol 'EMAIL'. Buttons for 'Edit', 'Delete', 'Request confirmation', 'Confirm subscription', and 'Create subscription' are visible above the table. The footer contains links for 'CloudShell', 'Feedback', 'Language', and copyright information for Amazon Web Services India Private Limited.

ID	Endpoint	Status	Protocol
b30d9a05-353f-42dc-a5fc...	santhoshkumar.d89@gmail.com	Confirmed	EMAIL

The screenshot shows the Amazon SNS console interface. The left sidebar contains navigation links: Dashboard, Topics, Subscriptions, Mobile, Push notifications, Text messaging (SMS), and Origination numbers. The main content area displays the details for a specific subscription: b30d9a05-353f-42dc-a5fc-7997634b66ce. The details include the ARN, Endpoint (santhoshkumar.d89@gmail.com), Topic (YahooSNS), and Subscription Principal. The status is 'Confirmed' with a green checkmark. Below the details, there are tabs for 'Subscription filter policy' and 'Redrive policy (dead-letter queue)'. The 'Subscription filter policy' tab is active, showing a policy that filters messages based on subscriber receipt.

<Insert Screenshot for b(2) here>



Simple Notification Service

## Subscription confirmed!

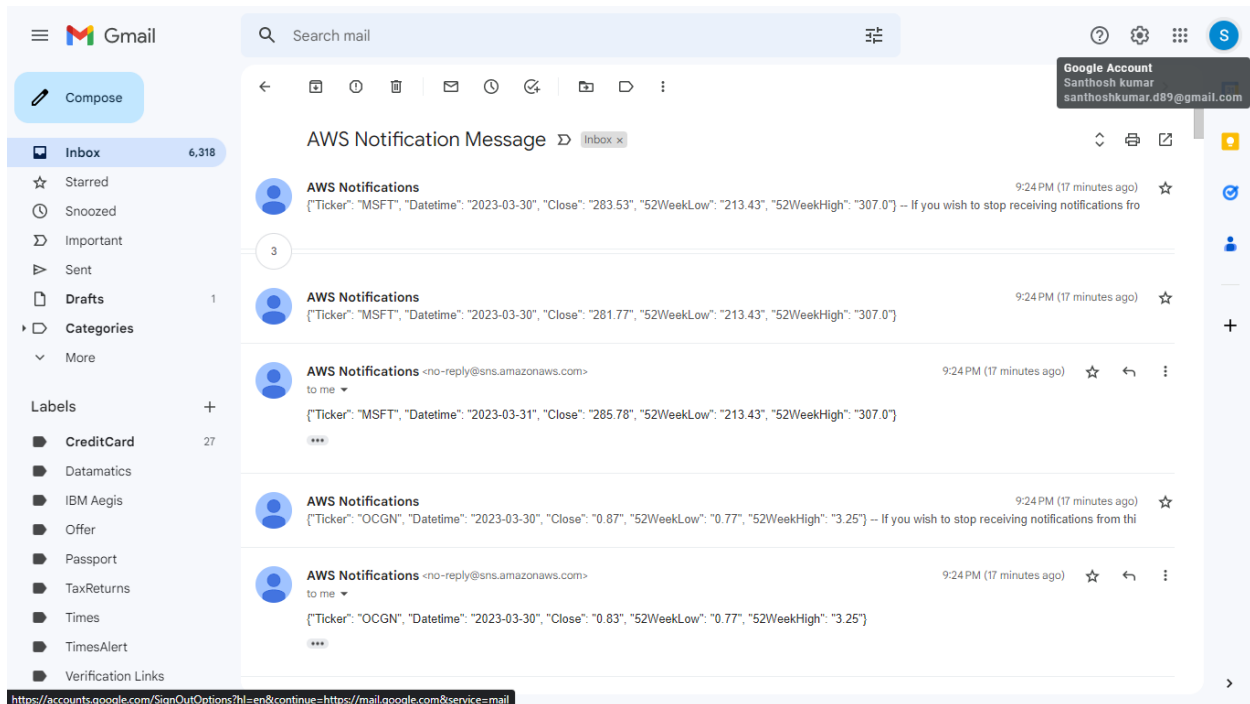
You have successfully subscribed.

Your subscription's id is:

**arn:aws:sns:us-east-1:195039156226:YahooSNS:b30d9a05-353f-42dc-a5fc-7997634b66ce**

If it was not your intention to subscribe, [click here to unsubscribe](#).

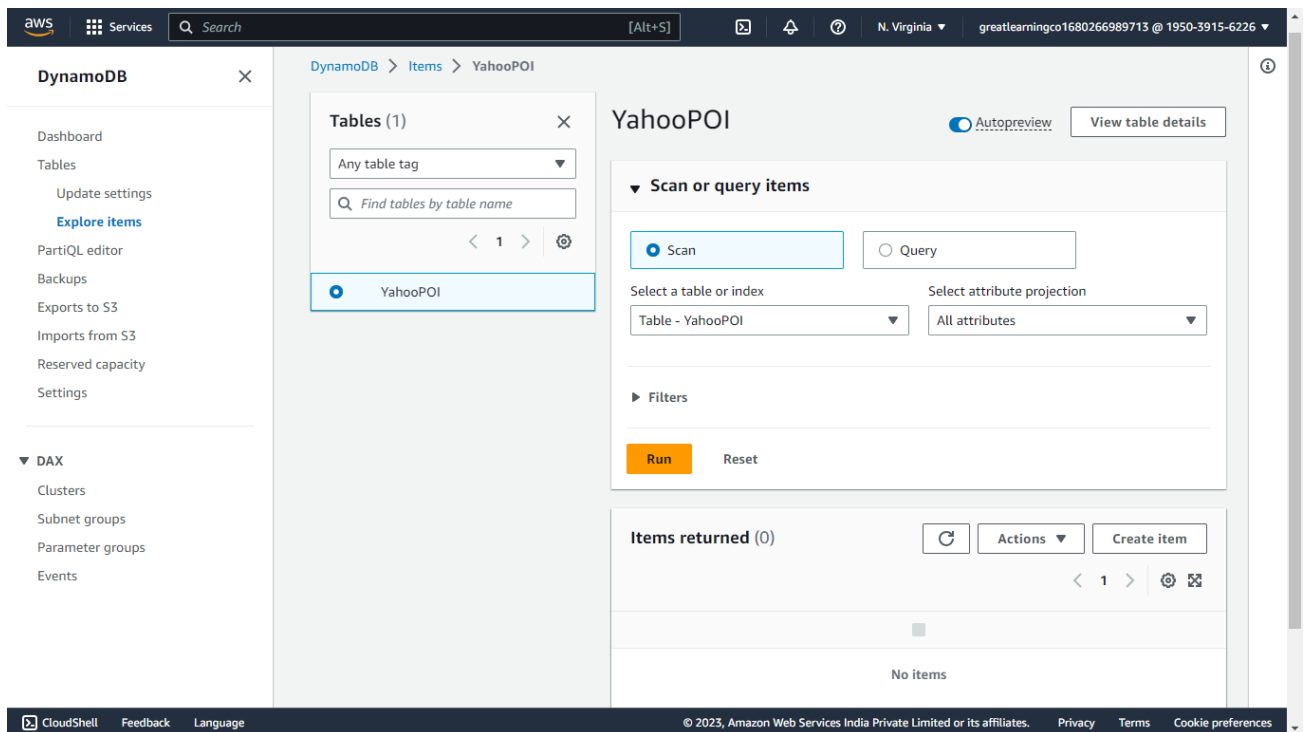
<Insert Screenshot for b(3) here>



santhoshkumar.d89@gmail.com  
F0AR54VQ8G

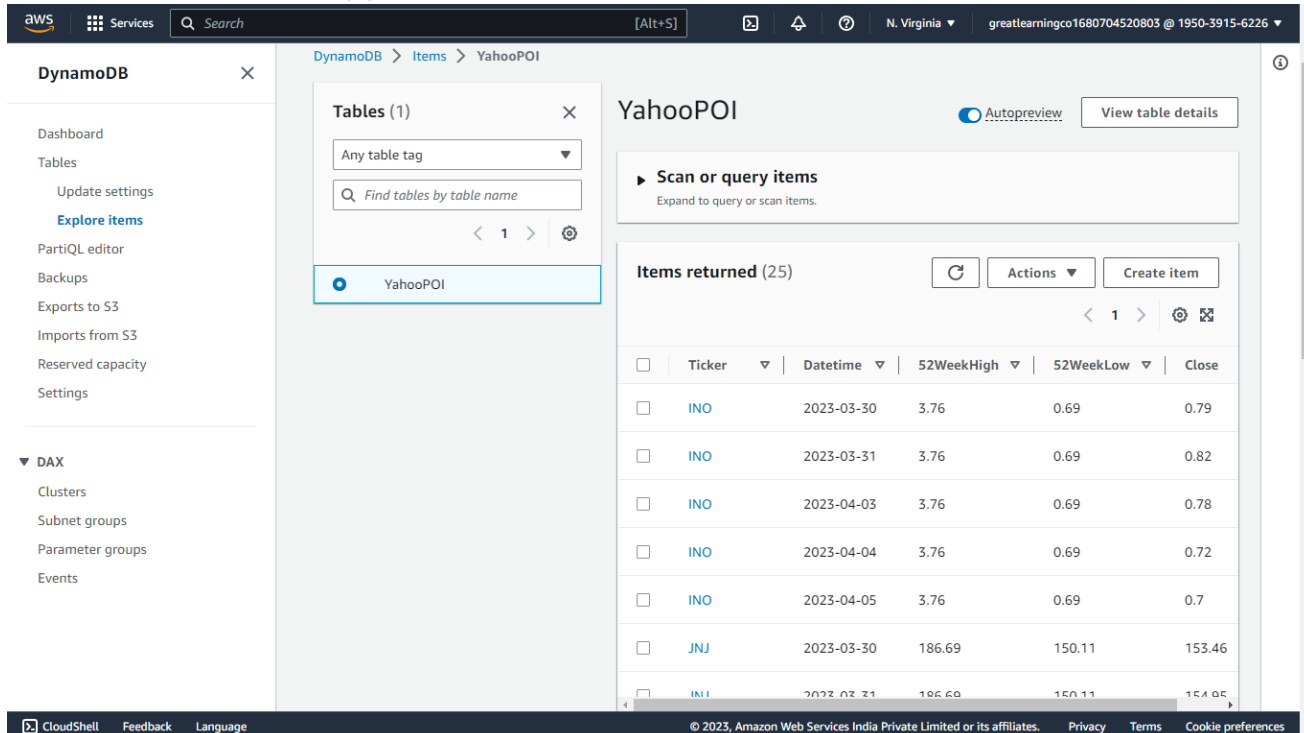
Step number	c
Step name	Dynamodb table creation
Instructions	<ol style="list-style-type: none"> <li>1) Goto AWS services and search for dynamodb</li> <li>2) Create a table by providing the partition key and sort key</li> </ol>
Expected screenshots	<ol style="list-style-type: none"> <li>1) Empty Table</li> <li>2) Table with stored data</li> </ol>

<Insert Screenshot for c(1) here >



The screenshot shows the AWS DynamoDB console interface. On the left is a navigation menu with options like Dashboard, Tables, Update settings, Explore items, PartiQL editor, Backups, Exports to S3, Imports from S3, Reserved capacity, Settings, DAX, Clusters, Subnet groups, Parameter groups, and Events. The main area displays the 'YahooPOI' table. Under the 'Scan or query items' section, the 'Scan' option is selected. The 'Table' dropdown is set to 'Table - YahooPOI' and the 'Attribute projection' is set to 'All attributes'. The 'Run' button is visible. Below this, the 'Items returned (0)' section shows 'No items'.

<Insert Screenshot for c(2) here>



The screenshot shows the AWS DynamoDB console interface with the 'YahooPOI' table. The 'Scan or query items' section is expanded, and the 'Run' button has been clicked. The 'Items returned (25)' section now displays a table with 25 rows of data. The table has columns: Ticker, Datetime, 52WeekHigh, 52WeekLow, and Close. The data includes stock prices for INO and JNJ.

	Ticker	Datetime	52WeekHigh	52WeekLow	Close
<input type="checkbox"/>	INO	2023-03-30	3.76	0.69	0.79
<input type="checkbox"/>	INO	2023-03-31	3.76	0.69	0.82
<input type="checkbox"/>	INO	2023-04-03	3.76	0.69	0.78
<input type="checkbox"/>	INO	2023-04-04	3.76	0.69	0.72
<input type="checkbox"/>	INO	2023-04-05	3.76	0.69	0.7
<input type="checkbox"/>	JNJ	2023-03-30	186.69	150.11	153.46
<input type="checkbox"/>	JNJ	2023-03-31	186.69	150.11	154.95