

# COS20019 – CLOUD COMPUTING ARCHITECTURE

Name: Nguyen Linh Dan

Student ID: 103488557

## ACA MODULE 13 CHALLENGE LAB

### IMPLEMENTING A SERVERLESS ARCHITECTURE FOR THE CAFÉ

In this challenging lab, I implemented a serverless architecture to automatically generate the sales daily reports for a café shop. This lab will need the knowledge of Lambda function and SNS service.

#### Task 1: Downloading the source code

This task only provides me with some Python source code for the Lambda function I will create in the next steps. Therefore, there is no configuration required here.

#### Task 2: Creating the DataExtractor Lambda function in the VPC

##### Create a security group for the Lambda function

At the beginning of the Lambda configuration stage, I need to create a **Lambda function** which could **extract data** from the RDS database. Therefore, a security group with rules to activate the connection between the RDS and the Lambda function is necessary.

The screenshot shows the AWS Management Console interface for creating a new security group. The 'Basic details' section is active, showing the following fields:

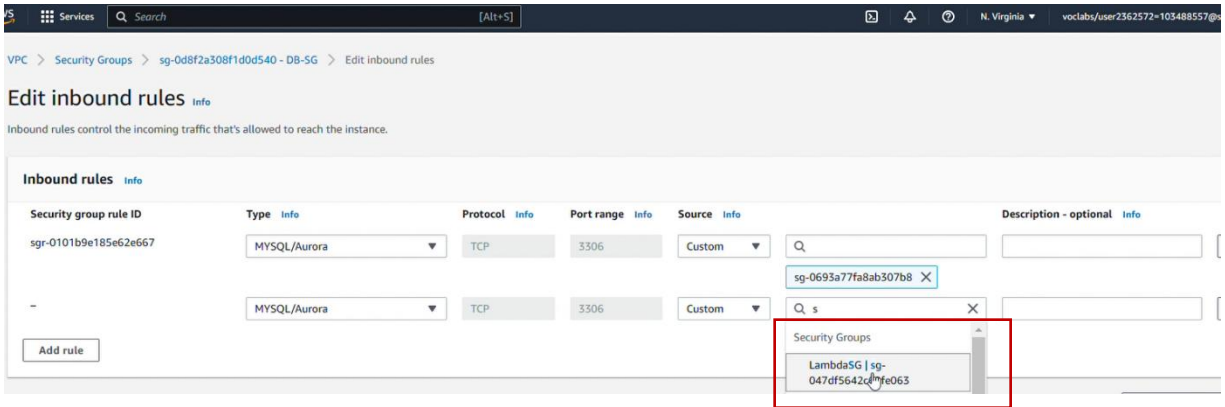
- Security group name:** LambdaSG
- Description:** Lambda security group
- VPC:** vpc-0266f01180a4387e

A text box on the right side of the console states: "This security group will be passed as a source of inbound rules of the RDS security group."

Below the 'Basic details' section, the 'Outbound rules' section is partially visible, showing a table with columns for Type, Protocol, Port range, Destination, and Description - optional.

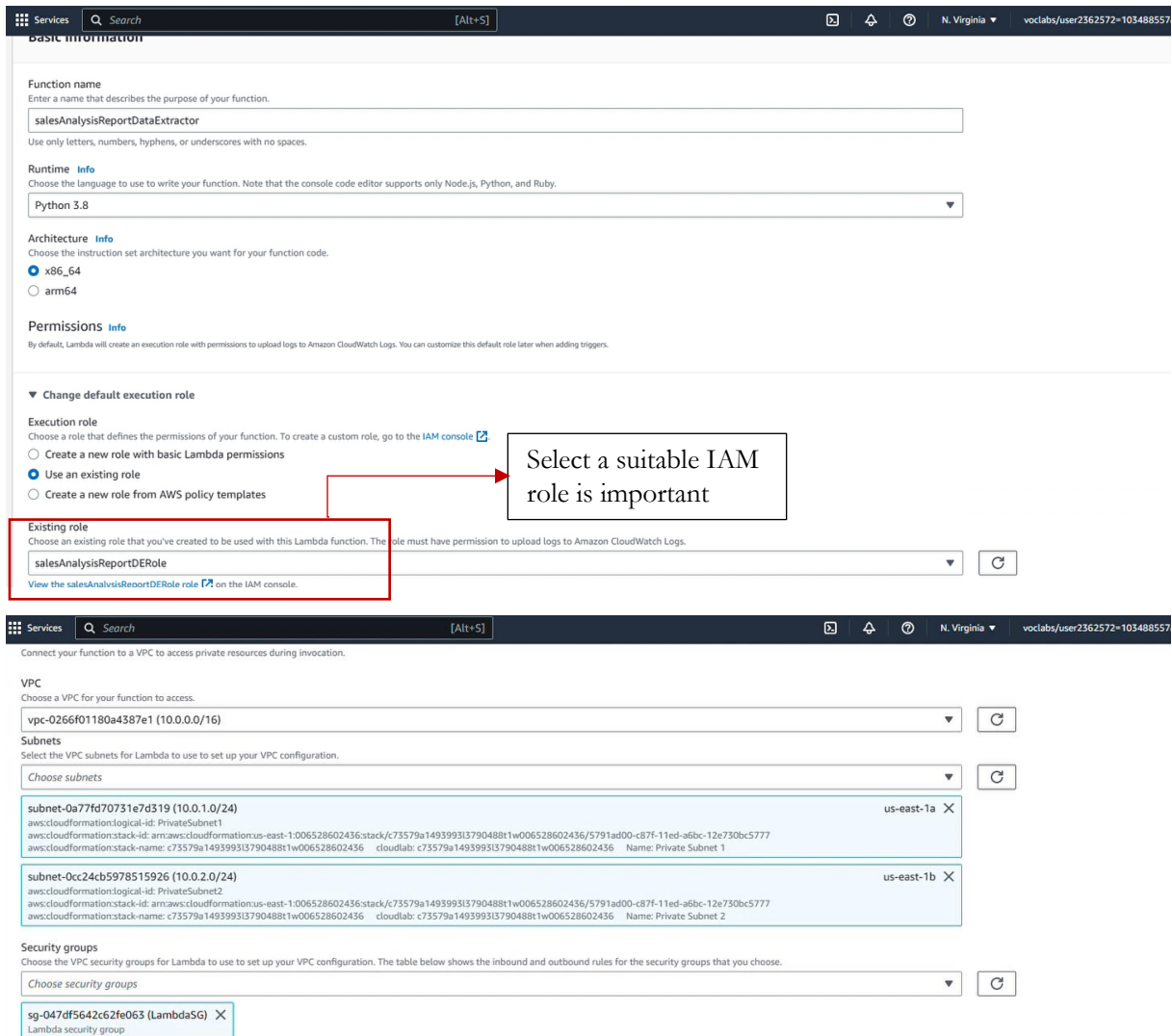
##### Update the DatabaseSG security group

Next, I also need to configure the RDS security group to establish the connection. I added a new **MySQL/Aurora** rule and defined the traffic source of this rule as the Lambda security group.

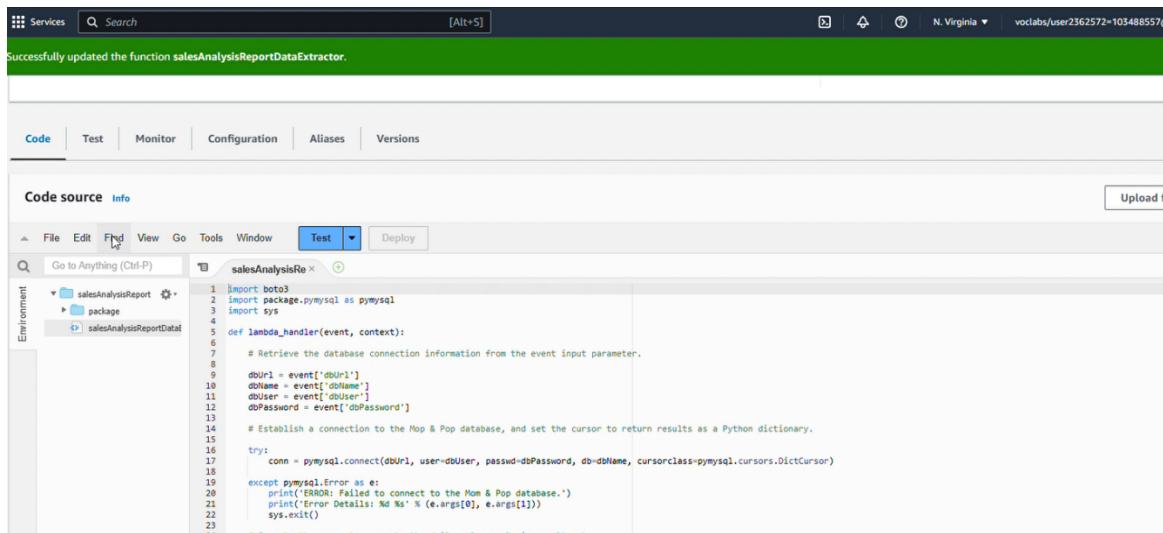


## Create a DataExtractor Lambda function

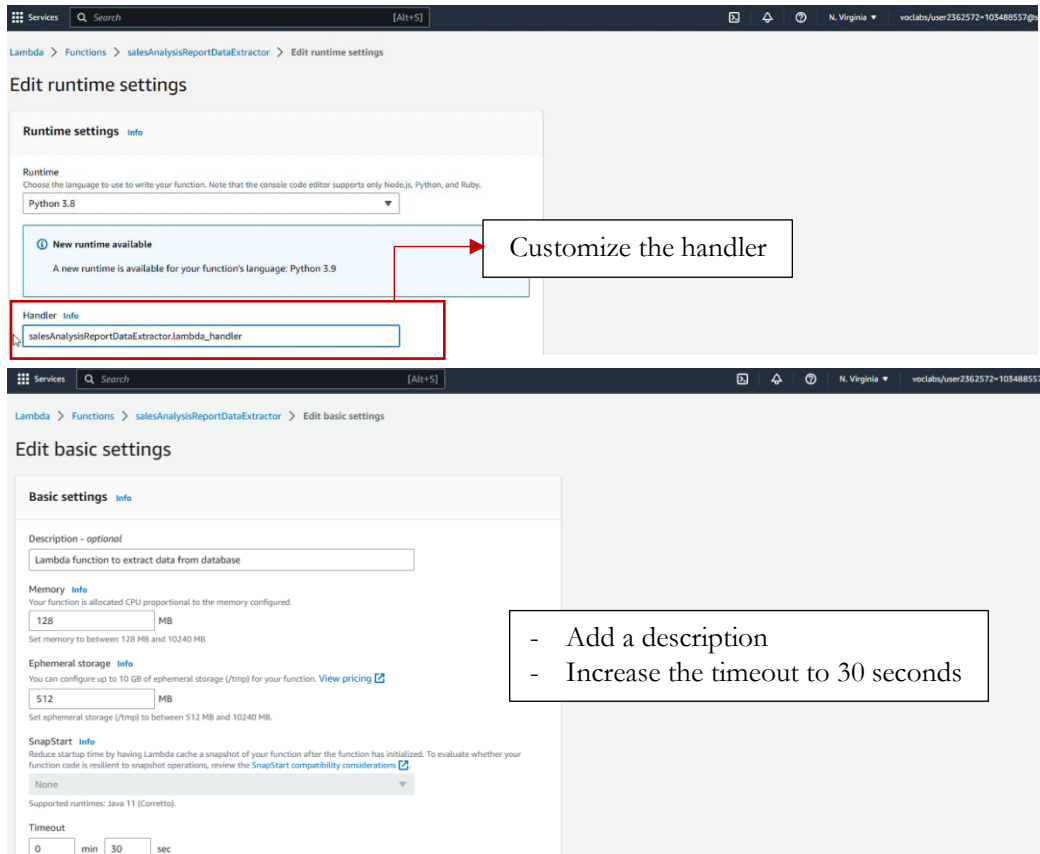
This Lambda will be used to extract data from the RDS database. The below Lambda function configurations followed the instructions (correct VPC, Security groups, IAM role, and subnets).



When the Lambda function is created, I uploaded the downloaded **zip package** to the **Code** tab.



I also modified the handler and other general information of this Lambda function.



### Task 3: Creating the salesAnalysisReport Lambda function

I created another Lambda function to automate the process of generating and sending daily sales reports. Most configurations of the current Lambda and the previous Lambda were the same. However, this Lambda function was not attached to a VPC, and it uses another IAM role.

The screenshot shows the 'Create new function' wizard in the AWS Lambda console. The 'Function name' field is set to 'salesAnalysisReport'. The 'Runtime' is set to 'Python 3.8'. The 'Architecture' is set to 'x86\_64'. Under 'Permissions', the option 'Use an existing role' is selected, and the 'Existing role' dropdown is set to 'salesAnalysisReportRole'. A button 'Change default execution role' is visible.

I modified the handler of this function

The screenshot shows the 'Edit runtime settings' page for the 'salesAnalysisReport' function. The 'Runtime' is 'Python 3.8'. A notification box states 'New runtime available: A new runtime is available for your function's language: Python 3.9'. The 'Handler' field is set to 'salesAnalysisReport.lambda\_handler', which is highlighted with a red box. A callout box with an arrow points to this field, containing the text 'Customize the handler'.

The general information of the 2 functions was identical.

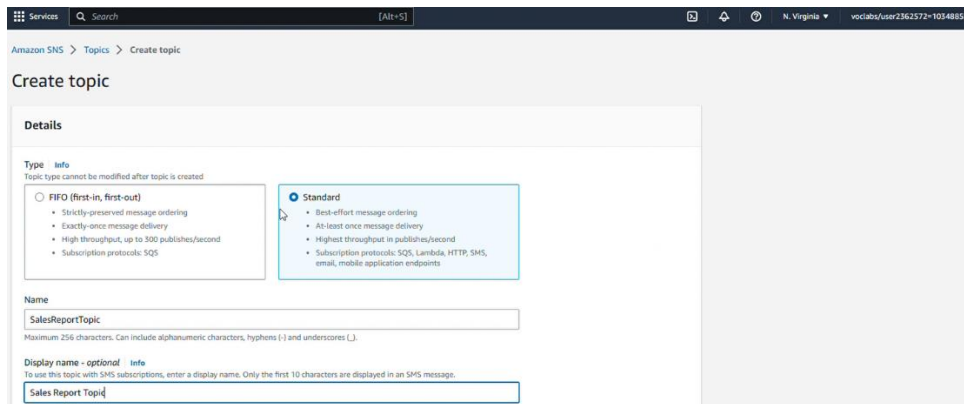
The screenshot shows the 'Edit basic settings' page for the 'salesAnalysisReport' function. The 'Description' is 'Lambda function to generate and send the daily sales report'. The 'Memory' is set to '128 MB'. The 'Ephemeral storage' is set to '512 MB'. The 'SnapStart' is set to 'None'. The 'Timeout' is set to '30 sec'. A callout box with an arrow points to the 'Timeout' field, containing the text: '- Add a description' and '- Increase the timeout to 30 seconds'.

## Task 4: Creating an SNS topic

### Create a SNS topic

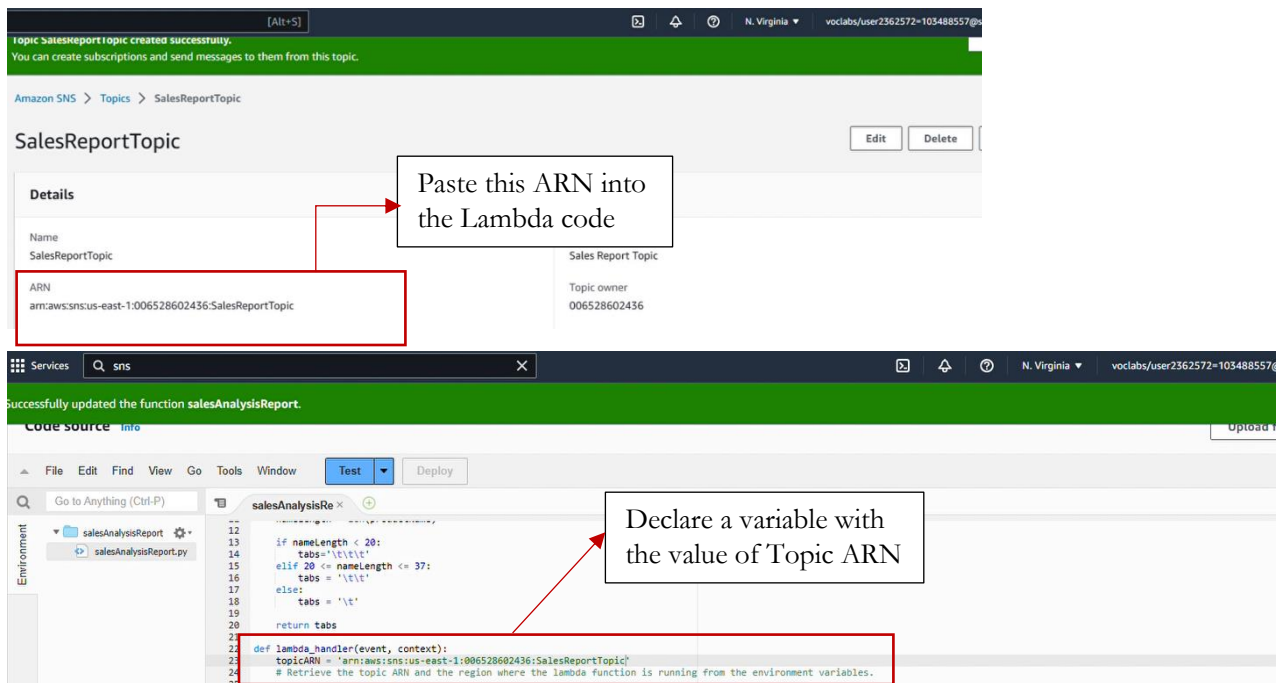
Tasks 4 and 5 will focus on using the **SNS service** to send the daily reports via email to the café managers. First, I created an **SNS topic** and added the topic ARN to the **salesAnalysisReport** Lambda function.

Access the **SNS** service → fill in the **Topic name** → **Next Step**. The topic configurations will be shown below.



### Update the salesAnalysisReport Lambda function

After the SNS topic had been created, I copied the **Topic ARN** and passed it into the Lambda function as a variable.



## Task 5: Creating an email subscription to the SNS topic

Besides the Topic, I also need to create an **SNS subscription** to receive the daily reports via email.

Amazon SNS > Subscriptions > Create subscription

### Create subscription

**Details**

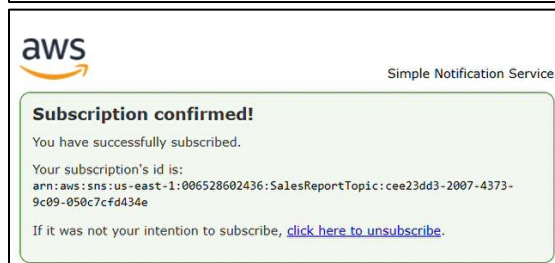
Topic ARN  
arn:aws:sns:us-east-1:006528602436:SalesReportTopic

Protocol  
The type of endpoint to subscribe  
Email

Endpoint  
An email address that can receive notifications from Amazon SNS.  
dannishw00303@fpt.edu.vn

Daily reports will be sent to this email

I had to confirm the subscription to activate it.



## Task 6: Testing the salesAnalysisReport Lambda function

After setting up all necessary things, I run a test to check whether my Lambda function and SNS configurations work correctly.

On the **Code** tab of the function → click on the button **Test** → **Configure test event**. Enter the **test event name** and leave the remaining part by default.

**Configure test event**

A test event is a JSON object that mocks the structure of requests emitted by AWS services to invoke a Lambda function. Use it to see the function's invocation result.

To invoke your function without saving an event, configure the JSON event, then choose Test.

Test event action  
☒ Create new event ☐ Edit saved event

Event name  
TestEvent

Maximum of 25 characters consisting of letters, numbers, dots, hyphens and underscores.

When I run the test, if my configurations are correct, I will receive a daily report via the email I specified on the Subscription setup page.

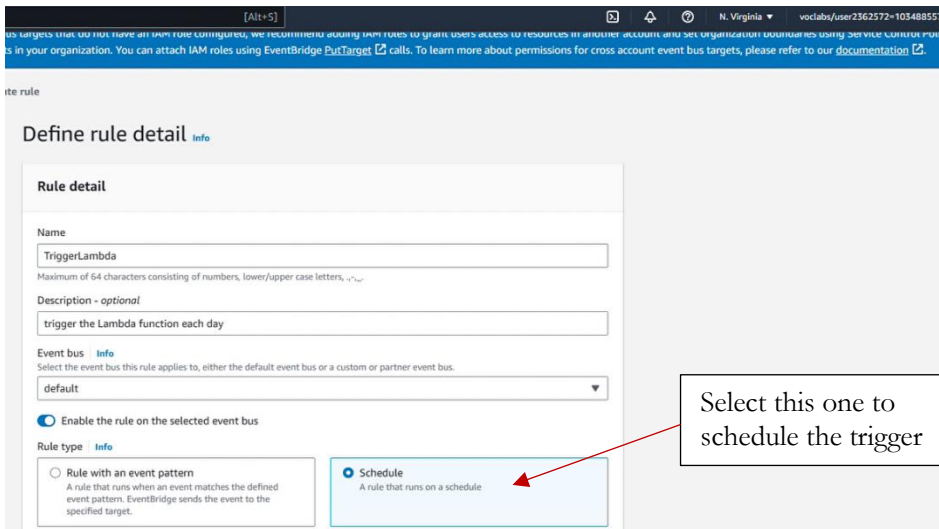


| Item Name                 | Quantity |
|---------------------------|----------|
| Croissant                 | 29       |
| Donut                     | 23       |
| Chocolate Chip Cookie     | 18       |
| Muffin                    | 6        |
| Strawberry Blueberry Tart | 34       |
| Strawberry Tart           | 33       |

## Task 7: Setting up an Amazon EventBridge event to trigger the Lambda function each day

The last issue of the café shop that I need to solve in this lab is setting up a trigger to send the daily reports via email every day without human intervention.

Access the **EventBridge** service → select **EventBridge rule** → **Create rule**



Define rule detail

**Rule detail**

Name: TriggerLambda

Description - optional: trigger the Lambda function each day

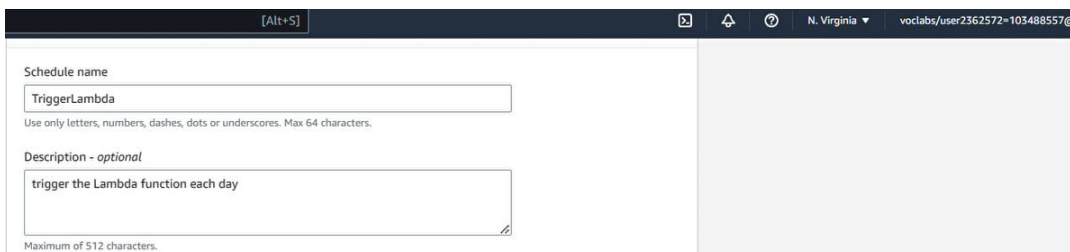
Event bus: default

☒ Enable the rule on the selected event bus

**Rule type**

☐ Rule with an event pattern

☒ Schedule



Schedule name: TriggerLambda

Description - optional: trigger the Lambda function each day



[Alt+S] N. Virginia voclabs/user2362572-103488557@

### Schedule pattern

**Occurrence** [Info](#)  
You can define an one-time or recurrent schedule.

☐ One-time schedule ☒ **Recurring schedule**

**Schedule type**  
Choose the schedule type that best meets your needs.

☒ **Cron-based schedule**  
A schedule set using a cron expression that runs at a specific time, such as 8:00 a.m. PST on the first Monday of every month.

☐ Rate-based schedule  
A schedule that runs at a regular rate, such as every 10 minutes.

**Cron expression** [Info](#)  
Define the cron expression for the schedule [Copy](#) [Clear](#)

**cron** (       )  
Minutes Hours Day of month Month Day of the week Year

**Next 10 trigger dates**  
Date and time are displayed in your current time zone in UTC format, e.g. "Wed, Nov 9, 2022 09:00 (UTC - 08:00)" for Pacific time

Wed, 22 Mar 2023 15:00:00 (UTC +07:00)  
Thu, 23 Mar 2023 15:00:00 (UTC +07:00)  
Fri, 24 Mar 2023 15:00:00 (UTC +07:00)  
Sat, 25 Mar 2023 15:00:00 (UTC +07:00)  
Sun, 26 Mar 2023 15:00:00 (UTC +07:00)  
Mon, 27 Mar 2023 15:00:00 (UTC +07:00)  
Tue, 28 Mar 2023 15:00:00 (UTC +07:00)  
Wed, 29 Mar 2023 15:00:00 (UTC +07:00)  
Thu, 30 Mar 2023 15:00:00 (UTC +07:00)  
Fri, 31 Mar 2023 15:00:00 (UTC +07:00)

This rule will invoke a trigger at 15:00 every day

[Alt+S] N. Virginia voclabs/user2362572-103488557@

**Timezone - optional**  
The timezone for the schedule.

(UTC +07:00) Asia/Bangkok

**Start date and time - optional**  
The start date and time of the schedule.

2023/03/22   
YYYY/MM/DD Use 24-hour format timestamp (hh:mm)

[Alt+S] N. Virginia voclabs/user2362572-103488557@

|                                  |                                   |
|----------------------------------|-----------------------------------|
| Kinesis Data Firehose PutRecord  | <b>AWS Lambda Invoke</b>          |
| SageMaker StartPipelineExecution | Amazon SNS Publish                |
| Amazon SQS SendMessage           | AWS Step Functions StartExecution |

**Invoke**  
AWS Lambda

**Lambda function**  
salesAnalysisReport

Choose the Lambda as the trigger target  
Select a Lambda function to invoke everyday