



COS20019

Cloud Computing Architecture

Week 9 – ACA Module 14

Challenge Lab: Implementing a Serverless Architecture for the Café

Truong Ngoc Gia Hieu
105565520

Challenge Lab: Implementing a Serverless Architecture for the Café

Scenario

The café's business is thriving. Frank and Martha want to get daily sales reports for products that are sold from the café's website. They will use this report to plan ingredient orders and monitor the impact of product promotions.

Sofia and Nikhil's initial idea is to use one of the Amazon Elastic Compute Cloud (Amazon EC2) web server instances to generate the report. Sofia sets up a cron job on the web server instance, which sends email messages that report daily sales. However, the cron job reduces the performance of the web server because it's resource intensive.

Nikhil mentions the cron job and how it reduces the web application's performance to Olivia. Olivia advises Sofia and Nikhil to separate non-business-critical reporting tasks from the production web server instance. After Sofia and Nikhil review the advantages and disadvantages of their current approach, they decide that they don't want to slow down the web server. They also consider running a separate EC2 instance, but they are concerned about the cost of running an instance 24/7 when it's only needed for a short time each day.

Sofia and Nikhil decide that running the report generation code as an AWS Lambda function would work, and it would also lower costs. The report itself could be sent to Frank and Martha's email address through Amazon Simple Notification Service (Amazon SNS). In this lab, you take on the role of Sofia to implement the daily report code as a Lambda function.

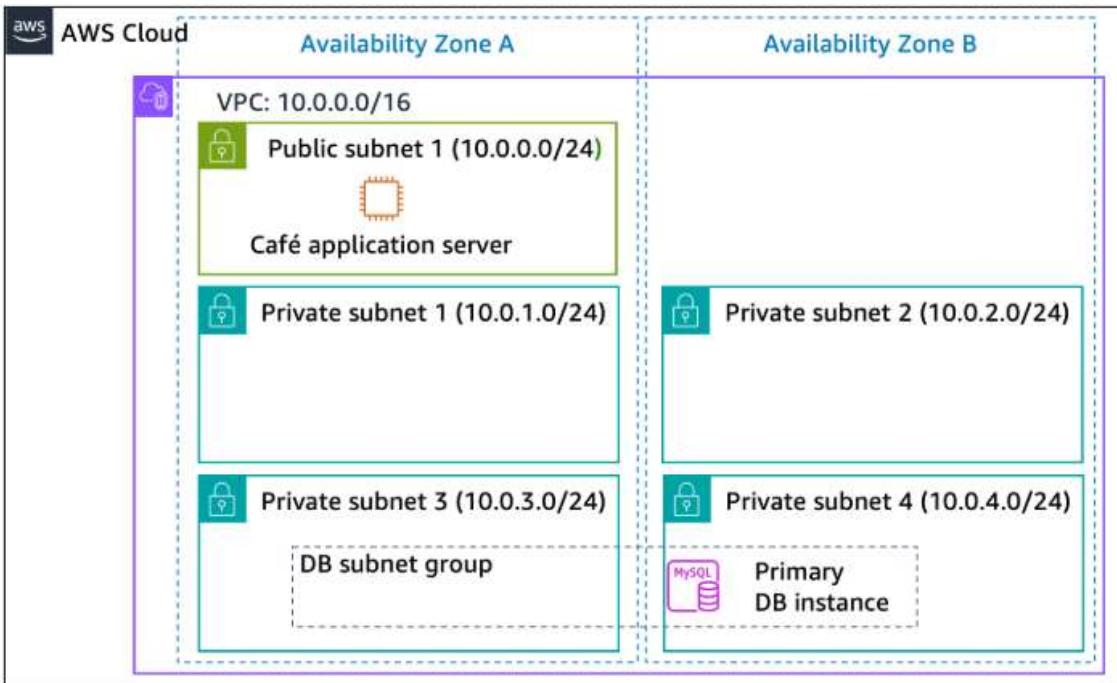
A. Lab overview and objectives

In this lab, you use AWS Lambda to create a café sales report that's emailed each day through Amazon SNS.

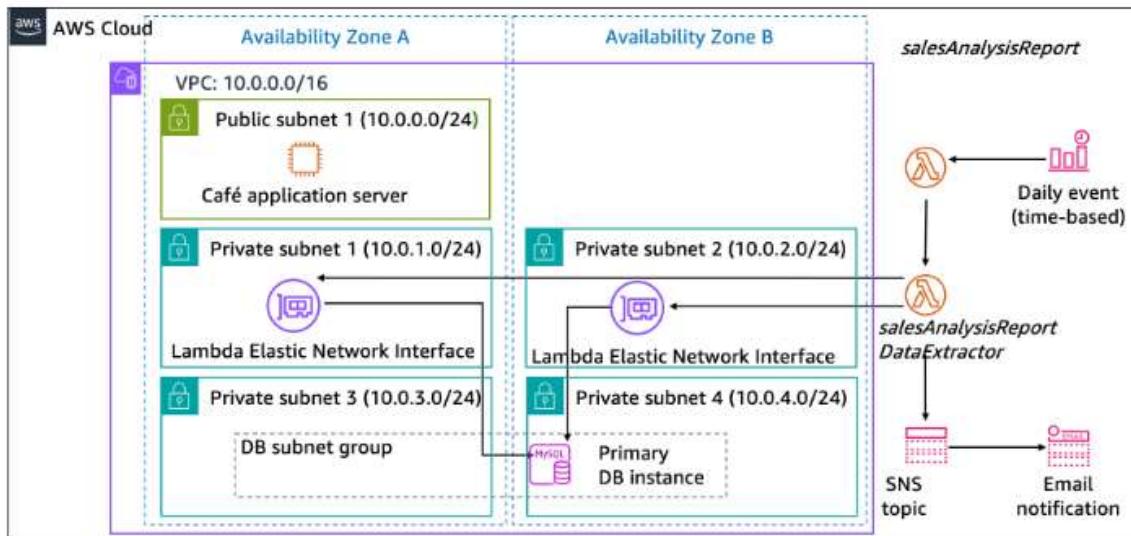
After completing this lab, you should be able to implement a serverless architecture to generate a daily sales report that features the following:

- A Lambda function within a virtual private cloud (VPC) that connects to an Amazon Relational Database Service (Amazon RDS) database with the café's sales data
- A Lambda function that generates and runs the sales report
- A scheduled event that initiates the sales report Lambda function each day

When you *start* the lab, your architecture looks like the following example:



At the *end* of this lab, your architecture will look like the following example:



Duration

This lab requires approximately **90 minutes** to complete.

AWS service restrictions

In this lab environment, access to AWS services and service actions might be restricted to the ones that are needed to complete the lab instructions. You might encounter errors if you attempt to access other services or perform actions beyond the ones that are described in this lab.

B. Accessing AWS Management Console



Figure 1: AWS Activated

C. Task 1: Downloading the source code

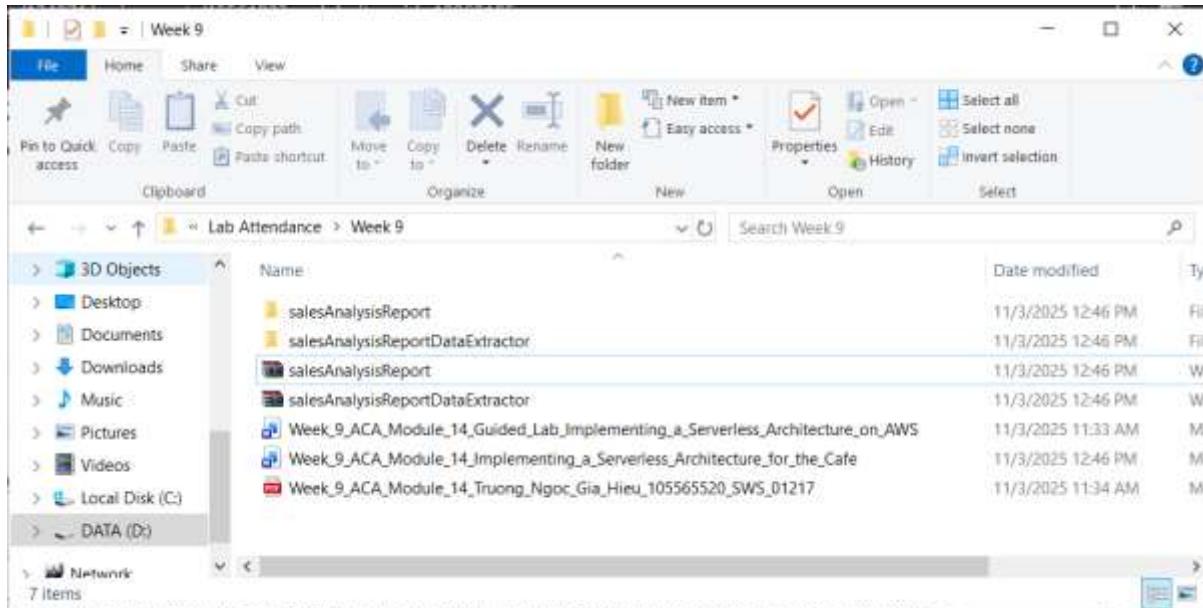


Figure 2: Materials downloaded

D. Task 2: Creating the DataExtractor Lambda function in the VPC

In this task, you create the *DataExtractor* Lambda function that extracts the café's sales data from an Amazon RDS database. So the Lambda function can access the RDS database instance, you must update the database security group with a rule to allow connections from the Lambda function. To enable this communication, you create a security group for the Lambda function and add it as an inbound rule to the security group of the RDS instance.

Step 2.1: Search and select EC2 service. Then, create security group with following configurations:

- Security group name: LambdaSG
- VPC: *Lab VPC*
- Outbound Rules: *All traffic* to all addresses

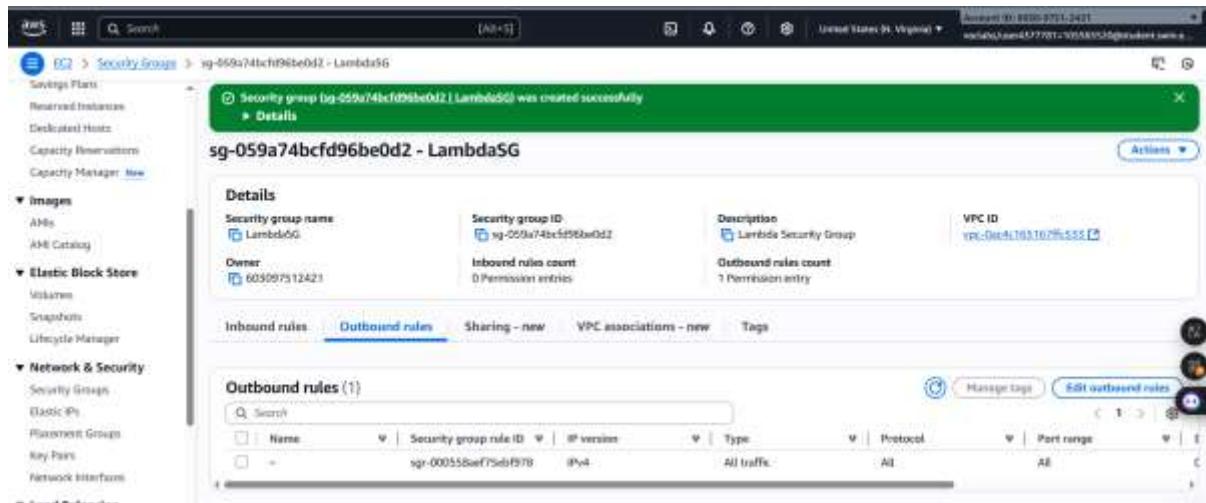


Figure 3: LambdaSG

Step 2.2: Update the DatabaseSG security group.

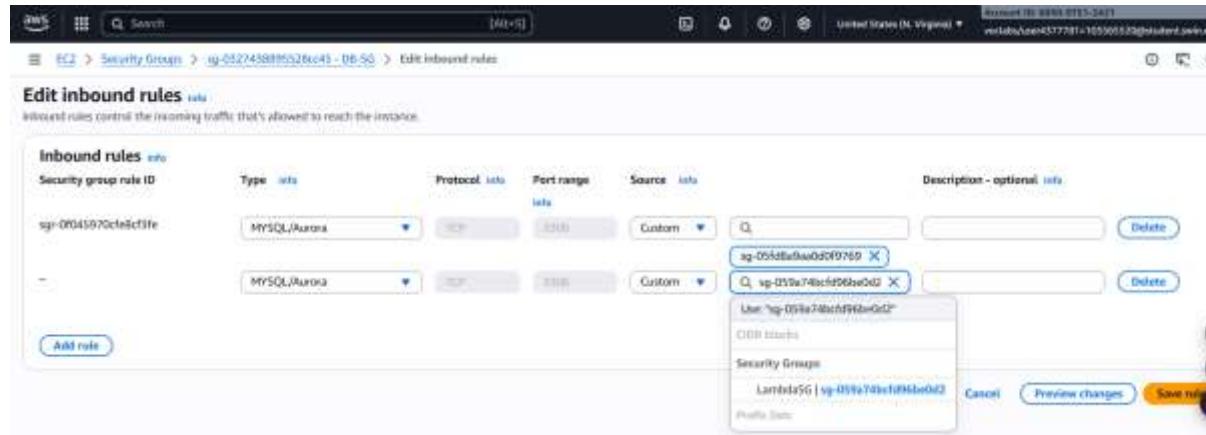


Figure 4: Inbound rule configuration

Step 2.3: Create a Lambda function with the following settings:

- **Function name:** salesAnalysisReportDataExtractor
 - **Runtime:** Python 3.11
 - **Role:** salesAnalysisReportDERole
 - **VPC:**
 - **VPC:** Lab VPC
 - **Subnets:** Private subnet 1 and Private subnet 2
 - **Security Group:** The Lambda function security group that you created
- Tip:** It will take several minutes for the function to be created.

Basic information**Function name**

Enter a name that describes the purpose of your function.

salesAnalysisReportDataExtractor

Function name must be 1 to 64 characters, must be unique to the region, and can't include spaces. Valid characters are a-z, A-Z, 0-9, hyphens (-), and underscores (_).

Runtime 

Choose the language to use to write your function. Note that the console code editor supports only Node.js, Python, and Ruby.

Python 3.11

**Architecture** 

Choose the instruction set architecture you want for your function code.

 arm64 x86_64**Permissions** 

By default, Lambda will create an execution role with permissions to upload logs to Amazon CloudWatch Logs. You can customize this default role later when adding triggers.

▼ Change default execution role**Execution role**Choose a role that defines the permissions of your function. To create a custom role, go to the [IAM console](#).

- Create a new role with basic Lambda permissions.
- Use an existing role
- Create a new role from AWS policy templates

Existing role

Choose an existing role that you've created to be used with this Lambda function. The role must have permission to upload logs to Amazon CloudWatch Logs.

salesAnalysisReportDERole

View the salesAnalysisReportDERole role  or the IAM console**Figure 5: Basic Information configuration****VPC** 

Select a VPC for your resource to access.

vpc-0ac4c165167fffc533 (10.0.0.0/16)

 Allow IPv6 traffic for dual-stack subnets

You can allow outbound IPv6 traffic to subnets that have both IPv4 and IPv6 CIDR blocks.

Subnets

Select the VPC subnets for Lambda to use to set up your VPC configuration.

Choose subnets



subnet-0fb73ab2b13ebc916 (10.0.1.0/24)

us-east-1a aws:cloudformation:logical-id: PrivateSubnet1
aws:cloudformation:stack-id: arn:aws:cloudformation:us-east-1:603097512421:stack/c177567a4588214112437959t1w603097512421/410c3230-b678-11f0-90bd-12259049e4af

aws:cloudformation:stack-name: c177567a4588214112437959t1w603097512421 cloudlab: c177567a4588214112437959t1w603097512421 Name: Private Subnet 1

subnet-0be15e03711df5f8e (10.0.2.0/24)

us-east-1b aws:cloudformation:logical-id: PrivateSubnet2
aws:cloudformation:stack-id: arn:aws:cloudformation:us-east-1:603097512421:stack/c177567a4588214112437959t1w603097512421/410c3230-b678-11f0-90bd-12259049e4af

aws:cloudformation:stack-name: c177567a4588214112437959t1w603097512421 cloudlab: c177567a4588214112437959t1w603097512421 Name: Private Subnet 2

Security groups

Select security groups for your VPC configuration. The following table shows the inbound and outbound rules for your selected security group.

Choose security groups

sg-059a74bcfd96be0d2 (LambdaSG) 

Lambda Security Group

Figure 6: Additional configuration**Step 2.4:** Configure the *DataExtractor* Lambda function as follows:

- **Code:** Upload the *salesAnalysisReportDataExtractor.zip* file
- **Description:** Lambda function to extract data from database
- **Handler:** *salesAnalysisReportDataExtractor.lambda_handler*
- **Memory Size:** 128 MB
- **Timeout (seconds):** 30

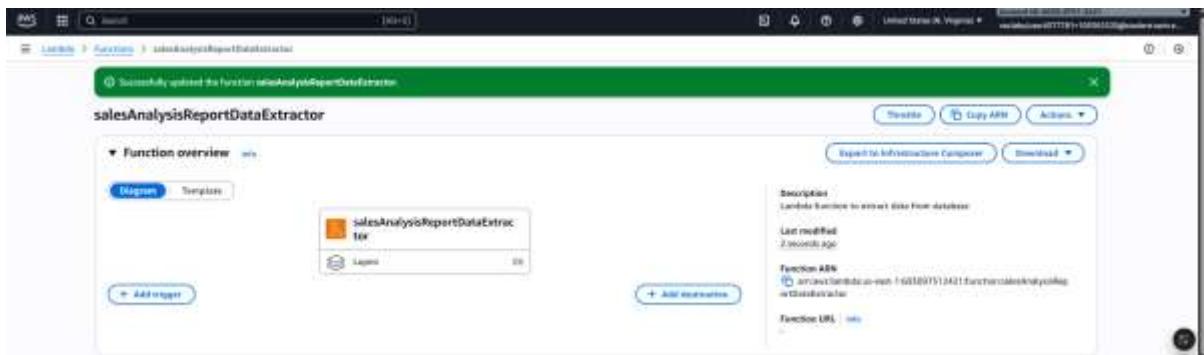


Figure 7: Configuration completed

E. Task 3: Creating the SalesAnalysisReport Lambda function

In this task, you create the Lambda function that generates and sends the daily sales analysis report.

Step 3.1: Create a second Lambda function with the following settings:

- **Function name:** salesAnalysisReport
- **Runtime:** Python 3.11
- **Role:** salesAnalysisReportRole

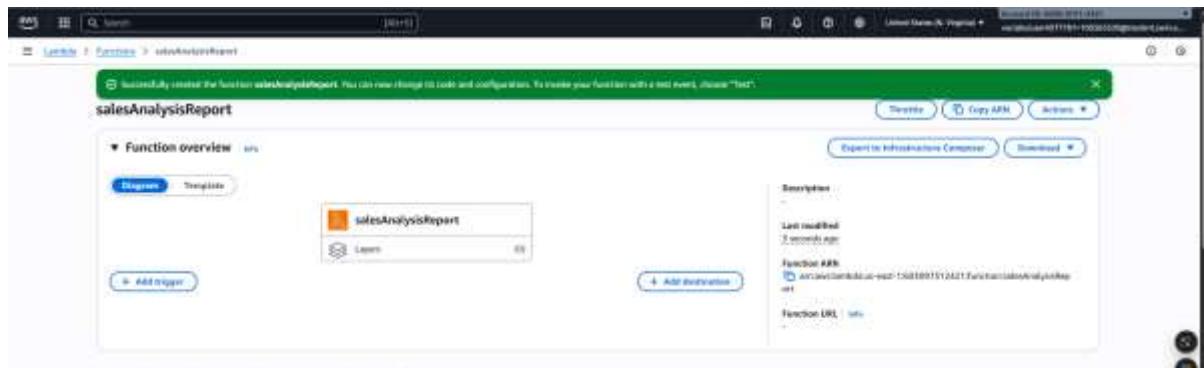


Figure 8: Function created successfully

Step 3.2: Configure the *salesAnalysisReport* Lambda function as follows:

- **Code:** Upload the *salesAnalysisReport.zip* file
- **Description:** Lambda function to generate and send the daily sales report
- **Handler:** *salesAnalysisReport.lambda_handler*
- **Memory Size:** 128 MB
- **Timeout (seconds):** 30

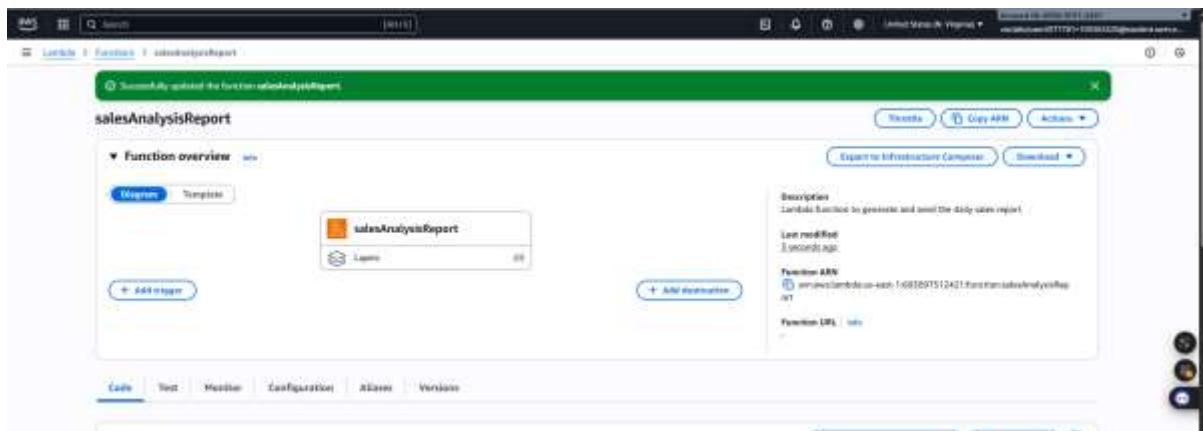


Figure 9: Configuration completed

F. Task 4: Creating an SNS topic

The sales analysis report uses an SNS topic to send the report to email subscribers. In this task, you create an SNS topic and update the environment variables of the *salesAnalysisReport* Lambda function to store the topic Amazon Resource Name (ARN).

Step 4.1: Create a standard SNS topic with the following configuration:

- **Name:** SalesReportTopic
- **Display Name:** Sales Report Topic

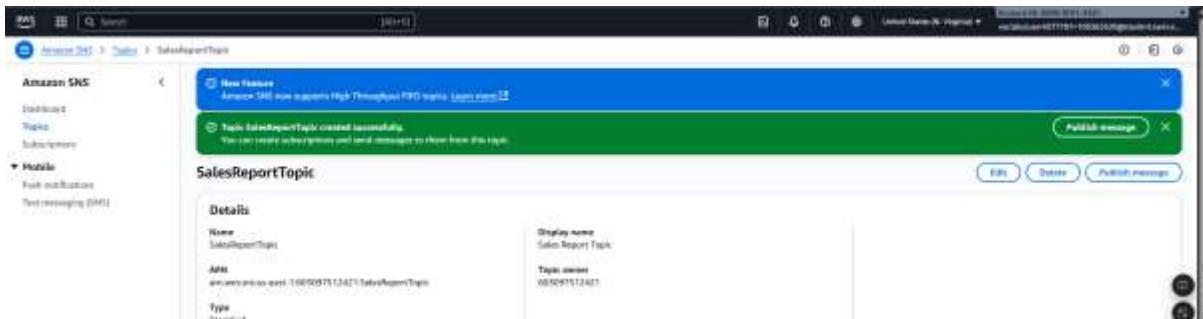


Figure 10: SalesReportTopic created successfully

Step 4.2: Update the *salesAnalysisReport* Lambda function by adding the following environment variable:

- **Variable Name:** topicARN
- **Variable Value:** The ARN of the topic you just created

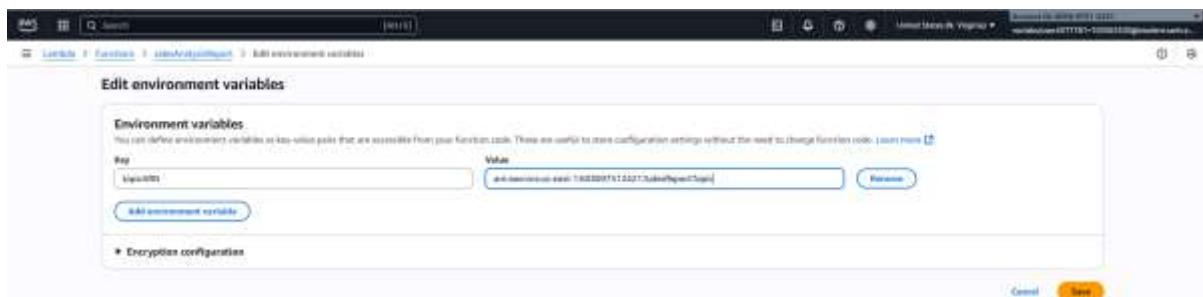


Figure 11: Variable configuration

G. Task 5: Creating an email subscription to the SNS topic

To receive the sales report through email, you must create an email subscription to the topic that you created in the previous task.

Step 5.1: Create a new email subscription to the topic. Use an email address that you can easily access for this lab.

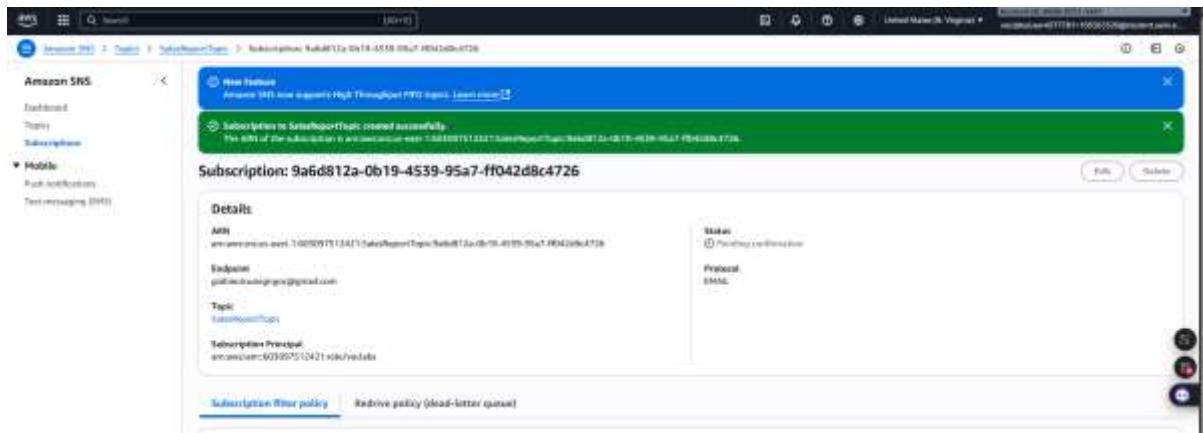


Figure 12: Subscription configured successfully

Step 5.2: Confirm the email subscription from your email client. Note: If you don't receive an email confirmation, check your Junk or Spam folder.

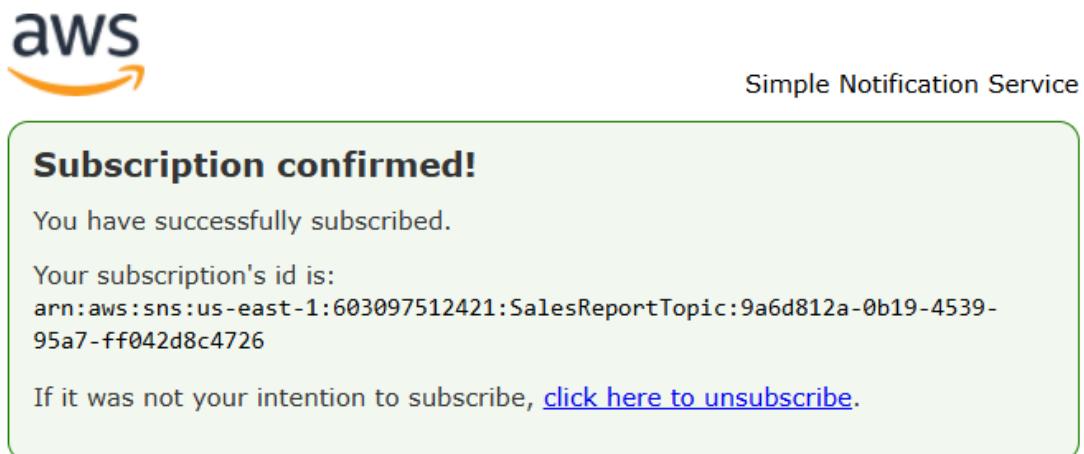


Figure 13: Subscription confirmed

H. Task 6: Testing the salesAnalysisReport Lambda function

Before creating the daily reporting event, you must test that the *salesAnalysisReport* Lambda function works correctly.

Step 6.1: Create a test for the *salesAnalysisReport* Lambda function.

Step 6.2: Run the *salesAnalysisReport* test. If the test succeeds, you should have an email report in a couple of minutes.

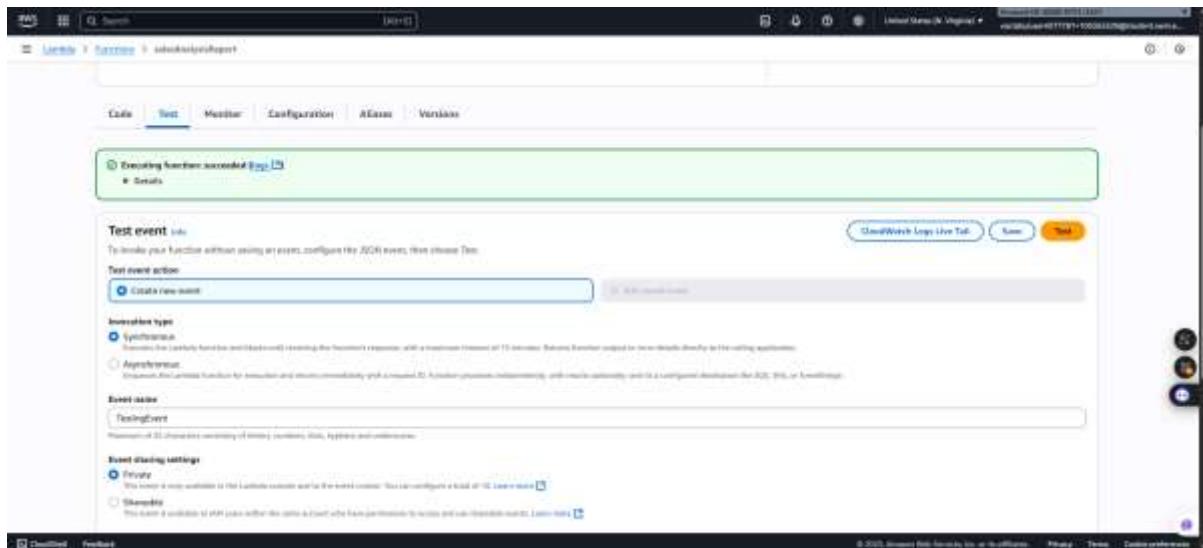


Figure 14: Test success

Sales Analysis Report	
Date: 2025-11-03	
Product Group: Pastries	
Item Name	Quantity
*****	*****
Croissant	29
Donut	23
Chocolate Chip Cookie	18
Muffin	6
Strawberry Blueberry Tart	34
Strawberry Tart	33
Product Group: Drinks	
Item Name	Quantity
*****	*****
Coffee	33
Hot Chocolate	17
Latte	24

Figure 15: Sales Anaylsis Report

I. Task 7: Setting up an Amazon EventBridge event to initiate the Lambda function each day

The last step in this challenge is to set up an event to run the report each day.

Step 7.1: Create a new EventBridge rule that runs the *salesAnalysisReport* Lambda function each day at a specific time. Make sure to choose Continue to create rule.

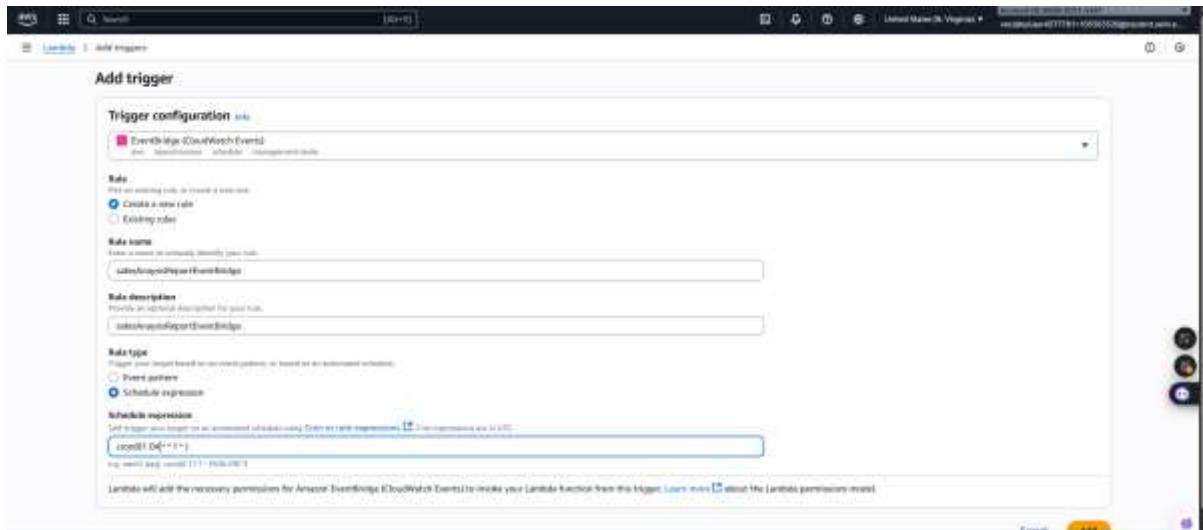


Figure 16: Trigger configuration

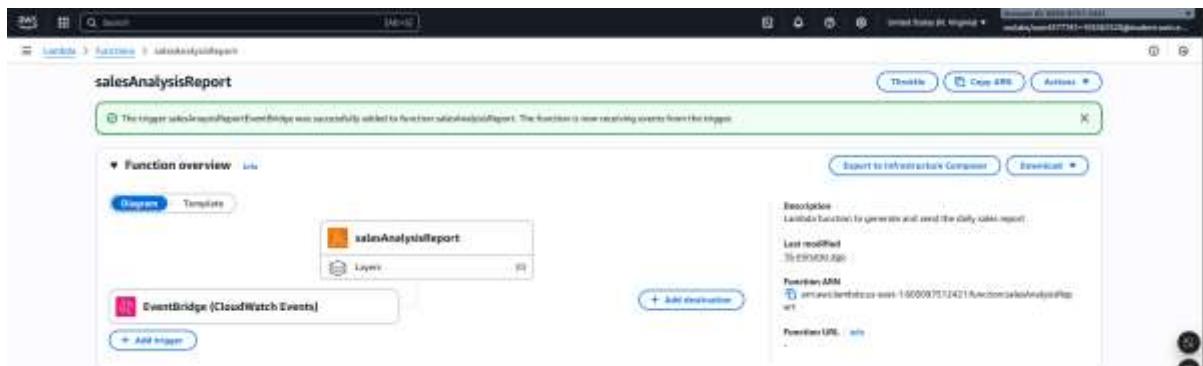


Figure 17: Add trigger successfully

J. Questions

Question 1: Why does the salesAnalysisReportDataExtractor.zip file have a package folder?

- It's an optional folder to improve scaling within the Lambda function.
- The folder contains any Python packages that are used by the Lambda function.
- The folder contains debugging information in Python.
- The folder is a required folder for Lambda functions that are deployed to a VPC.

Question 2: Why must the salesAnalysisReportDataExtractor file be in a VPC?

- The Lambda function must be able to communicate with the rest service instance.
- The Lambda function must be able to communicate with the SNS instance.
- The Lambda function must be set up differently than the other Lambda functions.
- The Lambda function must be able to communicate with an email server.

Question 3: Could the topic ARN be stored as an AWS Systems Manager parameter instead of as an environment variable (assuming that the code could be updated)?

- Yes
- No

Question 4: Will you receive an email message if you do not confirm the topic subscription?

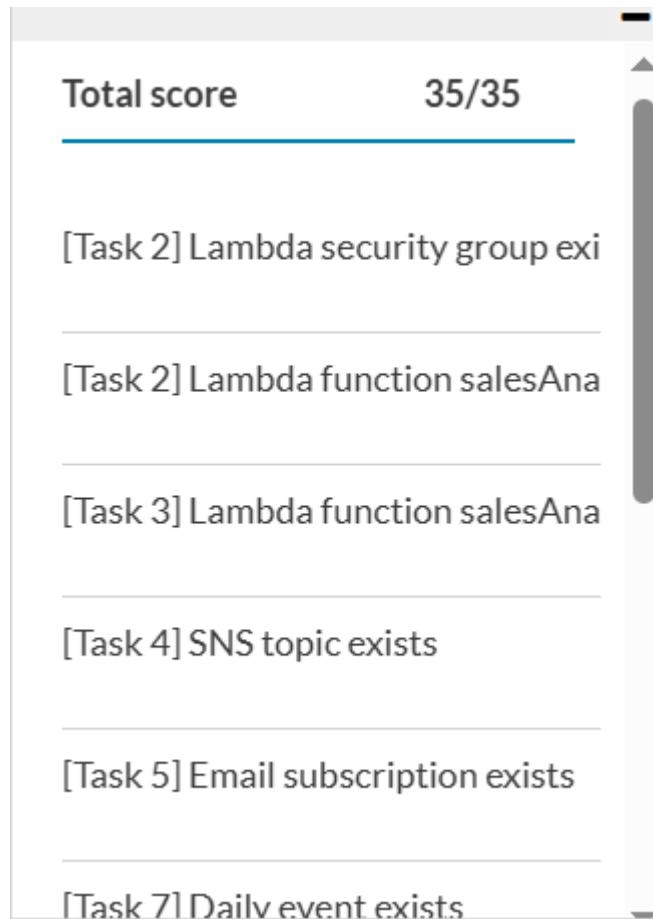
- Yes
- No

Question 5: Frank tells you that he hasn't received an email report in the last few days. What could you do to troubleshoot this issue?

- Restart the Lambda function because it might be stuck.
- Update the Python version.
- Use the logs from Amazon CloudWatch Logs and services that it uses.
- Use the AWS CloudTrail logs and review them for errors.

Figure 18: Questions

K. Task Completed

*Figure 19: Task Completed*

L. Conclusion

This challenge successfully addressed the performance limitations of generating resource-intensive daily sales reports directly on the production web server's EC2 instance. The initial reliance on a local cron job negatively impacted application performance and was not cost-optimized for a once-a-day task.

To resolve this, a modern, cost-effective serverless architecture was implemented utilizing AWS services. The core solution involved decoupling the task and using AWS Lambda (for secure RDS data extraction and report formatting), Amazon SNS (for reliable email distribution via the SalesReportTopic), and Amazon EventBridge (to schedule the daily report generation automatically).

By leveraging AWS Lambda, the solution successfully transitioned a business-critical yet resource-heavy task to a fully managed, pay-per-use environment, significantly reducing operational costs compared to running a dedicated EC2 instance 24/7. This implementation improves the overall performance and stability of the main café application while ensuring Frank and Martha receive their sales analysis reports reliably each morning.