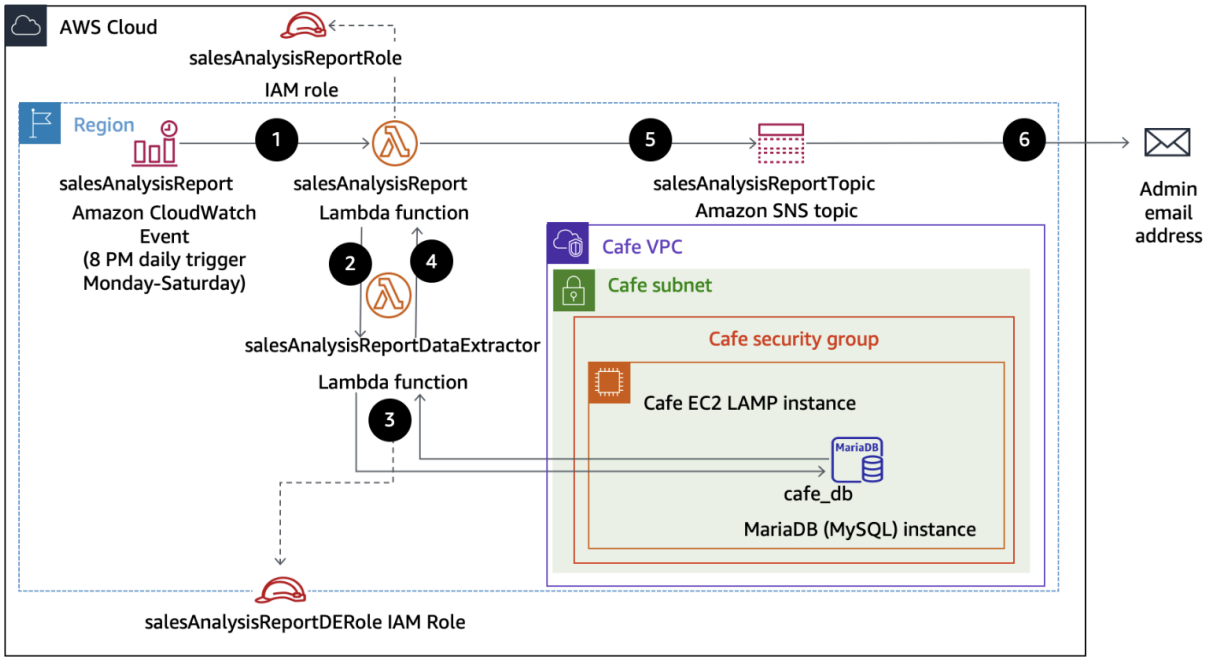
Working with AWS Lambda

Lab overview

In this lab, you deploy and configure an AWS Lambda based serverless computing solution. The Lambda function generates a sales analysis report by pulling data from a database and emailing the results daily. The database connection information is stored in Parameter Store, a capability of AWS Systems Manager. The database itself runs on an Amazon Elastic Compute Cloud (Amazon EC2) Linux, Apache, MySQL, and PHP (LAMP) instance.

The following diagram shows the architecture of the sales analysis report solution and illustrates the order in which actions occur.



The diagram includes the following function steps:

Step	Details
1	An Amazon CloudWatch Events event calls the salesAnalysisReport Lambda function at 8 PM every day Monday through Saturday.
2	The salesAnalysisReport Lambda function invokes another Lambda function, salesAnalysisReportDataExtractor, to retrieve the report data.
3	The salesAnalysisReportDataExtractor function runs an analytical query against the café database (cafe_db).

4	The query result is returned to the salesAnalysisReport function.
5	The salesAnalysisReport function formats the report into a message and publishes it to the salesAnalysisReportTopic Amazon Simple Notification Service (Amazon SNS) topic.
6	The salesAnalysisReportTopic SNS topic sends the message by email to the administrator.

In this lab, the Python code for each Lambda function is provided to you so that you can focus on the SysOps tasks of deploying, configuring, and testing the serverless solution components.

Objectives

After completing this lab, you will be able to do the following:

- Recognize necessary AWS Identity and Access Management (IAM) policy permissions to facilitate a Lambda function to other Amazon Web Services (AWS) resources.
- Create a Lambda layer to satisfy an external library dependency.
- Create Lambda functions that extract data from database, and send reports to user.
- Deploy and test a Lambda function that is initiated based on a schedule and that invokes another function.
- Use CloudWatch logs to troubleshoot any issues running a Lambda function.

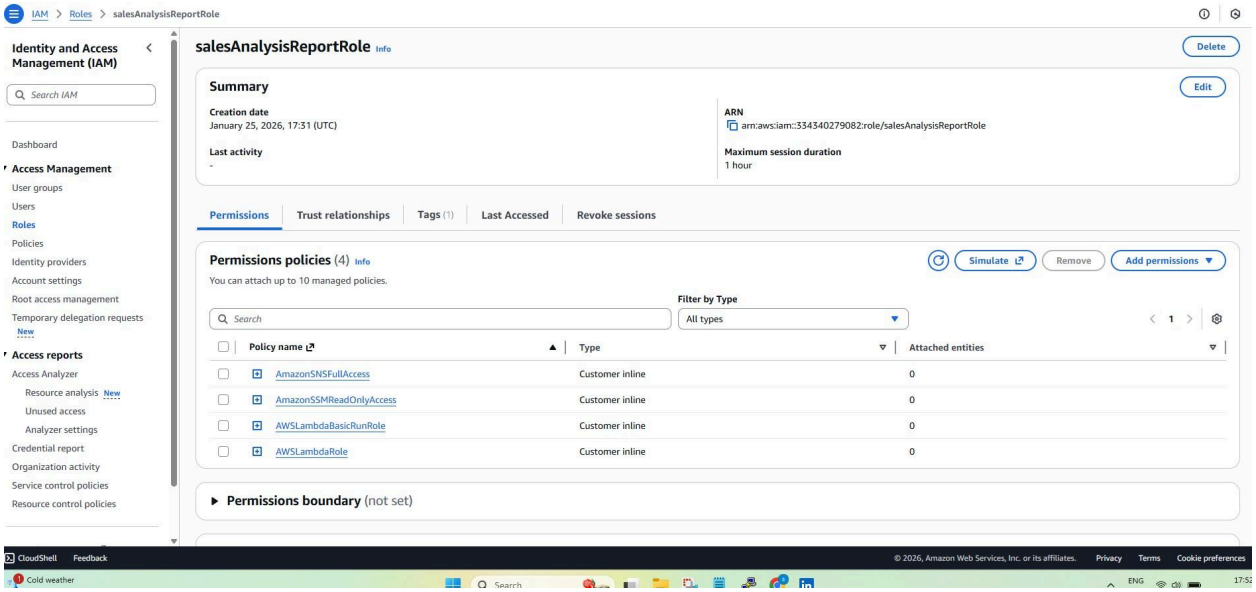
Task 1: Observing the IAM role settings

In this lab, you create two Lambda functions. Each function requires permissions to access the AWS resources with which they interact.

In this task, you analyze the IAM roles and the permissions that they grant to the salesAnalysisReport and salesAnalysisReportDataExtractor Lambda functions that you create later.

Task 1.1: Observing the salesAnalysisReport IAM role settings

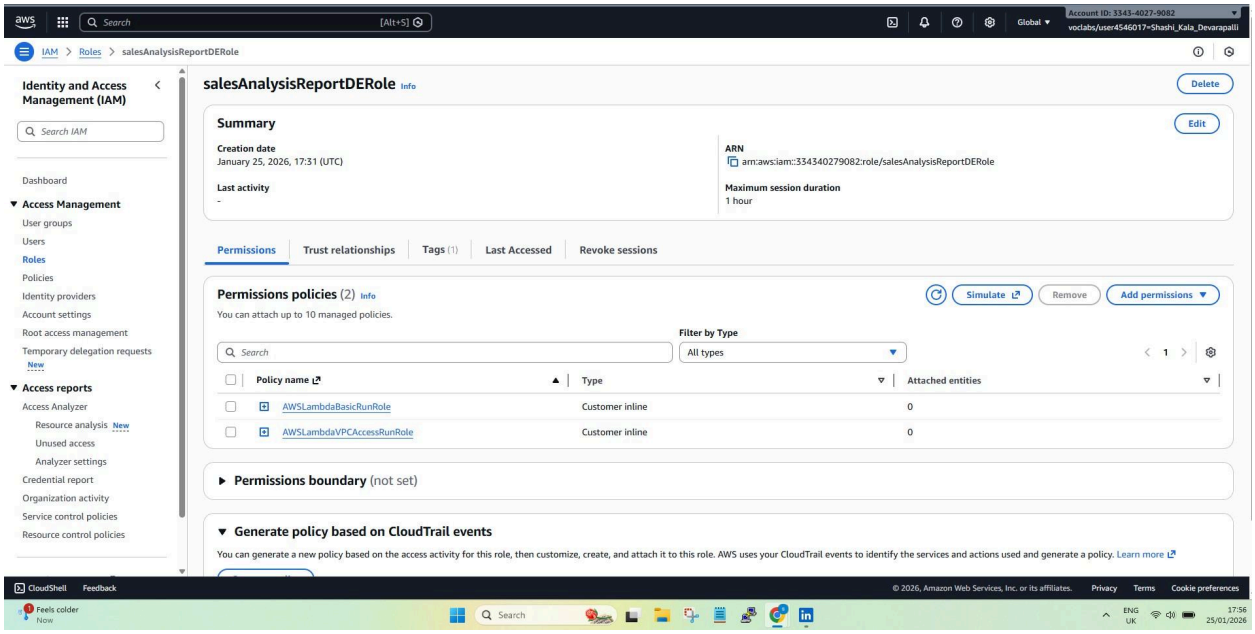
5. In the AWS Management Console, choose **Services** > **Security, Identity, & Compliance** > **IAM**.
6. In the navigation pane, choose **Roles**.
7. In the search box, enter `sales`
8. From the filtered results, choose the **salesAnalysisReportRole** hyperlink.
9. Choose the **Trust relationships** tab, and notice that lambda.amazonaws.com is listed as a trusted entity, which means that the Lambda service can use this role.
10. Choose the **Permissions** tab, and notice the four policies assigned to this role. To expand each role and analyze the permissions that each policy grants, choose the **+** icon next to each role:
    - **AmazonSNSFullAccess** provides full access to Amazon SNS resources.
    - **AmazonSSMReadOnlyAccess** provides read-only access to Systems Manager resources.
    - **AWSLambdaBasicRunRole** provides write permissions to CloudWatch logs (which are required by every Lambda function).
    - **AWSLambdaRole** gives a Lambda function the ability to invoke another Lambda function.
11. The salesAnalysisReport Lambda function that you create later in this lab uses the salesAnalysisReportRole role.



Task 1.2: Observing the salesAnalysisReportDERole IAM role settings

11. Choose **Roles** again.
12. In the search box, enter `sales`
13. From the filtered results, choose the **salesAnalysisReportDERole** hyperlink.
14. Choose the **Trust relationships** tab, and notice that lambda.amazonaws.com is listed as a trusted entity.
15. Choose the **Permissions** tab, and notice the permissions granted to this role:
    - **AWSLambdaBasicRunRole** provides write permissions to CloudWatch logs.
    - **AWSLambdaVPCAccessRunRole** provides permissions to manage elastic network interfaces to connect a function to a virtual private cloud (VPC).

The salesAnalysisReportDataExtractor Lambda function that you create next uses the salesAnalysisReportDERole role.



Task 2: Creating a Lambda layer and a data extractor Lambda function

In this task, you first create a Lambda layer, and then you create a Lambda function that uses the layer.

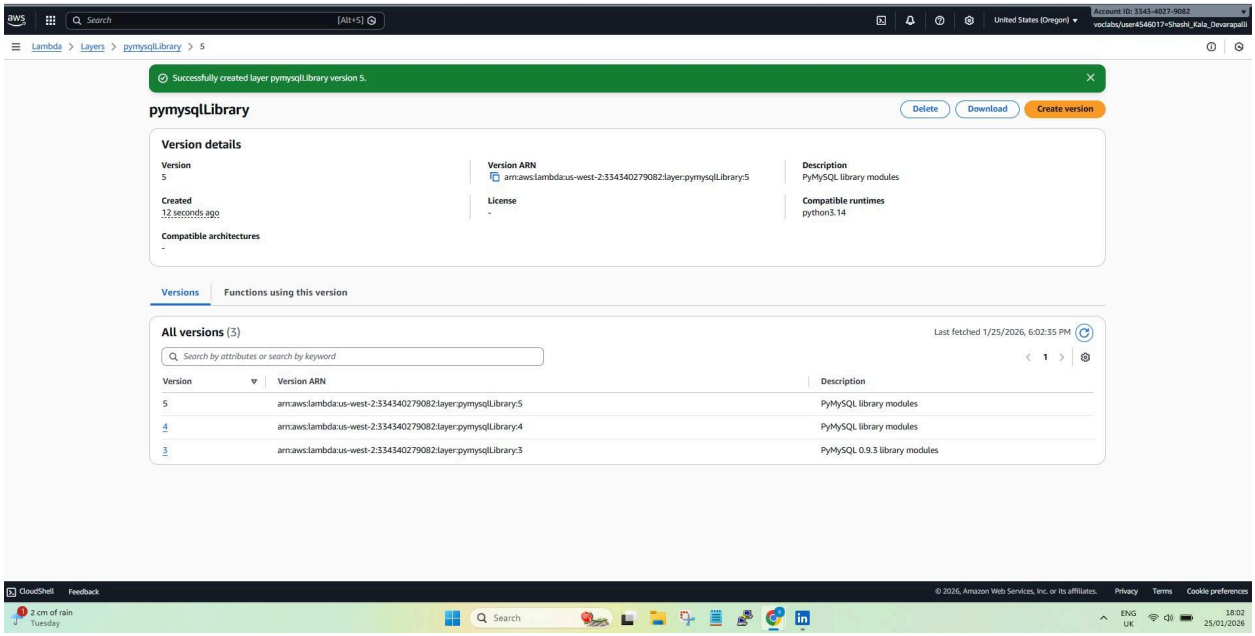
Start by downloading two required files.

16. To download the lab files required by this task to your local machine, choose the following links:
    pymysql-v3.zip
    salesAnalysisReportDataExtractor-v3.zip
    Note: The salesAnalysisReportDataExtractor-v3.zip file is a Python implementation of a Lambda function that makes use of the PyMySQL open-source client library to access the MySQL café database. This library has been packaged into the pymysql-v3.zip which is uploaded to Lambda layer next.

Task 2.1: Creating a Lambda Layer

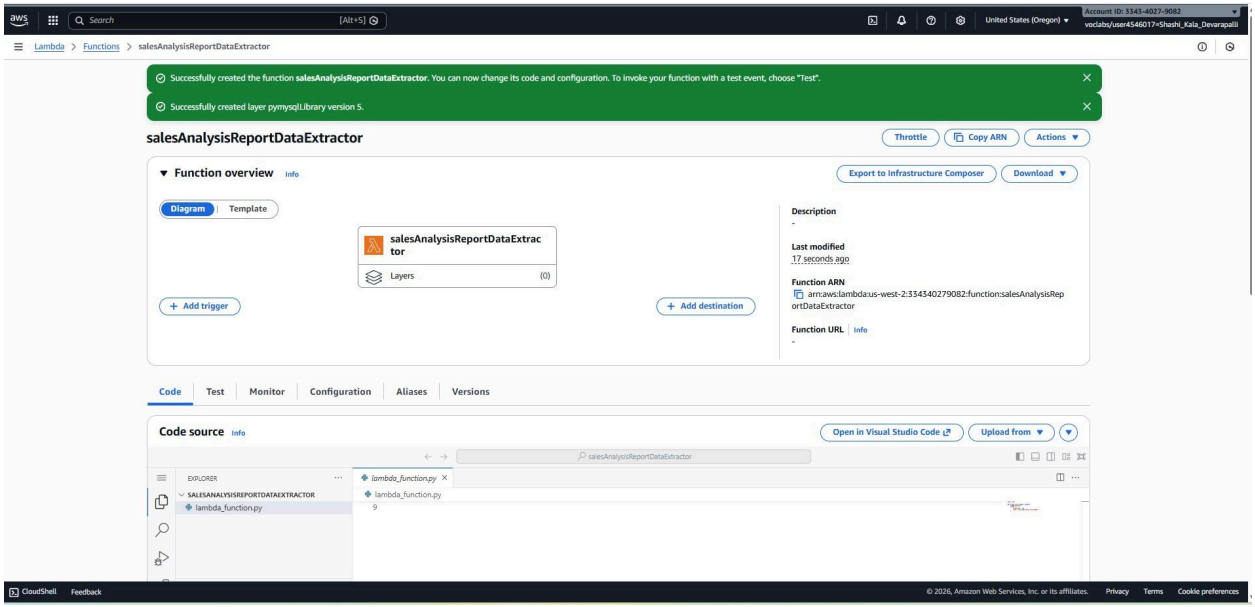
In the next steps, you create a Lambda layer named pymysqlLibrary and upload the client library into it so that it can be used by any function that requires it. Lambda layers provide a flexible mechanism to reuse code between functions so that the code does not have to be included in each function's deployment package.

17. In the AWS Management Console, choose **Services** > **Compute** > **Lambda**.
    Tip: If the navigation panel is closed, choose the collapsed menu icon (three horizontal lines) to open the **AWS Lambda** panel.
18. Choose **Layers**.
19. Choose **Create layer**.
20. Configure the following layer settings:
    - For **Name**, enter `pymysqlLibrary`
    - For **Description**, enter `PyMySQL library modules`
    - Select **Upload a .zip file**. To upload the pymysql-v3.zip file, choose **Upload**, navigate to the folder where you downloaded the pymysql-v3.zip file, and open it.
    - For **Compatible runtimes**, choose **Python 3.9**.
21. Choose **Create**.
    The message "Successfully created layer pymysqlLibrary version 1" is displayed.
    Tip: The Lambda layers feature requires that the .zip file containing the code or library conform to a specific folder structure. The pymysqlLibary.zip file used in this lab was packaged using the following folder structure:



22.

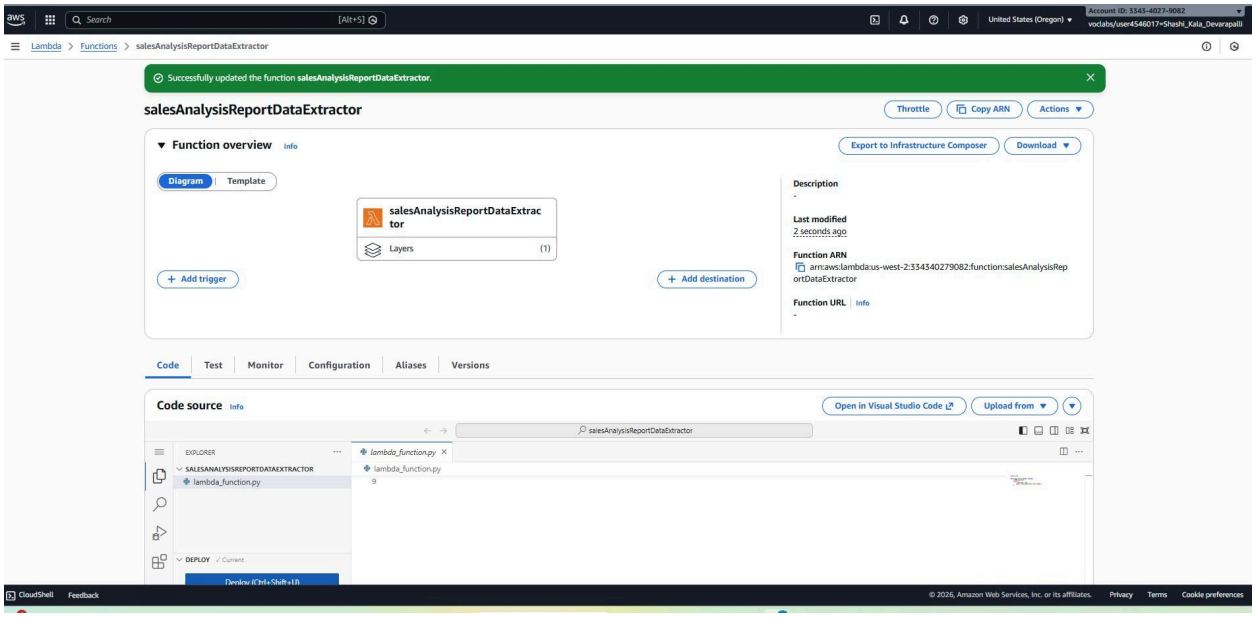
Task 2.2: Creating a data extractor Lambda function

22. In the navigation pane, choose **Functions** to open the **Functions** dashboard page.
23. Choose **Create function**, and configure the following options:
    - At the top of the **Create function** page, select **Author from scratch**.
    - For **Function name**, enter `salesAnalysisReportDataExtractor`
    - For **Runtime**, choose **Python 3.9**.
    - Expand **Change default execution role**, and configure the following options:
        - For **Execution role**, choose **Use an existing role**.
        - For **Existing role:**, choose **salesAnalysisReportDERole**.
24. Choose **Create function**.
    A new page opens with the following message: "Successfully created the function salesAnalysisReportDataExtractor."
23. 

Task 2.3: Adding the Lambda layer to the function

25. In the **Function overview** panel, choose **Layers**.
26. At the bottom of the page, in the **Layers** panel, choose **Add a layer**.
27. On the **Add layer** page, configure the following options:

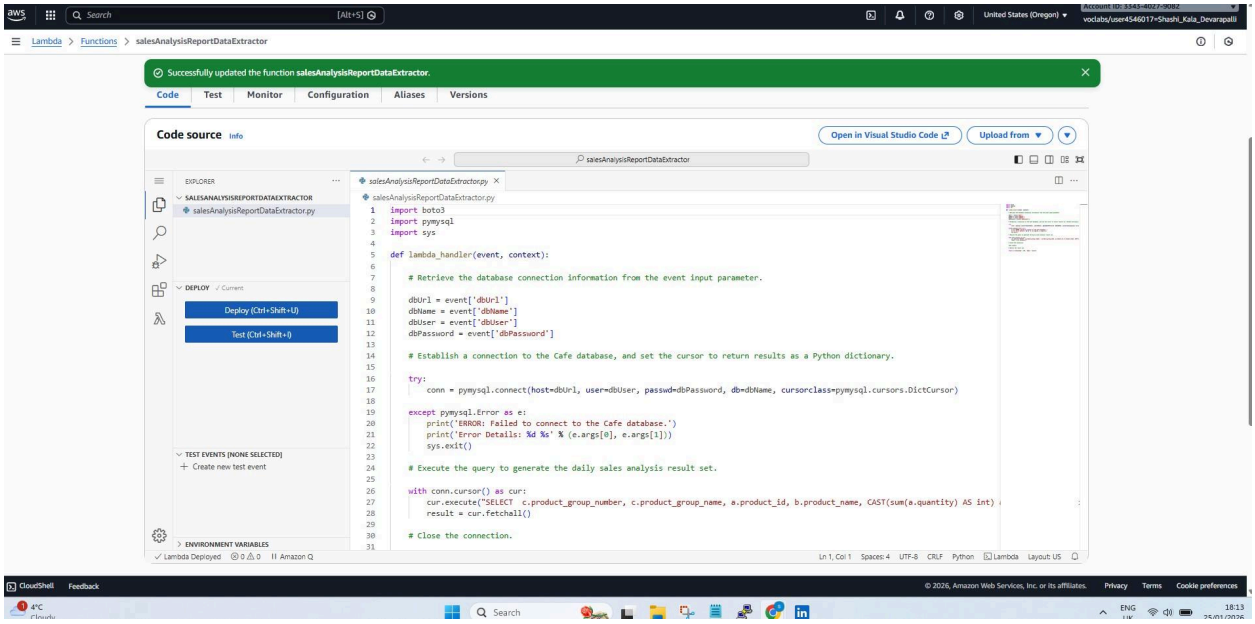
- **Choose a layer**: Choose **Custom layers**.
- **Custom layers**: Choose **pymysqlLibrary**.
- **Version**: Choose **1**.

28. Choose **Add**.
    The **Function overview** panel shows a count of **(1)** in the **Layers** node for the function.
24. 

Task 2.4: Importing the code for the data extractor Lambda function

29. Go to the **Lambda** > **Functions** > **salesAnalysisReportDataExtractor** page.
30. In the **Runtime settings** panel, choose **Edit**.
31. For **Handler**, enter `salesAnalysisReportDataExtractor.lambda_handler`
32. Choose **Save**.
33. In the **Code source** panel, choose **Upload from**.
34. Choose **.zip file**.
35. Choose **Upload**, and then navigate to and select the **salesAnalysisReportDataExtractor-v3.zip** file that you downloaded earlier.
36. Choose **Save**.
    The Lambda function code is imported and displays in the **Code source** panel. If necessary, in the **Environment** navigation pane, double-click **salesAnalysisReportDataExtractor.py** to display the code.
37. Review the Python code that implements the function.
    Note: If the code does not yet display in the function code editor, refresh the

console so that it displays.
Read the comments included in the code to gain an understanding of its logic flow. Notice that the function expects to receive the database connection information (dbURL, dbName, dbUser, and dbPassword) in the event input parameter.

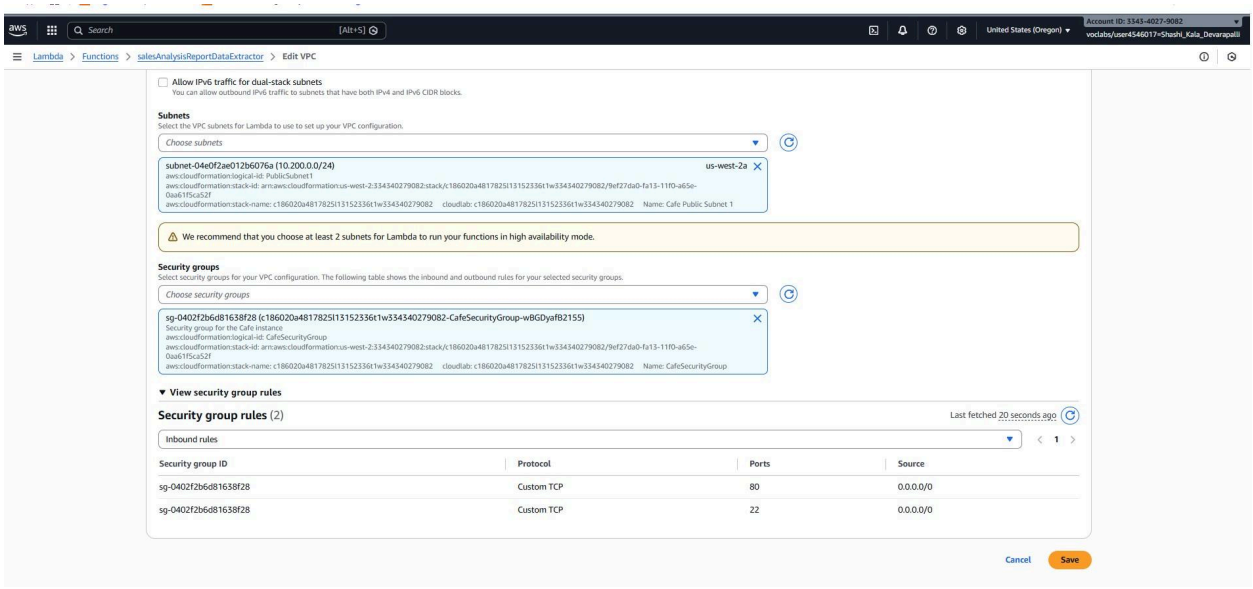


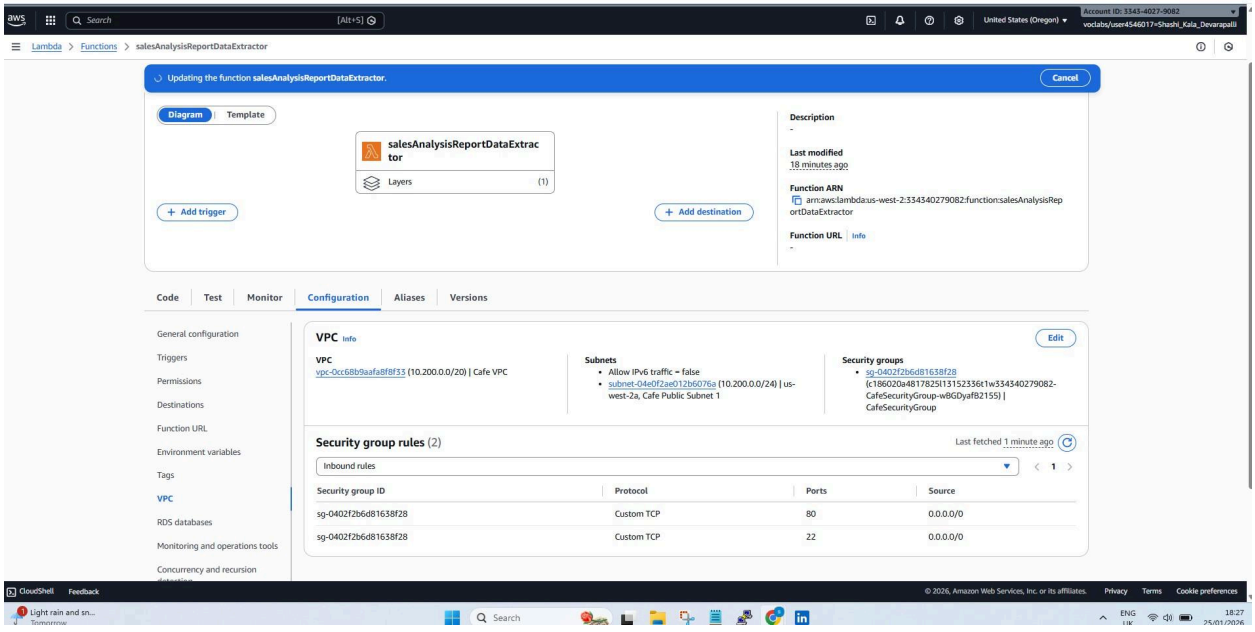
Task 2.5: Configuring network settings for the function

The final step before you can test the function is to configure its network settings. As the architecture diagram at the start of this lab shows, this function requires network access to the café database, which runs in an EC2 LAMP instance. Therefore, you need to specify the instance's VPC, subnet, and security group information in the function's configuration.

38. Choose the **Configuration** tab, and then choose **VPC**.
39. Choose **Edit**, and configure the following options:
    - **VPC**: Choose the option with **Cafe VPC** as the **Name**.
    - **Subnets**: Choose the option with **Cafe Public Subnet 1** as the **Name**.
      Tip: You can ignore the warning (if any) that recommends choosing at least two subnets to run in high availability mode because it is not applicable to the function.
    - **Security groups**: Choose the option with **CafeSecurityGroup** as the **Name**.

40. Notice that the security group's inbound and outbound rules are automatically displayed following the field.
41. Choose **Save**.

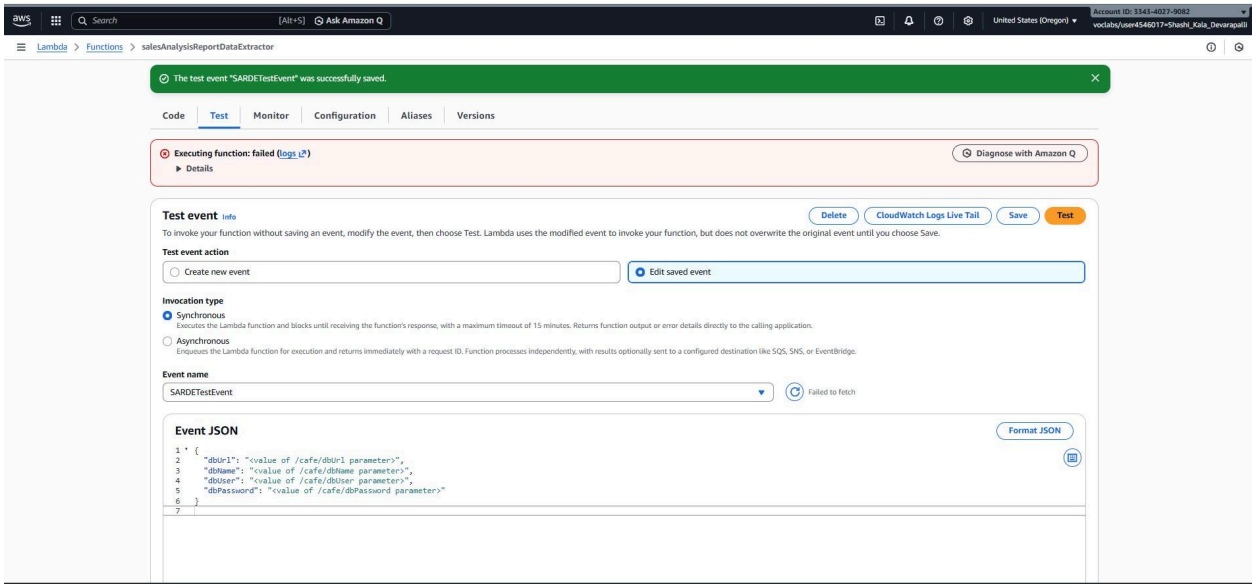


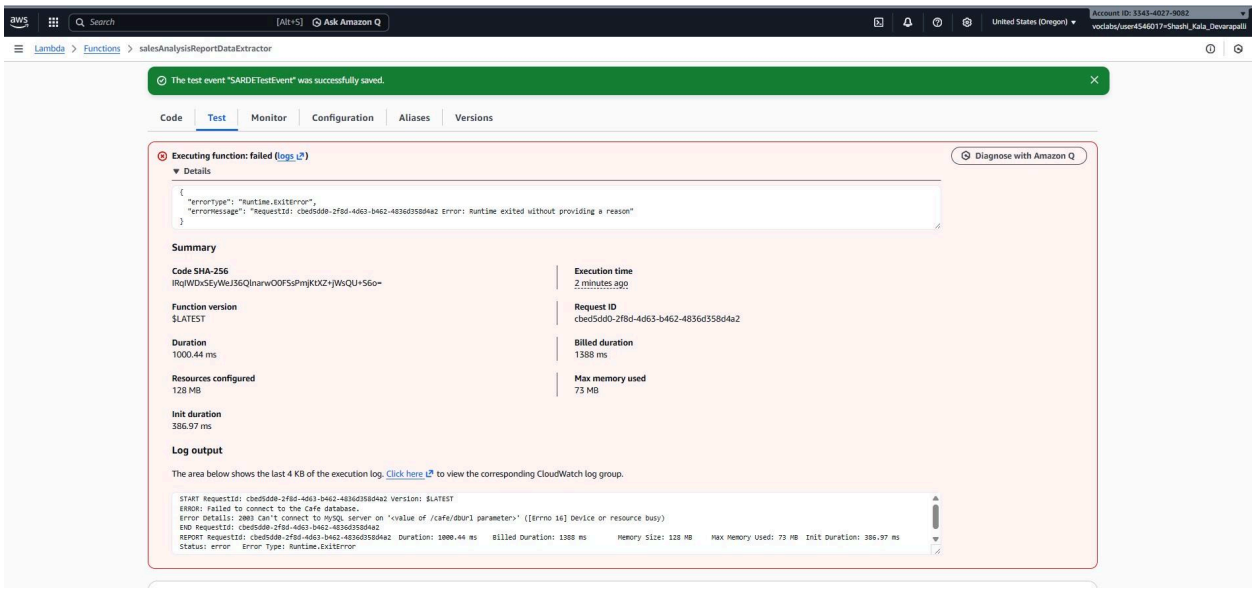


Task 3: Testing the data extractor Lambda function

Task 3.1: Launching a test of the Lambda function

You are now ready to test the salesAnalysisReportDataExtractor function. To invoke it, you need to supply values for the café database connection parameters. Recall that these are stored in Parameter Store.

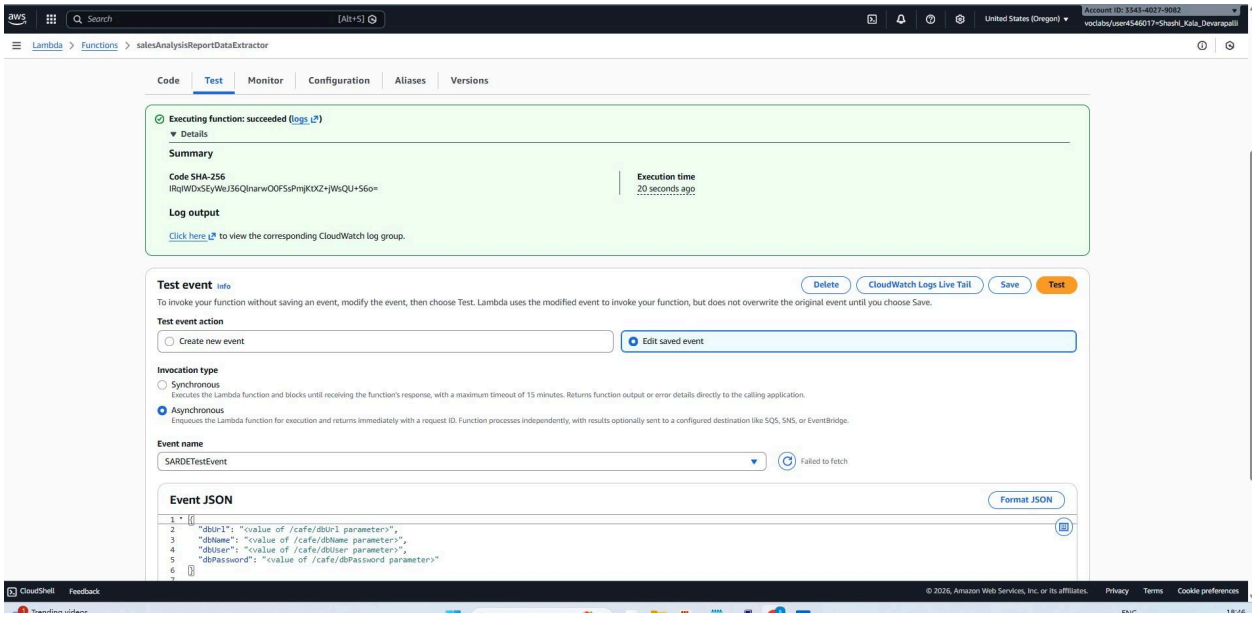
41. On a new browser tab, open the AWS Management Console, and choose **Services** > **Management & Governance** > **Systems Manager**.
42. In the navigation pane, choose **Parameter Store**.
43. Choose each of the following parameter names, and copy and paste the **Value** of each one into a text editor document:
    - /cafe/dbUrl
    - /cafe/dbName
    - /cafe/dbUser
    - /cafe/dbPassword
44. Return to the **Lambda Management Console** browser tab. On the **salesAnalysisReportDataExtractor** function page, choose the **Test** tab.
45. Configure the **Test event** panel as follows:
    - For **Test event action**, select **Create new event**.
    - For **Event name**, enter `SARDETestEvent`
    - For **Template**, choose **hello-world**.
    - In the **Event JSON** pane, replace the JSON object with the following JSON object:
    - ```
      {
        "dbUrl": "<value of /cafe/dbUrl parameter>",
        "dbName": "<value of /cafe/dbName parameter>",
        "dbUser": "<value of /cafe/dbUser parameter>",
        "dbPassword": "<value of /cafe/dbPassword parameter>"
      }
      ```
    - In this code, substitute the value of each parameter with the values that you pasted into a text editor in the previous steps. Enclose these values in quotation marks.
46. Choose **Save**.
      ```

47. Choose **Test**.
    After a few moments, the page shows the message "Execution result: failed".



Task 3.2: Troubleshooting the data extractor Lambda function

48. In the **Execution result** pane, choose **Details** to expand it, and notice that the error object returned a message similar to the following message after the function ran:
49. ```
    {
      "errorMessage": "2019-02-14T04:14:15.282Z ff0c3e8f-1985-44a3-8022-519f883c8412 Task timed out after 3.00 seconds"
    }
    ```
50. This message indicates that the function timed out after 3 seconds.
    The **Log output** section includes lines starting with the following keywords:
    - **START** indicates that the function started running.
    - **END** indicates that the function finished running.
    - **REPORT** provides a summary of the performance and resource utilization statistics related to when the function ran.
51. What caused this error?
    ```



Task 3.3: Analyzing and correcting the Lambda function

49. In this task, you analyze and correct the issue observed when you tested the Lambda function.
    Here are a few hints to help you find the solution:
    - One of the first things that this function does is connect to the MySQL database running in a separate EC2 instance. It waits a certain amount of time to establish a successful connection. After this time passes, if the connection is unsuccessful, the function times out.
    - By default, a MySQL database uses the MySQL protocol and listens on port number 3306 for client access.
    - Choose the **Configuration** tab again, and choose **VPC**. Notice the **Inbound rules** for the security group that are used by the EC2 instance running the database. Is the database port number (3306) listed? You can choose the security group link if you want to edit and add an inbound rule to it.
50. Once you have corrected the problem, return to browser tab with the **salesAnalysisReportDataExtractor** function page. Choose the **Test** tab, and choose **Test** again.
    You should now see a green box showing the message “Execution result: succeeded (logs).” This message indicates that the function ran successfully.
51. Choose **Details** to expand it.
    The function returned the following JSON object:

52.
```
{
  "statusCode": 200,
  "body": []
}
```
53. The body field, which contains the report data that the function extracted, is empty because there is no order data in the



Task 3.4: Placing an order and testing again

In this task, you access the café website and place some orders to populate data in the database.

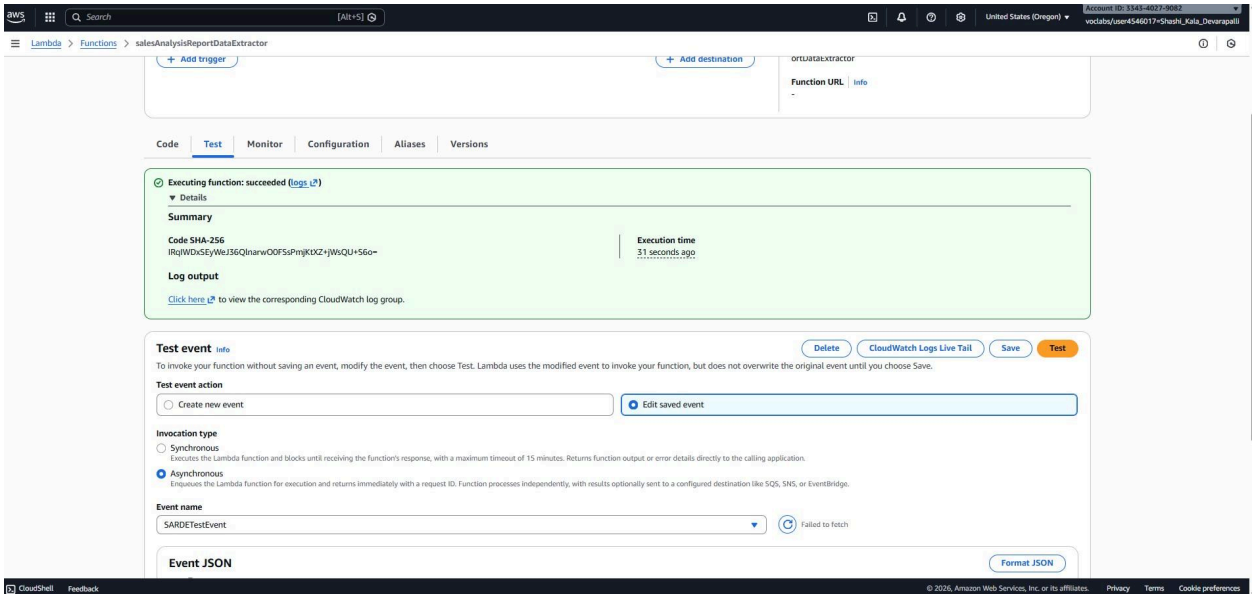
52. To open the café website on a new browser tab, find the public IP address of the café EC2 instance.
    The URL for the website has the format http://publicIP/cafe where **publicIP** is the public IP address of the café EC2 instance. There are two ways to find the public IP address:
    Option 1:
    - On the AWS Management Console, choose **Services** > **Compute** > **EC2**.
    - In the navigation pane, choose **Instances**.
    - Choose **CafeInstance**.
    - Copy the **Public IPv4 address** to a text editor.
    - On a new browser tab, enter http://publicIP/cafe, and replace *publicIP* with the public IPv4 address that you just copied to a text editor.
    - Press Enter to load the café website.

53. Option 2:
    - At the top of these instruction, choose Details, and then choose Show.
    - From the **Credentials** window, copy and paste the **CafePublicIP** to a text editor.
    - On a new browser tab, enter http://publicIP/cafe, and replace *publicIP* with the public IPv4 address that you just copied to a text editor.
    - Press Enter to load the café website.
54. On the café website, choose **Menu**, and place some orders to populate data in the database.

    Now that there is order data in the database, you test the function again.
55. Go to the browser tab with the **salesAnalysisReportDataExtractor** function page.
56. Choose the **Test** tab, and choose **Test**.

    The returned JSON object now contains product quantity information in the body field similar to the following:
57.
```
{
  "statusCode": 200,
  "body": [
    {
      "product_group_number": 1,
      "product_group_name": "Pastries",
      "product_id": 1,
      "product_name": "Croissant",
      "quantity": 1
    },
    {
      "product_group_number": 2,
      "product_group_name": "Drinks",
      "product_id": 8,
      "product_name": "Hot Chocolate",
      "quantity": 2
     }
    ]
}
```

58. Congratulations! You have successfully created the salesAnalysisReportDataExtractor Lambda function.



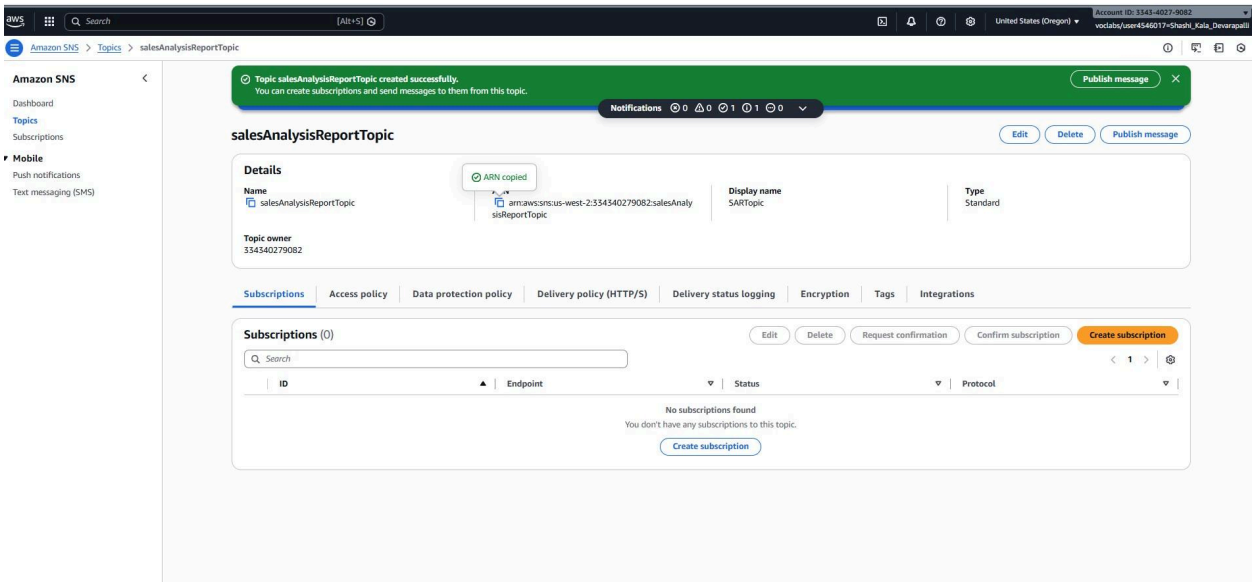
Task 4: Configuring notifications

In this task, you create an SNS topic and then subscribe an email address to the topic.

Task 4.1: Creating an SNS topic

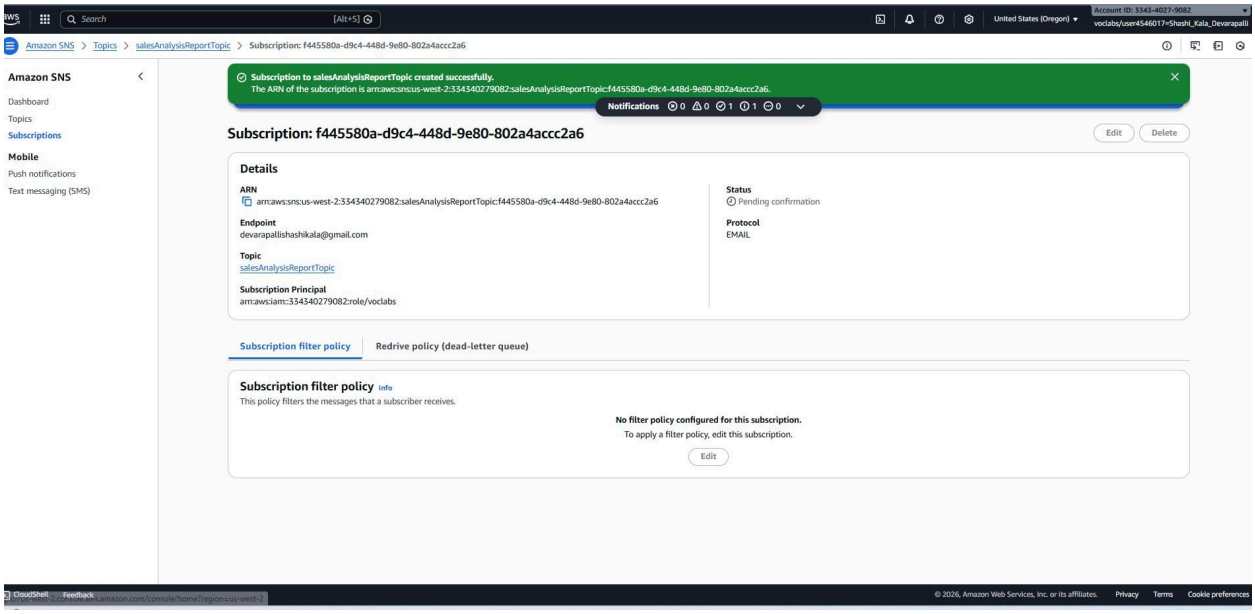
In this task, you create the SNS topic where the sales analysis report is published and subscribe an email address to it. The topic is responsible for delivering any message it receives to all of its subscribers. You use the Amazon SNS console for this task.

56. On the AWS Management Console, choose **Services** > **Application Integration** > **Simple Notification Service**.
57. In the navigation pane, choose **Topics**, and then choose **Create topic**.
    Note: If the **Topics** link is not visible, choose the three horizontal lines icon, and then choose **Topics**.
    The **Create topic** page opens.
58. Configure the following options:
    - **Type**: Choose **Standard**.
    - **Name**: Enter `salesAnalysisReportTopic`
    - **Display name**: Enter `SARTopic`
59. Choose **Create topic**.
60. Copy and paste the **ARN** value into a text editor document.
    You need to specify this ARN when you configure the next Lambda function.



Task 4.2: Subscribing to the SNS topic

61. Choose **Create subscription**, and configure the following options:
    - **Protocol**: Choose **Email**.
    - **Endpoint**: Enter an email address that you can access.
62. Choose **Create subscription**.
    The subscription is created and has a **Status** of *Pending confirmation*.
63. Check the inbox for the email address that you provided.
    You should see an email from SARTopic with the subject "AWS Notification - Subscription Confirmation."
64. Open the email, and choose **Confirm subscription**.
    A new browser tab opens and displays a page with the message "Subscription confirmed!"



Task 5: Creating the salesAnalysisReport Lambda function

Next, you create and configure the salesAnalysisReport Lambda function. This function is the main driver of the sales analysis report flow. It does the following:

- Retrieves the database connection information from Parameter Store
- Invokes the salesAnalysisReportDataExtractor Lambda function, which retrieves the report data from the database
- Formats and publishes a message containing the report data to the SNS topic

Task 5.1: Connecting to the CLI Host instance

In this task, you use EC2 Instance Connect to log in to the CLI Host instance running in your AWS account that already has the AWS Command Line Interface (AWS CLI) installed and the the Python code needed to create the next Lambda function. You then then run an AWS CLI command to create the Lambda function. Finally, you unit test the new function using the Lambda management console.

65. On the **EC2 Management Console**, in the navigation pane, choose **Instances**.
66. From the list of EC2 instances, choose the check box for the **CLI Host** instance.
67. Choose **Connect**.
68. On the **EC2 Instance Connect** tab, choose **Connect** to connect to the CLI Host.

Task 5.2: Configuring the AWS CLI

Amazon Linux instances have the AWS CLI pre-installed; however, you still need to supply credentials to connect the AWS CLI client to an AWS account.

69. In the EC2 Instance Connect terminal window, run the following command to update the AWS CLI software with the credentials:
70. `aws configure`
71. At the prompts, enter the following information:
    - **AWS Access Key ID**: At the top of these instructions, choose the Details dropdown menu, and then choose Show. A **Credentials** window opens. From this window, copy the **AccessKey** value, paste it into the terminal window, and press Enter.
    - **AWS Secret Access Key**: From the same **Credentials** window, copy the **SecretKey** value, paste it into the terminal window, and press Enter.
    - **Default region name**: Enter the Region code for the Region where you created the previous Lambda function. For this lab, enter the `us-west-2` Region code into the terminal window, and press Enter. To find this code, in the upper-right menu of the Lambda management console, choose the Region dropdown menu.
    - **Default output format**: Enter `json` and press Enter.

Task 5.3: Creating the salesAnalysisReport Lambda function using the AWS CLI

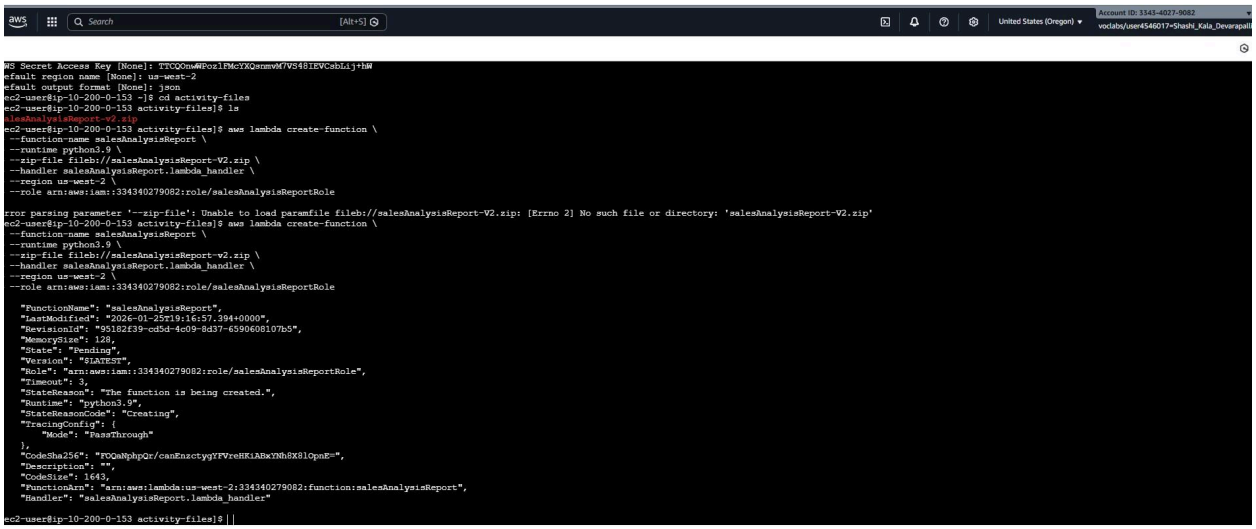
71. To verify that the salesAnalysisReport-v2.zip file containing the code for the salesAnalysisReport Lambda function is already on the CLI Host, run the following commands in the terminal:
72. `cd activity-files`
    `ls`
73. **Note:** Before you create the function, you must retrieve the ARN of the salesAnalysisReportRole IAM role. You specify it in the following steps.
74. To find the ARN of an IAM role, open the IAM management console, and choose **Roles**.
75. In the search box, enter **salesAnalysisReportRole**, and choose the role name. The **Summary** page includes the **ARN**.
76. Copy and paste the ARN to a text editor document.
77. Next, you use the Lambda create-function command to create the Lambda function and configure it to use the salesAnalysisReportRole IAM role.

To do this, at the terminal window command prompt, paste the following command. Replace *<salesAnalysisReportRoleARN>* with the value of the salesAnalysisReportRole ARN that you copied in a previous step, and replace *<region>* with the `us-west-2` Region code. This is the Region where you created the previous Lambda function. To find this code, in the upper-right menu of the Lambda management console, choose the Region dropdown menu.

```
aws lambda create-function \

--function-name salesAnalysisReport \

--runtime python3.9 \
```

```
--zip-file fileb://salesAnalysisReport-v2.zip \

--handler salesAnalysisReport.lambda_handler \

--region <region> \

--role <salesAnalysisReportRoleARN>
```

Once the command completes, it returns a JSON object describing the attributes of the function. You now complete its configuration and unit test it.

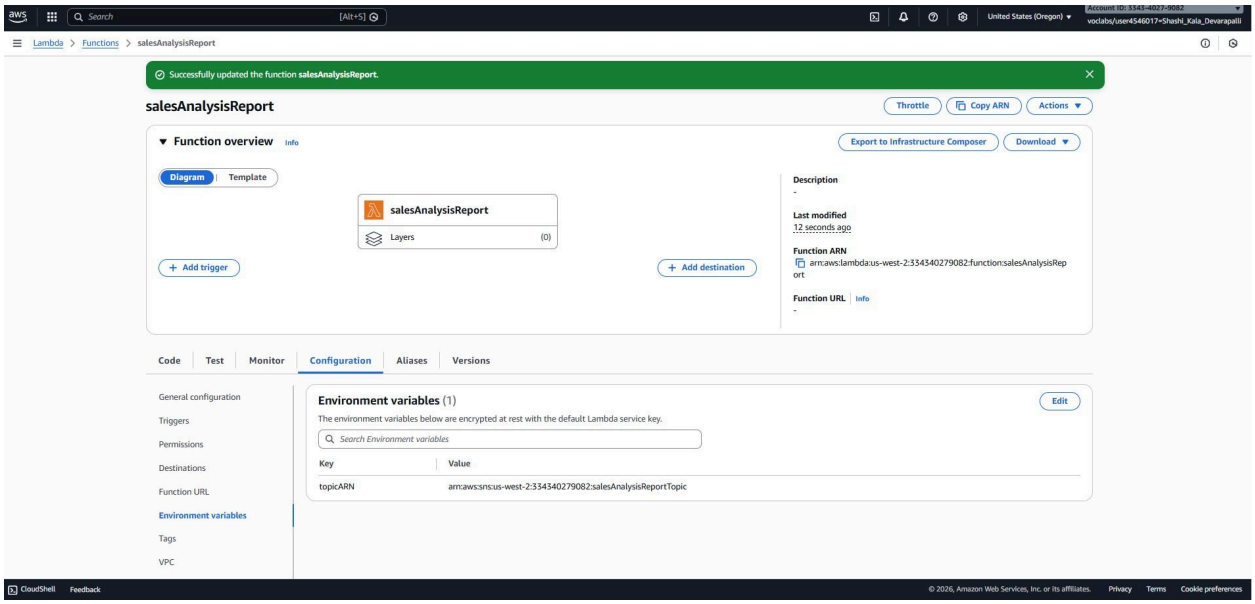


Task 5.4: Configuring the salesAnalysisReport Lambda function

76. Open the Lambda management console.
77. Choose **Functions**, and then choose **salesAnalysisReport**.
    The **Details** page for the function opens.
78. Review the details in the **Function overview** and **Code source** panels for the created function.
    In particular, read through the function code, and use the embedded comments to help you understand the logic.
    Notice on line 26 that the function retrieves the ARN of the topic to publish to, from an environment variable named **topicARN**. Therefore, you need to define that variable in the **Environment variables** panel.
79. Choose the **Configuration** tab, and choose **Environment variables**.
80. Choose **Edit**.
81. Choose **Add environment variable**, and configure the following options:

- **Key**: Enter `topicARN`
- **Value**: Paste the ARN value of the salesAnalysisReportTopic SNS topic that you copied earlier.

82. Choose **Save**.
    The following message appears: "Successfully updated the function salesAnalysisReport."



Task 5.5: Testing the salesAnalysisReport Lambda function

You are now ready to test the function.

83. Choose the **Test** tab, and configure the test event as follows:
    - For **Test event action**, choose **Create new event**.
    - For **Event name**, enter `SARTestEvent`
    - For **Template**, choose **hello-world**.
84. The function does not require any input parameters. Leave the default JSON lines as is.
85. Choose **Save**.
86. Choose **Test**.
    A green box with the message “Execution result: succeeded (logs)” appears.
    Tip: If you get a timeout error, choose the **Test** button again. Sometimes, when you first run a function, it takes a little longer to initialize, and the Lambda default timeout value (3 seconds) is exceeded. Usually, you can run the function again,

and the error will go away. Alternatively, you can increase the timeout value. To do so, follow these steps:
    - Choose the **Configuration** tab.
    - Choose **General configuration**.
    - Choose **Edit**.
    - Adjust the **Timeout** as needed.
    - Choose **Save**.
87. Choose **Details** to expand it.

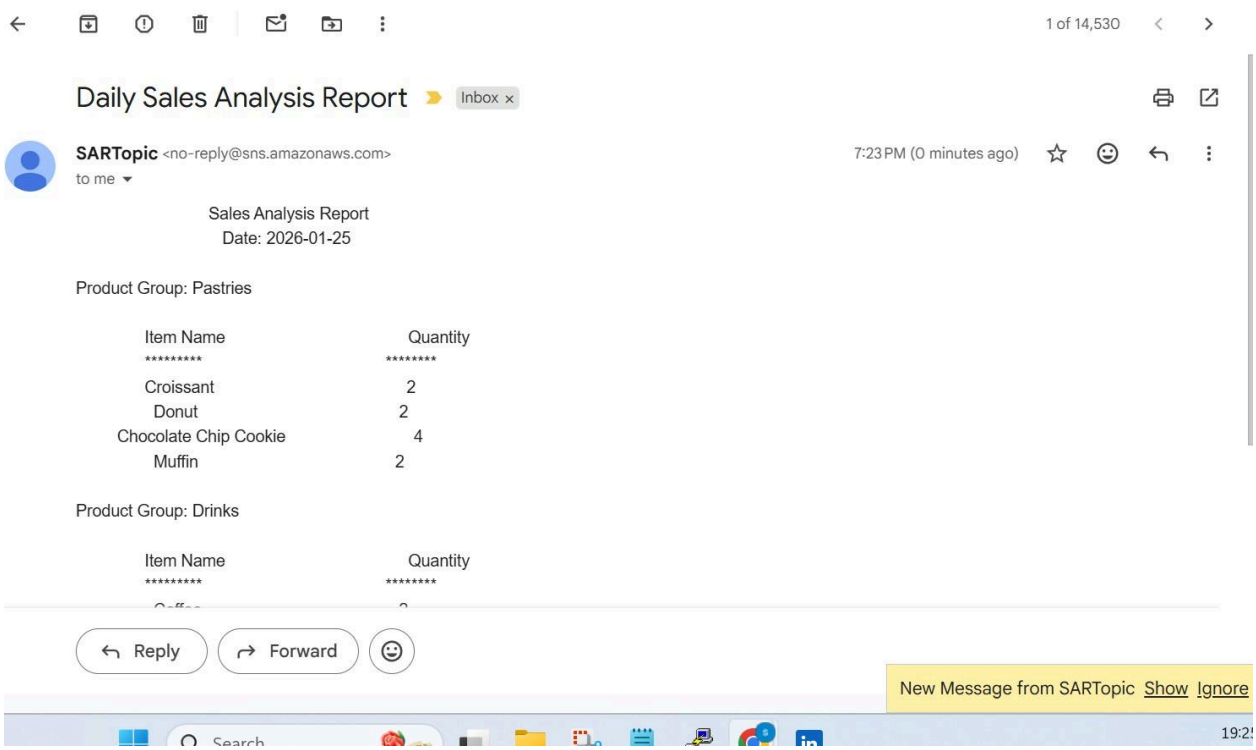
The function should have returned the following JSON object:

```
{
   "statusCode": 200,
   "body": "\"Sale Analysis Report sent.\""
}
```

87. Check your email inbox.
    If there were no errors, you should receive an email from AWS Notifications with the subject "Daily Sales Analysis Report."
    The email should contain a report that is similar to the following image depending on the orders that you placed on the café website:

88. You can place more orders on the café website and test the function to see the changes in the report that you receive.
    Great job! You have successfully unit tested the salesAnalysisReport Lambda function.



Task 5.6: Adding a trigger to the salesAnalysisReport Lambda function

To complete the implementation of the salesAnalysisReport function, configure the report to be initiated Monday through Saturday at 8 PM each day. To do so, use a CloudWatch Events event as the trigger mechanism.

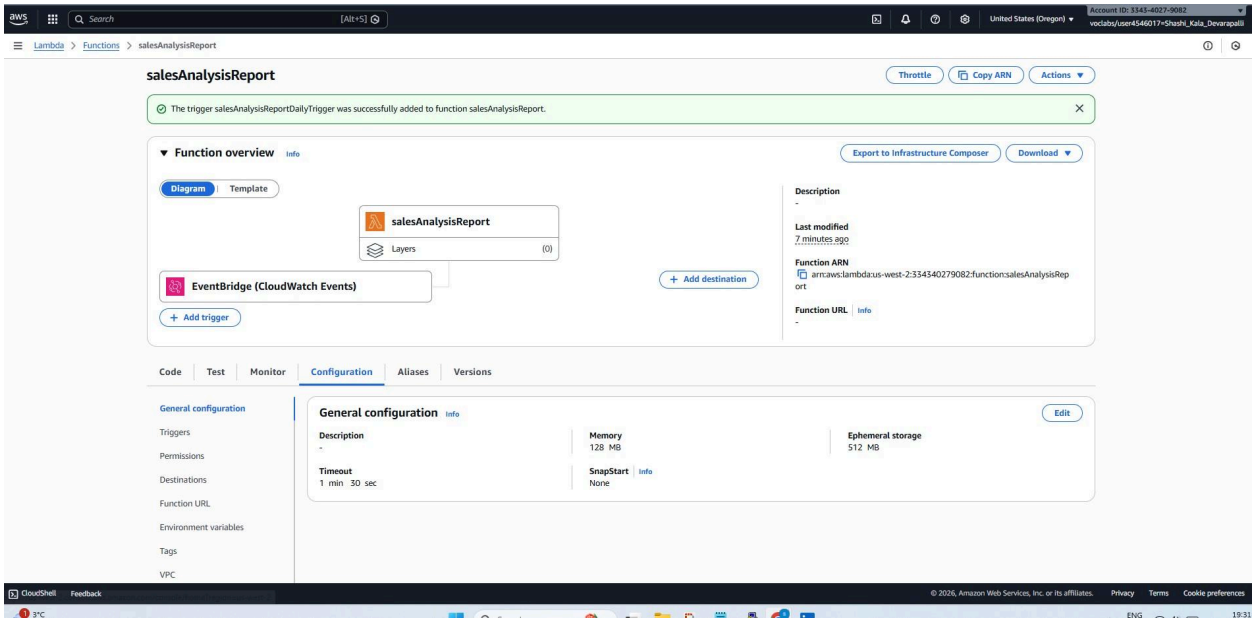
89. In the **Function overview** panel, choose **Add trigger**. The **Add trigger** panel is displayed.
90. In the **Add trigger** panel, configure the following options:
    - In the **Trigger configuration** pane, from the dropdown list, choose **EventBridge (CloudWatch Events)**.
    - For **Rule**, choose **Create a new rule**.
    - For **Rule name**, enter `salesAnalysisReportDailyTrigger`
    - For **Rule description**, enter `Initiates report generation on a daily basis`
    - For **Rule type**, choose **Schedule expression**.
    - For **Schedule expression**, specify the schedule that you would like by using a Cron expression. The general syntax of a Cron expression requires six fields separated by spaces as follows:

        - cron(Minutes Hours Day-of-month Month Day-of-week Year): In addition, all times in a Cron expression are based on the UTC time zone.
    - **Note:** For testing purposes, enter an expression that schedules the trigger 5 minutes from the current time. You can use the following examples:
        - If you are in London (UTC time zone), and the current time is 11:30 AM, enter the following expression:
          `cron(35 11 ? * MON-SAT *)`
        - If you are in New York (UTC time zone -5 during Eastern Standard Time), and the current time is 11:30 AM, enter the following expression:
          `cron(35 16 ? * MON-SAT *)`
    - This Cron expression schedules the event to be invoked at 11:35 AM Monday through Saturday.
91. **Tip**: For more information about the syntax of schedule expressions for rules, see Schedule Expressions for Rules.
    To get the correct UTC time, navigate to any browser and enter `UTC time`
92. Choose **Add**.

The new trigger is created and displayed in the **Function overview** panel and **Triggers** pane.

92. Consider the following challenge question, and adjust the Cron expression as needed: What should the Cron expression be when you deploy the function in production? Remember that you need to schedule the trigger every day, Monday through Saturday. Assume the you are in the UTC time zone.
93. Wait until 5 minutes have elapsed, and then check your email inbox.

If there were no errors, you should see a new email from AWS Notifications with a subject of "Daily Sales Analysis Report." The CloudWatch Events event invoked this message at the time that you specified in the Cron expression.



Conclusion

Congratulations! You now have successfully done the following:

- Recognized necessary IAM policy permissions to enable a Lambda function to other AWS resources
- Created a Lambda layer to satisfy an external library dependency
- Created Lambda functions that extract data from database, and send reports to user.
- Deployed and tested a Lambda function that is initiated based on a schedule and that invokes another function
- Used CloudWatch logs to troubleshoot any issues running a Lambda function